Very Deep Convolutional Networks for Large-Scale Image Recognition

Karen Simonyan **Andrew Zisserman**
Visual Geometry Group, University of Oxford
{karen,az}@robots.ox.ac.uk

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, 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 the state-of-the-art results. Importantly, 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

Convolutional networks (ConvNets) have recently enjoyed a great success in large-scale visual recognition [13, 20, 21, 25] which has become possible due to the large public image repositories, such as ImageNet [4], and high-performance computing systems, such as GPUs or large-scale distributed clusters [3]. In particular, an important role in the advance of deep visual recognition architectures has been played by the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [18], which has served as a testbed for a few generations of large-scale image classification systems, from high-dimensional shallow feature encodings [16] (the winner of ILSVRC-2011) to deep ConvNets [13] (the winner of ILSVRC-2012).

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 [13] in a bid to achieve better accuracy. For instance, the best-performing submissions to the ILSVRC-2013 [20, 25] 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 [10, 20]. 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 (3×3) convolution filters in all layers.

As a result, we come up with significantly more accurate ConvNet architectures, which not only achieve the state-of-the-art accuracy on ILSVRC classification and localisation tasks, but are also applicable to other image recognition datasets, where they achieve excellent performance even when combined with a linear SVM classifier without fine-tuning. We have released our two best-performing models [1] to facilitate further research.

The rest of the paper is organised as follows. In Sect. 2, we describe our ConvNet configurations. The details of the image classification training and evaluation are then presented in Sect. 3, and the configurations are compared on the ILSVRC classification task in Sect. 4. For completeness, we also

[1] http://www.robots.ox.ac.uk/~vgg/research/very_deep/

describe and assess our ILSVRC object localisation system in Sect. 5, and discuss the generalisation of very deep features to other datasets in Sect. 6. Sect. 7 concludes the paper, and Appendix A contains the list of major paper revisions.

2 ConvNet Configurations

To measure the improvement brought by the increased ConvNet depth in a fair setting, all our ConvNet layer configurations are designed using the same principles, inspired by [2, 13]. In this section, we first describe a generic layout of our ConvNet configurations (Sect. 2.1) and then detail out the specific configurations used in the evaluation (Sect. 2.2). Our design choices are then discussed and compared to the prior art in Sect. 2.3.

2.1 Architecture

The input to a ConvNet 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). We also experimented with 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 5 max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is carried out over a 2×2 pixel window, with stride 2.

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

All weight layers (except for the last fully-connected classification layer) are equipped with the rectification (ReLU [13]) non-linearity. We note that our networks (apart from one) do not contain Local Response Normalisation (LRN) normalisation [13]: 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 the same as in [13].

2.2 Configurations

The ConvNet configurations, evaluated in this paper, are outlined in Table 1, 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.

In Table 2 we report the number of parameters for each configuration. In spite of a large depth, the number of weights in our nets is not greater than the number of weights in a more shallow net with larger conv. layer widths and receptive fields (144M weights in [20]).

2.3 Discussion

Our ConvNet configurations are quite different from the ones used in the top-performing entries to ILSVRC-2012 and ILSVRC-2013 competitions [13, 20, 25]. Rather than using relatively large receptive fields in the first conv. layers (e.g. 11×11 with stride 4 in [13], or 7×7 with stride 2 in [20, 25]), we use very small 3×3 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 3×3 conv. layers (without spatial pooling in between) has an effective receptive field of 5×5; three such layers have a 7×7 effective receptive field. So what have we gained by using, for instance, a stack of three 3×3 conv. layers instead of a single 7×7 layer? First, we incorporate three non-linear rectification layers instead of a single one, which makes the decision function more discriminative. Second, we

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.

