

Very Deep Convolutional Networks for Large-Scale Image Recognition

<https://arxiv.org/abs/1409.1556>

VGG: Visual Geometry Group(옥스포드 대학의 연구팀)

목차

1. Quick Review

2. 배경

3. 모델 핵심 구조

4. 모델 학습

5. 모델 테스트

6. 모델 성능

7. 결론

1. Quick Review

VGGNet는 기존 ([Convolutional Layers → Pooling layers](#))의 반복 → [Fully connected Layers](#) 의 전통적인 CNN 구조를 크게 벗어나지 않으면서, 레이어를 깊게 쌓아 2014 ILSVRC 이미지 분류 대회에서 2위를 달성하였다. (1위 GoogleNet)

VGGNet의 핵심은 기존 CNN에서 사용되었던 7×7 , 5×5 크기의 필터들을 사용하지 않고 여러개의 3×3 필터로 쪼개어 사용함으로써 레이어를 더욱 깊게 쌓을 수 있었다는 것이다. 모든 Convolutional layers에서 비교적 작은 여러개의 3×3 필터만을 사용하게 되면 적은 파라미터 수로 깊게 레이어를 쌓을 수 있음과 동시에 여러 개의 비선형 함수를 사용할 수 있게 되므로 이를 통해 모델의 성능을 높일 수 있었다.

3×3 필터 사용의 장점

- 파라미터 수의 감소로 모델을 더욱 깊게 쌓을 수 있었다.
- 비선형성 증가로, 모델의 이미지 특징 식별성을 높인다.(*Makes the decision function more discriminative.*)

작은 필터 크기를 사용하여 모델의 깊이를 점차 늘려간다면, 파라미터의 수는 어쨌든 증가할 것이다. 이로 인해 한정된 학습 데이터에 대해 과적합 문제가 일어날 수 있고, 모델 깊이가 깊어짐에 따라 gradient vanishing/exploding 문제가 발생할 수 있을 것이다. 따라서 저자는 다음과 같은 기법을 사용하여 이러한 문제를 해결하였다고 한다.

- 과적합 문제 → **Multi-Scale training(Scale Jittering)**이라는 data augmentation 기법을 적용
- gradient 불안정 문제 → 얇은 모델에서 어느정도 학습된 가중치를 더욱 깊은 모델의 초기 가중치로 사용

정리하자면, VGGNet은 3×3 이라는 작은 필터 크기로 모델을 깊게 쌓아 학습을 진행하였고, 깊어진 모델로 인해 발생할 수 있는 과적합 문제와 gradient 불안정 문제를 각각 data augmentation과 가중치 초기화 전략으로 해결한 것입니다.

VGGNet은 간단한 구조와, 단일 네트워크에서 GoogleNet보다 좋은 성능을 보여 지금까지도 많은 주목을 받고 있다.

2. 배경(Abstract & Introduction)

2013년까지 나왔던 CNN 모델들은 레이어의 수가 10개 미만이었다.(ex. AlexNet: 8 Layers) CNN 모델의 성능을 높일 수 있는 가장 직접적인 방식은 레이어를 깊게 쌓는 것이다. 모델을 깊게 쌓을수록, 이미지에서 더욱 복잡한 특징을 추출할 수 있게 되기 때문이다.

따라서 본 논문의 저자는 **네트워크의 깊이가 모델 성능에 어떤 영향을 주는지 초점을 맞춰 연구**를 진행했다고 한다. 아래 ConvNet configurations를 보면 receptive field(conv.layer의 크기)는 가장 간단한 3x3으로 모두 고정시키고, 점차 모델의 깊이를 늘린 6개의 모델 구조에 대해 실험하였음을 알 수 있다.

이때 기존의 CNN 구조에서 사용되었던 7x7 5x5 등의 conv.layers를 **여러 개의 3x3 conv.layers**로 바꾸게 되면서, **파라미터의 수의 감소**로 학습 속도 향상과 과적합을 방지할 수 있었다. 또한 여러개의 conv.layers 사용함으로써 그 뒤에 따라오는 비선형 함수 ReLU로 인해 **non-linearity**를 증가시켜 모델 정확도를 향상시키는 효과를 얻을 수 있었다.

