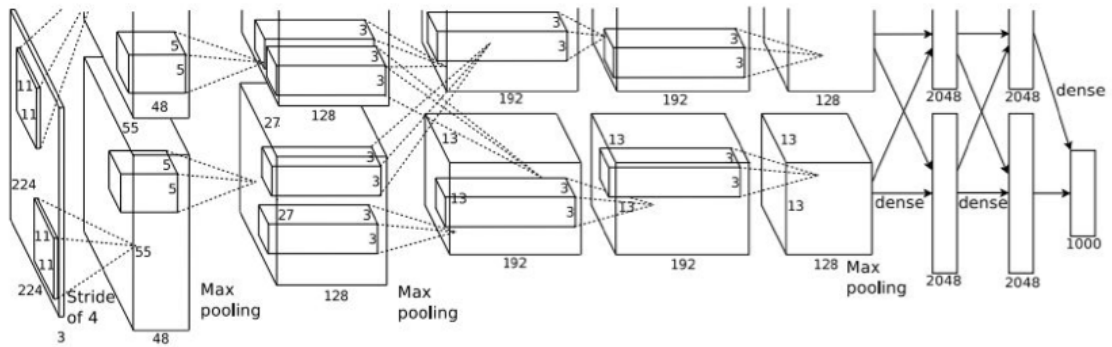


CNN Architectures

Major Architectures

1. AlexNet



a. First deep learning based ConvNet (8 layers)

b. At first layer (CONV1)

i. input image size: 227 x 227 x 3

ii. output volume size: 55 x 55 x 96

- 96개의 11 x 11 filters with stride 4

iii. total number of parameters: $11 * 11 * 3 * 96 = 35K$

c. At second layer (POOL1)

i. output volume size: 27 x 27 x 96

- 3 x 3 filters with stride 2

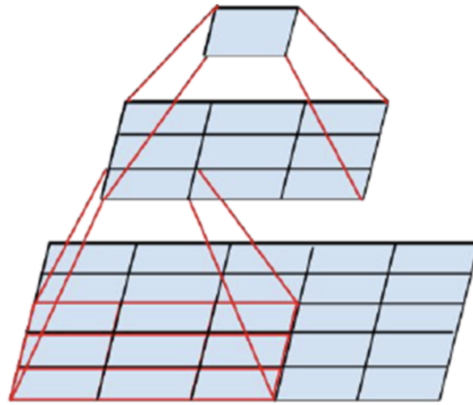
- preserve the depth

ii. total number of parameters: 0

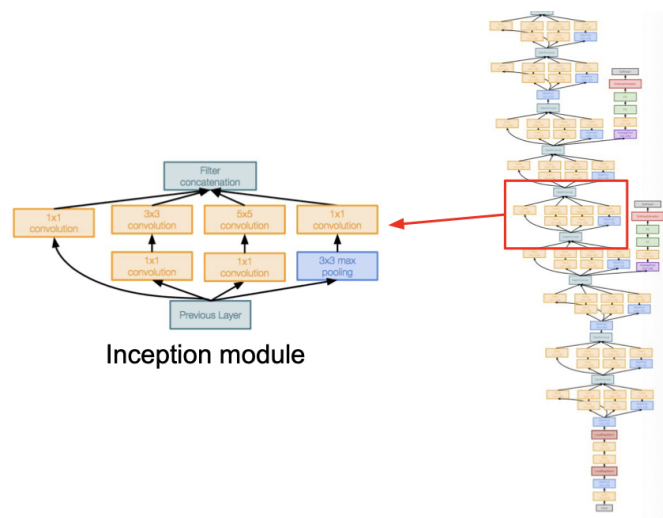
- parameters are the weights we try to learn

