FastCampus Pytorch

Ch6. Addvanced CNN

HARRY KIM

Lecture Content

1. AlexNet
2. VGGNet
3. GoogLeNet
4. ResNet
5. Inception
6. Transfer Learning

강의 자료

- Papers
  - ImageNet Classification with Deep Convolutional Neural Networks
  - VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION
  - Going deeper with convolutions
  - Deep Residual Learning for Image Recognition
  - Rethinking the Inception Architecture for Computer Vision

1. AlexNet

AlexNet

- **AlexNet**
  - "ImageNet Classification with Deep Convolutional Neural Networks"
  - Alex Krizhevsky et al.
  - ILSVRC 2012 1st

Fast campus

AlexNet

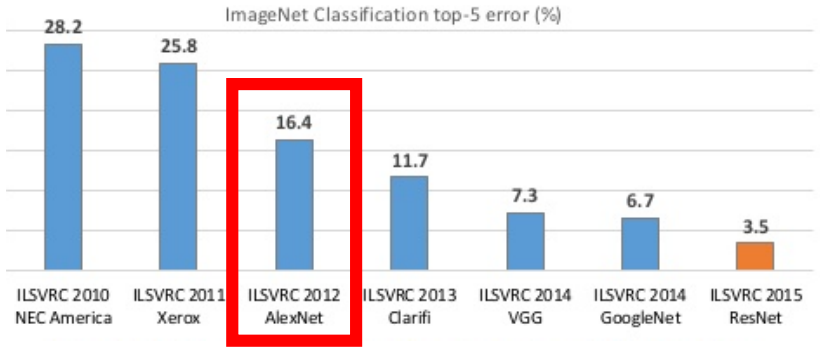
- **Imagenet**
  - 그러던 중 2009년 Imagenet Dataset이 등장함
    - 총 1.6천만개의 데이터
    - 100,000개의 Class로 이루어진 이미지들
    - 응급차, 달마시안, 이집트고양이 등등
  - Imagenet Large Scale Visual Recognition Challenge (ILSVRC)
    - 1백만개의 이미지
    - 1000개의 Class

ImageNet Challenge

IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.

- **AlexNet**
  - 2012년 이후 딥러닝 방식이 ILSVRC 최우수 팀으로 선정됨



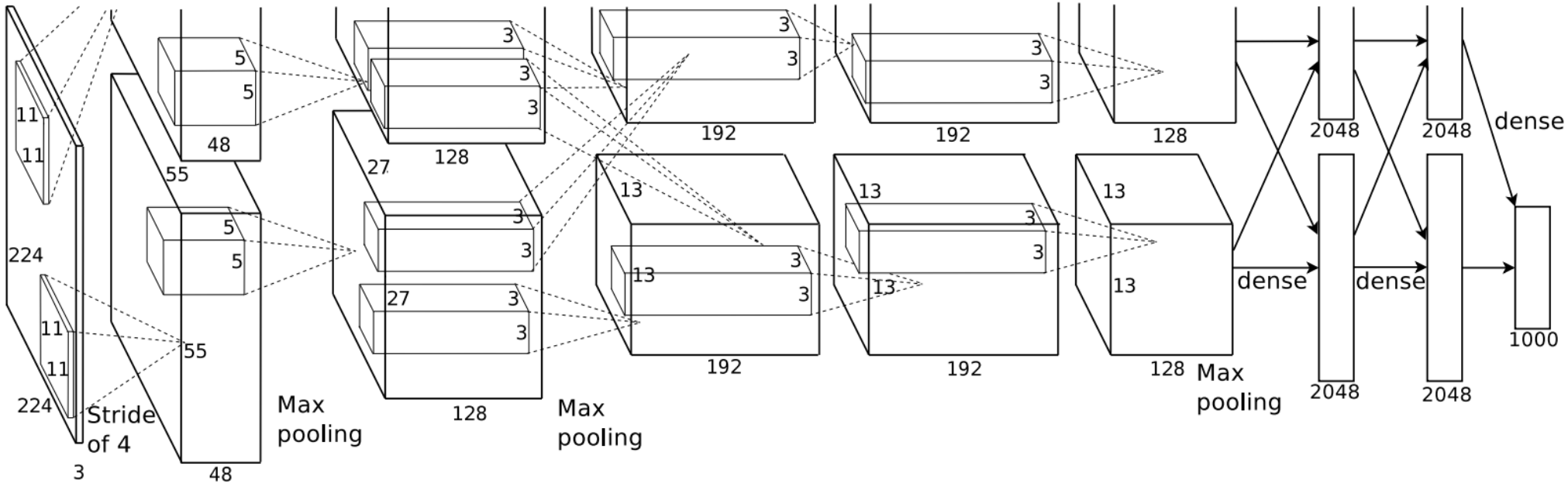
AlexNet

- **AlexNet**
  - **Input :** 227 × 227 RGB
  - **Preprocessing :** subtracting the mean RGB Value
  - **Data Augmentation** : Crop, Rescaling, Flip / PCA
  - **Optimizer** : SGD with Momentum 0.9
  - **Activation** : Relu
  - **Conv :** 5 × 5, 3 × 3
  - **Pooling :** Max pooling 2 × 2, stride 2
  - **Dropout** : 50%