<table>
<tr><td colspan="6">ConvNet Configuration</td></tr>
<tr><td>A</td><td>A-LRN</td><td>B</td><td>C</td><td>D</td><td>E</td></tr>
<tr><td>11 weight layers</td><td>11 weight layers</td><td>13 weight layers</td><td>16 weight layers</td><td>16 weight layers</td><td>19 weight layers</td></tr>
<tr><td colspan="6">input (224×224 RGB image)</td></tr>
<tr><td>conv3-64</td><td>conv3-64
LRN</td><td>conv3-64
conv3-64</td><td>conv3-64
conv3-64</td><td>conv3-64
conv3-64</td><td>conv3-64
conv3-64</td></tr>
<tr><td colspan="6">maxpool</td></tr>
<tr><td>conv3-128</td><td>conv3-128</td><td>conv3-128
conv3-128</td><td>conv3-128
conv3-128</td><td>conv3-128
conv3-128</td><td>conv3-128
conv3-128</td></tr>
<tr><td colspan="6">maxpool</td></tr>
<tr><td>conv3-256
conv3-256</td><td>conv3-256
conv3-256</td><td>conv3-256
conv3-256</td><td>conv3-256
conv3-256
conv1-256</td><td>conv3-256
conv3-256
conv3-256</td><td>conv3-256
conv3-256
conv3-256
conv3-256</td></tr>
<tr><td colspan="6">maxpool</td></tr>
<tr><td>conv3-512
conv3-512</td><td>conv3-512
conv3-512</td><td>conv3-512
conv3-512</td><td>conv3-512
conv3-512
conv1-512</td><td>conv3-512
conv3-512
conv3-512</td><td>conv3-512
conv3-512
conv3-512
conv3-512</td></tr>
<tr><td colspan="6">maxpool</td></tr>
<tr><td>conv3-512
conv3-512</td><td>conv3-512
conv3-512</td><td>conv3-512
conv3-512</td><td>conv3-512
conv3-512
conv1-512</td><td>conv3-512
conv3-512
conv3-512</td><td>conv3-512
conv3-512
conv3-512
conv3-512</td></tr>
<tr><td colspan="6">maxpool</td></tr>
<tr><td colspan="6">FC-4096</td></tr>
<tr><td colspan="6">FC-4096</td></tr>
<tr><td colspan="6">FC-1000</td></tr>
<tr><td colspan="6">soft-max</td></tr>
</table>

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

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

decrease the number of parameters: assuming that both the input and the output of a three-layer 3×3 convolution stack has C channels, the stack is parametrised by $3\left(3^2C^2\right) = 27C^2$ weights; at the same time, a single 7×7 conv. layer would require $7^2C^2 = 49C^2$ parameters, i.e. 81% more. This can be seen as imposing a regularisation on the 7×7 conv. filters, forcing them to have a decomposition through the 3×3 filters (with non-linearity injected in between).

The incorporation of 1×1 conv. layers (configuration C, Table 1) is a way to increase the non-linearity of the decision function without affecting the receptive fields of the conv. layers. Even though in our case the 1×1 convolution is essentially a linear projection onto the space of the same dimensionality (the number of input and output channels is the same), additional non-linearity is introduced by the rectification function. It should be noted that the 1×1 conv. layers have recently been utilised in the "Network in Network" architecture of [15].

Small-size convolution filters have been previously used in [2], but their nets are significantly less deep than ours, and they did not evaluate on the large-scale ILSVRC dataset. GoogLeNet [23], another top-performing entry of ILSVRC-2014 which was developed independently of our work, is similar to our approach in that it is based on very deep ConvNets (22 weight layers) and small convolution filters (apart from 3×3, they also use 1×1 and 5×5 convolutions). Their network topology is, however, more complex than ours, and the spatial resolution of the feature maps is reduced more aggressively in the first layers to decrease the amount of computation.

3 Classification Framework

In the previous section we presented the details of our network configurations. In this section, we describe the details of classification ConvNet training and evaluation.

3.1 Training

The ConvNet training procedure generally follows [13] (except for using the whole training images at multiple scales, as explained later). Namely, the training is carried out using mini-batch gradient descent (based on back-propagation [14]) 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 [13], 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 will stall learning due to a large number of rectification (cut-off) non-linearities. 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 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. We note that it might be possible to appropriately initialise learning without pre-training, e.g. by adjusting the random initialisation procedure [19]. We plan to address this in the future work.

To obtain 224×224 input images, they were randomly cropped from the full-size (non-cropped) training images, isotropically rescaled so that the smallest side equals $S \geq 224$. To further augment the training set, the crops underwent random horizontal flipping and random RGB colour shift [13].

