


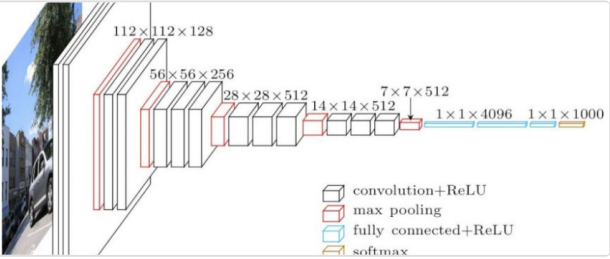
# DNN\_02.7 VGG


🕒 생성일	@2022년 6월 23일 오전 12:34
🏷️ 유형	머신러닝/딥러닝
👤 작성자	 동훈 오

**VGG16 논문 리뷰-Very Deep Convolutional Networks for Large-Scale Image Recognition**

VGG-16 모델은 ImageNet Challenge에서 Top-5 테스트 정확도를 92.7% 달성하면서 2014년 컴퓨터 비전을 위한 딥러닝 관련 대표적 연구 중 하나로 자리매김하였다. Very Deep Convolutional Networks for Large-Scale Image Recognition 논문의 내용을 살펴보면서 VGG 모델이 어떻게 이미지 분류 문제에서 이토록 좋은 성과에 도달할 수 있었는

 <https://medium.com/@msmapark2/vgg16-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-very-deep-convolutional-networks-for-large-scale-image-recognition-6f748235242a>

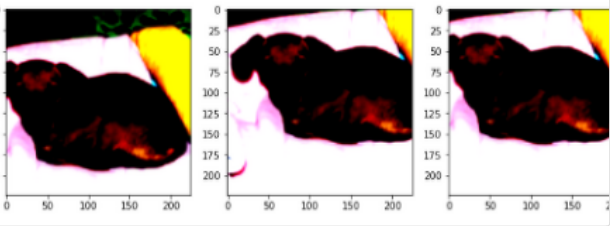


 <https://arxiv.org/pdf/1409.1556.pdf>

**[논문 구현] PyTorch로 VGGnet(2014) 구현하기**

VGGnet 논문 리뷰는 아래 포스팅에서 확인하실 수 있습니다. 전체 코드는 에서 확인하실 수 있습니다! 스타 눌러주신다면 감사하겠습니다! 아직 내용이 많이 빈약하지만 도움이 될 수 있도록 꾸준히 갱신하겠습니다...!! 데이터셋은 torchvision 패키지에서 제공하는 STL10 dataset을 이용하겠습니다. STL10 dataset은 10개의 label을 갖습니다. 작업 환경은 구글