3. 모델 핵심 구조(Section 2)

총 6개의 모델에 대해 학습을 진행하였고, 구성은 다음과 같다.

- 11 ~ 19개의 Convolutional Layers + 3개의 Fully-Connected Layers
- 3×3 Convolutional filters 사용. (Stride=1, Zero-Padding=1)
- 2×2 Max Pooling(Stride=2)
- ReLU 활성화 함수 사용

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					

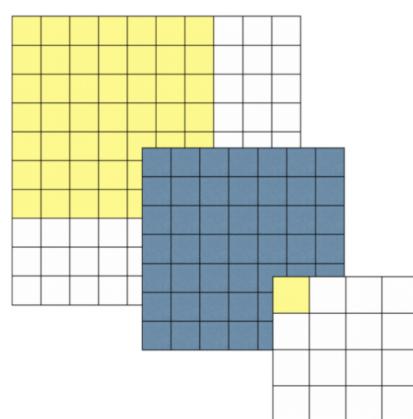
Table 2: **Number of parameters** (in millions).

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

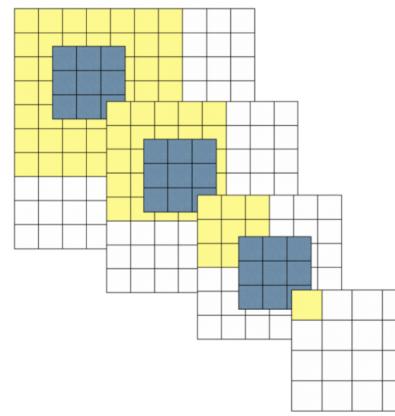
VGG 모델 아키텍처. 11 ~ 19까지 레이어 수를 점차 늘려갔다.

3 × 3 크기의 필터 사용

- 하나의 7×7 필터 대신 3개의 3×3 필터를 사용하였다.



1번의 7×7 필터 적용했을 경우



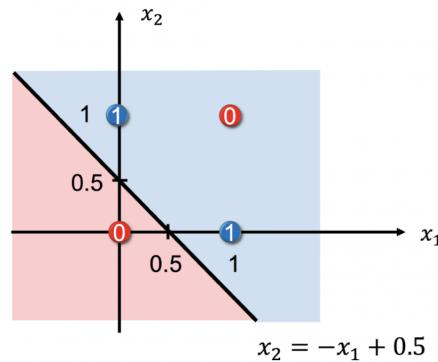
3번의 3×3 필터 적용했을 경우

3차례의 3×3 conv 필터링을 반복한 특징맵은 한번의 7×7 conv 필터링을 적용한 효과를 볼 수 있다. 출처: [1]

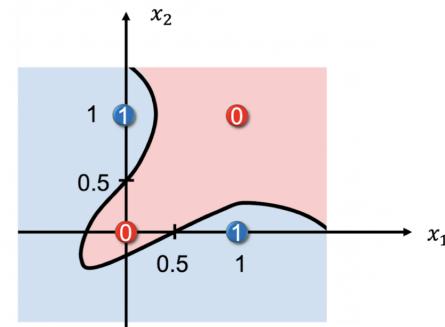
- 이처럼 작은 receptive field를 여러개 사용했을 때의 장점은 아래와 같다.

1. Decision function의 비선형성 증가

- 한개의 7×7 필터를 사용하게 되면 하나의 ReLU를 사용해야하지만, 3차례의 3×3 필터를 사용한다면 3번의 ReLU 활성함수를 적용할 수 있게 된다.



Impossible to divide into two regions completely with a line (or a hyperplane)



Possible to divide into two regions completely with a nonlinear curve

비선형함수 ReLU를 통과했을 경우, 선형함수만으로는 풀 수 없었던 XOR 문제를 해결할 수 있게 된다.

