Return of the Devil in the Details: Delving Deep into Convolutional Nets

Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman
Visual Geometry Group, Department of Engineering Science, University of Oxford
{ken,karen,vedaldi,az}@robots.ox.ac.uk

Abstract—The latest generation of Convolutional Neural Networks (CNN) have achieved impressive results in challenging benchmarks on image recognition and object detection, significantly raising the interest of the community in these methods. Nevertheless, it is still unclear how different CNN methods compare with each other and with previous state-of-the-art shallow representations such as the Bag-of-Visual-Words and the Improved Fisher Vector. This paper conducts a rigorous evaluation of these new techniques, exploring different deep architectures and comparing them on a common ground, identifying and disclosing important implementation details. We identify several useful properties of CNN-based representations, including the fact that the dimensionality of the CNN output layer can be reduced significantly without having an adverse effect on performance. We also identify aspects of deep and shallow methods that can be successfully shared. A particularly significant one is data augmentation, which achieves a boost in performance in shallow methods analogous to that observed with CNN-based methods. Finally, we are planning to provide the configurations and code that achieve the state-of-the-art performance on the PASCAL VOC Classification challenge, along with alternative configurations trading-off performance, computation speed and compactness.

1 Introduction

Perhaps the single most important design choice in current state-of-the-art image classification and object recognition systems is the choice of visual features, or image representation. In fact, most of the quantitative improvements to image understanding obtained in the past dozen years can be ascribed to the introduction of improved representations, from the *Bag-of-Visual-Words* (BoVW) [1], [2] to the *(Improved) Fisher Vector* (IFV) [3]. A common characteristic of these methods is that they are largely *handcrafted*. They are also relatively simple, comprising dense sampling of local image patches, describing them by means of visual descriptors such as SIFT, encoding them into a high-dimensional representation, and then pooling over the image. Recently, these handcrafted approaches have been substantially outperformed by the introduction of the latest generation of *Convolutional Neural Networks (CNNs)* [4] to the computer vision field. These networks have a substantially more sophisticated structure than standard representations, comprising several layers of non-linear feature extractors, and are therefore said to be *deep* (in contrast, classical representation will be referred to as *shallow*). Furthermore, while their structure is handcrafted, they contain a very large number of parameters learnt from data. When applied to standard image classification and object detection benchmark datasets such as ImageNet ILSVRC [5] and PASCAL VOC [6] such networks have demonstrated excellent performance [7], [8], [9], [10], [11], significantly better than standard image encodings [12].

Despite these impressive results, it remains unclear how different deep architectures compare to each other and to shallow computer vision methods such as IFV. Most papers did not test these representations extensively on a common ground, so a systematic evaluation of the effect of different design and implementation choices remains largely missing. As previously noted by Chatfield *et al*. [12] in their comparison of shallow visual encodings, the *performance of computer vision systems depends significantly on implementation details*. For example, state-of-the-art methods such as [13] not only involve the use of a CNN, but also include other improvements such as the use of very large scale datasets, GPU computation, and data augmentation (also known as data jittering or virtual sampling). These improvements could also transfer to shallow representations such as the IFV, potentially explaining a part of the performance gap [14].

In this study we analyse and empirically clarify these issues, conducting a large set of rigorous experiments (Sect. 4), in many ways picking up the story where it last ended in [12] with the comparison of shallow encoders. We focus on methods to construct *image representations*, *i.e.* encoding functions ϕ mapping an image I to a vector $\phi(I) \in \mathbb{R}^d$ suitable for analysis with a linear classifier, such as an SVM. We consider **three scenarios** (Sect. 2, Sect. 3): shallow image representations, deep representations pre-trained on outside data, and deep representation pre-trained and then fine-tuned on the target dataset. As part of our tests, we explore **generally-applicable best practices** that are nevertheless more often found in combination with CNNs [13] or, alternatively, with shallow encoders [12], porting them with mutual benefit. These are (Sect. 2): the use of *colour information*, feature *normalisation*, and, most importantly, the use of *substantial data augmentation*. We also determine **scenario-specific best-practices**, improving the ones in [12], [15] and others, including dimensionality reduction for deep features. Finally, we achieve the **new state-of-the-art performance** on PASCAL VOC classification using techniques significantly simpler than competing approaches [8], [16]. As in [12], the source code to reproduce all experiments in this paper will be made publicly available.

2 Scenarios

This section introduces the three types of image representation $\phi(I)$ considered in this paper, describing them within the context of three different scenarios. Having outlined details specific to each, general methodologies which apply to all three scenarios are reviewed, such as data augmentation and feature normalisation, together with the linear classifier (trained with a standard hinge loss) that is common to all three scenarios. We also specify here the benchmark datasets used in the evaluation.

2.1 Scenario 1: Shallow representation (IFV)

Our reference shallow image representation is the IFV [3]. Our choice is motivated by the fact that IFV usually outperforms related encoding methods such as BoVW, LLC [12], and VLAD [17]. Given an image I, the IFV $\phi_{\mathrm{FV}}(I)$ is obtained by extracting a dense collection of patches and corresponding local descriptors $\mathbf{x}_i \in \mathrm{R}^D$ (*e.g.* SIFT) from the image. Each descriptor $\mathbf{x}_i$ is then soft-quantized using a Gaussian Mixture Model with K components. First and second order differences between each descriptor $\mathbf{x}_i$ and its Gaussian cluster mean μ_k are accumulated in corresponding blocks $\mathbf{u}_k$, $\mathbf{v}_k$ in the vector $\phi_{\mathrm{FV}}(I) \in \mathrm{R}^{2KD}$, appropriately weighed by the Gaussian soft-assignments and covariance, leading to a $2KD$-dimensional image representation $\phi_{\mathrm{FV}}(I) = [\mathbf{u}_1^\top, \mathbf{v}_1^\top, \ldots \mathbf{u}_K^\top, \mathbf{v}_K^\top]^\top$. The *improved* version of the Fisher vector involves post-processing ϕ_{FV} by computing the signed square-root of its scalar components and normalising the result to a unit ℓ^2 norm. The details of this construction can be found in [3]; here we follow the notation of [12].