🔗 <https://deep-learning-study.tistory.com/521>

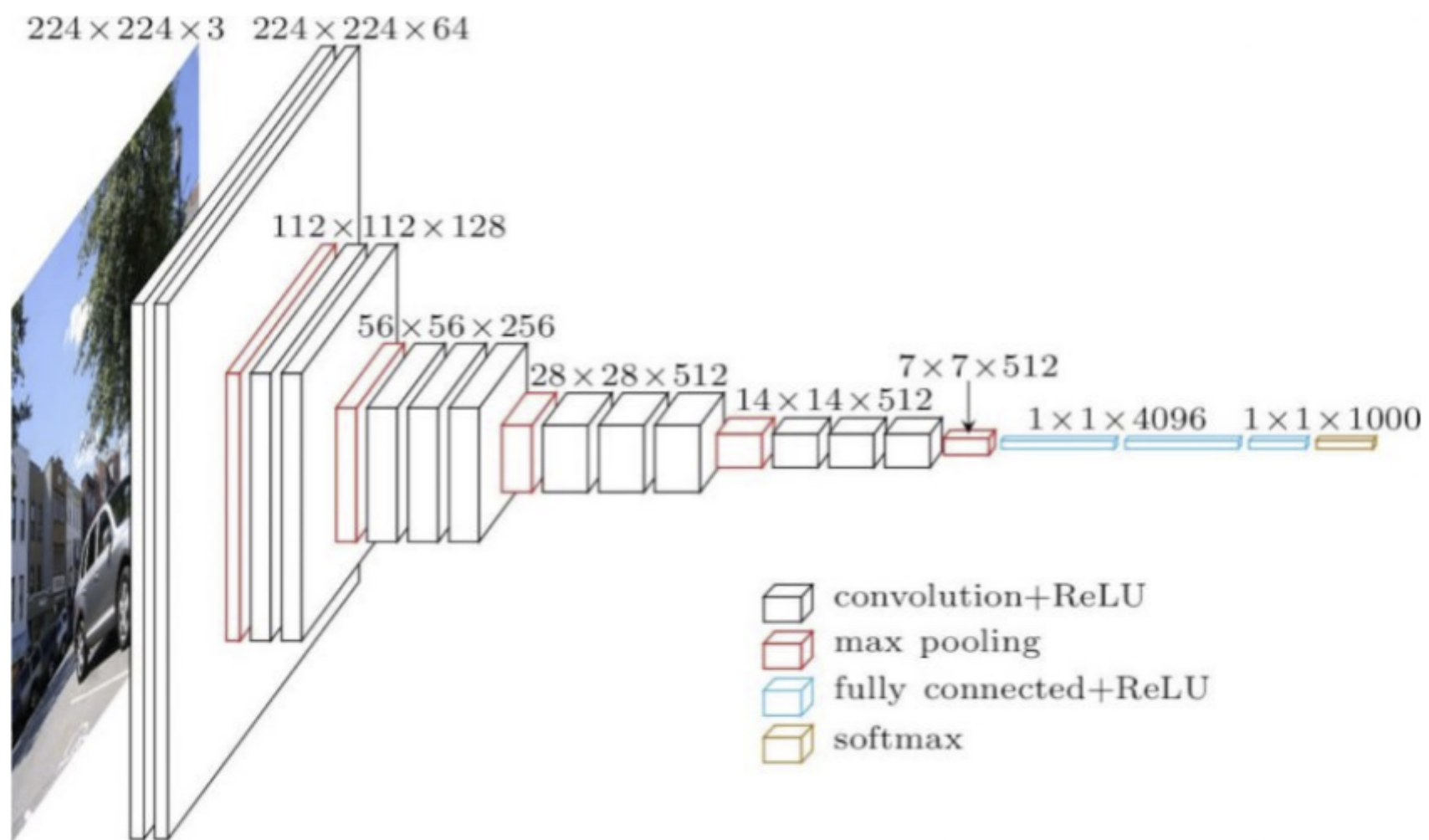


VGG 네트워크 구조는 2015 ICLR conference paper 에서 소개된 네트워크 이다. 2015년 이후 발표된 여러 컴퓨터 비전 네트워크, 기법들에서 back bone 네트워크로 VGG가 다양하게 활용되었을 만큼 의미가 있다.

논문의 제목은 ‘Very Deep Convolutional Networks for Large-Scale Image Recognition’ 이다.

VGG 네트워크란 말은 Visual Geometry Group 의 줄임말로, 논문의 저자들이 소속된 옥스포드 대학의 Engineering Science 부서의 특정 group 을 의미하는 것 같다.

VGG 모델은 AlexNet(2012) 의 8-layers 모델보다 깊이가 2배 이상 깊은 16-layers 의 네트워크 학습에 성공했다. VGG 모델이 16 또는 19 레이어에 달하는 깊은 신경망을 학습할 수 있었던 것은 모든 conv layer 에서 3\*3 필터를 사용했기 때문이다.



VGG16 Architecture

## Abstract

In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small ( $3 \times 3$ ) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively. We also show that our representations generalise well to other datasets, where they achieve state-of-the-art results. We have made our two best-performing ConvNet models publicly available to facilitate further research on the use of deep visual representations in computer vision.

