밑바닥부터 시작하는 딥러닝

딥러닝 모델

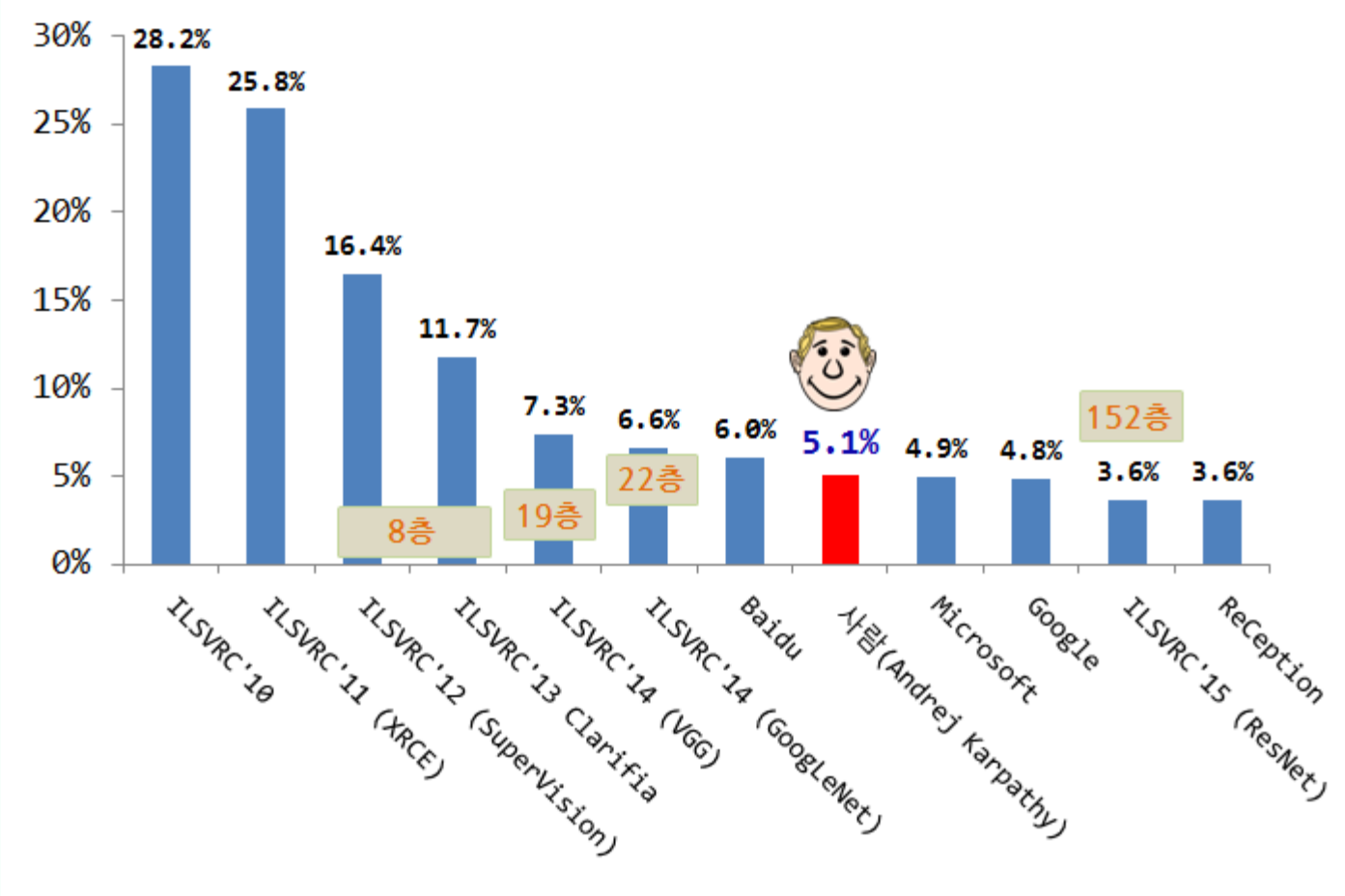
INDEX

1 LeNet

CNN의 원조

00 Overview

딥러닝의 발전과 오류율의 감소



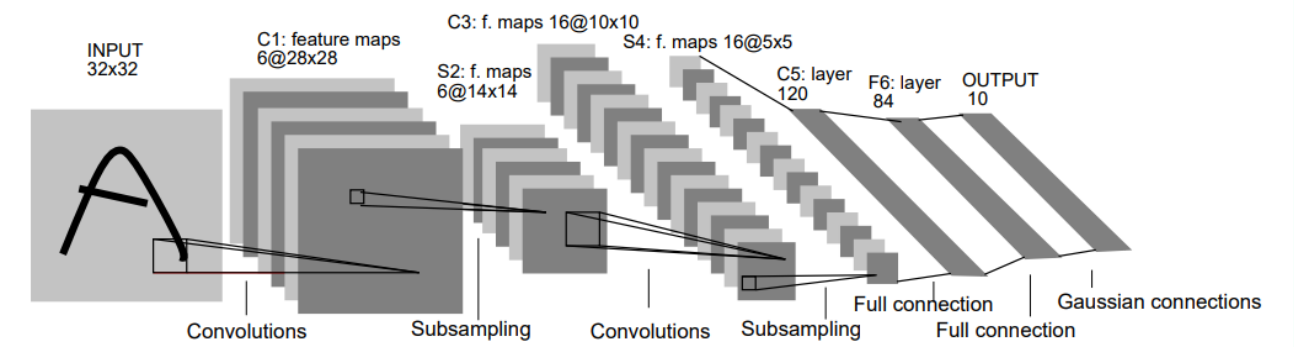


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

LeNet

발표시기 1998년 **제안자** Yann LeCun

항목

- 손글씨 숫자를 인식하는 네트워크
- 이 때는 풀링 계층이 단순히 원소 줄이기만 하는 서브샘플링 계층이었다.

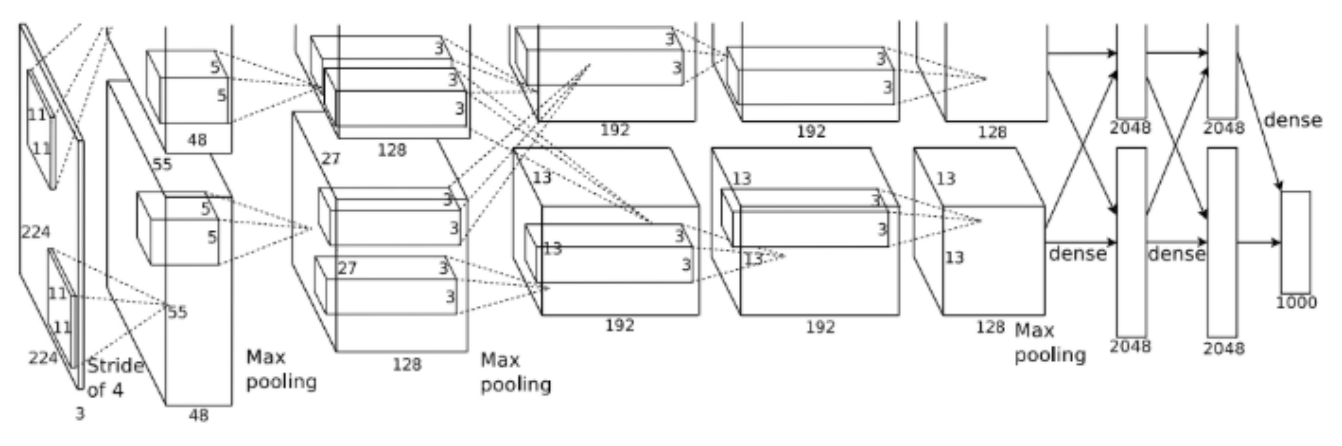
- 현재의 CNN과 비교하면 차이가 있다.
- 1. 이 네트워크는 시그모이드를 사용
- 2. 서브 샘플링을 해서 데이터 크기를 줄인다.

- 첫 CNN이라는 것에 의의를 가진다.

- 관련 논문 - http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
- 관련 코드 - https://github.com/sujaybabruwad/LeNet-in-Tensorflow/blob/master/LeNet-Lab.ipynb

2 AlexNet

딥러닝의 열풍을 불어 일으킨 네트워크



AlexNet

발표시기 2012년

제안자 Alex Krizhevsky 외 2인

항목

- 구성적으로 LeNet과 많은 차이를 보이지 않는다.
- 합성곱 계층과 풀링 계층을 거듭하며, 마지막으로 완전완결 계층을 거쳐 결과를 출력하는 CNN

- LeNet과 차이점
- ReLU 사용
- LRN이라는 국소적 정규화를 실시하는 계층을 이용
- DropOut을 사용

- 대량의 연산을 고속으로 수행
- 이는 GPU, 빅데이터 등의 발전으로 가능해졌음.
- 관련 논문 - https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- 관련 코드 - https://github.com/gholomia/AlexNet-Tensorflow/blob/master/src/alexnet.py