Fast campus

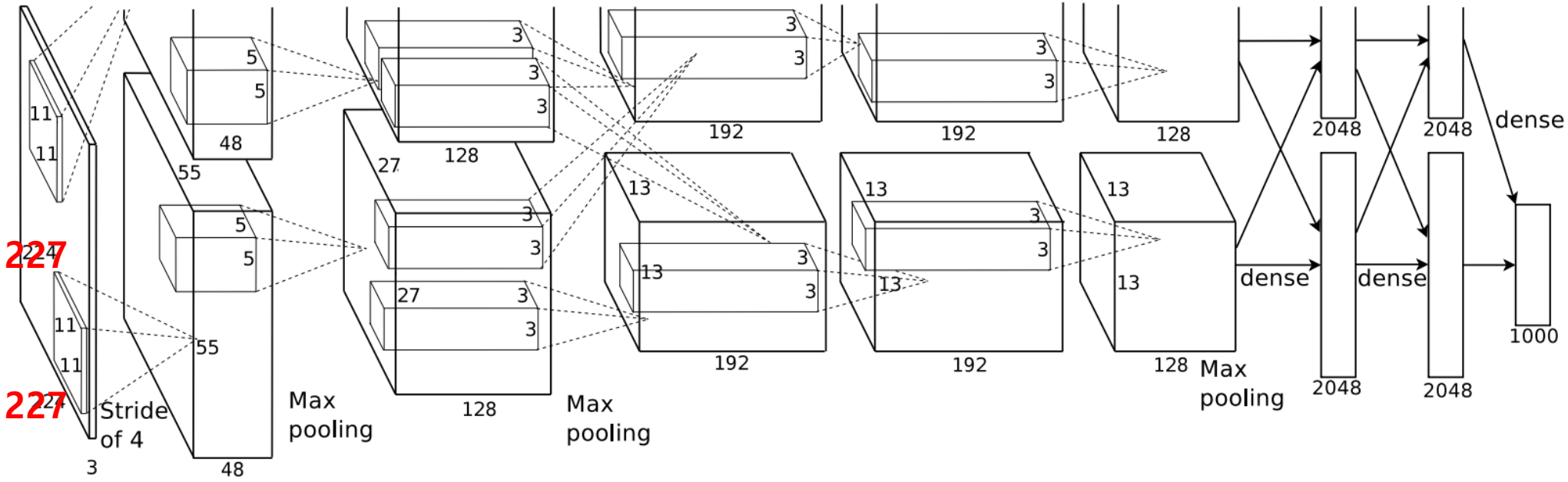
AlexNet

- **AlexNet**



AlexNet

- **AlexNet**



2. VGGNet

VGGNet

- **VGGNet**
  - "Very Deep Convolutional Networks For Large-Scale Image Recognition"
  - Karen Simonyan & Andrew Zisserman
  - ILSVRC 2014 2nd

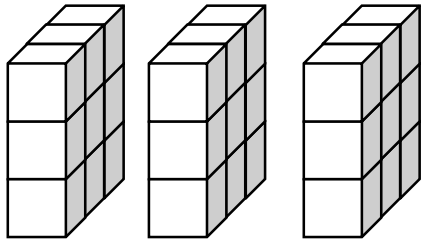
VGGNet

- **VGGNet**
  - **Input :** 224 × 224 RGB
  - **Preprocessing :** subtracting the mean RGB Value
  - **Data Augmentation** : Crop, Rescaling
  - **Optimizer** : SGD with Momentum 0.9
  - **Activation** : Relu
  - **Conv :** 3 × 3
  - **Pooling :** Max pooling 2 × 2, stride 2
  - **Dropout** : 50%

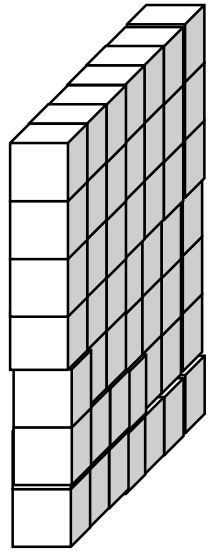
VGGNet

- **VGGNet**
  - **Feature** : 3×3 v.s. 7×7
  - 두 필터를 적용했을 때, 결과는 똑같다
  - 하지만 두 연산에 필요한 변수의 개수는 다르다