2. 학습 파라미터 수의 감소

- Input, Output 모두 C 개의 채널을 갖는다고 하자. 3×3 필터 세 개의 경우 $3 \times 3^2 C^2 = 27C^2$ 개의 파라미터를 갖게 되고, 7×7 필터 한 개의 경우 $7^2 C^2 = 49C^2$ 개의 파라미터를 갖게 된다.

같은 결과에 대해 더 적은 파라미터 수를 사용하여 overfitting을 줄이는 효과가 있고(**regularisation**), 비선형 함수를 더 많이 통과함으로써 모델이 의사 결정을 더 잘할 수 있도록 한다.

4. 모델 학습(Section 3.1)

학습 초기 설정

- 하이퍼 파라미터 설정

Cost Function	Multinomial logistic regression
Optimizer	Momentum(관성계수 $m=0.9$)
Mini-batch size	256
Regularisation	L2-norm(0.0005), Dropout(0.5)
Learning rate	초기값 0.01로 설정, 이후 validation accuracy 증가하지 않는 경우 0.1 만큼 감소

학습은 learning rate가 세번 감소($0.01 \rightarrow 0.00001$)하고 난 후 74 epoch에서 종료되었다. 여기서 주목할만한 부분은, AlexNet보다 더 깊은 레이어와 더 많은 파라미터 수임에도 불구하고 더 적은 epoch으로 빠르게 수렴했다는 것이다. 그 이유를 논문의 저자는 아래 두 가지로 설명하

고 있다. (참고: AlexNet은 8개의 레이어와 61,000,000(61 millions)개의 파리미터 수를 가진다. VGG는 VGG16 기준 16개의 레이어와 138,000,000(138 millions)개의 파라미터 수를 가진다.)

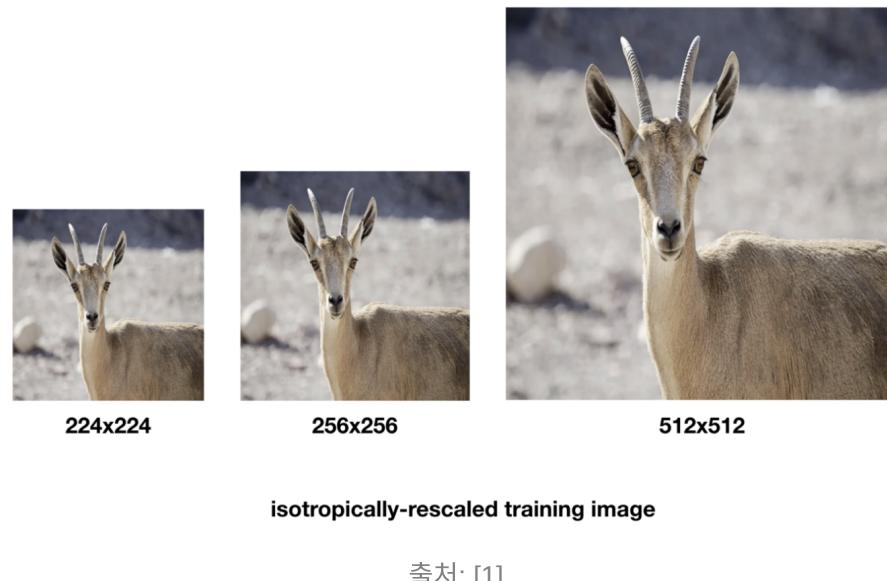
1. Implicit Regularisation - 3x3 convolution filters의 사용

2. Pre-initialisation of certain layers - 깊은 신경망을 가진 모델일수록, 가중치 초기화 방법에 따라 학습이 불안정하게 이루어질 수도 있다. 따라서 저자는 이러한 문제를 해결하고자 비교적 얕은 신경망을 가진 모델 A로 학습된 가중치를 이후 더 깊은 모델 학습 시 초기 가중치로 사용하였다. (더욱 정확하게는 모델 A의 처음 4개의 convolutional layer와 마지막 3개의 fully-connected layer의 가중치를 사용하였다. 일종의 transfer learning을 진행한 것.) 이때 모델 A 학습시 초기 가중치는 $N(0, 0.1^2)$ 로 초기화하였다.

