

8

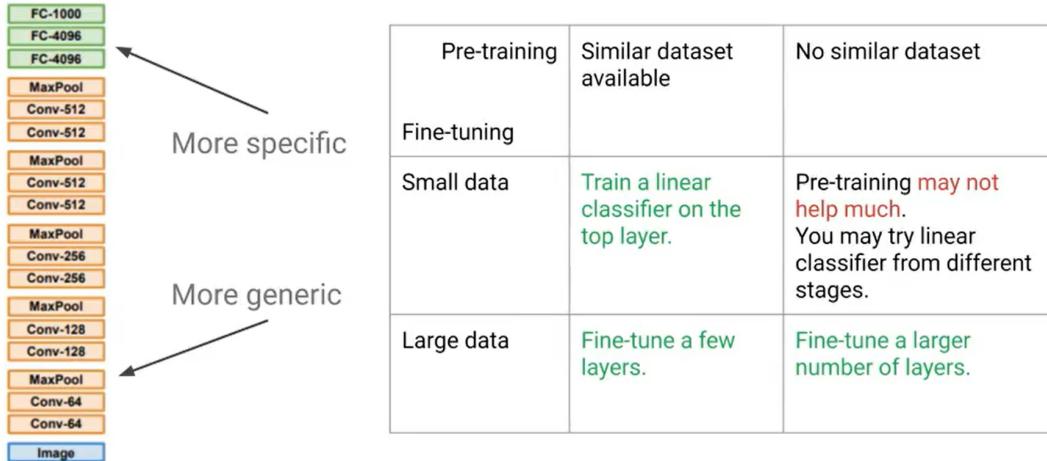
Lecture 8

Transfer Learning

- Transfer Learning
 - 한 도메인에서 학습한 모델을 다른 도메인에서 사용하는 것
 - 제일 high level에 있는 layer만 초기화하고 나머지는 그대로 사용
 - Pre-training : 타깃 데이터가 아닌 다른 큰 데이터셋에 train
 - Fine-tuning : pre train되어 있는 모델의 일부 high layer를 타깃 데이터에 train시킴
 - 예시



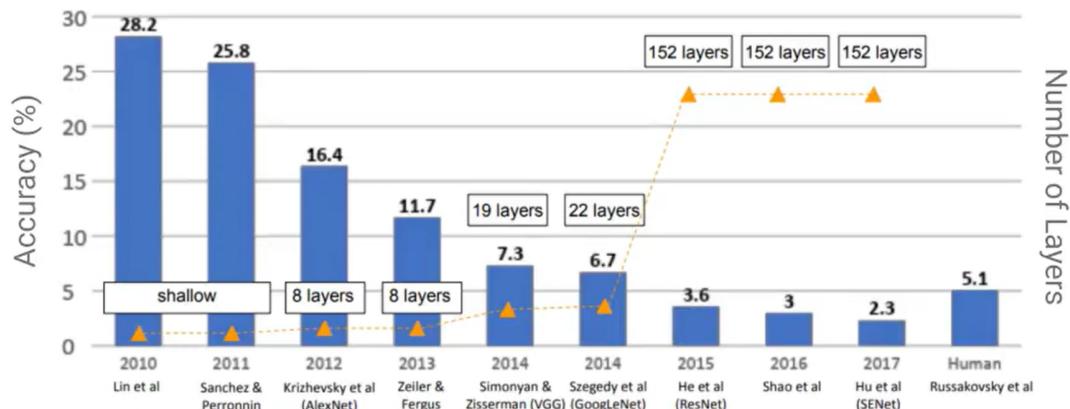
- 정리



- Pre-train은 빨리 수렴하는 것에 도움을 주는 것이지 성능을 더 높이는 것은 아니다.