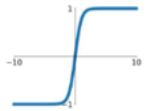
02 AlexNet

딥러닝의 열풍을 불러 일으킨 네트워크

1. ReLU Nonlinearity

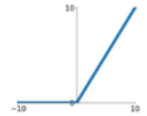
tanh

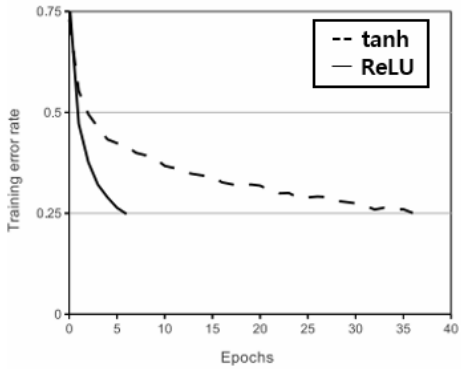
$\tanh(x)$



ReLU

$\max(0, x)$



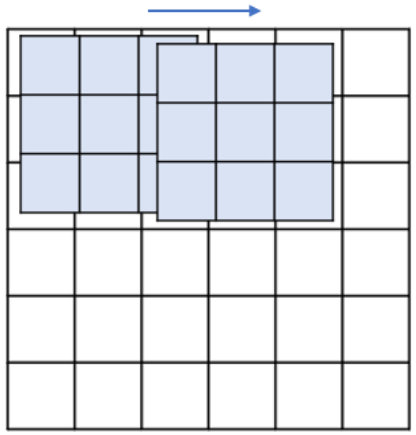


2. Local Response Normalization

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

3. Overlapping Pooling

3x3 pooling with stride=2



AlexNet

항목

- LeNet과 차이점
- ReLU 사용 - tanh보다 빠르다!
- LRN이라는 국소적 정규화를 실시하는 계층을 이용 – 최근엔 거의 이용하지 않음.
- Overlapping Pooling – kernel size를 stride보다 크게 하는 기법
- DropOut을 사용

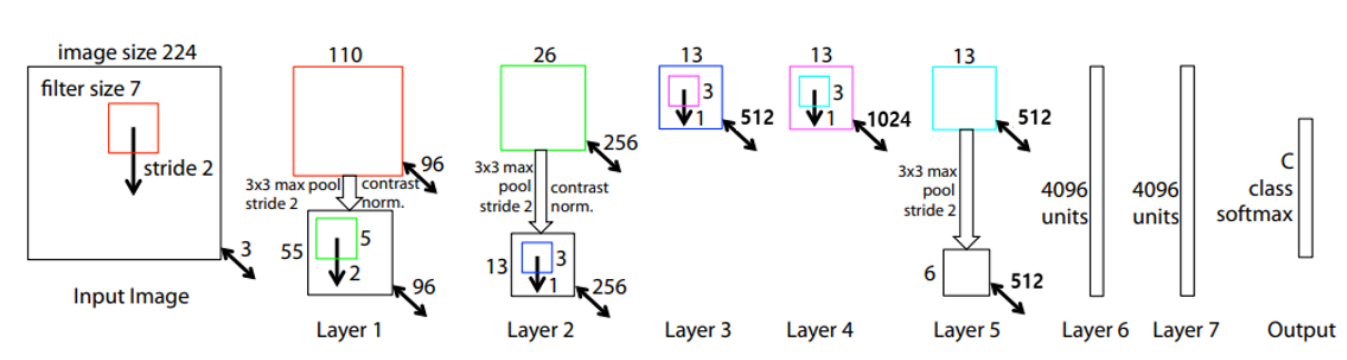
ZFNet

발표시기 2013년

제안자 Matthew D. Zeiler, Rob Fergus

항목

- AlexNet 기반



- 필터 사이즈와 stride, 그 뒤 합성곱 계층들의 filter 개수를 키워주는 등의 약간의 튜닝
- Architecture에 집중하기 보다, 학습이 진행됨에 따라 feature map을 시각화하는 방법과, 모델이 어느 영역을 보고 예측을 하는지 관찰하기 시각화 측면에 집중