Multi-scale training. Since objects in images can appear at different scales, it is beneficial to take this into account during training by considering images of multiple size (with the ConvNet input having a fixed size of 224×224). In particular, we consider two approaches. The first is to train separate models for different image sizes, and then combine their soft-max class scores at test time. In our experiments we trained separate models at two scales ($S = 256$ and $S = 384$) using the following procedure. Given a ConvNet configuration, we first trained the network using $S = 256$, i.e. on 224×224 images cropped out of $256 \times N$ images ($N \geq 256$). We then trained another network with the same layer configuration, but using 224×224 crops from larger images ($S = 384$). To speed-up training, it was initialised with the weights pre-trained with $S = 256$, and we used a smaller initial learning rate of 10^{-3}.

An alternative way of multi-scale training is to rescale each training image individually by randomly sampling the smallest image side S from a certain range $[S_{min}; S_{max}]$, independently for each training image (we used $S_{min} = 256$ and $S_{max} = 512$). This can be seen as training set augmentation by scale jittering. This approach is different from the previous one in that a single model is trained to recognise objects over a wide range of scales. We trained such multi-scale models by fine-tuning all layers of the corresponding single-scale model, pre-trained with $S = 384$.

3.2 Testing

At test time, given a trained ConvNet and an input image, it is classified in the following way. First, it is isotropically rescaled so that the smallest side equals $Q \geq 224$, which is not necessarily equal to S (as we will show in Sect. 4, using several values of Q for each S leads to improved performance). Then, the network is applied densely over the rescaled test image in a way similar to [20]. Namely, the fully-connected layers are first converted to the convolutional layers (the first FC layer – to a 7×7 conv. layer, the last two FC layers – to 1×1 conv. layers). The resulting net, which now contains only conv. layers, is applied to the whole (uncropped) image by convolving the filters in each layer with the full-size input. The resulting output feature map is a class score map with the number of channels equal to the number of classes, and the variable spatial resolution, dependent on the input image size. Finally, to obtain a fixed-size vector of class scores for the image, the class score map is spatially averaged (sum-pooled). Since the network is applied over the whole image, there is no need to sample multiple crops at test time [13], which is less efficient as it requires re-computation of conv. layer activations.

We also use the test set augmentation by horizontal flipping of the images; the soft-max class posteriors of the original and flipped images are averaged to obtain the final scores for the image.

3.3 Implementation Details

Our implementation is derived from the publicly available C++ Caffe toolbox [11] (branched out in December 2013), but contains a number of significant modifications, allowing us to perform training and evaluation on multiple GPUs installed in a single system, as well as train and evaluate on full-size (uncropped) images at multiple scales (as described above). Multi-GPU training exploits data parallelism, and is carried out by splitting each batch of training images into several GPU batches, processed in parallel on each GPU. After the GPU batch gradients are computed, they are averaged to obtain the gradient of the full batch. Gradient computation is synchronous across the GPUs, so the result is exactly the same as when training on a single GPU.

While more sophisticated methods of speeding up ConvNet training have been recently proposed [12], which employ model and data parallelism for different layers of the net, we have found that our conceptually much simpler scheme already provides a speedup of 3.75 times on an off-the-shelf 4-GPU system, as compared to using a single GPU. On a system equipped with four NVIDIA Titan Black GPUs, training a single net took 2–3 weeks depending on the architecture.

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 2012–2014 ILSVRC 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 ratio of incorrectly classified images; the latter is the main evaluation criterion used in the ILSVRC, and is computed as the ratio of images such that the ground-truth category is not within the top-5 categories.

For the majority of experiments, we used the validation set as the test set. Certain experiments were also carried out on the test set and submitted to the official ILSVRC server as a "VGG" team entry to the ILSVRC-2014 competition [18].

4.1 Configuration Comparison

We begin with evaluating the performance of individual ConvNet models with the layer configurations described in Sect. 2.2. The results of single-scale testing are shown in Table 3.

First, we note that using local response normalisation (A-LRN network) does not improve on the model A without any normalisation layers. We thus do not employ normalisation in the deeper architectures (B–E).