$3 \times 3 \times 3 = 27$



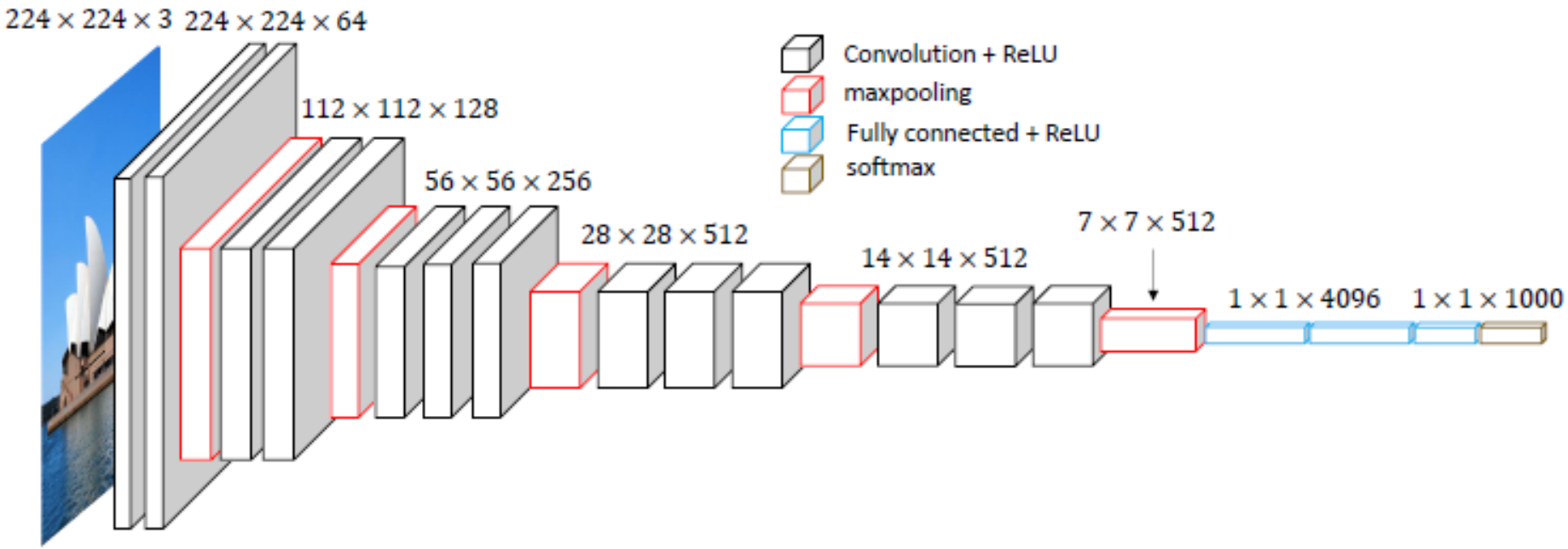
v.s.



$7 \times 7 = 49$

VGGNet

- VGGNet



VGGNet

- **VGGNet**
  - 네트워크의 깊이가 성능을 향상시키는 데에 매우 중요한 결정 요소
  - VGG-16과 VGG-19가 주로 사용
  - 매우 간단
  - 하지만 많은 메모리 사용 (144M)

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

3. GoogLeNet

GoogLeNet

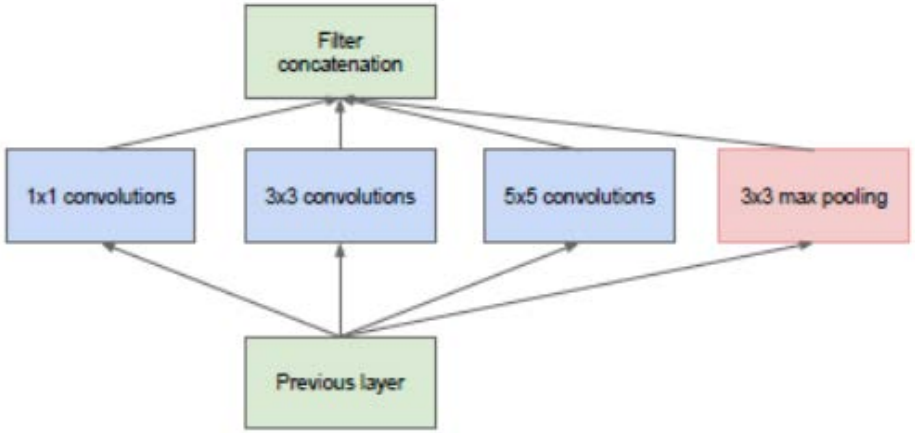
- **GoogLeNet**
  - "Going Deeper with Convolutions"
  - Szegedy et al.
  - ILSVRC 2014 1st