2.2 Scenario 2: Deep representation (CNN) with pre-training

Our deep representations are inspired by the success of the CNN of Krizhevsky *et al.* [13]. As shown in [7], [16], the vector of activities $\phi_{\mathrm{CNN}}(I)$ of the penultimate layer of a deep CNN, learnt on a large dataset such as ImageNet [5], can be used as a powerful image descriptor applicable to other datasets. Numerous CNN architectures that improve the previous state of the art obtained using shallow representations have been proposed, but choosing the best one remains an open question. Many are inspired by [13]: DeCAF [7], [11], Caffe [18], Oquab *et al.* [8]. Others use larger networks with a smaller stride of the first convolutional layer: Zeiler and Fergus [16] and OverFeat [10], [9]. Other differences include the CNN pre-training protocols. Here we adopt a single learning framework and experiment with architectures of different complexity exploring their performance-speed trade-off.

2.3 Scenario 3: Deep representation (CNN) with pre-training and fine-tuning

In Scenario 2 features are trained on one (large) dataset and applied to another (usually smaller). However, it was demonstrated [11] that fine-tuning a pre-trained CNN on the target data can significantly improve the performance. We consider this scenario separately from that of Scenario 2, as the image features become dataset-specific after the fine-tuning.

2.4 Commonalities

We now turn to what is in common across the scenarios.

2.4.1 Data augmentation

Data augmentation is a method applicable to shallow and deep representations, but that has been so far mostly applied to the latter [13], [16]. By augmentation we mean perturbing an image I by transformations that leave the underlying class unchanged (*e.g.* cropping and flipping) in order to generate additional examples of the class. Augmentation can be applied at training time, at test time, or both. The augmented samples can either be taken as-is for use during training or can be combined to form a single feature, e.g. using sum-pooling, max-pooling, or stacking.

2.4.2 Linear predictors

All the representations $\phi(I)$ in the three scenarios are used to construct *linear predictors* $\langle \mathbf{w}, \phi(I)\rangle$ for each class to be recognized. These predictors are learnt using Support Vector Machines (SVM) by fitting $\mathbf{w}$ to the available training data by minimizing an objective function balancing a quadratic regularizer and the hinge-loss. The parameter C in the SVM, trading-off regularizer and loss, is determined using an held-off validation subset of the data. Here we use the same learning framework with all representations. It is common experience that linear classifiers are particularly sensitive to the *normalisation of the data* and that, in particular, SVMs tend to benefit from ℓ^2 normalisation [3] (an interpretation is that after normalisation the inner product corresponds to the cosine similarly).

2.5 Benchmark data

As reference benchmark we use the PASCAL VOC [6] data as already done in [12]. The **VOC-2007** edition contains about 10,000 images split into train, validation, and test sets, and labelled with twenty object classes. A one-vs-rest SVM classifier for each class is learnt and evaluated independently and the performance is measured as mean Average Precision (mAP) across all classes. The **VOC-2012** edition contains roughly twice as many images and does not include test labels; instead, evaluation uses the official PASCAL Evaluation Server. To train deep representations we use the **ILSVRC-2012** challenge dataset. This contains 1,000 object categories from ImageNet [5] with roughly 1.2M training images, 50,000 validation images, and 100,000 test images. Performance is evaluated using the top-5 classification error. Finally, we also evaluate over the **Caltech-101** image classification benchmark [19] using the same protocol as in [12]. Namely, we considered three random splits into training and testing data, each of which comprises 30 training and up to 30 testing images per class, and performance is measured using mean class accurracy.

3 Details

This section gives the implementation details of the methods introduced in Sect. 2.

3.1 Improved Fisher Vector details

Our IFV representation uses a slightly improved setting compared to the best result of [12].

Computation starts by upscaling the image I by a factor of 2 [20], followed by SIFT features extraction with a stride of 3 pixels at 7 different scales with $\sqrt{2}$ scale increments. These features are square-rooted as suggested by [21], and decorrelated and reduced in dimension from $128D$ to $80D$ using PCA. A GMM with $K = 256$ components is learnt from features sampled from the training images. Hence the Fisher Vector $\phi_{\mathrm{FV}}(I)$ has dimension $2KD = 40,960$. Before use in classification, the vector is signed-square-rooted and l^2-normalised (square rooting correspond to the Hellinger's kernel map [22]). As in [12], square-rooting is applied twice, once to the raw encodings, and once again after sum pooling and normalisation. In order to capture the weak geometrical information, the IFV representation is used in a *spatial pyramid* [23]. As in [12], the image is divided into 1×1, 3×1, and 2×2 spatial subdivisions and corresponding IFV are computed and stacked with an overall dimension of $8 \times 2KD = 327,680$ elements.

In addition to this standard formulation, we experiment with a few modifications. The first one is the use of *intra-normalisation* of the descriptor blocks, an idea recently proposed for the VLAD descriptor [24]. In this case, the ℓ^2 normalisation is applied to the individual sub-blocks $(\mathbf{u}_k, \mathbf{v}_k)$ of the vector $\phi_{\mathrm{FV}}(I)$, which helps to alleviate the local feature burstiness [25]. In the case of the improved intra-normalised features, it was found that applying the square-rooting only once to the final encoding produced the best results.