Second, we observe that the classification error decreases with the increased ConvNet depth: from 11 layers in A to 19 layers in E. Notably, in spite of the same depth, the configuration C (which contains three 1×1 conv. layers), performs worse than the configuration D, which uses 3×3 conv. layers throughout the network. This indicates that while the additional non-linearity does help (C is better than B), it is also important to capture spatial context by using conv. filters with non-trivial receptive fields (D is better than C).

Finally, scale jittering at training time ($S \in [256; 512]$) leads to significantly better results that training on images with fixed smallest side ($S = 256$ or $S = 384$), even though a single scale is used at test time. This confirms that training set augmentation by scale jittering is indeed helpful for learning multi-scale image statistics.

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**

4.2 Multi-Scale Computation

Having evaluated the ConvNet models at a single scale ($Q = S$ for fixed S, and $Q = 0.5(S_{min} + S_{max})$ for jittered S), we now assess the effect of scale jittering at test time. It consists in running a model over several rescaled versions of a test image (corresponding to different Q), followed by averaging the resulting class posteriors. Considering that large discrepancy between training and testing scales leads to a drop in performance, the models trained with fixed S were evaluated over three test image sizes, close to the 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 $Q = \{S_{min}, 0.5(S_{min} + S_{max}), S_{max}\}$.

The results, presented in Table 4, indicate that scale jittering at test time leads to better performance (as compared to evaluating the same model at a single scale, shown in Table 3). As before, the deepest configurations (D and E) perform the best, and scale jittering is better than training with a fixed smallest side S. Our best single-network performance on the validation set is $24.8\%/7.5\%$ top-1/top-5 error (highlighted in bold in Table 4). On the test set, the configuration E achieves 7.3% top-5 error.

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**

4.3 ConvNet Fusion

Up until now, we evaluated the performance of individual ConvNet models. In this part of the experiments, we combine the outputs of several models by averaging their soft-max class posteriors. This improves the performance due to complementarity, and was used in the top ILSVRC submissions in 2012 [13] and 2013 [20, 25].

As can be seen from Table 5, combining two best-performing models (configurations D and E, multi-scale training) reduces the error from $24.8\%/7.5\%$ (single model, Table 4) to $24.0\%/7.1\%$. By the time of ILSVRC submission, however, we had only trained the single-scale networks, as well as the multi-scale model D (by fine-tuning only the fully-connected layers of a single-scale model). For reference, in Table 5 (last row) we report the results of combining these models. As can be seen, in spite of combining the predictions of 7 models, the accuracy is worse compared to the combination of two fully-trained multi-scale models.

Table 5: **Multiple ConvNet fusion results.** Combined models are denoted as "(configuration name/train image size/test image sizes)" (see Table 4 for individual model results).

Combined ConvNet models	Error		
	top-1 val	top-5 val	top-5 test
(D/[256;512]/256,384,512), (E/[256;512]/256,384,512)	**24.0**	**7.1**	**7.0**
(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

4.4 Comparison with the State of the Art

Finally, we compare our best-performing single-network and multiple-network results with the state of the art (Table 6). In the classification task of ILSVRC-2014 challenge [18], our "VGG" team secured the 2nd place with 7.3% test error (which was achieved using a combination of 7 models as explained in Sect. 4.3). After the submission, we decreased the error rate to 7.0% to by utilising two models, trained at multiple scales.

As can be seen from Table 6, our very deep ConvNets significantly outperform the previous generation of models, which achieved the best results in ILSVRC-2012 and ILSVRC-2013 competitions. Our result is also competitive with respect to the classification task winner (GoogLeNet with 6.7% error) and substantially outperforms the ILSVRC-2013 winning submission Clarifai, which achieved 11.2% with outside training data and 11.7% without it. This is remarkable, considering that our best result is achieved by combining just two models – significantly less than used in most ILSVRC submissions. In terms of the single-net performance, our architecture achieves the best result (7.3% test error), outperforming a single GoogLeNet by 0.6%. Notably, we did not depart from the classical ConvNet architecture [14], but pushed it to the limit by substantially increasing the depth.