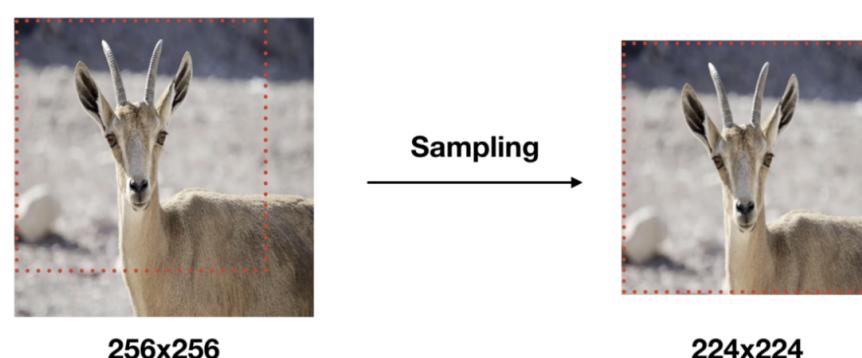
Data Augmentation

다음으로는 학습 시 이미지의 크기에 관한 부분이다. 인풋 이미지의 크기는 224×224 로 고정하였다. 이때 저자는 이러한 고정된 크기의 이미지를 만들기 위해 **single-scale trianing**, **multi-scale training** 두 가지 방법을 고려했다고 한다.

먼저 기존 이미지의 비율을 유지하며 가로 세로 크기를 $S \geq 224$ 로 rescale해준다. (비율을 유지하며 사이즈를 rescaling하는 것을 isotropically-rescale이라고 한다.)



그 후, 모델의 인풋 크기(224×224)에 맞게 랜덤하게 crop하여 데이터를 생성한다.



아래와 같이 같은 256×256 , 혹은 512×512 이미지라 하더라도, 각기 다른 인풋 이미지가 생성되는 것을 확인할 수 있다.



여기서 **single-scale training**이란 S 를 하나의 값으로 고정시켜놓고 데이터를 생성하는 것을 의미하고, **multi-scale training**은 $S \in [S_{min}, S_{max}]$ 로 범위를 설정하여 여러 S 값을 사용하는 것을 의미한다. 논문에서는 multi-scale training 시 $S_{min} = 256$, $S_{max} = 512$ 로 설정했다. 또한 single-scale training을 사용했을 때보다 multi-scale training을 사용했을 때 모델 성능이 더 좋았다고 한다.(이후 설명 관련 부분에서 자세한 설명)

이러한 방법을 통해 data augmentation 효과를 얻을 수 있었고, 하나의 이미지에 대한 다양한 측면(ex. 사슴의 머리, 몸통, 다리 등)을 학습시킬 수 있어 overfitting을 방지할 수 있다.

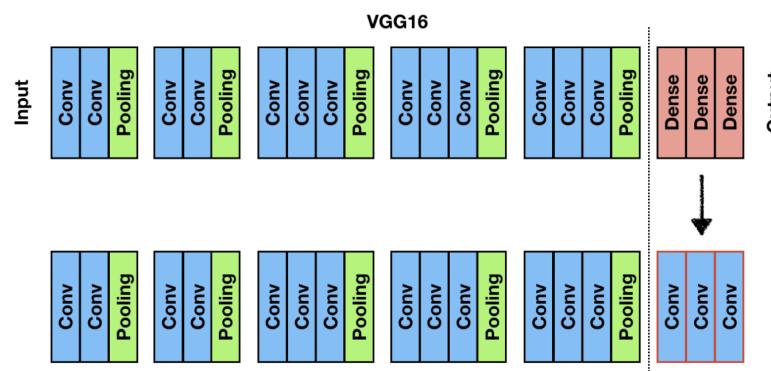
5. 모델 테스트(Section 3.2)

'Test'라고 하지만, 저자가 논문에서 `we used the validation as the test set.` 라고 언급했듯이, validation을 의미하는 것 같다. 앞으로 테스트(test)를 validation이라고 생각하자.