CNN case study

- ImageNet Challenge



- AlexNet

Layer

CONV1

POOL1

NORM1

CONV2

POOL2

NORM2

CONV3

CONV4

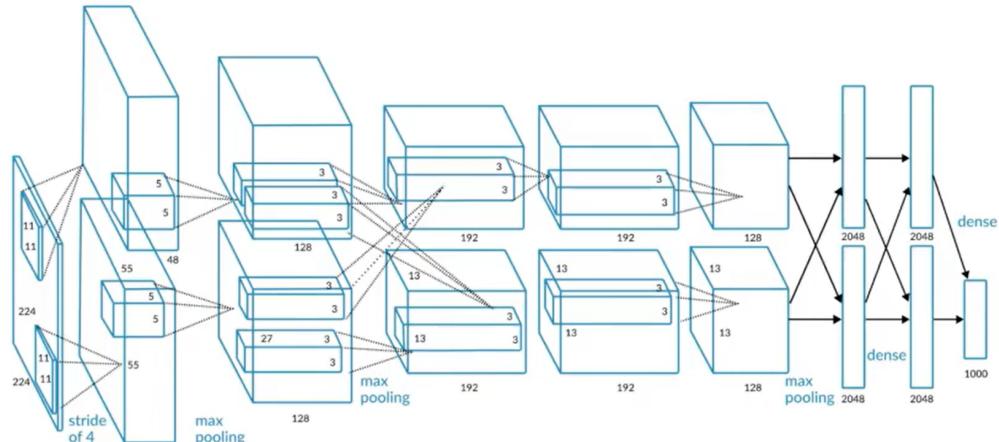
CONV5

POOL3

FC6

FC7

FC8



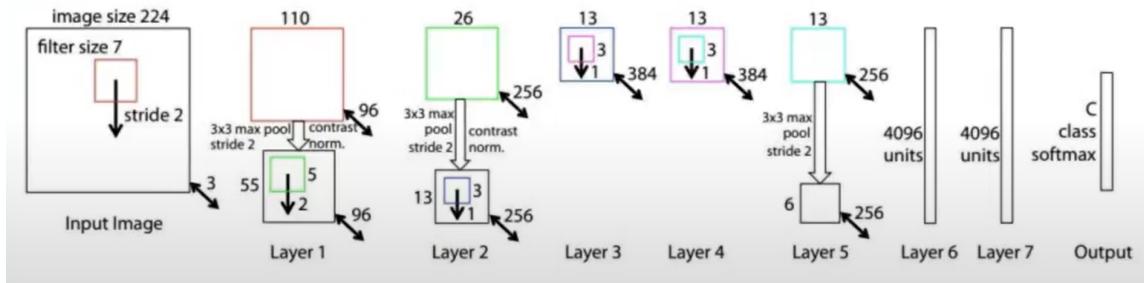
The upper and lower layers are identical in architecture. Two GPUs were used to train, running each of them, with minimal communication.

- 의의

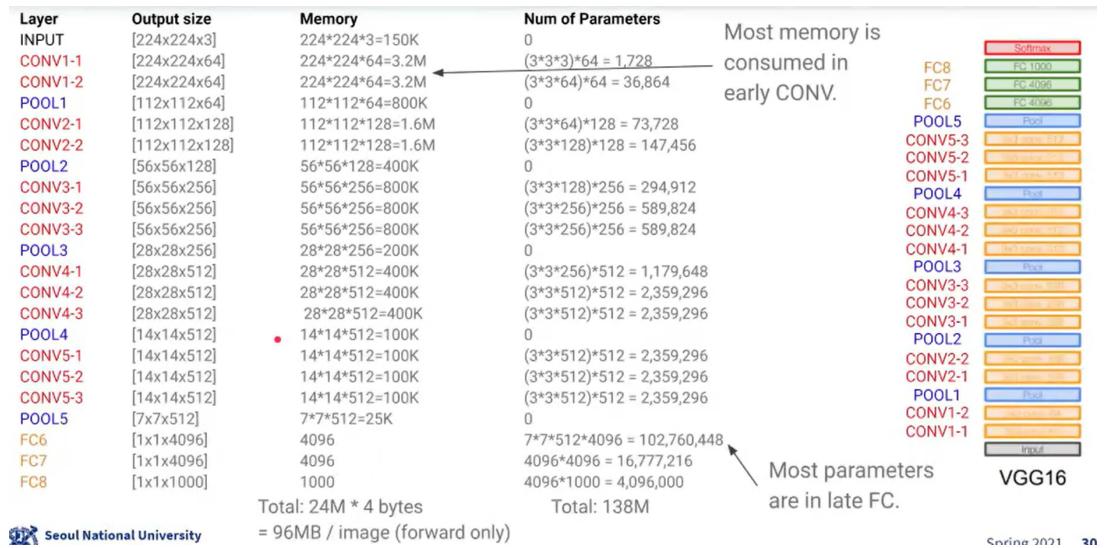
- ReLU를 처음 사용한 모델
- normalization layer를 사용함 (요즘은 성능차이가 딱히 없어서 사용 X)
- heavy data augmentation 사용
- dropout 0.5 사용 / batchsize 128로 사용 / optimization : SGD+Momentum 0.9
- initial learning rate : 0.01로 사용하고 validation accuracy가 plateau되면 1/10으로 줄임
- L2 regularization에서 weight decay (lambda 값)을 5e-4 사용