Table 6: **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)	24.0	7.1	7.0
VGG (1 net)	24.8	7.5	7.3
VGG (ILSVRC submission, 7 nets)	24.7	7.5	7.3
VGG (ILSVRC submission, 1 net)	24.9	8.0	-
GoogLeNet [23] (1 net)	-	7.9	
GoogLeNet [23] (7 nets)	-	6.7	
MSRA [9] (11 nets)	-	-	8.1
MSRA [9] (1 net)	27.9	9.1	9.1
Clarifai [18] (multiple nets)	-	-	11.7
Clarifai [18] (1 net)	-	-	12.5
Zeiler & Fergus [25] (6 nets)	36.0	14.7	14.8
Zeiler & Fergus [25] (1 net)	37.5	16.0	16.1
OverFeat [20] (7 nets)	34.0	13.2	13.6
OverFeat [20] (1 net)	35.7	14.2	-
Krizhevsky et al. [13] (5 nets)	38.1	16.4	16.4
Krizhevsky et al. [13] (1 net)	40.7	18.2	-

5 Localisation

In the previous sections we have considered the classification task of the ILSVRC challenge, and performed a thorough evaluation of ConvNet architectures of different depth. In this section, we turn to the localisation task of the challenge. It can be seen as a special case of object detection, where a single object bounding box should be predicted for each of the top-5 classes, irrespective of the actual number of objects of the class. For this we adopt the approach of [20], the winners of

ILSVRC-2013 localisation challenge, with a few modifications. Our method is described in Sect. 5.1 and evaluated in Sect. 5.2.

5.1 Localisation ConvNet

To perform object localisation, we use a very deep ConvNet, where the last fully connected layer predicts the bounding box location instead of predicting class scores. A bounding box is represented by a 4-D vector storing its center coordinates, width, and height. There is a choice of whether the bounding box prediction is shared across all classes (single-class regression, SCR [20]) or is class-specific (per-class regression, PCR). In the former case, the last layer is 4-D, while in the latter it is 4000-D (since there are 1000 classes in the dataset).

Apart from the last bounding box prediction layer, we use the ConvNet architecture D (Table 1), which contains 16 weight layers and was found to be the best-performing in the classification task (Sect. 4). This allows us to speed-up training by initialising with a pre-trained classification net.

Training. Training of localisation ConvNets is similar to that of the classification ConvNets (Sect. 3.1). The main difference is that we replace the logistic regression objective with a Euclidean loss, which penalises the deviation of the predicted bounding box parameters from the ground-truth. We trained two localisation models, each on a single scale: $S = 256$ and $S = 384$ (due to the time constraints, we did not use training scale jittering for our ILSVRC-2014 submission). As noted above, training was initialised with the corresponding classification models (trained on the same scales), and the initial learning rate was set to 10^{-3}. We explored both fine-tuning all layers and fine-tuning only the first two fully-connected layers, as done in [20]. The last fully-connected layer was initialised randomly and trained from scratch.

Testing. We consider two testing protocols. The first is used for comparing different network modifications on the validation set, and considers only the bounding box prediction for the ground truth class (to factor out the classification errors). The bounding box is obtained by applying the network only to the central crop of the image.

The second, fully-fledged, testing procedure is based on the dense application of the localisation ConvNet to the whole image, similarly to the classification task (Sect. 3.2). The difference is that instead of the class score map, the output of the last fully-connected layer is a set of bounding box predictions. To come up with the final prediction, we utilise the greedy merging procedure of [20], which first merges spatially close predictions (by averaging their coordinates), and then rates them based on the class scores, obtained from the classification ConvNet. When several localisation ConvNets are used, we first take the union of their sets of bounding box predictions, and then run the merging procedure on the union. We did not use the multiple pooling offsets technique of [20], which allows to increase the spatial resolution of the bounding box predictions by compensating for the loss of resolution after spatial pooling.

5.2 Localisation Experiments

In this section we first determine the best-performing localisation setting, and then evaluate it in a fully-fledged setting, where the labels are not known and are predicted by the classification ConvNet, and the bounding box prediction is done based on the whole image. The localisation error is measured according to the ILSVRC criterion [18], that is the bounding box prediction is deemed correct if its intersection over union ratio with the ground-truth bounding box is above 0.5.

