

VGG

김도영 이채영 함희정

Abstract

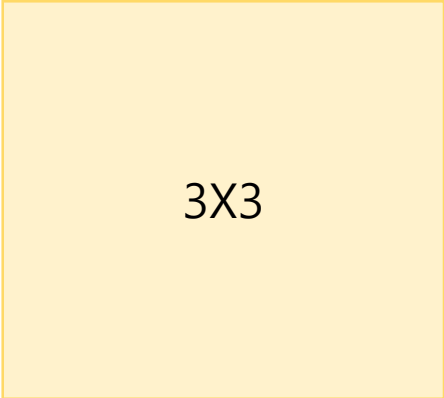
In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with **very small (3x3) convolution filters**, which shows that 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.

Configuration

During training, the input to our ConvNets is a fixed-size 224×224 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: 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations we also utilise 1×1 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 3×3 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 2×2 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.

All hidden layers are equipped with the rectification (ReLU (Krizhevsky et al., 2012)) non-linearity. We note that none of our networks (except for one) contain Local Response Normalisation (LRN) normalisation (Krizhevsky et al., 2012): as will be shown in Sect. 4, such normalisation does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time. Where applicable, the parameters for the LRN layer are those of (Krizhevsky et al., 2012).



3X3

																	Number of Parameters (millions)	Top-5 Error Rate (%)		
Image	Conv3-64	Max pool	Conv3-128	Max pool	Conv3-256	Conv3-256	Max pool	Conv3-512	Conv3-512	Max pool	FC-4096	FC-4096	FC-1000	Soft-max	133	10.4				
VGG-11																				
Image	Conv3-64	LRN	Max pool	Conv3-128	Max pool	Conv3-256	Conv3-256	Max pool	Conv3-512	Conv3-512	Max pool	FC-4096	FC-4096	FC-1000	Soft-max	133	10.5			
VGG-11 (LRN)																				
Image	Conv3-64	Conv3-64	Max pool	Conv3-128	Conv3-128	Max pool	Conv3-256	Conv3-256	Max pool	Conv3-512	Conv3-512	Max pool	FC-4096	FC-4096	FC-1000	Soft-max	133	9.9		
VGG-13																				
Image	Conv3-64	Conv3-64	Max pool	Conv3-128	Conv3-128	Max pool	Conv3-256	Conv3-256	Conv1-256	Max pool	Conv3-512	Conv3-512	Conv1-512	Max pool	FC-4096	FC-4096	FC-1000	Soft-max	134	9.4
VGG-16 (Conv1)																				
Image	Conv3-64	Conv3-64	Max pool	Conv3-128	Conv3-128	Max pool	Conv3-256	Conv3-256	Conv3-256	Max pool	Conv3-512	Conv3-512	Conv3-512	Max pool	FC-4096	FC-4096	FC-1000	Soft-max	138	8.8
VGG-16																				
Image	Conv3-64	Conv3-64	Max pool	Conv3-128	Conv3-128	Max pool	Conv3-256	Conv3-256	Conv3-256	Max pool	Conv3-512	Conv3-512	Conv3-512	Max pool	FC-4096	FC-4096	FC-1000	Soft-max	144	9.0
VGG-19																				

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>-<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).**