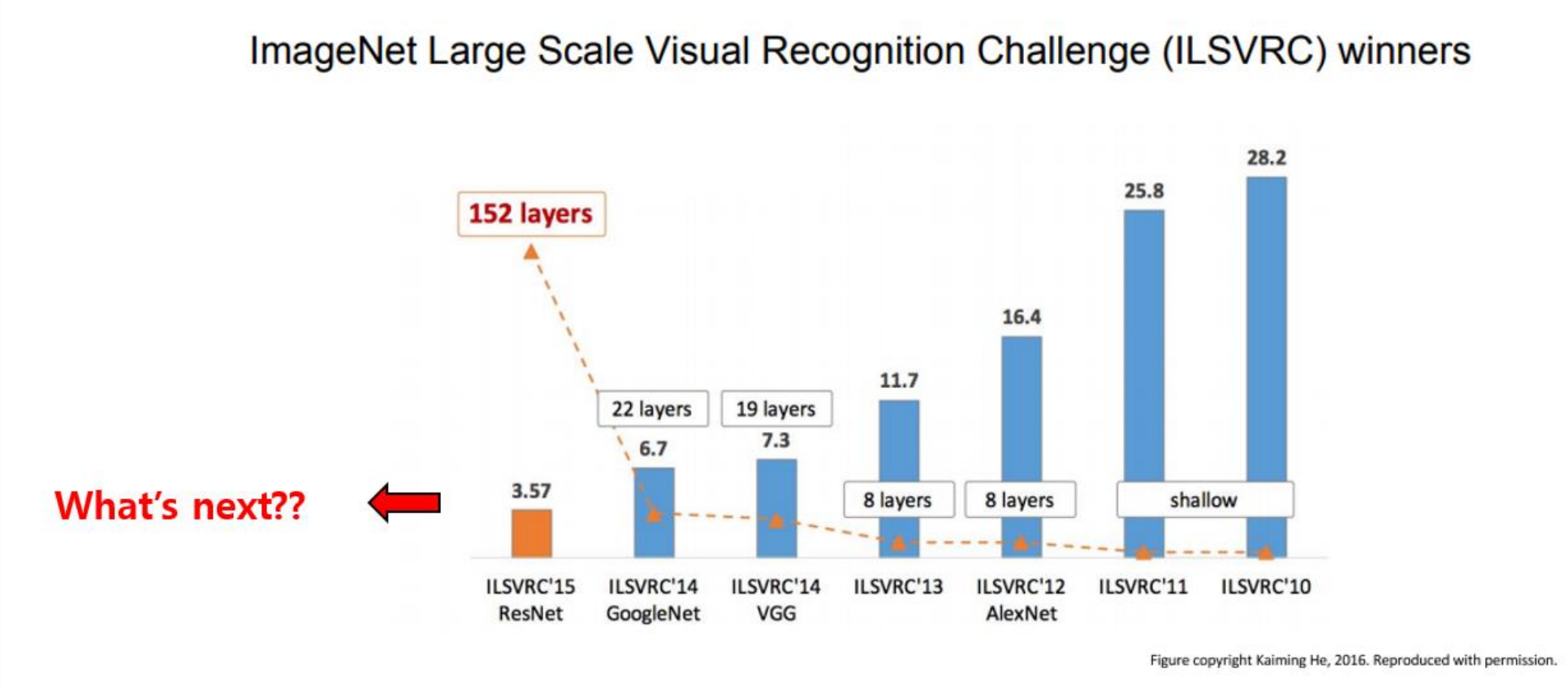
- 논문 - https://arxiv.org/pdf/1311.2901.pdf

3 VGGNet

네트워크의 깊이가 깊어지다.