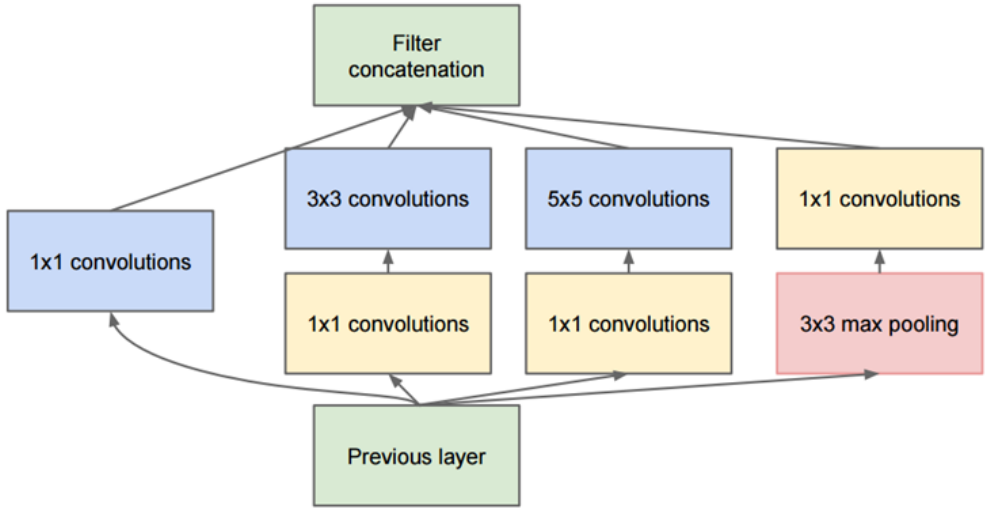
GoogLeNet

- **GoogLeNet**
  - **Input :** 224 × 224 RGB
  - **Preprocessing :** subtracting the mean RGB Value
  - **Data Augmentation** : Crop, Rescaling
  - **Optimizer** : SGD with Momentum 0.9
  - **Activation** : Relu
  - **Conv :** 7 × 7, 3 × 3, Inception
  - **Pooling :** Max pooling 3 × 3, stride 2 / Average Pooling 7 × 7, stride 1
  - **Dropout** : 40%

GoogLeNet

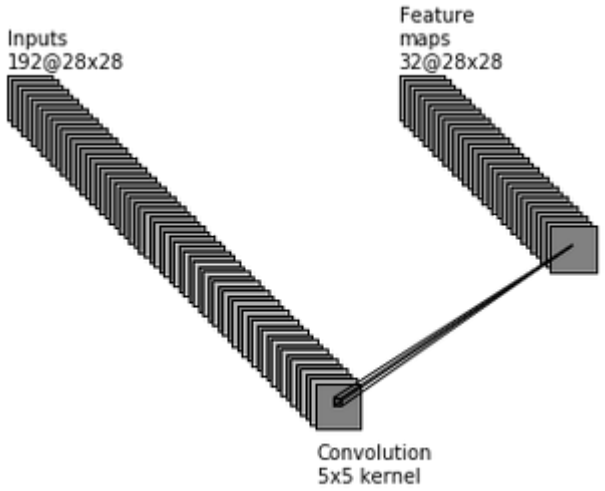
- **GoogLeNet**
  - **Feature** : Inception Module
  - 신경망은 깊을수록 좋지만, 깊으면 학습이 어려워짐 (Overfitting, Gradient Vanishing)
  - 따라서 Sparce한 신경망을 위해 Channel을 줄임
  - 1×1을 통해 이를 달성, 계산량도 대폭 감소 (5M)



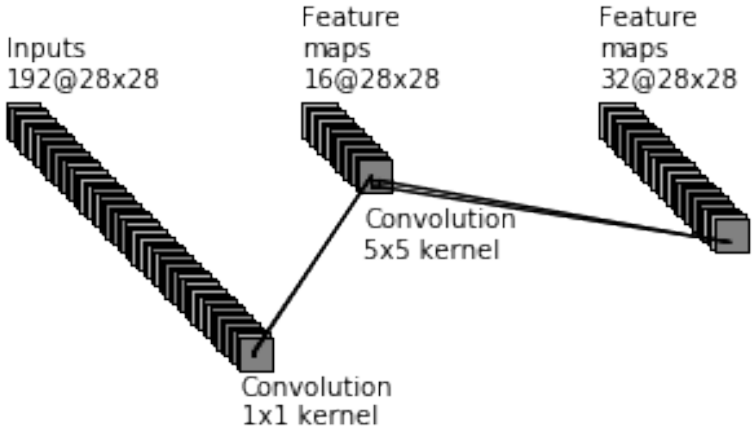


GoogLeNet

- **GoogLeNet**
  - **Feature** : Inception Module
  - 신경망은 깊을수록 좋지만, 깊으면 학습이 어려워짐 (Overfitting, Gradient Vanishing)
  - 따라서 Sparce한 신경망을 위해 Channel을 줄임
  - 1×1을 통해 이를 달성, 계산량도 대폭 감소 (5M)



