Deep Fishing: Gradient Features from Deep Nets

Albert Gordo[1]
albert.gordo@xrce.xerox.com
Adrien Gaidon[1]
adrien.gaidon@xrce.xerox.com
Florent Perronnin[2]
perronnin@fb.com

[1] Computer Vision Group
Xerox Research Centre Europe
[2] Facebook AI Research*

Abstract

Convolutional Networks (ConvNets) have recently improved image recognition performance thanks to end-to-end learning of deep feed-forward models from raw pixels. Deep learning is a marked departure from the previous state of the art, the Fisher Vector (FV), which relied on gradient-based encoding of local hand-crafted features. In this paper, we discuss a novel connection between these two approaches. First, we show that one can derive gradient representations from ConvNets in a similar fashion to the FV. Second, we show that this gradient representation actually corresponds to a structured matrix that allows for efficient similarity computation. We experimentally study the benefits of transferring this representation over the outputs of ConvNet layers, and find consistent improvements on the Pascal VOC 2007 and 2012 datasets.

1 Introduction

Image classification involves describing images with pre-determined labels. One of the first breakthroughs towards solving this problem was the bag-of-visual-words (BOV) [7, 35]. While the BOV simply involves counting the number of occurrences of quantized local features, approaches that encode higher order statistics such as the the Fisher Vector (FV) [26, 28] led to state-of-the-art image classification results [5, 31]. Especially, such higher-order encodings were used by the leading teams in the 2010 and 2011 editions of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [8, 30]. FV-based approaches were however outperformed in 2012 by the work of Krizhevsky *et al*. [20] based on Convolutional Networks (ConvNets) [21] trained in a supervised fashion on large amounts of labeled data. These models are feed-forward architectures involving multiple computational layers that alternate linear operations, *e.g*. convolutions, and non-linear operations, *e.g*. rectified linear units (ReLU). The end-to-end training of the large number of parameters inside ConvNets from pixels to the specific end-task is a key to their success. Since then, ConvNets, including improved architectures [32, 33, 44], have consistently outperformed all other alternatives in subsequent editions of ILSVRC. Also, ConvNets have remarkable *transferability properties* when used as "universal" feature extractors [43]: if one feeds an image to a ConvNet, the output of intermediate layers might be used as a representation of this image and typically fed

* Work done while FP was at the Computer Vision Group of the Xerox Research Centre Europe.

© 2015. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

Pages 111.1-111.12
DOI: https://dx.doi.org/10.5244/C.29.111

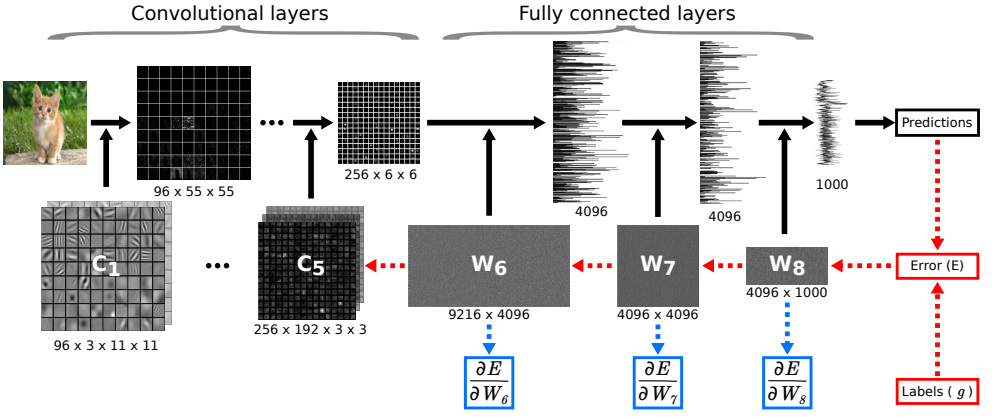

Figure 1: AlexNet architecture [20]. C_k are the parameters (4D tensors) of the convolutional layers. W_k are the parameters (matrices) of the fully connected layers. Black (resp. red) arrows represent the information flow during the forward (resp. backward) pass. Inspired by the Fisher Kernel [17], we study the use of gradient-related information (the blue matrices) as transferable representations.

to linear classifiers. To the best of our knowledge, this heuristic is not based on a strong theoretical ground, but has been experimentally shown to work well in practice [6, 9, 22, 29, 44].