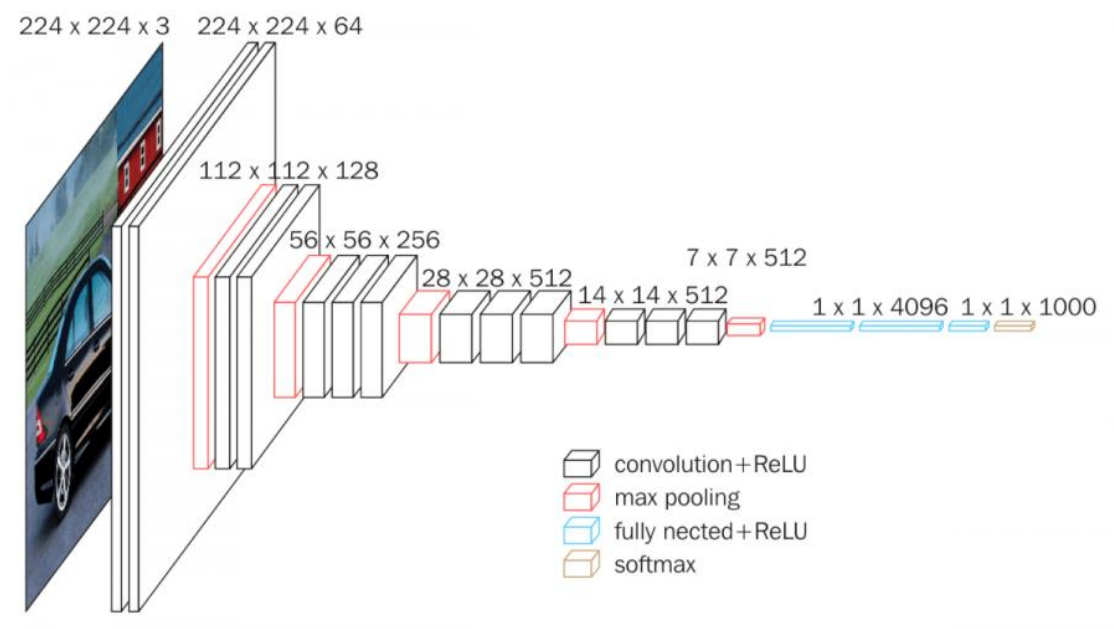
03 VGGNet

네트워크의 깊이가 깊어지다.



03 VGGNet

네트워크의 깊이가 깊어지다.



VGGNet

발표시기 2014년

제안자 옥스포드 Visual Geometry Group

항목

- 이전 방식들과 다르게, 비교적 작은 크기(여기서는 3x3)의 합성곱 필터를 깊게 쌓는 방식이다.
- 즉 계층 수가 많아졌다.

- 그림에서 보듯이 합성곱 계층을 2~4회 연속으로 풀링 계층을 두어, 크기를 절반으로 줄이는 처리를 반복한다.
- 마지막으로 완전연결 계층을 통과시켜 결과를 출력

- 관련 논문 - https://arxiv.org/pdf/1409.1556.pdf
- 관련 코드 - https://eungbean.github.io/2018/10/25/udacity-2-CNN-transfer-learing-with-vggnet/

03 VGGNet

네트워크의 깊이가 깊어지다.

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

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					

VGGNet

항목

- 이전 방식들과 다르게 비교적 작은 3x3 합성곱 필터를 깊게 쌓는다.

- 이렇게 3x3 filter를 중첩하여 쌓는 이유
- 3개의 3x3 conv layer를 중첩하면 1개의 7x7 conv layer와 receptive field(수용 영역)가 같아지지만
- 활성화 함수를 더 많이 사용할 수 있어서 더 많은 비선형성을 얻을 수 있으며, parameter 수도 줄어드는 효과를 얻을 수 있습니다. (3x3x3 = 27 < 7x7 = 49)

4 GoogLeNet

가로의 방향도 깊다.

04 GoogLeNet

네트워크의 깊이가 깊어지다.

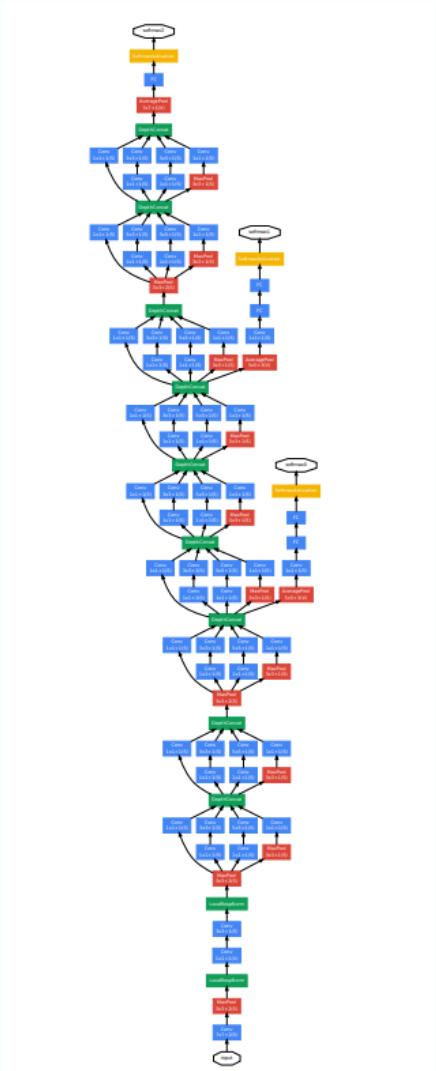


Figure 3: GoogLeNet network with all the bells and whistles.

GoogLeNet

발표시기 2014년

제안자 구글

항목

- VGGNet를 제치고 1등을 한 모델
- VGGNet보다 더 깊은 레이어
- 동시에 굉장히 길고 복잡한 구조

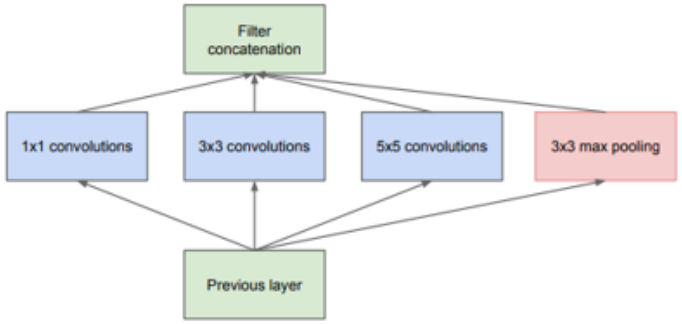
- 제일 주요 특징으로 오른쪽으로도 뭔가 있다!