120,422,400



12,443,648

https://hacktilldawn.com/2016/09/25/inception-modules-explained-and-implemented/

Fast campus

GoogLeNet

AlexNet

VGGNet

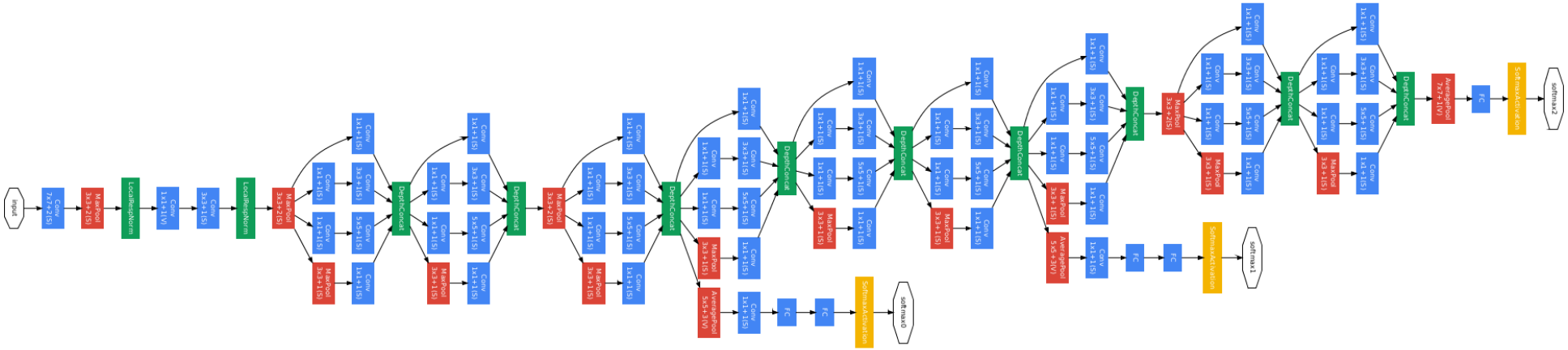
GoogLeNet

ResNet

Inception

Transfer Learning

- **GoogLeNet**
  - 이 외에도 Gradient Vanishing 방지를 위한 Auxiliary 도입 등 다양한 시도



GoogLeNet

- **GoogLeNet**
  - 총 19개의 Layer

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								

4. ResNet

ResNet

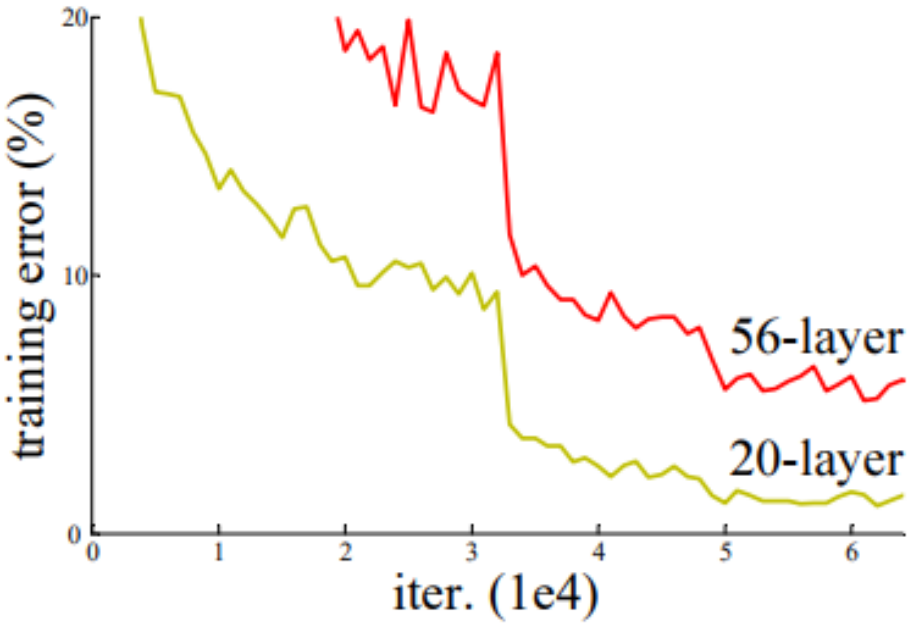
- **ResNet**
  - "Deep Residual Learning for Image Recognition"
  - Kaiming He et al.
  - ILSVRC 2015 1st

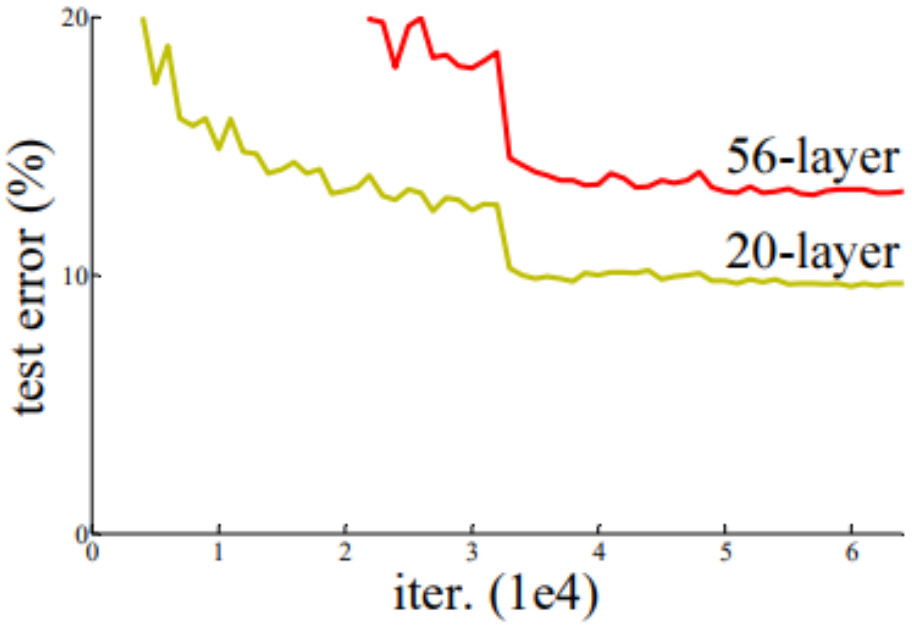
ResNet

- **ResNet**
  - **Input :** 224 × 224 RGB
  - **Preprocessing :** subtracting the mean RGB Value
  - **Data Augmentation** : Crop, Rescaling, Flip
  - **Optimizer** : SGD with Momentum 0.9
  - **Activation** : Relu
  - **Conv :** $7 \times 7, 3 \times 3$
  - **Pooling :** Max pooling 3×3, stride 2 / Average Pooling 7×7, stride 1
  - **Dropout** : Not used