Although ConvNets and FV approaches differ significantly, several works tried to combine their benefits [14, 27, 34, 38]. Our work also attempts to *get the best of both FV and ConvNet worlds*. Our **primary contribution** is a novel approach to extract a *transferable* representation of an image given a pre-trained ConvNet. We draw inspiration from the FV, which is based on the theoretically well-founded Fisher Kernel (FK) proposed by Jaakkola and Haussler [17]. The FK involves deriving a kernel from an underlying generative model of the data by taking the gradient of the log-likelihood with respect to the model parameters. In a similar manner, given an unlabeled image, we propose to compute the *gradient of a cross-entropy criterion measured between the predicted class probabilities and an equal probability output*. This gradient with respect to the parameters of the fully connected layers yields very high-dimensional representations (*cf*. Figure 1). Our **second contribution** consists in leveraging the special structure of this gradient representation to design an efficient kernel. We show that our representation actually corresponds to a rank-1 matrix, for which the trace kernel can be efficiently computed. Furthermore, this kernel decomposes in our case into the product of two simpler kernels: the standard one on forward-pass features, and a second one on quantities efficiently computed by back-propagation.

The remainder of this article is organized as follows. In section 2, we review related works. In section 3, we provide more background on the FK and ConvNets. In section 4, we introduce our novel hybrid ConvNet-gradient representation as well as our associated efficient kernel. Finally, we provide experimental results on the PASCAL VOC 2007 and 2012 benchmarks in section 5, showing that our representation consistently transfers better than the standard forward pass features.

2 Related Work

Hybrid techniques. Several works have proposed to combine the benefits of deep learning with "shallow" bag-of-patches representations based on higher-order statistics such as the

FV [26, 28] or the VLAD [18]. Simonyan *et al*. [34] propose to stack multiple FV layers, each defined as a set of five operations: i) FV encoding, ii) supervised dimensionality reduction, iii) spatial stacking, iv) ℓ_2 normalization and v) PCA dimensionality reduction. They show that, when combined with the original FV, such networks lead to significant performance improvements on ImageNet. Peng *et al*. [25] proposed a similar idea, but for action recognition. Alternatively, Sydorov *et al*. [38] improve on the FV framework by jointly learning the SVM classifier and the GMM visual vocabulary. Conceptually, this is similar to back-propagation as used to learn neural network parameters: the gradients corresponding to the SVM layer are back-propagated to compute the gradients with respect to the GMM parameters. Peng *et al*. [24] proposed a similar idea for the VLAD [18] descriptor. Finally, Gong *et al*. [14] address the lack of geometric invariance in ConvNets with a hybrid approach. They extract mid-level ConvNet features from large patches, embed them using the VLAD encoding, and aggregate them at multiple scales. This leads to competitive results on a number of classification tasks. While our goal – getting the best of the FV and deep frameworks – is shared with these previous works, we differ significantly, as we are the first to propose to derive gradient features from deep nets.

Deriving representations from pre-trained classifiers. Classemes [39, 40] is a common image representation from a set of classifiers obtained by simply stacking classifier scores. Dimensionality reduction is generally applied on classeme features [10], but learning separately the classification and dimensionality reduction is suboptimal [15]. Several works [2, 15, 42] learn an optimal embedding of images in a low-dimensional space via classifiers with an intermediate hidden layer. The first layer can be understood as a supervised dimensionality reduction step, while the second one can be interpreted as a set of classifiers in the intermediate space. A new image is represented as the output of this intermediate layer, discarding the classifiers. A natural extension is to learn deeper architectures, *i.e*. architectures with more than one hidden layer, and to use the output of these intermediate layers as features for the new tasks. Krizhevsky *et al*. [20] proposed to learn end-to-end a deep classifier based on the ConvNet architecture of LeCun *et al*. [21]. They showed qualitatively that the output of the penultimate layer could be used for image retrieval. This finding was quantitatively validated for a number of tasks, including image classification [6, 9, 22, 29, 44], image retrieval [1, 29], object detection [13], and action recognition [22]. The choice of the layer(s) whose output should be used for representation purposes depends on the problem at hand. As observed by Yosinski *et al*. [43], this choice should be driven by the distance between the base task (the one used to learn the classifier) and target task. In this paper, we show that this heuristic of using the output of an intermediate-level fully connected layer as image representation can be related to the application of the Fisher Kernel idea to ConvNets.