- ZFNet

- Minor architectural changes to AlexNet.
- ILSVRC challenge winner in the next year (2013)
 - Top 5 error 16.4% → 11.7%
- Original paper: <https://arxiv.org/pdf/1311.2901.pdf>



- VGGNet
 - 훨씬 deep한 모델을 만들고 정확도를 높인 모델
 - AlexNet과의 다른 점
 - 모두 3X3 convolutional filter 사용하는 대신 훨씬 deep하게 만들었다.
(8 → 16/19 layer)
 - WHY??
3x3 conv를 두 층 쌓는 것은 5x5 conv layer 하나와 같은 효과를 갖는다.
 - filter의 크기를 3X3으로 함으로써 파라미터 수를 더 적게 할 수 있음
 - 구조적인 이점 : non-linearity가 더 늘어나기 때문에 더 복잡한 것을 수용할 수 있음
 - 전체 architecture

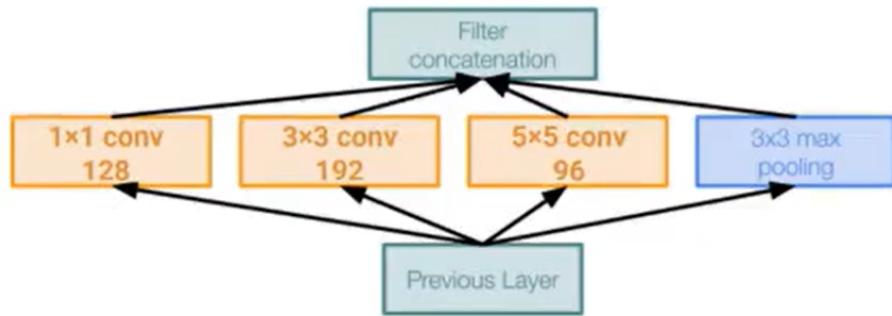


- detail

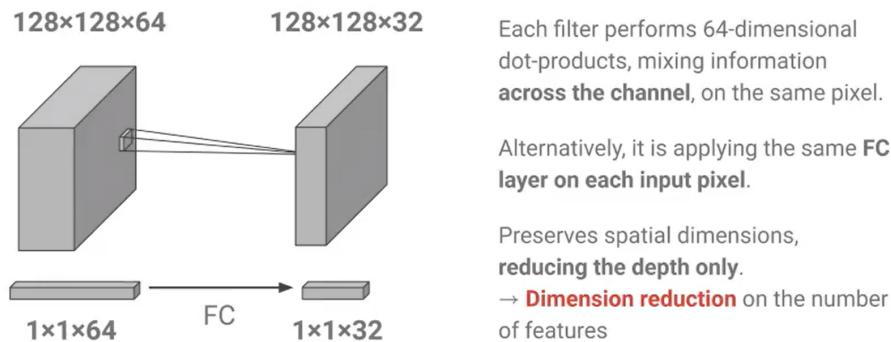
- normalization 사용 X
- VGG19가 VGG16보다 쪽금 더 좋다
- FC7이 다른 task에서도 generalization 잘됨
- FC6과 7에서 dropout은 0.5 / batchsize 256 / optimization도 Alexnet과 다르지 않음