ResNet

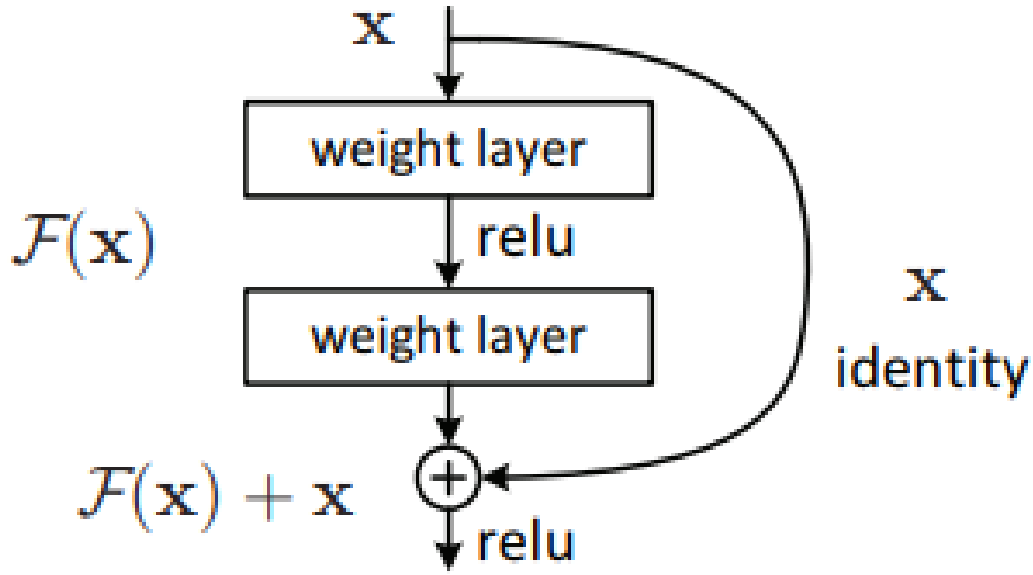
- **ResNet**
  - **Feature** : Residual Network
  - 깊게 쌓으면 좋아지는데, 여태까지는 고작 수십개가 한계
  - 깊으면 Overfitting으로 인해 안 좋아질 것 같았지만...
  - 실제로는 Degradation (정확도가 멈춰있는 상태)이 주요 원인





ResNet

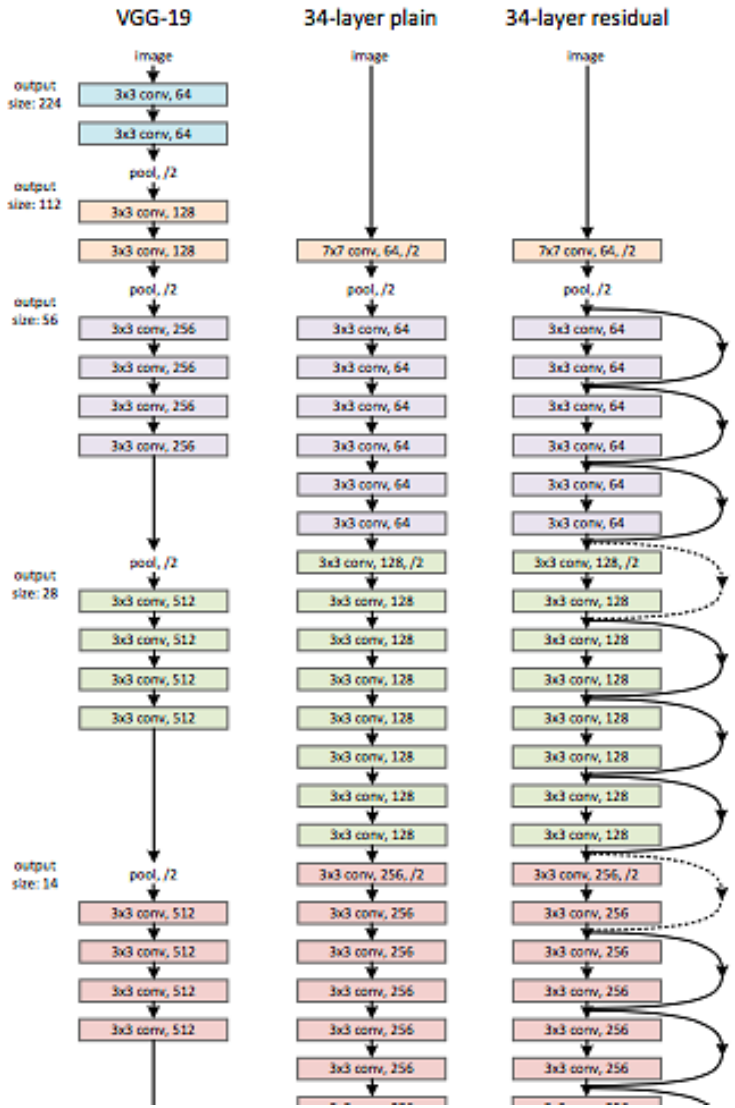
- **ResNet**
  - **Feature** : Residual Network
  - 이는 원래 값이 전달이 잘 되지 않기 때문 (특성을 너무 많이 잃어버림)
  - 따라서 특성을 더 잘 전달해주기 위해, 레이어를 건너뛰어 전달