- but pooling just take the max value in one region

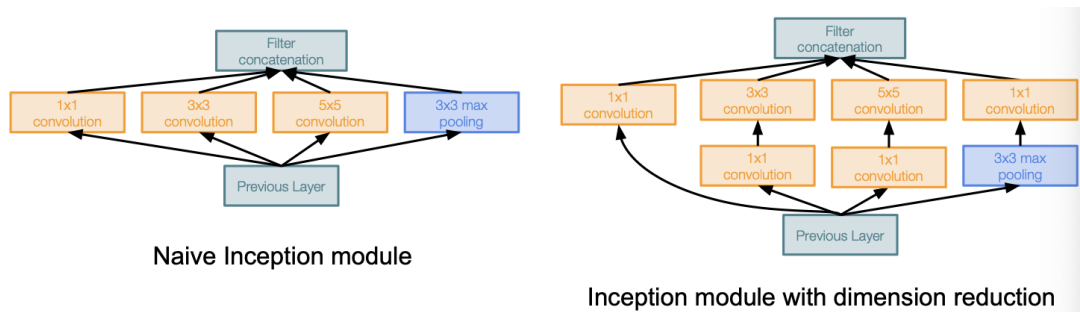
2. VGGNet



3. GoogLeNet

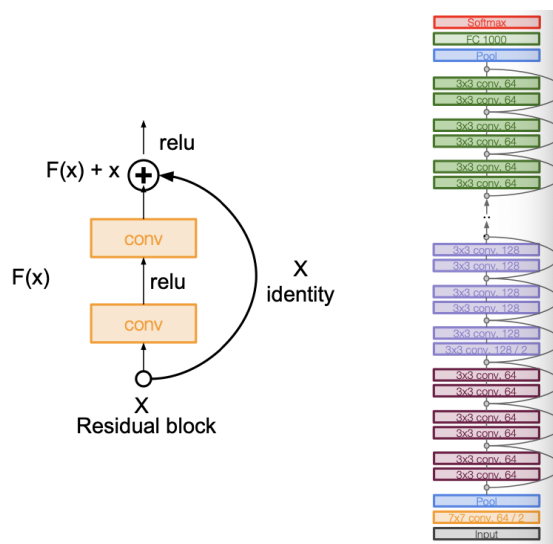


- a. Deeper network (22 layers) with computational efficiency
- b. Use ***inception module***
 - i. design a good local network topology (network within a network) and then stack these modules on top of each other
 - ii. apply parallel filter operations on the input from previous layer
 - iii. concatenate all filter outputs together depth-wise
- c. what is the problem with this?
 - i. high computational complexity
- d. Solution: use ***bottleneck layers*** (1 x 1 conv) to reduce feature depth



- i. preserve spatial dimension and only reduce depth
- ii. use zero-padding to preserve spatial dimension

4. ResNet



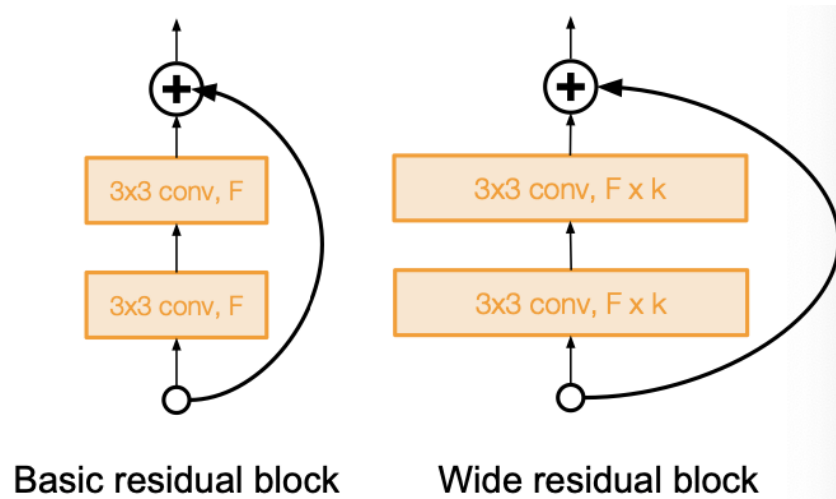
- a. Very deep networks using residual connections
 - i. what happens when we continue stacking deeper layers on a plain convolutional neural network?
 - not able to do better than shallow model
- b. Hypothesis: the problem is an optimization problem, deeper models are harder to optimize
- c. Solution: use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping
 - i. plain layers: $x \rightarrow F(x) \rightarrow H(x)$
 - ii. residual block: $x \rightarrow F(x) \rightarrow F(x) + x (= H(x))$
- d. For deeper networks, use bottleneck layer to improve efficiency

Minor Architectures

1. NiN (Network in Network)

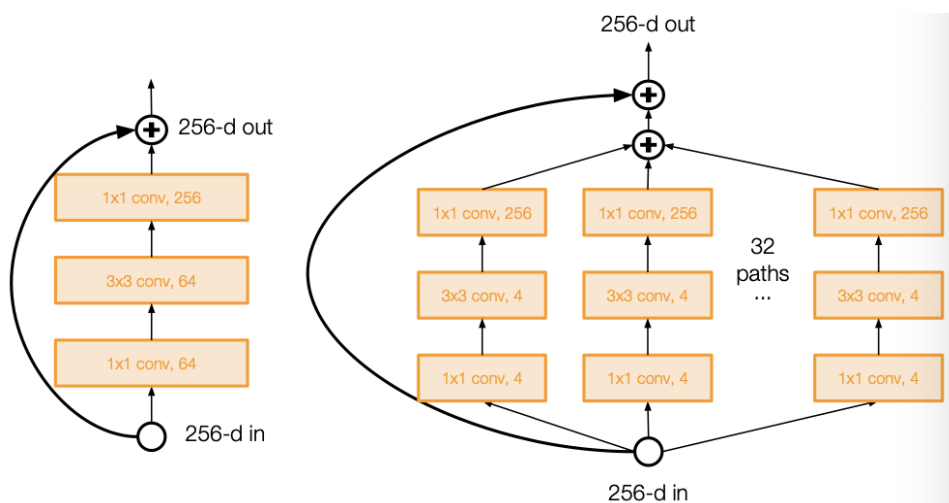
- a. Use micronetwork within each conv layer to compute more abstract features for local patches
 - i. micronetwork uses multilayer perceptron