- GoogleNet

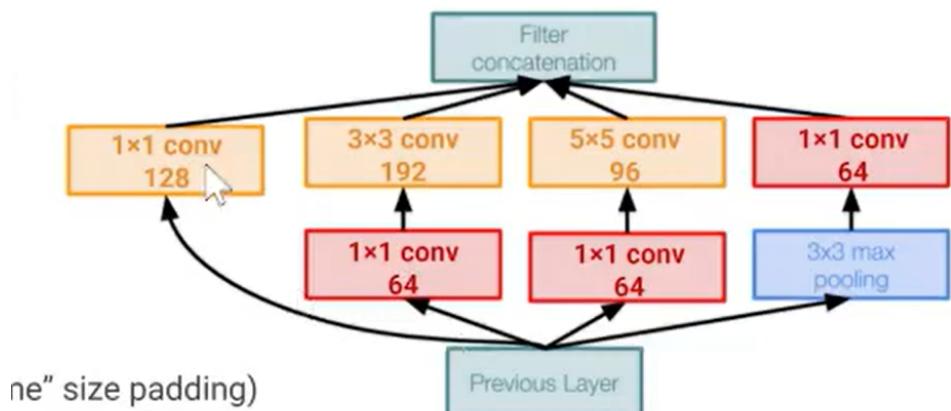
- layer가 22개 + 정확도가 좀 늘어남
- inception module을 여러 개 쌓아 만들
- 기본적인 IDEA
 - 각 layer에서 1x1, 3x3, 5x5 conv filter를 모두 사용 후 3x3 max pooling 진행 함
→ 더 flexible해지고 성능이 더 좋아진다는 장점 but 계산량이 많아진다는 단점



- 계산량이 많아지는 단점을 bottleneck layer를 이용해 해결
 - 1x1 layer은 channel-wise한 효과를 갖고 옴 (dimensional reduction)
- Solution: “bottleneck” layers that use 1×1 convolutions to reduce channel size.