The second modification is the use of *spatially-extended local descriptors* [20] instead of a spatial pyramid. Here descriptors $\mathbf{x}_i$ are appended with their image location (x_i, y_i) before quantization with the GMM. Formally, $\mathbf{x}_i$ is extended, after PCA projection, with its normalised spatial coordinates:

$[\mathbf{x}_i^\top, x_i/W - 0.5, y_i/H - 0.5]^\top$, where $W \times H$ are the dimensions of the image. Since the GMM quantizes both appearance and location, this allows for the spatial information to be captured directly by the soft-quantization process. This method is significantly more memory-efficient than using a spatial pyramid. Specifically, the PCA-reduced SIFT features are spatially augmented by appending (x, y) yielding $D = 82$ dimensional descriptors pooled in a $2KD = 41,984$ dimensional IFV.

The third modification is the use of colour features in addition to SIFT descriptors. While colour information is used in CNNs [13] and by the original FV paper [3], it was not explored in the comparison [12]. We do so here by adopting the same Local Colour Statistics (LCS) features as used by [3]. LCS is computed by dividing an input patch into a 4×4 spatial grid (akin to SIFT), and computing the mean and variance of each of the *Lab* colour channels for each cell of the grid. The LCS dimensionality is thus $4 \times 4 \times 2 \times 3 = 96$. This is then encoded in a similar manner to SIFT.

3.2 Convolutional neural networks details

The CNN-based features are based on three CNN architectures representative of the state of the art. To ensure a fair comparison between them, these networks are trained using the same training protocol and the same implementation, which we developed based on the open-source Caffe framework [18]. ℓ^2-normalising the CNN features $\phi_{\text{CNN}}(I)$ before use in the SVM was found to be important for performance. The architectures and the adaptation procedure are described next.

3.2.1 CNN architectures

The three CNN architectures (Table 1) explore different accuracy/speed trade-off points. The trade-off is mainly controlled by choosing the filter sizes and the downsampling factors in the convolutional and pooling layers, as detailed next. The comparison with the state-of-the-art CNNs is presented in Sect. 4.

Our **Fast (CNN-F)** architecture is similar to the one used by Krizhevsky *et al.* [13]. It comprises 8 learnable layers, 5 of which are convolutional, and the last 3 are fully-connected. The input image size is 224×224. Fast processing is ensured by the 4 pixel stride in the first convolutional layer. The main differences between our architecture and that of [13] are the reduced number of convolutional layers and the dense connectivity between convolutional layers ([13] used sparse connections to enable training on two GPUs).

Our **Medium (CNN-M)** architecture is similar to the one used by Zeiler and Fergus [16]. It is characterised by the decreased stride and smaller receptive field of the first convolutional layer, which was shown to be beneficial on the ILSVRC dataset. At the same time, conv2 uses larger stride (2 instead of 1) to keep the computation time reasonable. The main difference between our net and that of [16] is that we use less filters in the conv4 layer (512 instead of 1024).

Our **Slow (CNN-S)** architecture is related to the "accurate" network from the OverFeat package [10]. It also uses 7×7 filters with stride 2 in conv1. Unlike CNN-M and [16], the stride in conv2 is smaller (1 pixel), but the max-pooling window in conv1 and conv5 is larger (3×3) to compensate for the increased spatial resolution. Compared to [10], we use 5 convolutional layers as in the previous architectures ([10] used 6), and less filters in conv5 (512 instead of 1024); we also incorporate an LRN layer after conv1 ([10] did not use contrast normalisation).

3.2.2 CNN training

In general, our CNN training procedure follows that of [13], learning on ILSVRC-2012 using gradient descent with momentum. The hyper-parameters are the same as used by [13]: momentum 0.9; weight decay $5 \cdot 10^{-4}$; initial learning rate 10^{-2}, which is decreased by a factor of 10, when the validation error stop decreasing. The layers are initialised from a Gaussian distribution with a zero mean and variance equal to 10^{-2}. We also employ similar data augmentation in the form of random crops, horizontal flips, and RGB colour jittering. Test time crop sampling was discussed in Sect. 3.3 of the submission; at training time, 224×224 crops are sampled randomly, rather than deterministically. Thus, the only notable difference to [13] is that the crops are taken from the whole training image $P \times 256, P \geq 256$, rather than its 256×256 centre. This results in a better coverage of images with a large aspect ratio. When evaluating the networks on ILSVRC-2012 (see Sect. 4), the 10-fold augmentation of Sect. 3.3 is employed.

Training was performed on a single NVIDIA GTX Titan GPU using a modified version of the publicly-available Caffe framework [18]. The training time

Arch.	conv1	conv2	conv3	conv4	conv5	full1	full2	full3
CNN-F	64x11x11 st. 4, pad 0 LRN, x2 pool	256x5x5 st. 1, pad 2 LRN, x2 pool	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 -	256x3x3 st. 1, pad 1 x2 pool	4096 drop- out	4096 drop- out	1000 soft- max
CNN-M	96x7x7 st. 2, pad 0 LRN, x2 pool	256x5x5 st. 2, pad 1 LRN, x2 pool	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 x2 pool	4096 drop- out	4096 drop- out	1000 soft- max
CNN-S	96x7x7 st. 2, pad 0 LRN, x3 pool	256x5x5 st. 1, pad 1 x2 pool	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 -	512x3x3 st. 1, pad 1 x3 pool	4096 drop- out	4096 drop- out	1000 soft- max

TABLE 1: **CNN architectures.** Each architecture contains 5 convolutional layers (conv 1–5) and three fully-connected layers (full 1–3). The details of each of the convolutional layers are given in three sub-rows: the first specifies the number of convolution filters and their receptive field size as "num x size x size"; the second indicates the convolution stride ("st.") and spatial padding ("pad"); the third indicates if Local Response Normalisation (LRN) [13] is applied, and the max-pooling downsampling factor. For full 1–3, we specify their dimensionality, which is the same for all three architectures. Full1 and full2 are regularised using dropout [13], while the last layer acts as a multi-way soft-max classifier. The activation function for all weight layers (except for full3) is the REctification Linear Unit (RELU) [13].

varied from 5 days for CNN-F to 3 weeks for CNN-S (which required more iterations to converge).