테스트 시에도 multi-scale을 적용하였다. Train 시 rescale의 기준이 되는 값을 S 라고 했다면, test 시에는 기준 값을 Q 라고 부른다. (두 값이 의미하는 것은 동일하다.)

먼저 Test scale 파라미터 Q 를 조절함으로써 하나의 테스트 이미지에서 여러 개의 데이터를 추출한다. 이렇게 여러개의 데이터를 예측한 결과의 평균(Sum-pooling)으로 최종 분류 결과를 구할 수 있다. Rescaling 시 Q 는 꼭 S 와 같을 필요가 없고, 실제로 각각의 S 마다 여러 Q 를 사용했을 때 모델의 성능이 더 좋게 나왔다고 한다.(5. 모델 성능 파트에서 확인할 수 있다.)

또한 Test할 때의 모델 구조를 Train때와는 다르게 하여 검증을 진행하기도 했다. 훈련 후 test 시에는 모델의 마지막 3 Fully-connected layers를 Convolutional layers로 변환하였다. 첫 번째 Fully-connected layer는 7×7 Conv, 마지막 두 Fully-connected layers는 1×1 Conv로 변환하였는데, 이렇게 Fully-connected layers 없이 Convolutional layers로만 구성된 모델 구조를 **Fully-convolutional Network**라고 부른다.(이렇게 구조를 바꾸어 테스트한 방식을 **dense-evaluation**이라고 한다.[6])



출처: [3]

신경망이 Convolution layers로만 구성될 경우 입력 이미지의 크기 제약이 없어진다. 이에 따라 하나의 입력 이미지를 다양한 스케일로 사용한 결과들을 양상화하여 이미지 분류 정확도를 개선하는 것도 가능해진다.[1]

[이 부분에 대해서는 아직 명확하게 이해가 가지 않아, Fully-Convolutional Network에 대한 공부 후 다시 정리해보겠습니다.]

6. 모델 성능(Section 4)

모델의 성능은 ILSVRC-2012 dataset에서 **Top-1 Error**와 **Top-5 Error**를 통해 평가하였다.

Train Images	130만
Validation Images	5만
Test Images	10만
Classes	1000

5-1. Single-Test Scale 적용 결과

Table 3: ConvNet performance at a single test scale.

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	25.5	8.0

Single-test-scale 적용 시 결과

Q (test scale)를 고정시켰을 때의 모델 성능이다. 다음과 같은 결과를 확인할 수 있다.

- Layer가 깊어질수록 모델의 성능이 증가하고 있다.
- 같은 depth이지만, 1x1과 3x3 conv layer의 차이만 있었던 C와 D를 비교했을 때, D의 성능이 더 높았다. 여기서 모델 C를 모델B와 비교해봤을 때 1x1 conv.layer를 사용함으로써 non-linearity 추가한 것이 물론 도움이 되지만, 1x1에 비해 3x3을 사용했을 때의 성능이 더 높았던 이유는 더 큰 spatial context를 고려할 수 있어야 하는 것 역시 중요하다는 것을 알수 있다.
- 학습 시 single-scale보다 multi-scale을 사용하는 것이 모델 성능을 더 향상시켰다.
- Table3에는 나와있지 않지만, 저자는 모델 B 학습 시 2개의 3x3 conv.layer를 1개의 5x5 conv.layer로 바꿔서 실험했지만, 더 높은 error를 보였다고 한다. 즉 small filter를 사용하는 것이 모델 성능 향상에 더 도움되었던 것이다.

5-2. Multi-Test Scale 적용 결과

Table 4: ConvNet performance at multiple test scales.

