# VGG 논문 리뷰

< Very Deep Convolutional Networks For Large-Scale Image Recognition >

## 1. 개요

VGG 는 ImageNet Challenge 2014 에서 Top-5 테스트 정확도를 92.7%까지 끌어 올렸다. 우승작인 GoogleNet 과 0.1%의 근소한 차이로 2 위에 입상했다. 하지만 GoogleNet 보다 굉장히 단순한 구조를 가지고 있었고 이것은 신경망의 깊이가 정확도에 영향을 준다는 것을 잘 보여주었다.

## 2. 구조

VGG 모델은 신경망의 깊이가 딥러닝 정확도에 영향을 주는 지를 확인하기 위해서 11 - layer 부터 19 - layer 까지 총 5 개의 모델을 준비했다. 여기서 다룰 모델을 VGG - 16 으로 16 개의 layer 를 가진 모델이다.

| 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<br>**LRN** | conv3-64<br>**conv3-64** | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 | conv3-64<br>conv3-64 |
| maxpool | | | | | |
| conv3-128 | conv3-128 | conv3-128<br>**conv3-128** | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 | conv3-128<br>conv3-128 |
| maxpool | | | | | |
| conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256 | conv3-256<br>conv3-256<br>**conv1-256** | conv3-256<br>conv3-256<br>**conv3-256** | conv3-256<br>conv3-256<br>conv3-256<br>**conv3-256** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512 | conv3-512<br>conv3-512<br>**conv1-512** | conv3-512<br>conv3-512<br>**conv3-512** | conv3-512<br>conv3-512<br>conv3-512<br>**conv3-512** |
| maxpool | | | | | |
| FC-4096 | | | | | |
| FC-4096 | | | | | |
| FC-1000 | | | | | |
| soft-max | | | | | |

간략한 표현을 위해 Activation function 을 생략함

1) Input

위의 표를 보면 알 수 있겠지만 입력 이미지의 크기는 224 x 224x 3 으로 고정이다.

2) Convolutional layer (conv layer)

Conv3 - 64 가 의미하는 것은 3x3 의 kernel size 를 사용했고 64 개의 filter 를 가졌다는 의미이다. 여기서 3x3 의 작은 kernel size 를 사용한 이유는 3x3 이 상하좌우, 중앙을 표현할 수 있는 가장 작은 단위이기 때문이다.

첫번째 conv layer 의 Filter 개수는 64 개이고 Filter 의 개수가 512 에 도달할 때 까지 MaxPooling layer 이후에 2 배씩 계속 증가시켜준다.

Padding 을 적용하여 image size 을 유지한다.

- Kernel size = 3 x 3
- Stride = 1
- Padding = same
- Activation Function = ReLu

3) MaxPooling2D

conv layer 이후 Maxpooling 을 이용하여 출력의 크기를 절반으로 줄여준다.

이렇게 해주는 이유는 Feature map 에 있는 작고 사소한 특징이 출력에 큰 영향을 미치지 못하도록 하기 위함이다.

- Window size = 2 x 2
- Stride = 2

4) Fully Connected(FC) layer

1000 개의 클래스를 분류하기 위해 3 번째 FC layer 에는 1000 개의 channel 을 가진다.

출력이 각 클래스에 속할 확률이어야 하므로 soft-max function 을 이용한다.

- First Two layers = 4096 channels each
- Third layers = 1000 channels
- Last layer = soft-max function

## 3. 7 x 7 conv layer 대신에 3 x 3 conv layer 를 이용한 이유

Model: "sequential_4"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_2 (Conv2D) | (None, 224, 224, 64) | 1792 |
| conv2d_3 (Conv2D) | (None, 224, 224, 64) | 36928 |
| max_pooling2d_1 (MaxPooling2 | (None, 112, 112, 64) | 0 |
| conv2d_4 (Conv2D) | (None, 112, 112, 128) | 73856 |
| conv2d_5 (Conv2D) | (None, 112, 112, 128) | 147584 |
| max_pooling2d_2 (MaxPooling2 | (None, 56, 56, 128) | 0 |
| conv2d_6 (Conv2D) | (None, 56, 56, 256) | 295168 |
| conv2d_7 (Conv2D) | (None, 56, 56, 256) | 590080 |
| conv2d_8 (Conv2D) | (None, 56, 56, 256) | 590080 |
| max_pooling2d_3 (MaxPooling2 | (None, 28, 28, 256) | 0 |
| conv2d_9 (Conv2D) | (None, 28, 28, 512) | 1180160 |
| conv2d_10 (Conv2D) | (None, 28, 28, 512) | 2359808 |
| conv2d_11 (Conv2D) | (None, 28, 28, 512) | 2359808 |
| max_pooling2d_4 (MaxPooling2 | (None, 14, 14, 512) | 0 |
| conv2d_12 (Conv2D) | (None, 14, 14, 512) | 2359808 |
| conv2d_13 (Conv2D) | (None, 14, 14, 512) | 2359808 |
| conv2d_14 (Conv2D) | (None, 14, 14, 512) | 2359808 |
| max_pooling2d_5 (MaxPooling2 | (None, 7, 7, 512) | 0 |

Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0

< Three 3 x 3 conv layer >

Model: "sequential_5"