- 관련 논문 – https://static.googleusercontent.com/media/research.google.com/ko//pubs/archive/43022.pdf
- 관련 코드 - https://github.com/conan7882/GoogLeNet-Inception

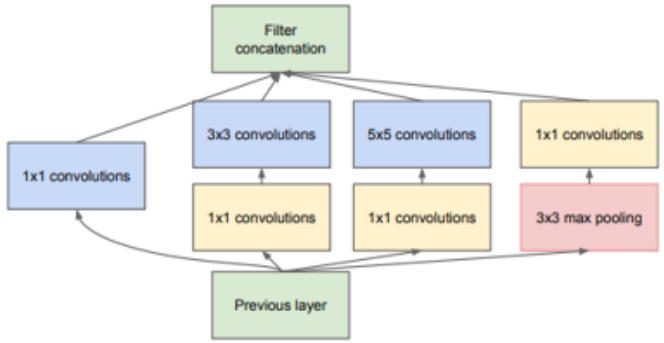
04 GoogLeNet

네트워크의 깊이가 깊어지다.

1. Inception Module



(a) Inception module, naïve version



(b) Inception module with dimensionality reduction

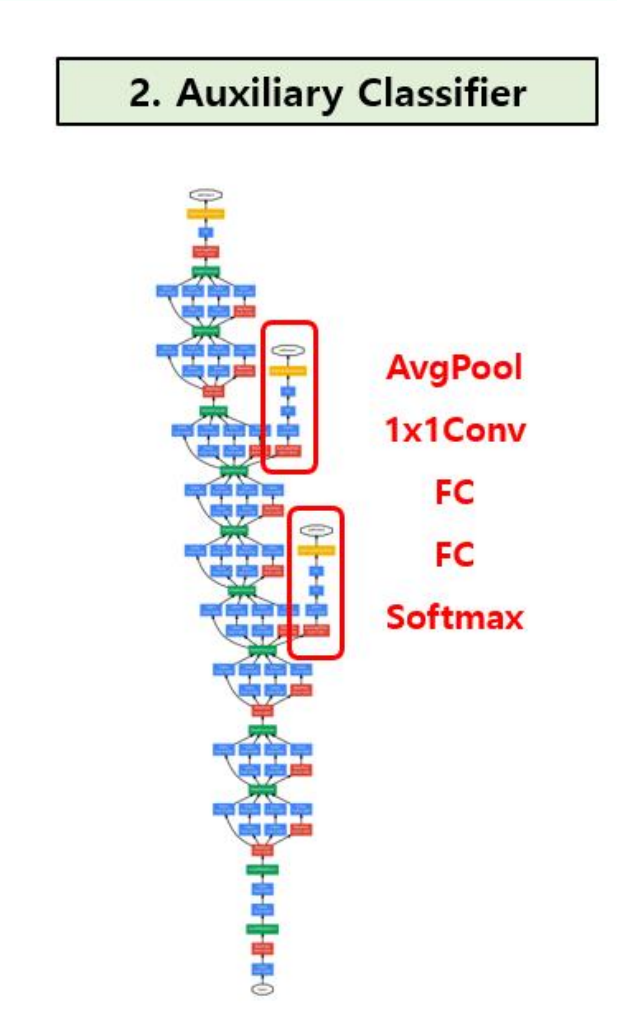
Figure 2: Inception module

GoogLeNet

항목

- Inception module
- Inception module이란, 블록 구조이다.
- 기존엔 각 계층간 하나의 합성곱 연산, 하나의 풀링 연산
- 하지만 이 모델은, 4가지 종류의 합성곱이나 풀링을 수행하고, 4개의 결과를 채널 방향으로 합친다.

- 여기서 맨 왼쪽 1x1 합성곱은 Inception module에서 필요한 연산양을 감소 시킨다.
- 방법은 일단 1x1 conv를 넣어 채널을 줄였다가, 3x3, 5x5 conv에서 확장하는 방식. 이를 통해서 채널 간의 연관성을 연산한다.