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
B	256	224,256,288	28.2	9.6
C	256	224,256,288	27.7	9.2
	384	352,384,416	27.8	9.2
	[256; 512]	256,384,512	26.3	8.2
D	256	224,256,288	26.6	8.6
	384	352,384,416	26.5	8.6
	[256; 512]	256,384,512	24.8	7.5
E	256	224,256,288	26.9	8.7
	384	352,384,416	26.7	8.6
	[256; 512]	256,384,512	24.8	7.5

Multi-test-scale 적용 시 결과

- Train뿐만 아니라 Test 시에도 Multi-scale을 적용했을 때 더 높은 모델 성능을 보여주었다. Test시에는 3개의 Q 에 대해 모델 예측값을 추출한 다음, 평균을 내는 식으로 계산했다.
- 이때 train 데이터 분포 test 데이터 분포 사이 너무 큰 차이가 발생하면 안되므로,
 - single-scale training 경우, test scale Q 를 $\{S - 32, S, S + 32\}$ 로 설정하였다.

- multi-scale training 경우, test scale Q 를 $\{S_{min}, 0.5(S_{min} + S_{max}), S_{max}\}$ 로 설정하였다.

5-3. Multi-Crop & dense evaluation 결과

Table 5: **ConvNet evaluation techniques comparison.** In all experiments the training scale S was sampled from [256; 512], and three test scales Q were considered: {256, 384, 512}.

ConvNet config. (Table 1)	Evaluation method	top-1 val. error (%)	top-5 val. error (%)
D	dense	24.8	7.5
	multi-crop	24.6	7.5
	multi-crop & dense	24.4	7.2
E	dense	24.8	7.5
	multi-crop	24.6	7.4
	multi-crop & dense	24.4	7.1

Dense evaluation은 Fully-Convolutional Network 구조로 변환하여 테스트한 것을 의미한다. Multi-Crop 과 Dense evaluation을 모두 사용했을 때 가장 좋은 성능을 보였다.

5-4. 모델 양상을 결과

Table 6: **Multiple ConvNet fusion results.**

Combined ConvNet models	Error		
	top-1 val	top-5 val	top-5 test
ILSVRC submission			
(D/256/224,256,288), (D/384/352,384,416), (D/[256;512]/256,384,512) (C/256/224,256,288), (C/384/352,384,416) (E/256/224,256,288), (E/384/352,384,416)	24.7	7.5	7.3
post-submission			
(D/[256;512]/256,384,512), (E/[256;512]/256,384,512), dense eval.	24.0	7.1	7.0
(D/[256;512]/256,384,512), (E/[256;512]/256,384,512), multi-crop	23.9	7.2	-
(D/[256;512]/256,384,512), (E/[256;512]/256,384,512), multi-crop & dense eval.	23.7	6.8	6.8

ILSVRC-2014에서 7개의 모델을 양상을하여 예측한 결과(7.3% test error)를 제출하였고, 그 후 최적의 2개의 모델을 양상을하여 에러를 6.8%까지 낮출 수 있었다고 한다.

5-5. ILSVRC 분류 대회에서 다른 SOTA 모델과의 비교

Table 7: **Comparison with the state of the art in ILSVRC classification.** Our method is denoted as “VGG”. Only the results obtained without outside training data are reported.

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	23.7	6.8	6.8
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	6.7	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

VGG는 2014 이전까지의 ILSVRC 이미지 분류 대회에서 우승했던 다른 어떤 모델의 성능보다 우수하다. 또한 2014 ILSVRC에서 우승했던 GoogleNet과 비교해봤을 때도 성능 측면에서 큰 차이가 없기 때문에 상당히 우수한 모델이라고 할 수 있다.

양상을 아닌 single-net 측면에서 봤을 때, 우승자인 GoogleNet(7.9%)보다도 낮은 에러(7.0%)를 보여주고 있다는 점은 주목할만하다.