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_20 (Conv2D) | (None, 224, 224, 64) | 1792 |
| conv2d_21 (Conv2D) | (None, 224, 224, 64) | 36928 |
| max_pooling2d_9 (MaxPooling2 | (None, 112, 112, 64) | 0 |
| conv2d_22 (Conv2D) | (None, 112, 112, 128) | 73856 |
| conv2d_23 (Conv2D) | (None, 112, 112, 128) | 147584 |
| max_pooling2d_10 (MaxPooling | (None, 56, 56, 128) | 0 |
| conv2d_24 (Conv2D) | (None, 56, 56, 256) | 1605888 |
| max_pooling2d_11 (MaxPooling | (None, 28, 28, 256) | 0 |
| conv2d_25 (Conv2D) | (None, 28, 28, 512) | 6423040 |
| max_pooling2d_12 (MaxPooling | (None, 14, 14, 512) | 0 |
| conv2d_26 (Conv2D) | (None, 14, 14, 512) | 12845568 |
| max_pooling2d_13 (MaxPooling | (None, 7, 7, 512) | 0 |

Total params: 21,134,656
Trainable params: 21,134,656
Non-trainable params: 0

< Single 7 x 7 conv layer >

1) 위 표처럼 3 개의 3 x 3 conv layer 를 사용하면 7 x 7 conv layer 1 개를 사용했을 때 보다 파라미터의 개수가 줄어 드는 것을 볼 수 있다. 파라미터 개수의 감소는 연산량 감소로 이어지므로 훈련 속도에 영향을 준다.

2) Single 7 x 7 conv layer 를 사용했을 때의 feature map 과 3 개의 3 x 3 conv layer 를 사용했을 때의 feature map 의 크기가 동일하다. 예를 들어 10 x 10 image 가 있다고 가정한다면 7 x 7 filter size(=receptive field)를 적용하면 4 x 4 가 출력으로 나온다. 3 x 3 filter size 를 3 번 적용하면 8x8 -> 6x6 -> 4x4 로 7 x 7 filter size 와 동일하다.

3) 3 번의 activation function 을 사용하므로 1 번 사용하는 것보다 비선형성이 높아지고 이것은 특징을 잘 식별할 수 있게한다.

## 4. 훈련

- Optimizing the multinomial logistic regression
- Mini-batch gradient descent
- Momentum (0.9)
- L2 norm
- Dropout (0.5)
- Learning rate (0.01

Learning rate 의 경우 validation accuracy 의 증가가 멈출 때까지 10 배씩 계속 감소시킨다.

1) 훈련 전 이미지를 S 의 크기로 rescale 한 후 224x224 크리고 잘라낸다. 여기서 S 는 하나의 고정된 값(Single scale)일 수도 있고 범위를 가진 값(Multi scale)일 수도 있다.

2) 입력 이미지를 잘라서 데이터 개수를 늘리므로 이것은 Data Augmentation 으로 볼 수 있다.

3) 먼저 층이 얕은 모델 A 를 적당히 훈련시킨 후 처음 4 개의 conv layer 와 마지막 3 개의 FC layer 를 다음에 학습할 네트워크의 초기값으로 활용했다. 모델 A 를 훈련 시킬 때 랜덤 초기화를 진행했는데 weight 는 평균 0, 분산은 0.01, 편향은 0 으로 초기화 시켰다.

## 5. 테스트

테스트 전 테스트 이미지를 Q 의 크기로 rescale 한다  S 와 의미는 같지만 Q 는 테스트에서만 사용하고 S 는 훈련에서만 사용한다.

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 Scale Evaluation>

1) 3 개의 1x1 conv-layer 가 더 추가됐다는 점을 제외하면 B 와 C 의 구조는 동일한데 C 의 성능이 B 보다 높다. 이것은 비선형성을 추가하는 것이 모델의 정확도를 높이는데 도움을 준다는 것을 의미한다.

2) 1x1 conv-layer 대신에 3x3 conv-layer 이 들어있는 D 가 C 보다 성능이 높다. 이것은 kermel size 가 정확도에 영향을 준다는 것을 의미한다.

3) Q 가 single scale 임에도 불과하고 S 가 single scale 때 보다 multi scale 일 때 높은 성능을 보여준다.

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 Scale Evaluation>

4) 위 표를 보면 범위를 무작위로 변경하면서 훈련하는 것이(=scale jittering)이 정확도 상승에 도움을 준다는 것을 알 수 있다.

5) VGG 모델은 19 - layer 에서 정확도가 수렴한다는 것을 알 수 있다.

## 6. 최종 성능

| 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 | - |

모델 D 와 E 를 앙상블하여 multi-crop & dense evaluation 이라는 메소드를 이용하여 Top-5 에러가 6.8%로 1 위 모델인 GoogleNet 과 0.1%차를 보여주는 것을 알 수 있다.

## 7. 결론

이 논문을 통해서 층의 깊이와 receptive field (=kernel size)가 정확도에 미치는 영향을 알 수 있었다. 층을 쌓을수록 정확도가 올라가는 것을 보아 데이터 셋이 더 큰 경우에는 더 깊은 층을 쌓을 수 있다는 것을 알 수 있다.
논문에서 여러가지 평가방법을 언급하였고 이 평가방법과 앙상블을 사용하여 정확도를 올렸다. 이런 점을 본다면 평가방법이 정확도에 끼치는 영향과 앙상블에 대해서 조금 더 공부할 필요가 있다고 생각한다.

**Reference**


- Very Deep Convolutional Networks For Large-Scale Image Recognition (Karen Simonyan & Andrew Zisserman , University of Oxford)