## 1. Introduction

With ConvNets becoming more of a commodity in the computer vision field, a number of attempts have been made to improve the original architecture of Krizhevsky et al. (2012) in a bid to achieve better accuracy. For instance, the best-performing submissions to the ILSVRC-2013 (Zeiler & Fergus, 2013; Sermanet et al., 2014) utilised smaller receptive window size and smaller stride of the first convolutional layer. Another line of improvements dealt with training and testing the networks densely over the whole image and over multiple scales (Sermanet et al., 2014; Howard, 2014). In this paper, we address another important aspect of ConvNet architecture design – its depth. To this end, we fix other parameters of the architecture, and steadily increase the depth of the network by adding more convolutional layers, which is feasible due to the use of very small ( $3 \times 3$ ) convolution filters in all layers.

## 2. ConvNet Configurations

### 2.1. Architecture

During training, the input to our ConvNets is a fixed-size  $224 \times 224$  RGB image. The only pre-processing we do is subtracting the mean RGB value, computed on the training set, from each pixel. The image is passed through a stack of convolutional (conv.) layers, where we use filters with a very small receptive field:  $3 \times 3$  (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations we also utilise  $1 \times 1$  convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for  $3 \times 3$  conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a  $2 \times 2$  pixel window, with stride 2.

A stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

### 2.2. Configurations

The ConvNet configurations, evaluated in this paper, are outlined in Table I, one per column. In the following we will refer to the nets by their names (A–E). All configurations follow the generic design presented in Sect. 2.1, and differ only in the depth: from 11 weight layers in the network A (8 conv. and 3 FC layers) to 19 weight layers in the network E (16 conv. and 3 FC layers). The width of conv. layers (the number of channels) is rather small, starting from 64 in the first layer and then increasing by a factor of 2 after each max-pooling layer, until it reaches 512.



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 ( $224 \times 224$ RGB image)					
conv3-64	conv3-64 <b>LRN</b>	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	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 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 conv3-256 <b>conv3-256</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
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

## 2.3. Discussion

Our ConvNet configurations are quite different from the ones used in the top-performing entries of the ILSVRC-2012 (Krizhevsky et al., 2012) and ILSVRC-2013 competitions (Zeiler & Fergus, 2013; Sermanet et al., 2014). Rather than using relatively large receptive fields in the first conv. layers (e.g.  $11 \times 11$  with stride 4 in (Krizhevsky et al., 2012), or  $7 \times 7$  with stride 2 in (Zeiler & Fergus, 2013; Sermanet et al., 2014)), we use very small  $3 \times 3$  receptive fields throughout the whole net, which are convolved with the input at every pixel (with stride 1). It is easy to see that a stack of two  $3 \times 3$  conv. layers (without spatial pooling in between) has an effective receptive field of  $5 \times 5$ ; three

such layers have a  $7 \times 7$  effective receptive field. So what have we gained by using, for instance, a stack of three  $3 \times 3$  conv. layers instead of a single  $7 \times 7$  layer? First, we incorporate three non-linear rectification layers instead of a single one, which makes the decision function more discriminative. Second, we decrease the number of parameters: assuming that both the input and the output of a three-layer  $3 \times 3$  convolution stack has  $C$  channels, the stack is parametrised by  $3(3^2C^2) = 27C^2$  weights; at the same time, a single  $7 \times 7$  conv. layer would require  $7^2C^2 = 49C^2$  parameters, i.e. 81% more. This can be seen as imposing a regularisation on the  $7 \times 7$  conv. filters, forcing them to have a decomposition through the  $3 \times 3$  filters (with non-linearity injected in between).

### 3. Classification Framework

#### 3.1. Training

The ConvNet training procedure generally follows [Krizhevsky et al. \(2012\)](#) (except for sampling the input crops from multi-scale training images, as explained later). Namely, the training is carried out by optimising the multinomial logistic regression objective using mini-batch gradient descent (based on back-propagation ([LeCun et al., 1989](#))) with momentum. The batch size was set to 256, momentum to 0.9. The training was regularised by weight decay (the  $L_2$  penalty multiplier set to  $5 \cdot 10^{-4}$ ) and dropout regularisation for the first two fully-connected layers (dropout ratio set to 0.5). The learning rate was initially set to  $10^{-2}$ , and then decreased by a factor of 10 when the validation set accuracy stopped improving. In total, the learning rate was decreased 3 times, and the learning was stopped after 370K iterations (74 epochs). We conjecture that in spite of the larger number of parameters and the greater depth of our nets compared to ([Krizhevsky et al., 2012](#)), the nets required less epochs to converge due to (a) implicit regularisation imposed by greater depth and smaller conv. filter sizes; (b) pre-initialisation of certain layers.