7. 결론(Section 5)

VGG 모델은 기존의 전통적인 CNN 모델 구조에서 크게 벗어나지 않았지만, 깊이(depth)를 19개의 레이어까지 깊게 쌓음으로써 이미지 분류 성능을 높일 수 있었다. 이를 통해 모델의 깊이가 이미지 특성을 학습하는데 중요한 역할을 하고 있음을 알 수 있다.

여기서는 19개의 레이어까지 실험을 진행했는데, 이는 19개 레이어에서 에러율이 수렴했기 때문이다. 더 큰 데이터셋의 경우 모델의 깊이를 더 깊게 한다면 효과적일 수도 있을 것이라고 논문의 저자는 말하고 있다.

Appendix

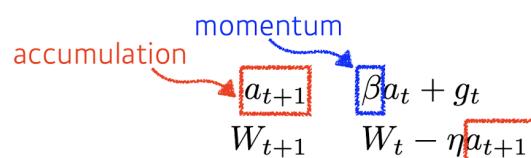
1. Receptive Field

- 출력 레이어의 뉴런 하나에 영향을 미치는 입력 뉴런들의 공간 크기. 필터의 크기와 같다.

2. Local Response Normalisation(LRN)

- 2012 ImageNet 대회에서 최초 CNN 우승 모델인 AlexNet에서 사용한 normalisation 방법 중 하나이다. ReLU는 Input 값이 양수일 경우, 입력 값을 그대로 사용한다. 그렇게 되면 Convolution이나 Pooling 레이어를 통과할 때 높은 픽셀값이 주변 픽셀값에 영향을 주게 된다. 이를 방지하기 위해 다른 Activation map의 같은 위치의 있는 픽셀끼리 정규화를 해주는게 바로 LRN인 것이다. 더 자세한 설명과 연산 방법은 [5]를 참고하자.
- 이 논문에서는 LRN이 성능을 높여주지 않았고, 메모리 사용량과 연산량만 증가시켰기 때문에 사용하지 않았다고 한다. 또한 요즘에는 LRN을 거의 사용하지 않고, 대부분 Batch Normalization으로 정규화를 진행한다.

3. Momentum



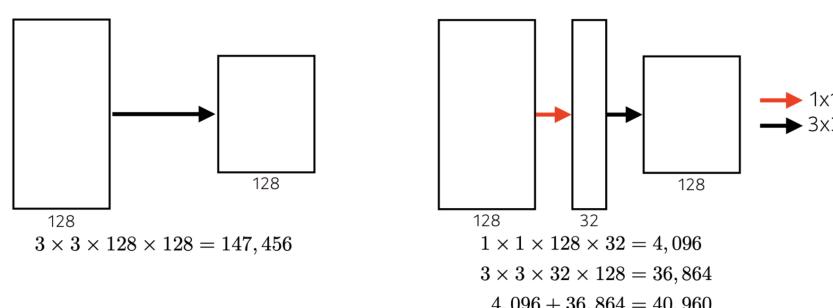
- 이전 batch에서 특정 방향으로 gradient가 흘러가면, 그 정보를 활용하는 옵티마이저이다.
- g_t 라는 gradient가 현재 들어왔다면, 모멘텀에 해당하는 β 하이퍼파라미터 * a_t 한 모멘텀이 포함되어있는 gradient(accumulation)으로 업데이트를 해주는 것이다.
- 한번 흘러가는 gradient를 어느 정도 유지시켜주기 때문에 학습이 잘 될 수 있다는 장점이 있다.

4. Multinomial Logistic Regression

- Cross Entropy Loss를 의미한다.

5. 1×1 Convolution을 사용하는 이유

- 채널 수 조절 가능
- 연산량의 감소(파라미터 수 감소)



똑같이 output의 채널 수를 128개로 만들지만, 1×1 Conv를 사용했을 때의 파라미터 수가 훨씬 적다. 이처럼 1×1 로 채널을 감소시키고 다시 늘리는 구조를 bottle neck 구조라고 한다. 출처: [7]

- 비선형성(Non-linearity) - 1×1 Conv를 사용할 때마다 비선형 활성 함수(ex. ReLU)를 사용하게 된다.