ResNet

- **ResNet**
  - 이외에도...
  - 모든 Convolution Layer 뒤에 Batch Nomalization 사용
  - Xaiver/2 초기화
  - 매우 좋은 성능 도출

  - 하지만, Residual Network에서의 두 차원이 동일해야 함



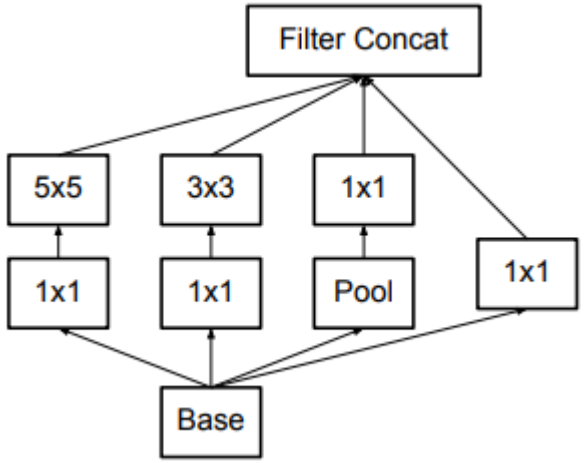
5. Inception

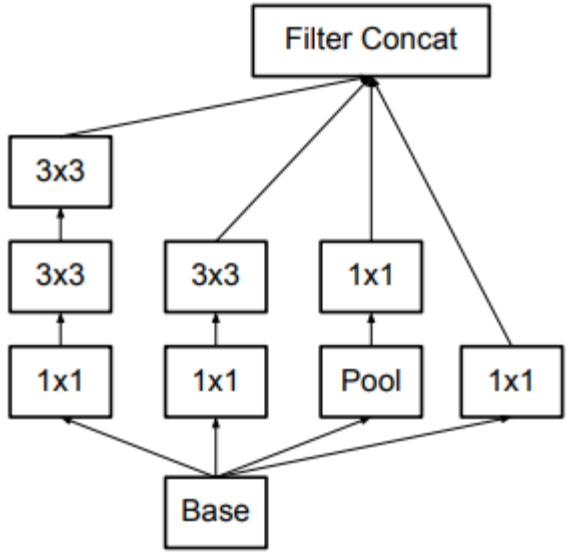
Inception

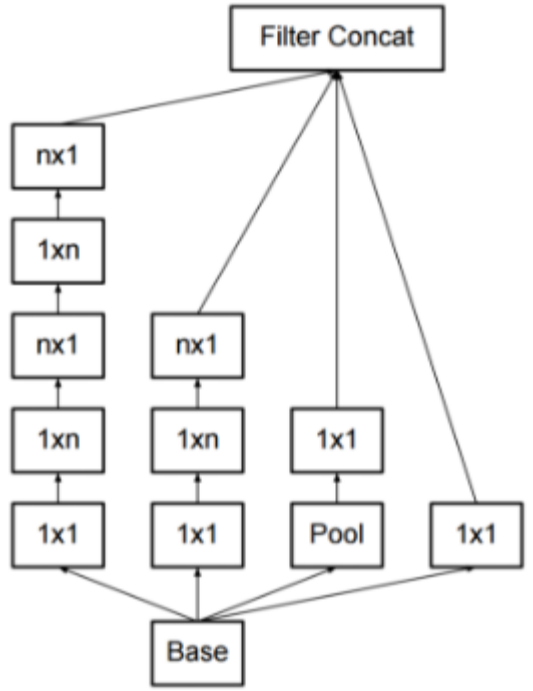
- **Inception v3**
  - "Rethinking the Inception Architecture for Computer Vision"
  - Szegedy et al.