3.2.3 *CNN fine-tuning on the target dataset*

Given a pre-trained CNN, it can be fine-tuned on a target dataset by using it as a starting point for training. In our experiments, we fine-tuned CNN-S using VOC-2007, VOC-2012 or Caltech-101 as the target data. Fine-tuning was carried out using the same framework (and the same data augmentation), as we used for CNN training on ILSVRC. The last fully-connected layer (conv8) has the output dimensionality equal to the number of classes (20 in the case of VOC datasets, 102 in the case of Caltech-101), so it could not be initialised with the corresponding layer of the ILSVRC net; instead, we initialised it from a Gaussian distribution (as used for CNN training above). Now we turn to dataset-specific fine-tuning details.

VOC-2007 and VOC-2012. Considering that PASCAL VOC is a multi-label dataset (*i.e.* a single image might have multiple labels), we replaced the softmax regression loss with a more appropriate loss function, for which we considered two options: one-vs-rest classification hinge loss (the same loss as used in the SVM experiments) and ranking hinge loss. Both losses define constraints on the scores of positive (I_{pos}) and negative (I_{neg}) images for each class: $w_c\phi(I_{pos}) > 1 - \xi, w_c\phi(I_{neg}) < -1 + \xi$ for the classification loss, $w_c\phi(I_{pos}) > w_c\phi(I_{neg}) + 1 - \xi$ for the ranking loss (w_c is the c-th row of the last fully-connected layer, which can be seen as a linear classifier on deep features $\phi(I)$; ξ is a slack variable). We implemented both losses as the layers of our CNN training framework. The resulting fine-tuned networks are denoted as "CNN S TUNE-CLS" (for the classification loss) and "CNN S TUNE-RNK" (for the ranking loss).

In the case of both VOC datasets, the training and validation subsets were combined to form a single training set. Considering the small size of the training data (2 orders of magnitude smaller than ILSVRC-2012), it is important to control over-fitting, which we achieved by using a smaller initial learning rate for the fine-tuned hidden layers of the net. Namely, the learning rate schedule for the last layer / hidden layers was as follows: $10^{-2}/10^{-4} \rightarrow 10^{-3}/10^{-4} \rightarrow 10^{-4}/10^{-4} \rightarrow 10^{-5}/10^{-5}$. The values of other hyper-parameters were kept the same as in ILSVRC-2012 training.

Caltech-101 dataset contains a single class label per image, so fine-tuning was performed using the softmax regression loss. Other settings (including the learning rate schedule) were the same as used for the VOC fine-tuning experiments.

3.2.4 *Low-dimensional CNN feature training*

Our baseline networks (Table 1) have the same dimensionality of the last hidden layer (full7): 4096. This design choice is in accordance with the state-of-the-art architectures [13], [16], [10], and leads to a 4096-D dimensional image representation, which is rather compact, compared to IFV. At the same time, for large-scale image recognition applications, even lower dimensionality can be required. To investigate the dependency of the performance on the feature dimensionality, we trained three modifications of the CNN-M network, which have a lower dimensionality of the full7 layer: 2048, 1024, and 128. The networks were learnt on ILSVRC-2012. To speed-up training, it was initialised by setting all layers except for full7 and full8 to those of the CNN-M net and using a lower initial learning rate (10^{-3}) for their training. Therefore, only full7 and full8 layers had to be learnt "from scratch", since their dimensionality is different from the CNN-M layers; their initial learning rate was set to 10^{-2}.

The performance of deep low-dimensional features is discussed in Sect. 4.

3.3 Data augmentation details

We explore three data augmentation strategies. The first strategy is to use **no augmentation**. In contrast to IFV, however, CNNs require images to be transformed to a fixed size (224×224) even when no augmentation is used. Hence the image is downsized so that the smallest dimension is equal to 224 pixels and a 224×224 crop is extracted from the centre.[1] The second strategy is to use **flip augmentation**, mirroring images about the y-axis producing two samples from each image. The third strategy, termed **C+F augmentation**, combines cropping and flipping. For CNN-based representations, the image is downsized so that the smallest dimension is equal to 256 pixels. Then 224×224 crops are extracted from the four corners and the centre of the image. Note that the crops are sampled from the whole image, rather than its 256×256 centre, as done by [13]. These crops are then flipped about the y-axis, producing 10 perturbed samples per input image. In the case of the IFV encoding, the same crops are extracted, but at the original image resolution.

4 Analysis

This section describes the experimental results, comparing different features and data augmentation schemes. The results are given in Tab. 2 for VOC-2007 and analyzed next, starting from generally applicable methods such as augmentation and then discussing the specifics of each scenario. We then move onto the other datasets and the state of the art in section Sect. 4.7.

4.1 Data augmentation

We experiment with no data augmentation (denoted *Image Aug=–* in Tab. 2), flip augmentation (*Image Aug=F*), and C+F augmentation (*Image Aug=C*). Augmented images are used as stand-alone samples (f), or by fusing the corresponding descriptors using sum (s) or max (m) pooling or stacking (t). So for example *Image Aug=(C) f s* in row [f] of Tab. 2 means that C+F augmentation is used to generate additional samples in training (f), and is combined with sum-pooling in testing (s).