Settings comparison. As can be seen from Table 7, per-class regression (PCR) outperforms the class-agnostic single-class regression (SCR), which, interestingly, differs from the findings of [20], where PCR was outperformed by SCR. We also note that fine-tuning all layers for the localisation task leads to noticeably better results than fine-tuning only the fully-connected layers (as done in [20]). In these experiments, the smallest images side was set to $S = 384$; the results with $S = 256$ exhibit the same behaviour and are not shown for brevity.

Fully-fledged evaluation. Having determined the best localisation setting (PCR, fine-tuning of all layers), we now apply it in the fully-fledged scenario, where the top-5 class labels are predicted using our best-performing classification system (Sect. 4.4), and multiple densely-computed bounding box predictions are merged using the method of [20]. As can be seen from Table 8, application of the localisation ConvNet to the whole image substantially improves the results compared to using

Table 7: **Localisation error for different modifications** with the simplified testing protocol: the bounding box is predicted from a single central image crop, and the ground-truth class is used. All ConvNet layers (except for the last one) have the configuration D (Table 1), while the last layer performs either single-class regression (SCR) or per-class regression (PCR).

Fine-tuned layers	regression type	GT class localisation error
1st and 2nd FC	SCR	36.4
	PCR	34.3
all	PCR	**33.1**

a center crop (Table 7), despite using the top-5 predicted class labels instead of the ground truth. Similarly to the classification task (Sect. 4), testing at several scales and combining the predictions of multiple networks further improves the performance.

Table 8: **Localisation error**

smallest image side		top-5 localisation error (%)	
train (S)	test (Q)	val.	test.
256	256	29.5	-
384	384	28.2	26.7
384	352,384	27.5	-
fusion: 256/256 and 384/352,384		**26.9**	**25.3**

Comparison with the state of the art. We compare our best localisation result with the state of the art in Table 9. With 25.3% test error, our "VGG" team has won the localisation challenge of ILSVRC-2014 [18]. Notably, our results are considerably better than those of the ILSVRC-2013 winner Overfeat [20], even though we used less scales and did not employ their prediction resolution enhancement technique. We envisage that better localisation performance can be achieved if this technique is incorporated into our method. This indicates the performance advancement brought by our very deep ConvNets – we got better results with a simpler localisation method, but a more powerful representation. For reference, we also report the results of our submission to ILSVRC-2013, achieved using a much smaller ConvNet model (akin to [13]) and with weak supervision, i.e. without using the ground-truth bounding boxes.

Table 9: **Comparison with the state of the art in ILSVRC localisation**. Our method is denoted as "VGG".

Method	top-5 val. error (%)	top-5 test error (%)
VGG	26.9	25.3
GoogLeNet [23]	-	26.7
OverFeat [20]	30.0	29.9
Krizhevsky et al. [13]	-	34.2
VGG-2013 [22] (weakly-supervised)	-	46.4

6 Generalisation of Very Deep Features

In the previous sections we have discussed training and evaluation of very deep ConvNets on the ILSVRC dataset. In this section, we evaluate our ConvNets, pre-trained on ILSVRC, as feature extractors on other, smaller, datasets, where training large models from scratch is not feasible due to over-fitting. Recently, there has been a lot of interest in such a use case [1, 5, 17, 25], as it turns out that deep image representations, learnt on ILSVRC, generalise well to other datasets, where they have outperformed hand-crafted representations by a large margin. Following that line of work, we investigate if our models lead to better performance than more shallow models utilised in the state-of-the-art methods. In this evaluation, we consider two models with the best classification performance on ILSVRC (Sect. 4) – configurations "Net-D" and "Net-E" (which we made publicly available).

To utilise the ConvNets, pre-trained on ILSVRC, for image classification on other datasets, we remove the last fully-connected layer (which performs 1000-way ILSVRC classification), and use the activations of the penultimate layer as image features, which are aggregated across multiple locations and scales. The resulting image descriptor is L_2-normalised and combined with a linear

SVM classifier, trained on the dataset in question. For simplicity, pre-trained ConvNet weights are kept fixed (no fine-tuning is performed).