GoogLeNet

항목

- Auxiliary Classifier
- 3, 6번째 Inception module 뒤에 추가적인 Classifier 를 사용하는 것을 볼 수 있는데, 이를 Auxiliary Classifier라 부른다.

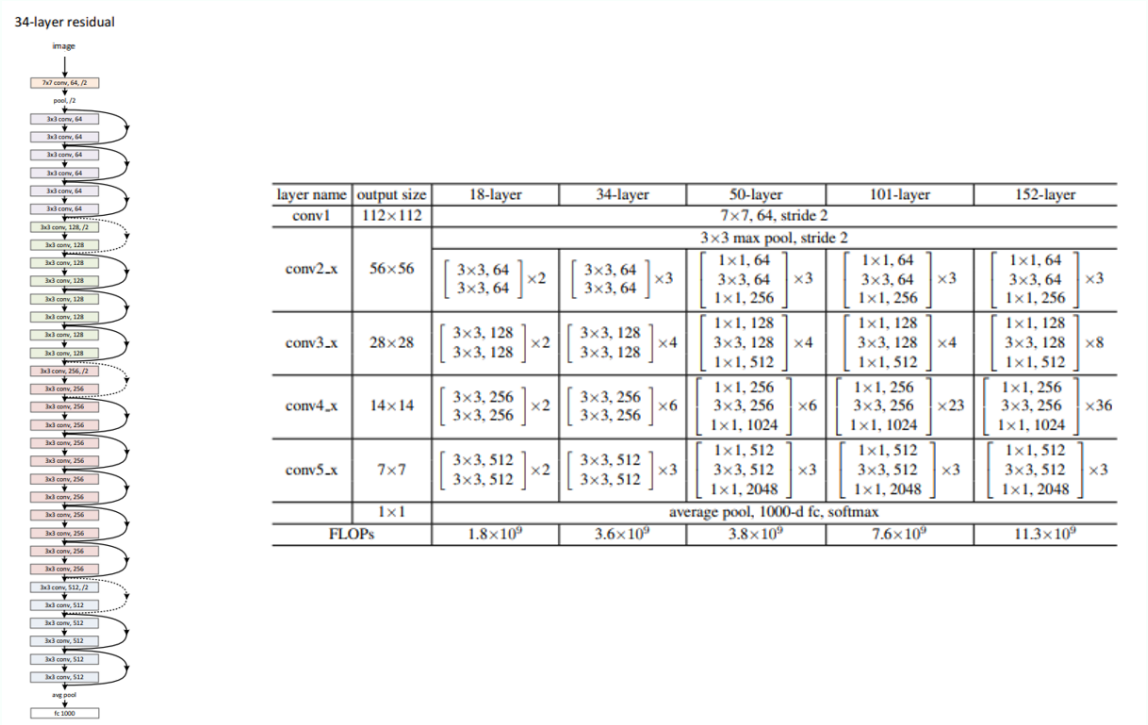
- 이 것의 역할을 가장 뒷부분에만 Classifier가 존재한다면, input과 가까운 부분에는 gradient가 잘 전파되지 않기 때문에 이 Classifier를 붙여 이를 완화한다.
- 보통 학습 단계에서만 사용하고 추론 단계에선 사용하지 않는데 이는 성능 향상이 미미하기 때문.