Augmentation consistently improves performance by $\sim 3\%$ for both IFV (*e.g.* [d] *vs.* [f]) and CNN (*e.g.* [o] *vs.* [p]). Using additional samples for training and sum-pooling for testing works best ([p]) followed by sum-pooling [r], max pooling [q], and stacking [s]. In terms of the choice of transformations, flipping improves only marginally ([o] *vs.* [u]), but using the more expensive C+F sampling improves, as seen, by about $2 \sim 3\%$ ([o] *vs.* [p]). We experimented with sampling more transformations, taking a higher density of crops from the centre of the image, but observed no benefit.

4.2 Colour

Colour information can be added and subtracted in CNN and IFV. In IFV replacing SIFT with the colour descriptors of [3] (denoted *COL* in *Method*) yields significantly worse performance ([j] *vs.* [h]). However, when SIFT and colour descriptors are combined by stacking the corresponding IFVs (*COL+*) there is a small but significant improvement of around $\sim 1\%$ in the non-augmented case (*e.g.* [h] *vs.* [k]) but little impact in the augmented case (*e.g.* [i] *vs.* [l]). For CNNs, retraining the network after converting all the input images to grayscale (denoted *GS* in *Methods*) has a more significant impact, resulting in a performance drop of $\sim 3\%$ ([w] *vs.* [p], [v] *vs.* [o]).

4.3 Scenario 1: Shallow representation (IFV)

The baseline IFV encoding using a spatial pyramid [a] performs slightly better than the results [I] taken from Chatfield *et al.* [12], primarily due to a larger number of spatial scales being used during SIFT feature extraction, and the resultant SIFT features

1. Extracting a 224×224 centre crop from a 256×256 image [13] resulted in worse performance.

Method	SPool	Image Aug.			Dim	mAP	✈	🚲	🐦	⛵	🍾	🚌
(I) FK BL	spm	–			327K	**61.69**	79.0	67.4	51.9	70.9	30.8	72.2
(II) DECAF	–	(C)	t	t	327K	**73.41**	87.4	79.3	84.1	78.4	42.3	73.7
(a) FK	spm	–			327K	**63.66**	83.4	68.8	59.6	74.1	35.7	71.2
(b) FK IN	spm	–			327K	**64.18**	82.1	69.7	59.7	75.2	35.7	71.3
(c) FK	(x,y)	–			42K	**63.51**	83.2	69.4	60.6	73.9	36.3	68.6
(d) FK IN	(x,y)	–			42K	**64.36**	83.1	70.4	62.4	75.2	37.1	69.1
(e) FK IN	(x,y)	(F)	f	-	42K	**64.35**	83.1	70.5	62.3	75.4	37.1	69.1
(f) FK IN	(x,y)	(C)	f	s	42K	**67.17**	85.5	71.6	64.6	77.2	39.0	70.8
(g) FK IN	(x,y)	(C)	s	s	42K	**66.68**	84.9	70.1	64.7	76.3	39.2	69.8
(h) FK IN 512	(x,y)	–			84K	**65.36**	84.1	70.4	65.0	76.7	37.2	71.3
(i) FK IN 512	(x,y)	(C)	f	s	84K	**68.02**	85.9	71.8	67.1	77.1	38.8	72.3
(j) FK IN COL 512	–	–			82K	**52.18**	69.5	52.1	47.5	64.0	24.6	49.8
(k) FK IN 512 COL+	(x,y)	–			166K	**66.37**	82.9	70.1	67.0	77.0	36.1	70.0
(l) FK IN 512 COL+	(x,y)	(C)	f	s	166K	**67.93**	85.1	70.5	67.5	77.4	35.7	71.2
(m) CNN F	–	(C)	f	s	4K	**77.15**	88.8	83.2	86.5	84.5	48.2	76.1
(n) CNN S	–	(C)	f	s	4K	**79.58**	90.7	85.9	88.3	86.2	50.9	80.1
(o) CNN M	–	–			4K	**77.42**	91.1	83.6	87.5	84.1	48.5	79.2
(p) CNN M	–	(C)	f	s	4K	**79.78**	92.0	85.1	89.1	86.6	52.2	79.1
(q) CNN M	–	(C)	f	m	4K	**79.40**	91.4	84.3	89.2	85.1	51.3	77.8
(r) CNN M	–	(C)	s	s	4K	**79.35**	91.7	84.7	89.1	86.3	51.7	78.0
(s) CNN M	–	(C)	t	t	41K	**79.01**	91.5	85.3	89.3	85.1	51.5	77.7
(t) CNN M	–	(C)	f	-	4K	**78.05**	91.6	84.1	87.7	85.3	49.2	79.3
(u) CNN M	–	(F)	f	-	4K	**77.42**	91.0	83.1	87.3	84.3	48.4	79.3
(v) CNN M GS	–	–			4K	**73.46**	87.6	79.9	82.4	80.2	44.4	75.3
(w) CNN M GS	–	(C)	f	s	4K	**76.82**	89.8	83.3	84.6	82.9	49.2	77.9
(x) CNN M 2048	–	(C)	f	s	2K	**80.13**	92.1	86.6	89.6	86.2	52.2	79.6
(y) CNN M 1024	–	(C)	f	s	1K	**79.79**	91.7	86.4	89.2	85.6	53.1	79.4
(z) CNN M 128	–	(C)	f	s	128	**78.16**	90.6	83.7	88.5	86.7	51.9	81.1
(α) FK+CNN F	(x,y)	(C)	f	s	88K	**77.95**	89.6	83.1	87.1	84.5	48.0	79.4
(β) FK+CNN M 2048	(x,y)	(C)	f	s	86K	**80.14**	90.9	85.9	88.8	85.5	52.3	81.4
(γ) CNN S TUNE-RNK	–	(C)	f	s	4K	**82.42**	95.3	90.4	92.5	89.6	54.4	81.9

TABLE 2: VOC 2007 results *(continued overleaf)*. See Sect. 4 for details.

being square-rooted. *Intra-normalisation*, denoted as *IN* in the *Method* column of the table, improves the performance by $\sim 1\%$ (*e.g.* [c] *vs.* [d]). More interestingly, switching from spatial pooling (denoted *spm* in the *SPool* column) to feature spatial augmentation (*SPool=(x,y)*) has either little effect on the performance or results in a marginal increase ([a] *vs.* [c], [b] *vs.* [d]), whilst resulting in a representation which is over $10\times$ smaller. We also experimented with augmenting with scale in addition to position as in [20] but observed no improvement. Finally, we investigate pushing the parameters of the representation setting $K = 512$ (rows [h]-[l]). Increasing the number of GMM centres in the model from $K = 256$ to 512 results in a further performance increase (e.g. [h] *vs.* [d]), but at the expense of higher-dimensional codes (125K dimensional).