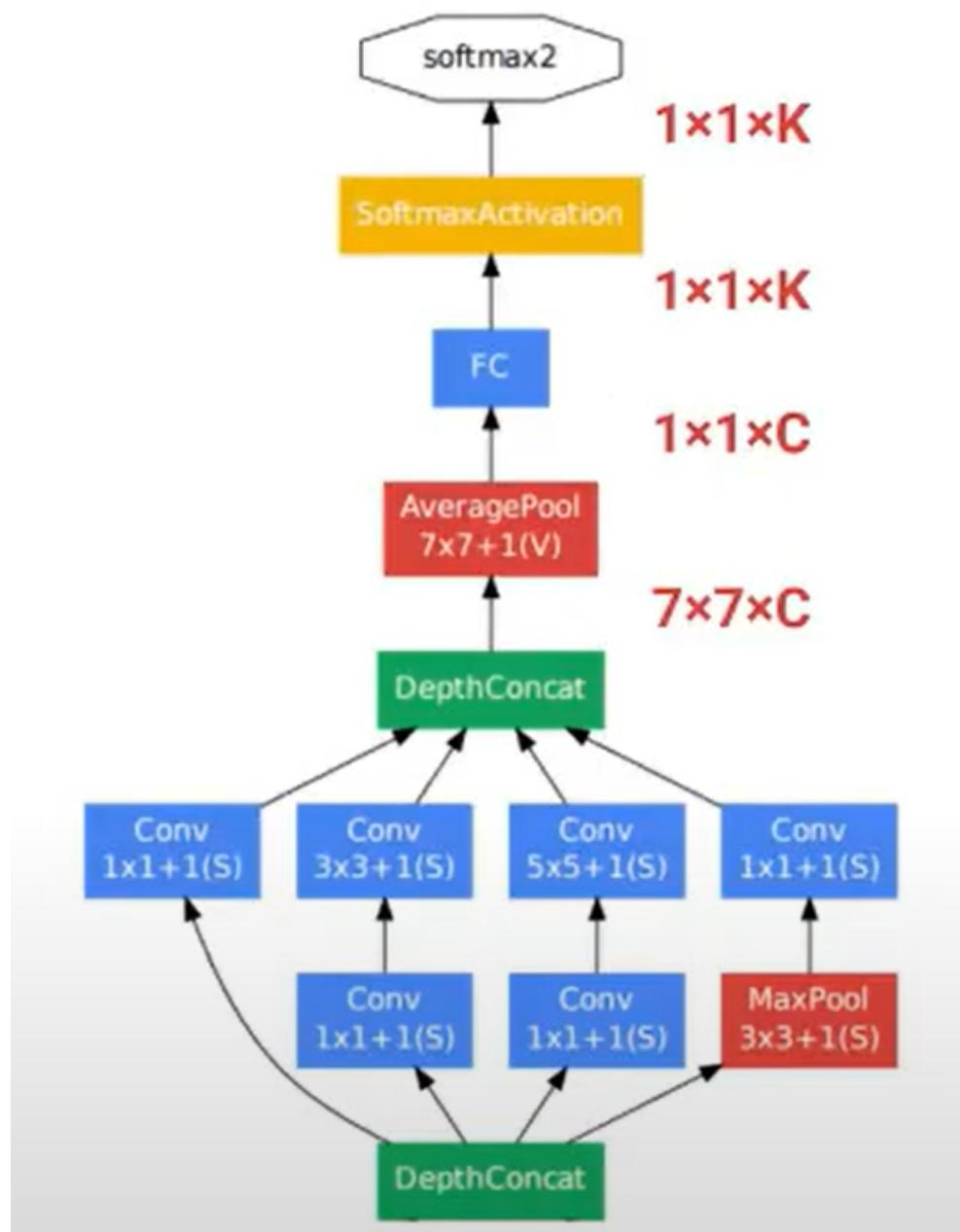
→ architecture



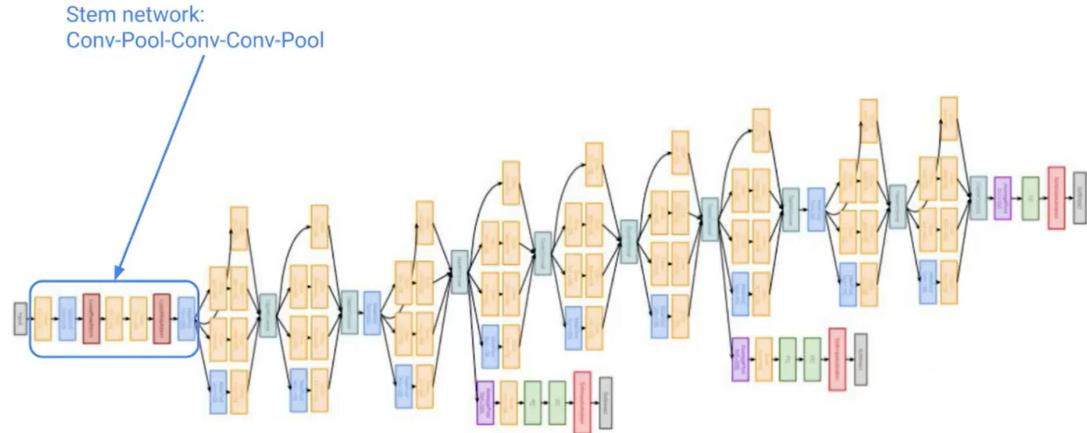
- 마지막엔 global average pooling을 진행함
 - VGGNet에서 했던 것처럼 마지막 FC layer 계산량이 heavy하다보니까 그냥 평균 냄