5 ResNet

성능이 떨어지는 부분은 과감히 스킵

05 ResNet

성능이 떨어지는 부분은 과감히 스킵



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ResNet

발표시기 2015년

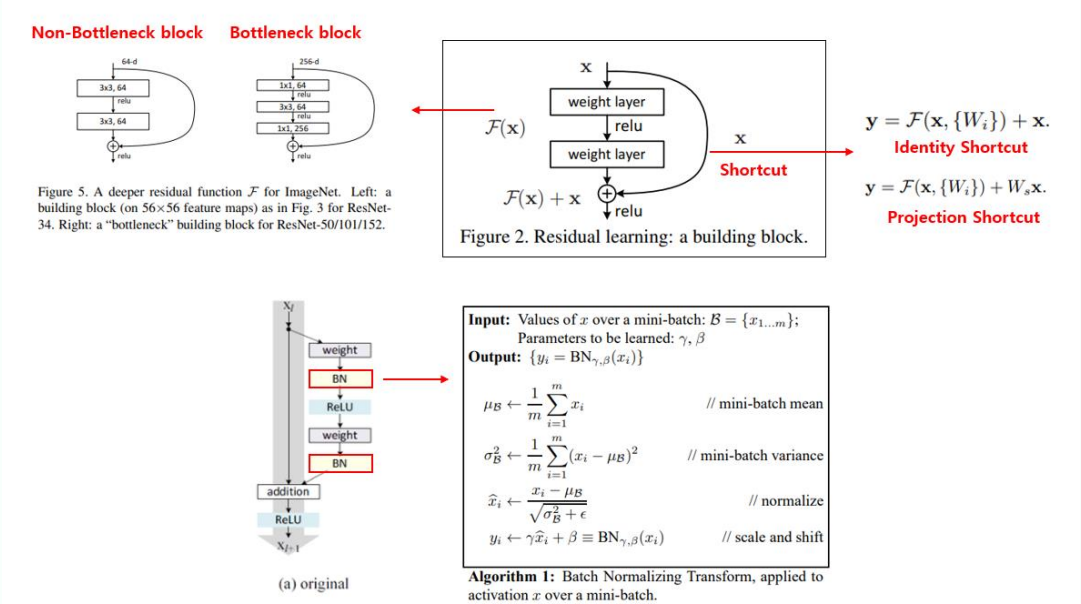
제안자 마이크로소프트

항목

- 굉장히 유명하고 제일 많이 쓰는 아키텍처
- 왼쪽의 표는 ResNet-34 구조이며, VGGNet처럼 3x3 합성곱이 반복된다.
- 레이어의 개수에 따라 ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152 같은 총 5개의 버전이 있다.
- 계층의 개수가 많이 사용될수록 연산량과 매개변수 개수는 커지지만, 정확도는 좋아진다는 효과를 얻을 수 있다.
- 관련 논문 - https://arxiv.org/pdf/1512.03385.pdf
- 관련 코드 - https://github.com/taki0112/ResNet-Tensorflow

05 ResNet

성능이 떨어지는 부분은 과감히 스킵



ResNet-34

항목

- ILSVRC 2015 대회에서 1위를 한 모델
- 층의 깊이를 깊게 하는 것이 성능 향상에 좋다는 것은 다들 알고 있는 이야기.
- 하지만 지나치게 깊으면 학습이 되지 않고, 성능또한 떨어진다.
- 이 모델은 층의 깊이에 비례해 성능을 향상시키는 것이 핵심이기에, 만약 성능이 떨어진다고 보이면, 스킵 연결이라는 것을 이용한다.

05 ResNet

성능이 떨어지는 부분은 과감히 스킵

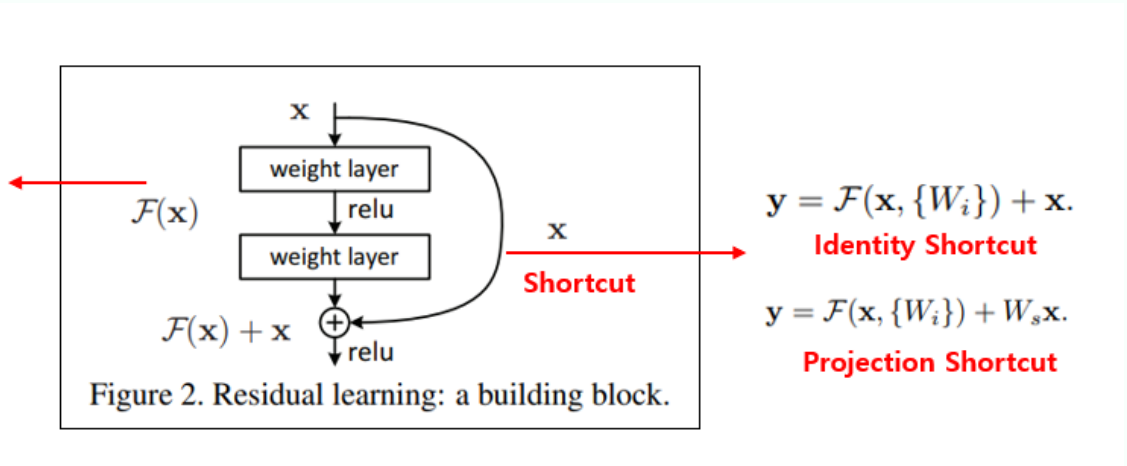


Figure 2. Residual learning: a building block.

ResNet-34

항목

- 스킵 연결

- 스킵 연결은 합성곱을 건너뛰어 출력에 바로 더하는 구조이다.
- 다음 사진처럼 입력 x를 연속한 두 합성곱 계층을 건너뛰어 출력에 바로 연결한다.
- 그래서 원래 출력은 f(x)이지만 여기서는 f(x) + x라는 출력을 받게 된다.

- 층이 깊어져도 학습을 효율적으로 할 수 있도록 하는데. 이를 통해 역전파 때 스킵 연결이 신호 감쇠를 막아준다.

Thank you

End Of Document