Inception

- **Inception v2**
  - **Features :** Increasing Conv

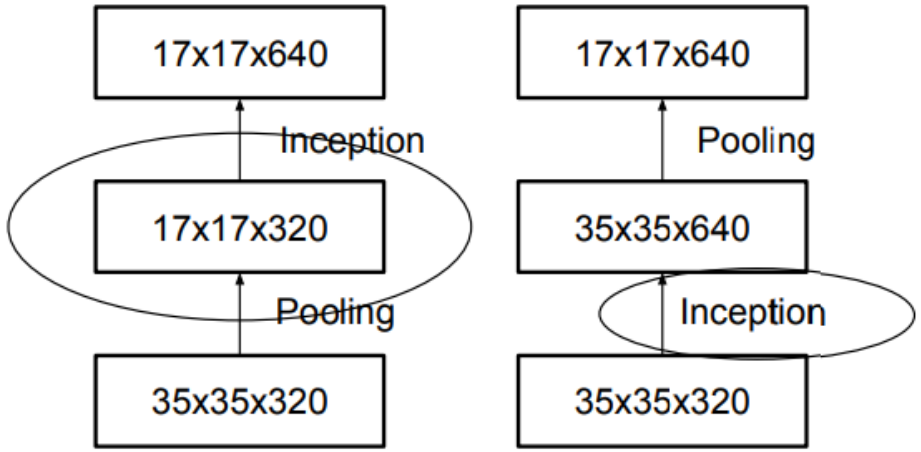


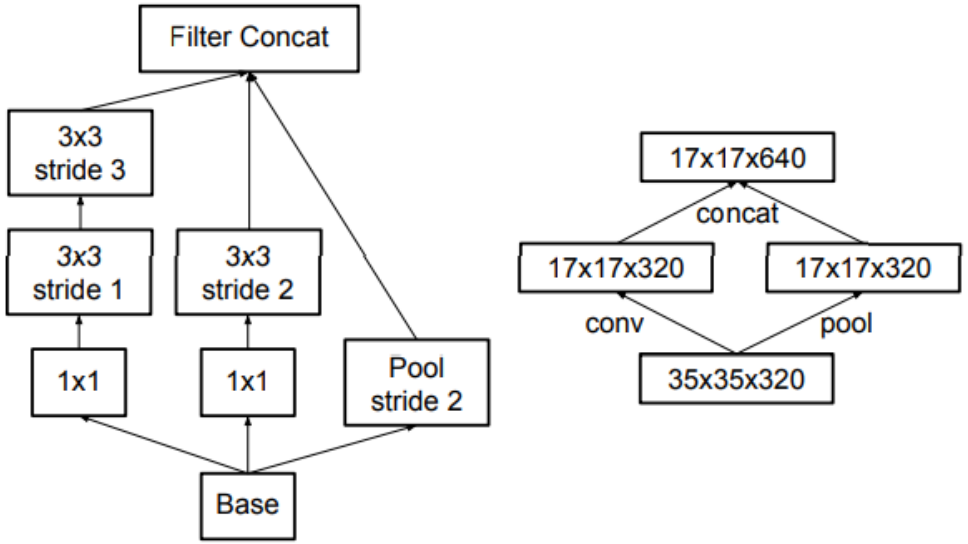




Inception

- **Inception v2**
  - **Features :** Inception v.s. Pooling





Inception

- **Inception v3**
  - **Optimizer :** RMSProp
  - **Label Smoothing :** [0, 0, 1]의 Label을 [0.0001, 0.0001, 0.9998]과 같이 변환
  - **Batch Normalization** : Last FC layer

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 4	35×35×288
5×Inception	As in figure 5	17×17×768
2×Inception	As in figure 6	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Inception

- **Inception v3**
  - 다른 모델에 비해서 성능이 개선됨

Network	Crops Evaluated	Top-1 Error	Top-5 Error
GoogLeNet [20]	10	-	9.15%
GoogLeNet [20]	144	-	7.89%
VGG [18]	-	24.4%	6.8%
BN-Inception [7]	144	22%	5.82%
PReLU [6]	10	24.27%	7.38%
PReLU [6]	-	21.59%	5.71%
Inception-v3	12	19.47%	4.48%
Inception-v3	144	**18.77%**	**4.2%**

6. Transfer Learning

Transfer Learning

- **Transfer Learning**
  - 앞선 모델들을 처음부터 학습시키는 것
  - = 오랜 시간 소요
  - = 많은 자원 소모

  - 그렇다면 이미 잘 훈련된 모델을 활용해 유사한 문제를 해결할 수는 없을까?
  - = Transfer Learning

Transfer Learning

- **Transfer Learning**
  - 훈련된 모델(Pre-trained Model)의 의의
    - Feature Extractor : 마지막 Layer를 제거하고 보면, 특성을 추출한다고 생각 가능
      - 추출된 특성 = CNN codes
      - 마지막에서 2번째 Layer가 4096차원이면
      - 4096-D codes
      - 이를 활용해 머신러닝 등 가능
    - Fine-tuning : 좋은 초기 가중치를 가지고 있는 모델이라 생각 가능
      - 가중치를 바탕으로 다른 데이터에 대해 학습
      - 좋은 가중치를 가지고 있으므로 좋은 결과로 다가갈 가능성 높음
      - 모든 Layer를 학습시키거나, 특정 Layer만 학습시킬 수 있음

Transfer Learning

- **Transfer Learning**
  - 훈련된 모델을 통한 새로운 학습
    - 데이터 유사시
      - 데이터가 적다면, 최종 FC layer만 학습 (Overfitting 방지)
      - 데이터가 많다면, 전체 네트워크를 학습
    - 데이터 다를 시
      - 데이터가 적다면, 앞단 Layer와 FC layer를 학습 (Data Manipulation & Overfitting 방지)
      - 데이터가 많다면, 전체 네트워크를 학습

Transfer Learning

- **Transfer Learning**
  - **import torchvision.models as models**
  - resnet18 = models.resnet18(pretrained=True)
  - alexnet = models.alexnet(pretrained=True)
  - squeezenet = models.squeezenet1_0(pretrained=True)
  - vgg16 = models.vgg16(pretrained=True)
  - densenet = models.densenet161(pretrained=True)
  - inception = models.inception_v3(pretrained=True)

실 습