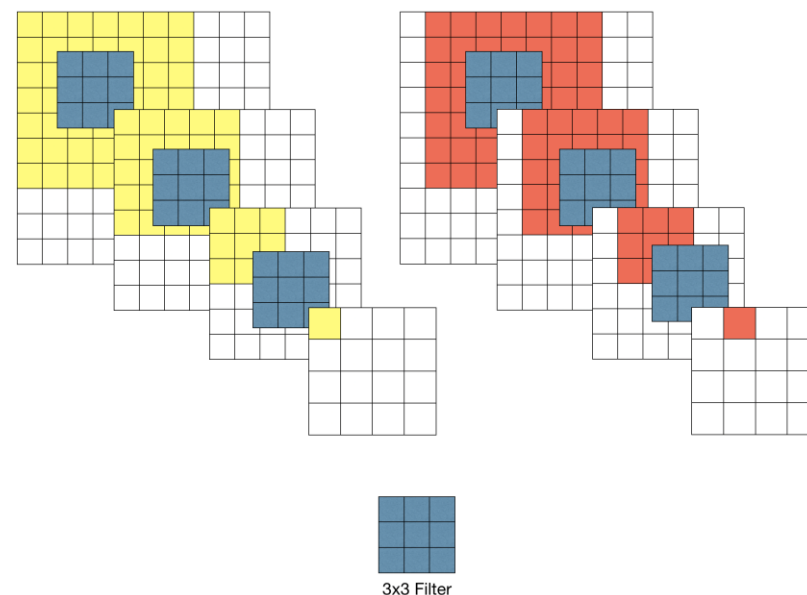
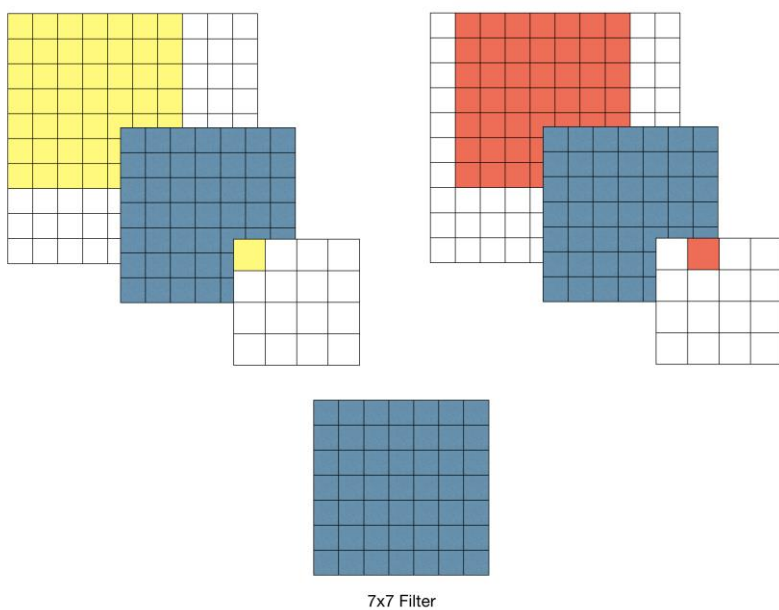
Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Why 3X3 Conv layer ?

- 7X7 conv layer 대신 3개의 3X3 Conv Layer
 - 1) 파라미터 수 의 감소
 - 2) 비 선형성의 증가

3X3 Conv layer_파라미터 수의 감소

- 7X7 conv layer 대신 3개의 3X3 Conv Layer



3X3 Conv layer_파라미터 수의 감소

- 7X7 conv layer 대신 3개의 3X3 Conv Layer

Suppose input is $H \times W \times C$ and we use convolutions with C filters to preserve depth (stride 1, padding to preserve H, W)

one CONV with 7×7 filters

Number of weights:

$$= C \times (7 \times 7 \times C) = 49 C^2$$

Number of multiply-adds:

$$\begin{aligned} &= (H \times W \times C) \times (7 \times 7 \times C) \\ &= 49 HWC^2 \end{aligned}$$