6. Fully-convolutional network

- FCN은 Semantic Segmentation task에 적합한 모델을 위해 기존에 이미지 분류에서 우수한 성능을 보인 CNN 기반 모델(AlexNet, VGG16, GoogLeNet)을 목적에 맞춰 변형시킨 것이다.[7] 대부분의 이미지 분류 모델들은 Convolutional layer에서 이미지의 특징을 추출하고, 추출된 특징으로 class를 분류를 위해 출력층이 fully-connected layer로 이루어져 있다.
- 이러한 fully-connected layer의 단점은, 이미지의 위치 정보를 사라지게 하고 입력 이미지의 크기가 고정되어야 한다는 단점이 있다. 이를 개선하기 위해 모델의 fully-connected layer를 모두 convolutional layer 바꾼 것이 FCN(Fully-convolutional Network)인 것이다.

7. Top-1 Error와 Top-5 Error

- Top-1 error와 Top-5 error는 이미지 분류 성능을 평가하기 위한 것들이다.[4]

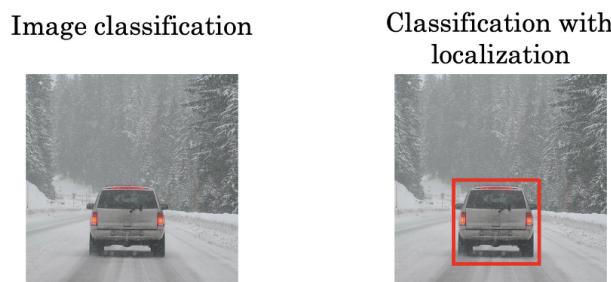
분류 모델의 경우, 모델의 output으로 각 클래스에 해당하는 확률(Softmax 함수 적용 시)이 반환된다. 예를 들어 {강아지, 고양이, 햄스터} 3개의 클래스를 분류한다고 해보자. 학습된 모델은 새로운 input 이미지가 들어왔을 때, [0.1, 0.2, 0.7] 과 같은 결과를 출력할 것이다. 이때 top-1 class는 햄스터(0.7)가 된다. 그리고 top-2 class는 고양이(0.2)가 된다.

이런 방식으로, 모델이 새로운 input 데이터에 대해 예측한 클래스, 즉 top-1 class가 실제 클래스와 같다면, top-1 error는 0이 될 것이다. 이를 확장하여 1000개의 클래스가 있을 경우, 모델의 top-5 class 중 실제 클래스가 존재한다면 top-5 error는 0이 된다.

이처럼 ImageNet과 같이 1000개라는 많은 클래스가 존재하는 경우, top-1 error뿐만 아니라 top-5 error도 같이 확인하는 경우가 많다.

8. Localisation

- Input 이미지의 class 뿐만 아니라 해당 물체가 어디 있는지까지 찾아내는 task이다. 물체의 위치를 찾기 위해 해당 예측 영역의 bounding box 좌표(박스 중심좌표 (x,y), width, height)도 함께 학습해야 하기 때문에, 기존의 cross entropy 대신 Euclidean Loss를 사용했다고 한다. 모델 구조는 VGG16을 사용하였고, 그 결과 2014 ILSVRC Localisation 대회에서 1위(25.3% error)를 달성했다.



References

- [1] <https://medium.com/@msmapark2/vgg16-논문-리뷰-very-deep-convolutional-networks-for-large-scale-image-recognition-6f748235242a>
- [2] <https://phil-baek.tistory.com/entry/1-Very-Deep-Convolutional-Networks-for-Large-Scale-Image-Recognition-VGGNet-논문-리뷰>
- [3] <https://medium.com/@msmapark2/fcn-논문-리뷰-fully-convolutional-networks-for-semantic-segmentation-81f016d76204>
- [4] <https://bskyvision.com/422>
- [5] <https://taeguu.tistory.com/29>
- [6] <https://89douner.tistory.com/61>
- [7] <https://jadon.tistory.com/26>