The initialisation of the network weights is important, since bad initialisation can stall learning due to the instability of gradient in deep nets. To circumvent this problem, we began with training the configuration A (Table I), shallow enough to be trained with random initialisation. Then, when training deeper architectures, we initialised the first four convolutional layers and the last three fully-connected layers with the layers of net A (the intermediate layers were initialised randomly). We did not decrease the learning rate for the pre-initialised layers, allowing them to change during learning. For random initialisation (where applicable), we sampled the weights from a normal distribution with the zero mean and  $10^{-2}$  variance. The biases were initialised with zero. It is worth noting that after the paper submission we found that it is possible to initialise the weights without pre-training by using the random initialisation procedure of [Glorot & Bengio \(2010\)](#).

To obtain the fixed-size  $224 \times 224$  ConvNet input images, they were randomly cropped from rescaled training images (one crop per image per SGD iteration). To further augment the training set, the crops underwent random horizontal flipping and random RGB colour shift ([Krizhevsky et al., 2012](#)). Training image rescaling is explained below.

#### About training image size :

We consider two approaches for setting the training scale  $S$ . The first is to fix  $S$ , which corresponds to single-scale training (note that image content within the sampled crops can still represent multi-scale image statistics). In our experiments, we evaluated models trained at two fixed scales:  $S = 256$  (which has been widely used in the prior art ([Krizhevsky et al., 2012](#); [Zeiler & Fergus, 2013](#); [Sermanet et al., 2014](#))) and  $S = 384$ . Given a ConvNet configuration, we first trained the network using  $S = 256$ . To speed-up training of the  $S = 384$  network, it was initialised with the weights pre-trained with  $S = 256$ , and we used a smaller initial learning rate of  $10^{-3}$ .

The second approach to setting  $S$  is multi-scale training, where each training image is individually rescaled by randomly sampling  $S$  from a certain range  $[S_{min}, S_{max}]$  (we used  $S_{min} = 256$  and  $S_{max} = 512$ ). Since objects in images can be of different size, it is beneficial to take this into account during training. This can also be seen as training set augmentation by scale jittering, where a single



## 4. Classification Experiments

**Dataset.** In this section, we present the image classification results achieved by the described ConvNet architectures on the ILSVRC-2012 dataset (which was used for ILSVRC 2012–2014 challenges). The dataset includes images of 1000 classes, and is split into three sets: training (1.3M images), validation (50K images), and testing (100K images with held-out class labels). The classification performance is evaluated using two measures: the top-1 and top-5 error. The former is a multi-class classification error, i.e. the proportion of incorrectly classified images; the latter is the

실험 수행 내용을 요약하면 다음과 같다.

### 4.1. Single Scale Evaluation

We begin with evaluating the performance of individual ConvNet models at a single scale with the layer configurations.

We observe that the classification error decreases with the increased ConvNet depth: from 11 layers in A to 19 layers in E.

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

### 4.5. Comparison with the State Of The Art(SOTA)

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.)	<b>23.7</b>	<b>6.8</b>	<b>6.8</b>
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)	-	<b>6.7</b>	
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	-

## 5. Conclusion

In this work we evaluated very deep convolutional networks (up to 19 weight layers) for large-scale image classification. It was demonstrated that the representation depth is beneficial for the classification accuracy, and that state-of-the-art performance on the ImageNet challenge dataset can be achieved using a conventional ConvNet architecture (LeCun et al., 1989; Krizhevsky et al., 2012) with substantially increased depth. In the appendix, we also show that our models generalise well to a wide range of tasks and datasets, matching or outperforming more complex recognition pipelines built around less deep image representations. Our results yet again confirm the importance of depth in visual representations.