한번만 fully connecteded함으로써 계산량 줄임

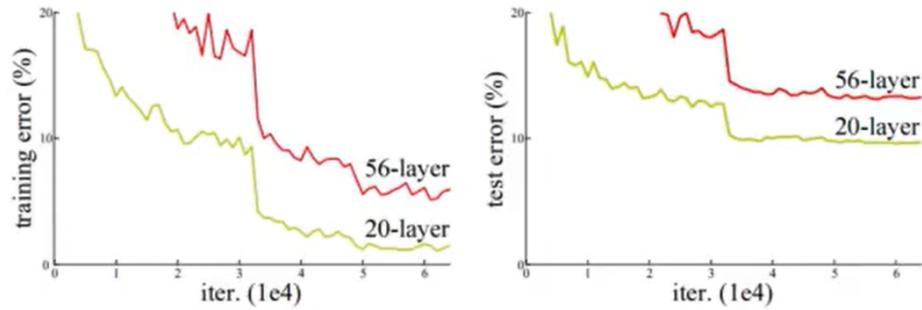
→ architecture



- 최종 architecture



- input layer
- stem network : conv-pool-conv-conv-pool
- 9개의 inception module
- 마지막으로 classification output layer
- 3층과 6층 끝났을 때 auxiliary classification output layer를 부가적으로 넣어 줌
→ WHY? gradient가 갈수록 줄어들기 때문에 6층과 9층에서 한번씩 loss 확인해봄
- 정리
 - 22층의 layer (9개의 inception module x2 + 3 conv in stem network + 1 FC)
 - 효율성이 굉장히 높아지고 inception module을 사용해 한 층을 다각화
 - 8번 epoch 돌릴 때마다 learning rate을 4퍼센트씩 줄임
- ResNet
 - 마이크로소프트에서 나온 모델
 - 22개에서 152개의 layer를 가짐 + error 또한 2배 가까이 줄어듦
 - ResNet의 아이디어는 거의 모든 곳에서 사용될 정도로 중요함
 - deeper한 모델들은 이론적으로 필요 없는 파라미터들은 사용하지 않으면 된다!
→ 근데 실험 결과 shallow할수록 더 좋은 성능을 갖는다



→ WHY?

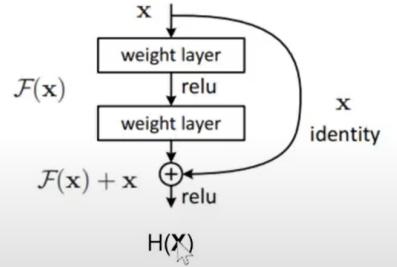
일단 overfitting 문제는 아님 (test error가 증가하는건 아니니까)

가설 : 더 깊어진 모델은 training하기가 어렵다 (hard to optimize)

- 해결 방안

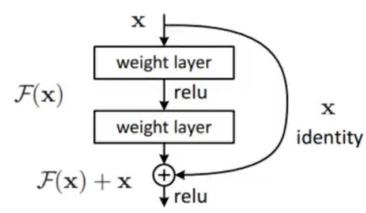
학습하는 모델의 패턴이 복잡하지 않는다면 identity shortcut을 만들어 줘서 넘어 가도록 함

- The input x is always added to the output, and the layers are to **fit the residual**, $H(x) - x$, instead of $H(x)$ directly.
- Much **easier to optimize** → easier to go **deeper** ($22 \rightarrow 152$ layers).



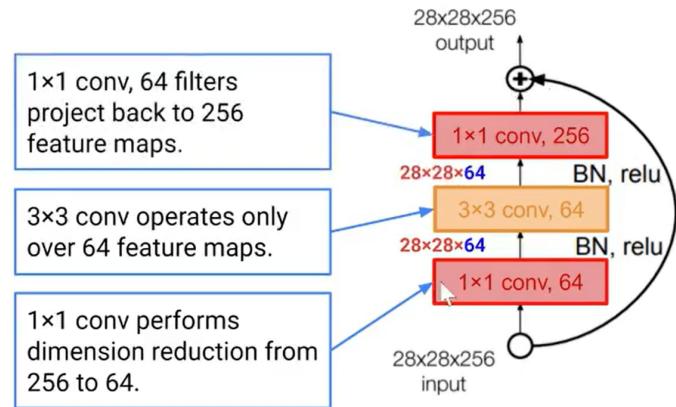
- 전체 architecture

- Stack **residual blocks** up to ~152 layers.
- Each residual blocks has 3×3 conv layers.
- Periodically, double the number of filters and downsample spatially using stride 2.
- Stem conv layers at the beginning (=GoogLeNet)
- No additional FC layers in the end (=GoogLeNet)