4.4 Scenario 2: Deep representation (CNN) with pre-training

CNN-based methods consistently outperform the shallow encodings, even after the improvements discussed above, by a large $\sim 10\%$ mAP margin ([i] *vs.* [p]). Our small architecture CNN-F, which is similar to DeCAF [7], performs significantly better than the latter ([II] *vs.* [s]), validating our implementation. Both medium CNN-M [m] and slow CNN-S [p] outperform the fast CNN-F [m] by a significant $2 \sim 3\%$ margin. Since the accuracy of CNN-S and CNN-M is nearly the same, we focus

	🚗	🐈	🪑	🐄	🍽	🐕	🐎	🏍	🧍	🪴	🐑	🛋	🚂	📺
(I)	79.9	61.4	56.0	49.6	58.4	44.8	78.8	70.8	85.0	31.7	51.0	56.4	80.2	57.5
(II)	83.7	83.7	54.3	61.9	70.2	79.5	85.3	77.2	90.9	51.1	73.8	57.0	86.4	68.0
(a)	80.7	64.4	53.8	53.8	60.2	47.8	79.9	68.9	86.1	37.3	51.1	55.8	83.7	56.9
(b)	80.6	64.8	53.9	54.9	60.7	50.5	80.4	69.5	86.2	38.3	54.4	56.3	82.7	56.7
(c)	81.1	64.2	51.1	53.4	61.9	50.0	80.0	67.5	85.3	35.7	51.9	53.8	83.5	58.9
(d)	80.5	66.9	50.9	53.9	62.1	51.5	80.5	68.5	85.9	37.2	55.2	54.3	83.3	59.2
(e)	80.5	66.8	51.0	54.1	62.2	51.5	80.4	68.2	86.0	37.3	55.1	54.2	83.3	59.2
(f)	82.4	71.6	52.8	62.4	63.4	57.1	81.6	70.9	86.9	41.2	61.2	56.9	85.2	61.5
(g)	81.9	71.0	52.8	61.6	62.2	56.8	81.8	70.0	86.5	41.5	61.0	56.5	84.3	60.9
(h)	81.1	67.9	52.6	55.4	61.4	51.2	80.5	69.1	86.4	41.2	56.0	56.2	83.7	59.9
(i)	82.5	73.2	54.7	62.7	64.5	56.6	82.2	71.3	87.5	43.0	62.0	59.3	85.7	62.4
(j)	66.1	46.6	42.5	35.8	41.1	45.5	75.4	58.3	83.9	39.8	47.3	35.6	69.2	49.0
(k)	80.0	65.9	52.8	56.1	61.0	56.9	81.4	69.6	88.4	49.0	59.2	56.4	84.7	62.8
(l)	81.6	70.8	52.9	59.6	63.1	59.9	82.1	70.5	88.9	50.6	63.7	57.5	86.1	64.1
(m)	86.2	85.4	57.9	69.2	72.7	82.0	87.2	80.4	91.8	59.1	77.0	66.1	89.0	71.8
(n)	87.6	88.6	60.5	74.6	74.0	86.9	88.7	83.8	92.3	58.9	79.8	69.2	90.6	74.0
(o)	85.9	87.3	58.8	72.1	74.1	83.3	85.9	81.5	91.1	56.7	78.1	59.6	88.9	71.0
(p)	87.5	88.7	60.3	78.0	74.2	85.5	88.1	84.2	92.1	60.4	80.6	67.3	91.3	73.6
(q)	87.4	88.6	60.8	77.9	73.2	85.8	87.7	83.4	92.5	58.9	80.9	66.7	91.1	74.0
(r)	87.4	88.6	59.6	77.7	74.3	85.2	88.1	84.0	92.0	59.0	79.5	66.0	90.9	73.2
(s)	86.8	88.1	59.8	75.3	74.8	85.1	87.7	82.8	92.0	58.3	79.3	66.0	90.4	73.3
(t)	86.2	87.3	58.8	74.2	75.2	84.0	86.6	82.6	91.3	56.4	78.5	62.7	88.9	71.1
(u)	86.1	87.2	58.8	70.7	75.0	83.2	86.0	80.9	91.2	56.8	78.8	59.9	89.1	71.2
(v)	84.7	84.1	58.1	65.1	69.7	77.5	83.1	77.8	89.7	44.2	69.5	60.9	87.3	67.5
(w)	87.1	86.8	59.9	72.7	72.9	81.6	86.1	81.6	90.9	48.3	74.3	66.2	89.4	71.0
(x)	87.5	88.8	61.0	76.9	74.3	85.3	88.7	83.6	92.3	60.6	82.2	68.9	91.5	74.7
(y)	87.4	88.7	59.3	76.7	72.8	85.5	88.3	83.3	91.7	60.4	82.1	67.9	92.4	74.0
(z)	86.5	87.0	58.5	67.3	72.6	84.3	86.0	82.8	89.2	57.1	80.7	64.8	90.6	73.5
(α)	86.8	85.6	59.9	72.0	73.4	81.4	88.6	80.5	92.1	60.6	77.3	66.4	89.3	73.3
(β)	87.7	88.4	61.2	76.9	76.6	84.9	89.1	82.9	92.4	61.9	80.9	68.7	91.5	75.1
(γ)	91.5	91.9	64.1	76.3	74.9	89.7	92.2	86.9	95.2	60.7	82.9	68.0	95.5	74.4