three CONV with 3×3 filters

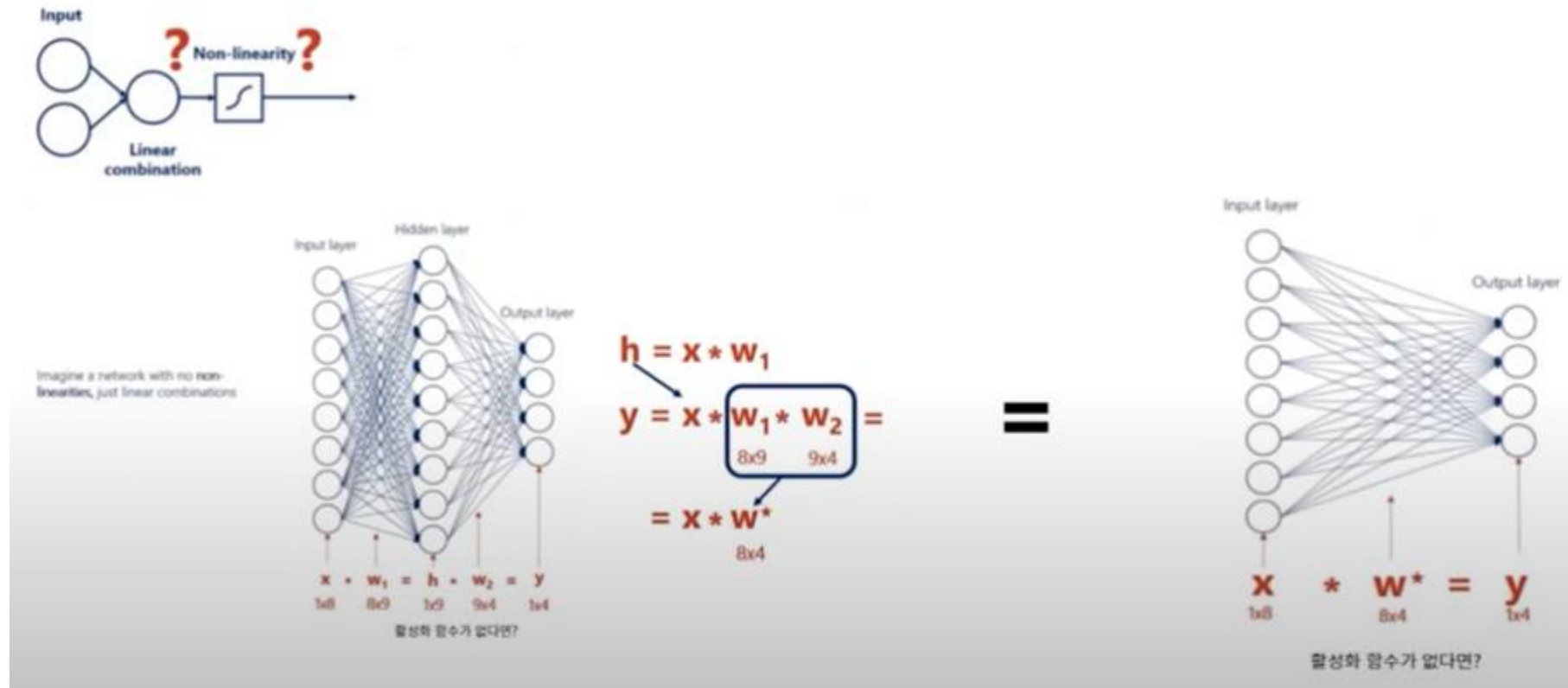
Number of weights:

$$= 3 \times C \times (3 \times 3 \times C) = 27 C^2$$

Number of multiply-adds:

$$\begin{aligned} &= 3 \times (H \times W \times C) \times (3 \times 3 \times C) \\ &= 27 HWC^2 \end{aligned}$$

3X3 Conv layer_비 선형성의 증가



Training

Optimization

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 L2 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.

Pre-training

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 1), shallow enough to be trained with random initialisation. Then, when training deeper architectures, we initialised the first four convolutional layers and the last three fullyconnected 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 224×224 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