Aggregation of features is carried out in a similar manner to our ILSVRC evaluation procedure (Sect. 3.2). Namely, an image is first rescaled so that its smallest side equals Q, and then the network is densely applied over the image plane (which is possible if all weight layers are treated as convolutional). We then perform global average pooling on the resulting feature map, which produces a 4096-D image descriptor. The descriptor is then averaged with the descriptor of a horizontally flipped image. As was shown in Sect. 4.2, evaluation over multiple scales is beneficial, so we extract features over several scales, e.g. $Q \in \{256, 384, 512\}$ (as used in ILSVRC experiments). The resulting multi-scale features can be either stacked or pooled across scales. Stacking allows a subsequent classifier to learn how to optimally combine image statistics over a range of scales; this, however, comes at the cost of the increased descriptor dimensionality. We return to the discussion of this design choice in the experiments below. We also assess late fusion of features, computed using two networks, which is performed by stacking their respective image descriptors.

Table 10: **Comparison with the state of the art on VOC-2007, VOC-2012, Caltech-101, and Caltech-256**. Our models are denoted as "VGG". Results marked with * were achieved using ConvNets pre-trained on the *extended* ILSVRC dataset (2000 classes).

Method	VOC-2007 (mean AP)	VOC-2012 (mean AP)	Caltech-101 (mean class recall)	Caltech-256 (mean class recall)
Zeiler & Fergus [25]	-	79.0	86.5 ± 0.5	74.2 ± 0.3
Chatfield et al. [1]	82.4	83.2	88.4 ± 0.6	77.6 ± 0.1
He et al. [9]	82.4	-	$\mathbf{93.4 \pm 0.5}$	-
Wei et al. [24]	81.5 (85.2*)	81.7 (**90.3***)	-	-
VGG Net-D (16 layers)	89.3	89.0	91.8 ± 1.0	85.0 ± 0.2
VGG Net-E (19 layers)	89.3	89.0	92.3 ± 0.5	85.1 ± 0.3
VGG Net-D & Net-E	**89.7**	**89.3**	$\mathbf{92.7 \pm 0.5}$	$\mathbf{86.2 \pm 0.3}$

Image Classification on VOC-2007 and VOC-2012. We begin with the evaluation on the image classification task of PASCAL VOC-2007 and VOC-2012 benchmarks [6]. These datasets contain 10K and 22.5K images respectively, and each image is annotated with one or several labels, corresponding to 20 object categories. The VOC organisers provide a pre-defined split into training, validation, and test data (the test data for VOC-2012 is not publicly available; instead, an evaluation server is provided). Recognition performance is measured using mean average precision (mAP) across classes.

Notably, by examining the performance on the validation sets of VOC-2007 and VOC-2012, we found that aggregating image descriptors, computed at multiple scales, by averaging performs similarly to the aggregation by stacking. We hypothesize that this is due to the fact that in the VOC dataset the objects appear over a variety of scales, so there is no particular scale-specific semantics which a classifier could exploit. Since averaging has a benefit of not inflating the descriptor dimensionality, we were able to aggregated image descriptors over a wide range of scales: $Q \in \{256, 384, 512, 640, 768\}$. It is worth noting though that the improvement over a smaller range of $\{256, 384, 512\}$ was rather marginal 0.3%.

The test set performance is reported and compared with other approaches in Table 10. Our networks "Net-D" and "Net-E" exhibit identical performance on VOC datasets, and their combination slightly improves the results. Our methods set the new state of the art across image representations, pre-trained on the ILSVRC dataset, outperforming the previous best result of [1] by more than 6%. It should be noted that the method of [24], which achieves 1% better mAP on VOC-2012, is pre-trained on an extended 2000-class ILSVRC dataset, which includes additional 1000 categories, semantically close to those in VOC datasets. It also benefits from the fusion with an object detection-assisted classification pipeline.