TABLE 2: VOC 2007 results *(continued from previous page)*

on the latter as it is simpler and marginally ($\sim 25\%$) faster. Remarkably, these good networks work very well even with no augmentation [o]. Another advantage of CNNs compared to IFV is the small dimensionality of the output features. We explored retraining the CNNs such that the final layer was of a lower dimensionality, and reducing from 4096 to 2048 actually resulted in a marginal performance boost ([x] *vs.* [p]). What is surprising is that we can reduce the output dimensionality further to 1024D [y] and even 128D [z] with only a drop of $\sim 2\%$ for codes that are $32\times$ smaller ($\sim 650\times$ smaller than our best performing IFV [i]). Note, ℓ^2-normalising the features accounted for up to $\sim 5\%$ of their performance over VOC 2007; normalisation should occur before input to the SVM and after pooling the augmented descriptors (where applicable).

4.5 Scenario 3: Deep representation (CNN) with pre-training and fine-tuning

We fine-tuned our CNN-S architecture on VOC-2007 using the ranking hinge loss, and achieved a significant improvement: 2.7% ([γ] *vs.* [n]). This demonstrates that in spite of the small amount of VOC training data (5,011 images), fine-tuning is able to adjust the learnt deep representation to better suit the dataset in question.

	ILSVRC-2012 (top-5 error)	**VOC-2007** (mAP)	**VOC-2012** (mAP)	**Caltech-101** (accuracy)
(a) FK IN 512	-	68.0	–	–
(b) CNN F	16.7	77.2	–	–
(c) CNN M	13.7	79.8	82.3	–
(d) CNN M 2048	13.5	80.1	82.3	–
(e) CNN S	**13.1**	79.6	82.7	**88.54 ± 0.33**
(f) CNN S TUNE-CLS	**13.1**	-	83.0	88.35 ± 0.56
(g) CNN S TUNE-RNK	**13.1**	**82.4**	**83.2**	–
(h) Zeiler & Fergus [16]	16.1	-	79.0	86.5 ± 0.5
(i) Razavian *et al.* [9], [10]	14.7	77.2	–	–
(j) Oquab *et al.* [8]	18	77.7	78.7 / 82.8	–

TABLE 3: **Comparison with the state of the art** on ILSVRC2012, VOC2007, VOC2012, and Caltech-101.

4.6 Combinations

For the CNN-M 2048 representation [x], stacking deep and shallow representations to form a higher-dimensional descriptor makes little difference ([x] *vs.* [β]). For the weaker CNN-F it results in a small boost of $\sim 0.8\%$ ([m] *vs.* [α]).

4.7 Comparison with the state of the art

In Table 3 we report our results on ILSVRC-2012, VOC-2007, VOC-2012, and Caltech-101 datasets, and compare them to the state of the art. First, we note that the ILSVRC error rates of our CNN-F, CNN-M, and CNN-S networks are better than those reported by [13], [16], and [10] for the related configurations. This validates our implementation, and the difference is likely to be due to the sampling of image crops from the uncropped image plane (instead of the centre). When using our CNN features on other datasets, the relative performance generally follows the same pattern as on ILSVRC, where the nets are trained. In particular, the CNN-F architecture exhibits the worst performance among our nets, while more powerful CNN-M and CNN-S perform considerably better.

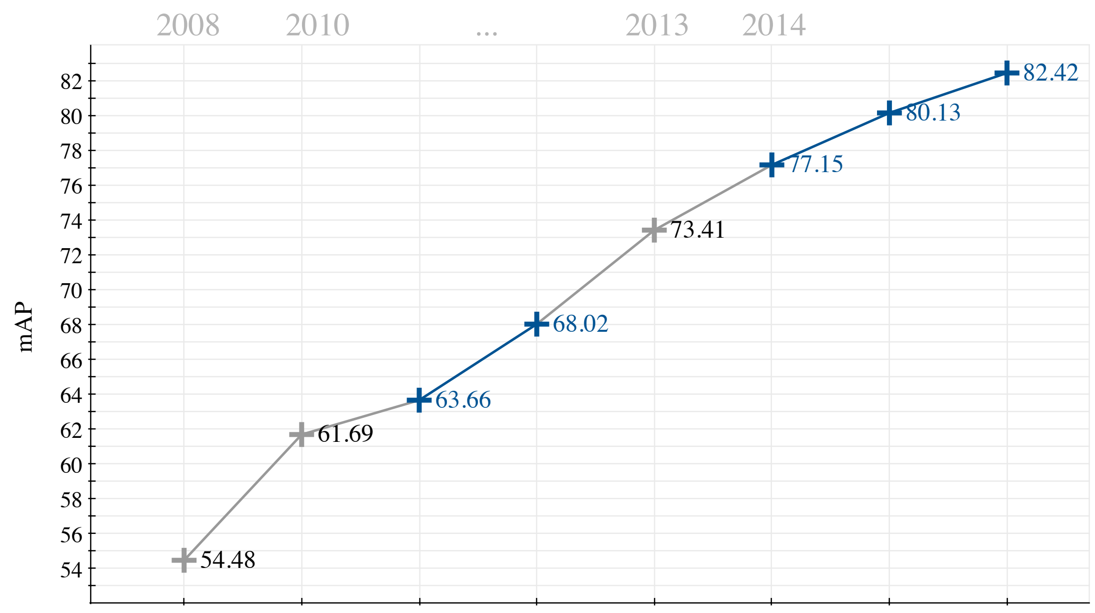
Further fine-tuning of CNN-S on the VOC datasets turns out to be beneficial; on VOC-2012, using the ranking loss is marginally better than the classification loss ([g] *vs.* [f]), which can be explained by the ranking-based VOC evaluation criterion. Interestingly, unlike the VOC datasets, fine-tuning on Caltech-101 did not improve the performance. This can be explained by a very limited amount of training data: just 30 samples per class. While it is sufficient to train an SVM classifier, it might not be enough to adjust the weights of a large deep architecture.