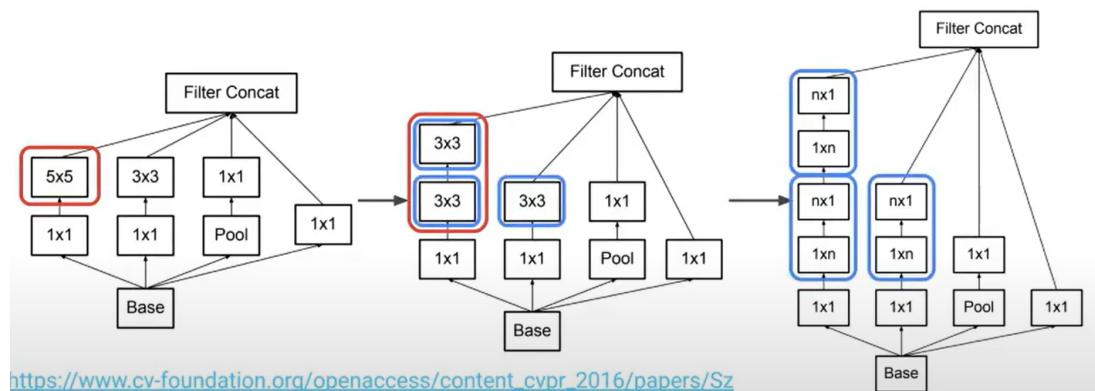


- conv feature map의 size는 2개로 줄어들지만 개수는 2배로 늘려주다가 다시 size 2배로 늘려주는 것 반복

- For deep (50+ layers) ResNets, a bottleneck layer is useful to improve efficiency.
- 1 3×3 conv per block, instead of 2.
 - This 3×3 conv is the bottleneck with smaller input/output.
- Similar to GoogLeNet.



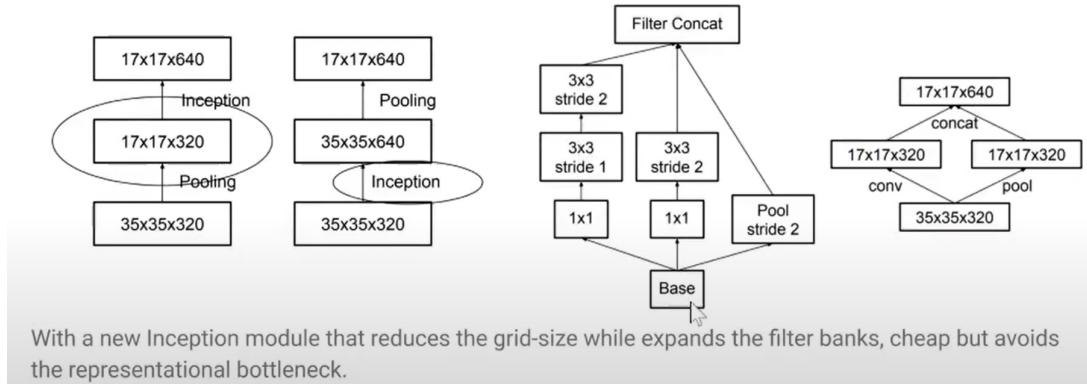
- resnet에서는 input과 output 사이즈가 무조건 같아야 함
 - 정리
 - 여러개의 버전 : 18, 34, 50, 101, 152개의 layer
 - batch normalization을 conv layer 끝날 때마다 적용
 - Xavier initializaiton
 - dropout 사용 X
- Inception-v2,3
 - 더 작은 conv layer 사용