Image Classification on Caltech-101 and Caltech-256. We also evaluated very deep features on Caltech-101 [7] and Caltech-256 [8] image classification benchmarks. Caltech-101 contains 9K images labelled into 102 classes (101 object categories and a background class), while Caltech-256 is larger with 31K images and 257 classes. A standard evaluation protocol on these datasets is to generate several random splits into training and test data and report the average recognition performance across the splits, which is measured by the mean class recall (which compensates for a different number of test images per class). Following [1, 9, 25], on Caltech-101 we generated 3 random splits into training and test data, so that each split contains 30 training images per class, and up to 50 test images per class. On Caltech-256 we also generated 3 splits, each of which contains 60 training images per class (and the rest is used for testing). In each split, 20% of training images were used as a validation set for hyper-parameter selection.

We found that unlike VOC, on Caltech datasets the stacking of descriptors, computed over multiple scales, performs better than averaging or max-pooling. This can be explained by the fact that in Caltech images objects typically occupy the whole image, so multi-scale image features are semantically different (capturing the whole object *vs*. object parts), and stacking allows a classifier to exploit such scale-specific representations. We used three scales $Q \in \{256, 384, 512\}$.

Our models are compared to each other and the state of the art in Table 10. As can be seen, the deeper 19-layer Net-E performs better than the 16-layer Net-D, and their combination further improves the performance. On Caltech-101, our representations are competitive with the approach of [9], which, however, performs significantly worse than our nets on VOC-2007. On Caltech-256, our features outperform the state of the art [1] by a large margin (8.6%).

Summary. Our experiments on PASCAL VOC and Caltech image classification datasets demonstrate that a better performance of very deep representations on ILSVRC translates into a better performance on other datasets. We note that our results are consistently high across different datasets despite simple training and evaluation procedure.

7 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 the state-of-the-art performance on the ImageNet challenge dataset can be achieved using a conventional ConvNet architecture [13, 14] with substantially increased depth. Namely, our object localisation system won the ILSVRC-2014 localisation challenge, while our classification system took the second place in the classification challenge. We have also shown that our models generalise well to other datasets, matching or outperforming more complex recognition pipelines built around less deep convolutional features. Our results yet again confirm the importance of depth in visual representations.

Acknowledgements

This work was supported by ERC grant VisRec no. 228180. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

A Paper Revisions

Here we present the list of major paper revisions, outlining the substantial changes for the convenience of the reader.

v1 Initial version. Presents the experiments carried out before the ILSVRC submission.

v2 Adds post-submission ILSVRC experiments with training set augmentation using scale jittering, which improves the performance.

v3 Adds generalisation experiments (Sect. 6) on PASCAL VOC and Caltech image classification datasets. The models used for these experiments are publicly available.

References

[1] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *Proc. BMVC.*, 2014.

[2] D. C. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Proc. CVPR*, pages 3642–3649, 2012.

[3] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, Q. V. Le, and A. Y. Ng. Large scale distributed deep networks. In *NIPS*, pages 1232–1240, 2012.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*, 2009.

[5] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *CoRR*, abs/1310.1531, 2013.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010.

[7] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE CVPR Workshop of Generative Model Based Vision*, 2004.

[8] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729v2, 2014.

[10] A. G. Howard. Some improvements on deep convolutional neural network based image classification. In *Proc. ICLR*, 2014.

[11] Y. Jia. Caffe: An open source convolutional architecture for fast feature embedding. `http://caffe.berkeleyvision.org/`, 2013.

[12] A. Krizhevsky. One weird trick for parallelizing convolutional neural networks. *CoRR*, abs/1404.5997, 2014.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[14] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[15] M. Lin, Q. Chen, and S. Yan. Network in network. In *Proc. ICLR*, 2014.

[16] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *Proc. ECCV*, 2010.

[17] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN Features off-the-shelf: an Astounding Baseline for Recognition. *CoRR*, abs/1403.6382, 2014.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[19] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *Proc. ICLR*, 2014.

[20] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In *Proc. ICLR*, 2014.

[21] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *CoRR*, abs/1406.2199, 2014.

[22] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Workshop at International Conference on Learning Representations*, 2014.

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[24] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan. CNN: Single-label to multi-label. *CoRR*, abs/1406.5726, 2014.

[25] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.