Training

- Single scale
- Multi scale
- Multi crop

Single scale

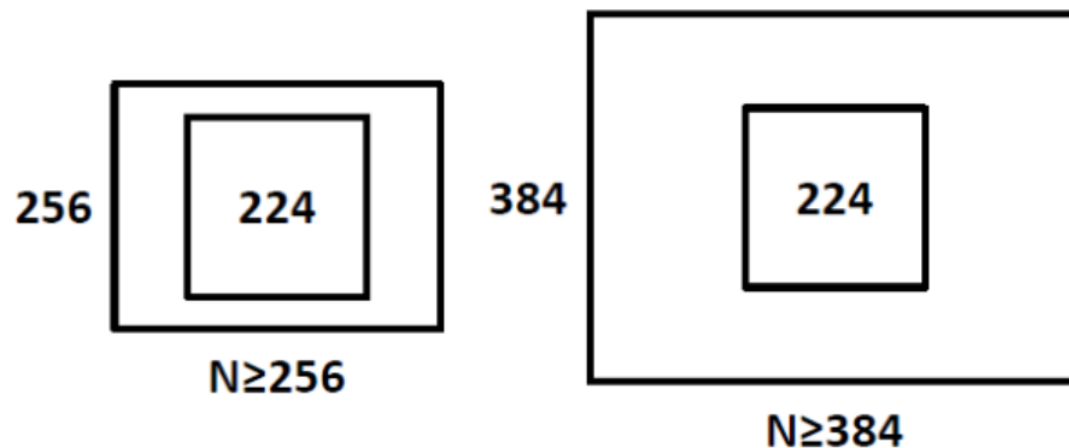
S : the smallest side of an isotropically-rescaled training image

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} .

1> single-scale training

$S = 256$

$S = 384$



Single-Scale Training with $S=256$ and $S=384$

Multi scale

S : the smallest side of an isotropically-rescaled training image

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 model is trained to recognise objects over a wide range of scales. For speed reasons, we trained multi-scale models by fine-tuning all layers of a single-scale model with the same configuration, pre-trained with fixed $S = 384$.

2> multi-scale training

$S = [256, 512]$ jittering

Multi scale



224x224



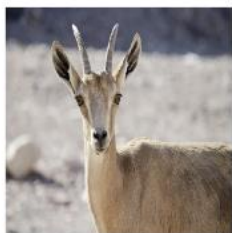
256x256



512x512

isotropically-rescaled training image

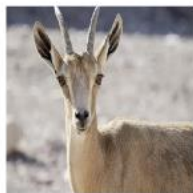
Multi crop



256x256



224x224



224x224



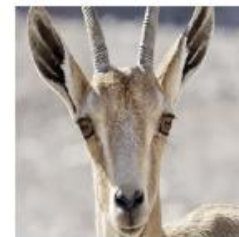
224x224



224x224



512x512



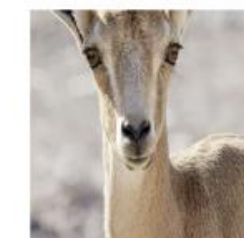
224x224



224x224



224x224



224x224

Test

Single scale

First, it is isotropically rescaled to a pre-defined smallest image side, denoted as Q (we also refer to it as the test scale).

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

Classification Framework - Multi scale

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

training one: $Q = \{S - 32, S, S + 32\}$. At the same time, scale jittering at training time allows the network to be applied to a wider range of scales at test time, so the model trained with variable $S \in [S_{\min}; S_{\max}]$ was evaluated over a larger range of sizes $Q = \{S_{\min}, 0.5(S_{\min} + S_{\max}), S_{\max}\}$.

Testing

Then, the network is applied densely over the rescaled test image in a way similar to (Sermanet et al., 2014). Namely, the fully-connected layers are first converted to convolutional layers (the first FC layer to a 7×7 conv. layer, the last two FC layers to 1×1 conv. layers).

Finally, to obtain a fixed-size vector of class scores for the image, the class score map is spatially averaged (sum-pooled).

Since the fully-convolutional network is applied over the whole image, there is no need to sample multiple crops at test time (Krizhevsky et al., 2012), which is less efficient as it requires network re-computation for each crop. While we believe that in practice the increased computation time of multiple crops does not justify the potential gains in accuracy, for reference we also evaluate our networks using 50 crops per scale (5×5 regular grid with 2 flips), for a total of 150 crops over 3 scales, which is comparable to 144 crops over 4 scales used by Szegedy et al. (2014).

Multi crop

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

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

t multiple test scales.

top-1 val. error (%)	top-5 val. error (%)
28.2	9.6
27.7	9.2
27.8	9.2
26.3	8.2
26.6	8.6
26.5	8.6
24.8	7.5
26.9	8.7
26.7	8.6
24.8	7.5

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

Conclusion

In this work we evaluated very deep convolutional networks (up to 19 weight layers) for largescale 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.

Q & A
감사합니다