Our fine-tuned CNN-S net [g] sets the new state of the art on VOC-2007, VOC-2012 and Caltech-101 datasets, outperforming the recently published CNN-based methods of [16], [9], [8]. Interestingly, we achieve superior results to [8] ([j]) with a conceptually simpler method, but with a more powerful deep architecture. In their case, they exploited ground-truth object bounding boxes, additional VOC-like ImageNet categories, and a large number of multi-scale crops, but used a less accurate CNN. Without extra categories (*i.e.* by using only ILSVRC data as we do), they report 78.7% on VOC-2012. The CNN architecture of [16] ([h]) is similar to ours, but their use of softmax regression loss is not well-suited to multi-label VOC data.

4.8 Performance Evolution on VOC-2007

A comparative plot of the evolution in the performance of the methods evaluated in this paper, along with a selection from the earlier review by Chatfield *et al.* [12] is presented in Fig. 1. The classification accuracy of image representations on PASCAL VOC was 54.48% mAP for the BoVW model in 2008, 61.7% for the IFV in 2010 [12], and 73.41% for DeCAF [7] and similar [8], [9] CNN-based methods introduced in late 2013.

In addition to setting the new state of the art with our CNN-based features, with the modifications outlined in the paper, primarily through the use of data augmentation similar to that employed in the CNN-based methods, we are able to improve the performance of shallow IFV descriptors to 68.02%.



Method	BOW	FK-BL	FK	FK-IN	DeCAF	CNN-F	CNN-M 2K	CNN-S
Dim.	32K	327K	327K	84K	4K	4K	2K	4K (TN)
Aug.	–	–	–	f s	t t	f s	f s	f s
Ref.		[I]	[a]	[i]	[II]	[m]	[y]	[γ]

Fig. 1: **Evolution of Performance on PASCAL VOC-2007 over the recent years.** Please refer to Table 2 for details and references.

4.9 Timings and dimensionality

One of our best-performing CNN representations CNN-M-2048 [x] is $\sim 42\times$ more compact than the best performing IFV [i] (84K vs. 2K) and is also $\sim 50\times$ faster to compute ($\sim 120s$ vs. $\sim 2.4s$ per image with augmentation enabled). Non-augmented CNN-M features [o] take around $0.3s$ per image.

5 Conclusion

In this paper we presented a rigorous empirical evaluation of CNN-based methods for image classification, along with a comparison with more traditional shallow feature encoding methods. We have demonstrated that the performance of shallow representations can be significantly improved by adopting data augmentation, typically used in deep learning. In spite of this improvement, deep architectures still outperform the shallow methods by a large margin. We have shown that the performance of deep representations on the ILSVRC dataset is a good indicator of their performance on other datasets, and that fine-tuning can further improve on already very strong results achieved using the combination of deep representations and a linear SVM. We plan to release the source code and CNN models for the experiments presented in the paper in the hope that it would provide good baselines for image representation research. In future work, we will further explore the space of the CNN configurations.

Acknowledgements

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

References

[1] G. Csurka, C. Bray, C. Dance, and L. Fan, “Visual categorization with bags of keypoints,” in *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22.

[2] J. Sivic and A. Zisserman, “Video Google: A text retrieval approach to object matching in videos,” in *Proc. ICCV*, vol. 2, 2003, pp. 1470–1477.

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

[4] 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*, vol. 1, no. 4, pp. 541–551, 1989.

[5] 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.

[6] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL Visual Object Classes (VOC) challenge,” *IJCV*, vol. 88, no. 2, pp. 303–338, 2010.

[7] 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*, vol. abs/1310.1531, 2013.

[8] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Learning and Transferring Mid-Level Image Representations using Convolutional Neural Networks,” in *Proc. CVPR*, 2014.

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

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

[11] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proc. CVPR*, 2014.

[12] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman, “The devil is in the details: an evaluation of recent feature encoding methods,” in *Proc. BMVC.*, 2011.

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

[14] M. Paulin, J. Revaud, Z. Harchaoui, F. Perronnin, and C. Schmid, “Transformation Pursuit for Image Classification,” in *Proc. CVPR*, 2014.

[15] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid, “Towards good practice in large-scale learning for image classification,” in *Proc. CVPR*, 2012, pp. 3482–3489.

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

[17] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, “Aggregating local images descriptors into compact codes,” *IEEE PAMI*, 2012.

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

[19] 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.

[20] J. Sánchez, F. Perronnin, and T. Emídio de Campos, “Modeling the spatial layout of images beyond spatial pyramids,” *Pattern Recognition Letters*, vol. 33, no. 16, pp. 2216–2223, 2012.

[21] R. Arandjelović and A. Zisserman, “Three things everyone should know to improve object retrieval,” in *Proc. CVPR*, 2012.

[22] A. Vedaldi and A. Zisserman, “Efficient additive kernels via explicit feature maps,” *IEEE PAMI*, 2011.

[23] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories,” in *Proc. CVPR*, 2006.

[24] R. Arandjelović and A. Zisserman, “All about VLAD,” in *Proc. CVPR*, 2013.

[25] H. Jégou, M. Douze, and C. Schmid, “On the burstiness of visual elements,” in *Proc. CVPR*, Jun 2009.