2. Wide ResNet



- a. Use wider residual blocks
 - i. $F \times k$ filters instead of F filters in each layer
- b. Increase width instead of depth for computational efficiency

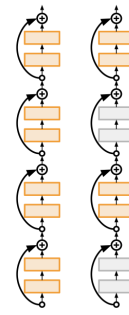
3. ResNeXt



- a. Increase width of residual block through multiple parallel pathways

4. Stochastic Depth

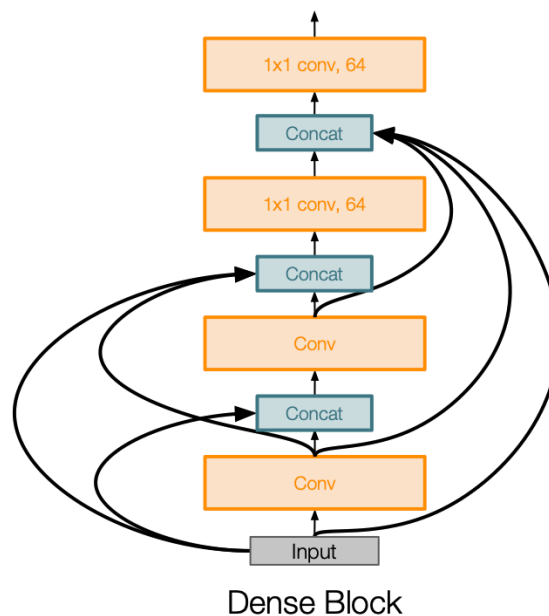
- a. Motivation: reduce vanishing gradients and training time through short networks during training
- b. Randomly drop a subset of layers during each training pass



5. FractalNet

- a. Use both shallow and deep paths to output
- b. Train with dropping out subpaths

6. DenseNet



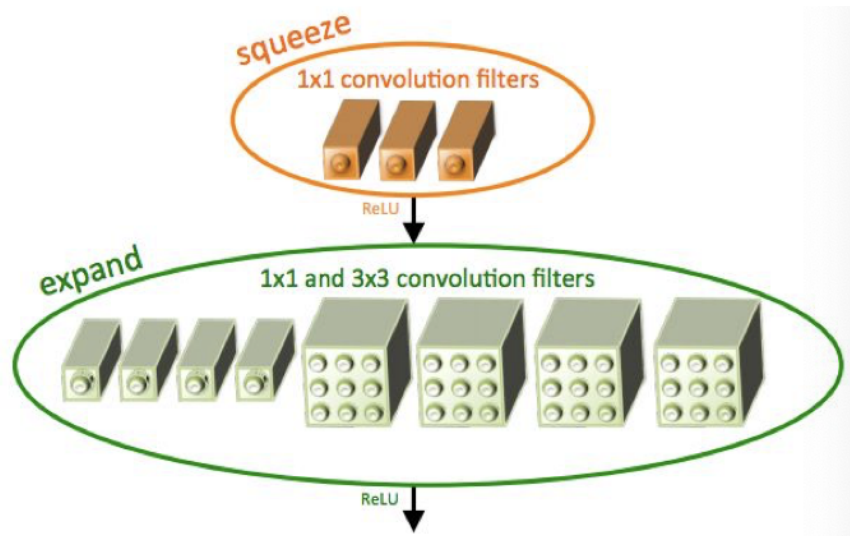
- a. Dense blocks of each layer are connected to every other layer in feedforward fashion
- b. Each layer obtain additional information from previous layers and feature maps are concatenated, not summed
 - i. 극도로 deep 하거나 wide 한 구조로부터 representational power 를 끌어내는 대신 feature 의 재사용을 통해 네트워크의 잠재력을 활용함
 - ii. 따라서 학습하기 쉽고 효율적인 parameter 를 가진 압축된 모델로 만들어졌고, 후속 layer의 input 에 variation 을 주면서 효율을 개선함

- iii. 위 부분들이 ResNet 과의 주요 차이점이며 inception module 보다 더 간단하고 효율적일 수 있는 이유라고 함

c. Advantages

- i. alleviate vanishing gradient
- ii. strengthen feature propagation
- iii. encourage feature reuse

7. SqueezeNet



- a. Consist of a squeeze layer with 1 x 1 filters feeding an expand layer with 1 x 1 and 3 x 3 filters