- Grid size reduction

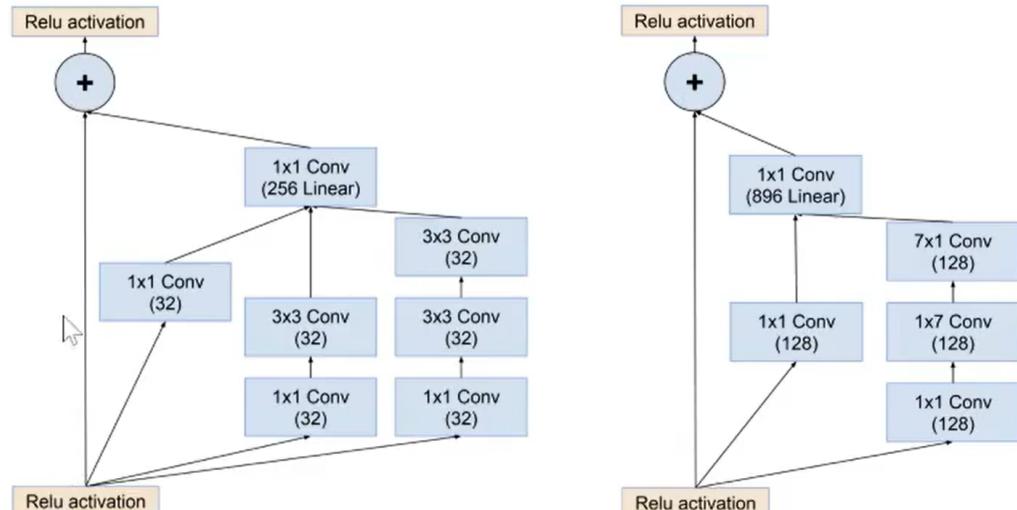
- inception을 먼저 하면 계산량 많아짐 / pooling을 먼저 하면 정보 잃음

→ 반씩 쪼개버림

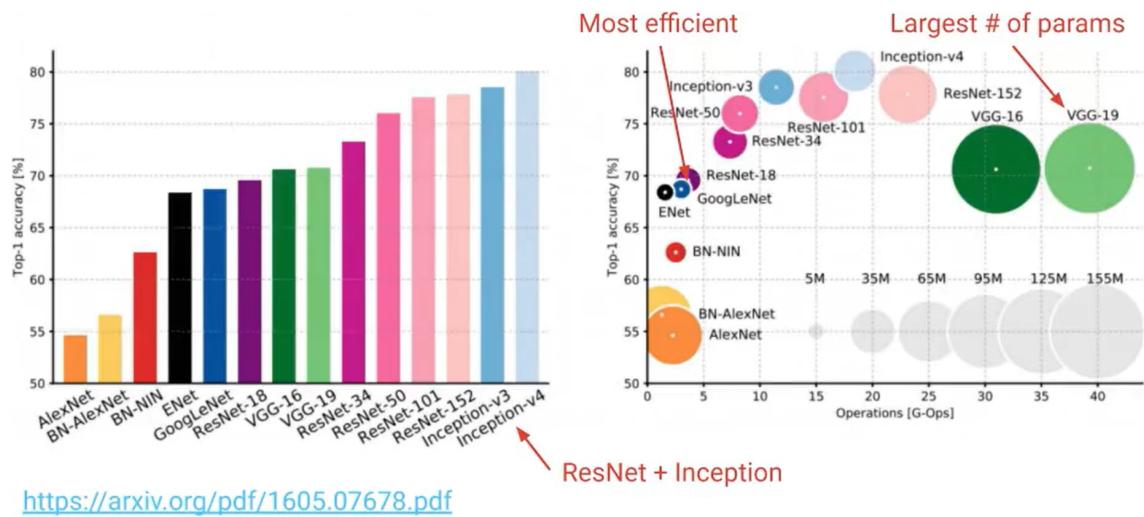


- Inception-v4

- GoogleNet과 ResNet의 combination



- Complexity 비교



<https://arxiv.org/pdf/1605.07678.pdf>