3 Background on the Fisher Kernel and ConvNets

3.1 Fisher Kernel

The Fisher Kernel (FK) is a generic principle introduced to combine the benefits of generative and discriminative models to pattern recognition. Let X be a sample, and let u_θ be a probability density function that models the generative process of X, where θ denotes the vector of parameters of u_θ. In statistics, the *score function* is given by the gradient of the log-likelihood of the data on the model:

$$\varphi_\theta^{FK}(X) = \nabla_\theta \log u_\theta(X). \tag{1}$$

This gradient describes the contribution of the individual parameters to the generative process. Jaakkola and Haussler [17] proposed to measure the similarity between two samples X and Y using the *Fisher Kernel* (FK) which is defined as:

$$K_{FK}(X,Y) = \varphi_\theta^{FK}(X)' F_\theta^{-1} \varphi_\theta^{FK}(Y) \tag{2}$$

where F_θ is the Fisher Information Matrix, usually approximated by the identity matrix [17]. One of the benefits of the FK framework is that it comes with guarantees. The FK is indeed asymptotically at least as good as the MAP decision rule, when assuming that the classification label is included in the generative model as a latent variable (theorem 1 in [17]). Some extensions make the dependence of the kernel on the classification labels explicit. This includes the likelihood kernel [12], which involves one generative model per class, and which consists in computing one FK for each generative model (and consequently for each class). This also includes the likelihood ratio kernel [36], which is tailored to the two-class problem, and which involves computing the gradient of the log-likelihood of the ratio between the two class likelihoods. Given two classes denoted c_1 and c_2 with class-conditional probability density functions $p(.|c_1)$ and $p(.|c_2)$ and with collective parameters θ, this yields:

$$\varphi_\theta^{LR}(X) = \nabla_\theta \log \frac{p(X|c_1)}{p(X|c_2)}. \tag{3}$$

The likelihood ratio kernel is supported by strong experimental evidence [36] and theory [37]. In section 4, we extend it to derive a gradient representation from a ConvNet model.

3.2 Convolutional Networks

Convolutional Networks (ConvNets) [21] are the de facto state-of-the-art models for image recognition since the work of Krizhevsky *et al.* [20]. This class of deep learning models relies on a feed forward architecture typically composed of a stack of convolutional layers followed by a stack of fully connected layers (see Figure 1 for the standard AlexNet [20] architecture). A convolutional layer is parametrized by a 4D tensor representing a stack of 3D filters. During the forward pass, these filters are run in a sliding window fashion across the output of the previous layer (or the image itself for the first layer) in order to produce a 3D tensor: the stack of per-filter activation maps. These activation maps then pass through a non-linearity (typically a Rectified Linear Unit, or ReLU [20]) and an optional pooling stage before being fed to the next layer. Both the standard AlexNet [20] and recent improved architectures like VGGNet [33] use a stack of fully connected layers to transform the output activation map of the convolutional layers into class-membership probabilities. A fully connected layer consists in a simple matrix vector multiplication followed by a non-linearity, typically ReLU for intermediate layers and a SoftMax for the last one.

Let x_k be the output of layer k, which is also the input of layer $k+1$ (for AlexNet, x_5 is the flattened activation map of the fifth convolutional layer). Layer k is parametrized by the 4D tensor C_k if it is a convolutional layer, and by the matrix W_k for a fully connected layer. A fully connected layer performs the operation $x_k = \sigma(W_k^T x_{k-1})$, where σ is the non-linearity. We note $y_k = W_k^T x_{k-1}$ the output of layer k before the non-linearity, and $\theta = \{C_1, \cdots, C_M\} \cup \{W_{M+1}, \cdots, W_L\}$ the parameters of all L layers of the network. Training such deep models consists in end-to-end learning of this vast number of parameters via the minimization of an error (or loss) function on a large training set of N image and ground-truth label pairs (I^i, g^i). The typical loss function used for classification is the cross-entropy:

$$E(I^i, g^i; \theta) = -\sum_{j=1}^{P} g^i_j \log(x^i_{L,j}) \tag{4}$$

where P is the number of labels (categories), $g^i \in \{0,1\}^P$ is the label vector of image I^i, and $x^i_{L,j}$ is the predicted probability of class j for image I^i resulting from the forward pass. The optimal network parameters θ^* are the ones minimizing the loss over the training set:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} E(I^i, g_i; \theta) \tag{5}$$

This optimization problem is typically solved using Stochastic Gradient Descent (SGD) [3], a stochastic approximation of batch gradient descent consisting in doing approximate gradient steps equal on average to the true gradient $\nabla_\theta E$. Each approximate gradient step is typically performed with a small batch of labeled examples in order to efficiently leverage the caching and vectorization mechanisms of modern hardware.

A particularity of deep networks is that the gradients with respect to all parameters θ can be computed efficiently in a stage-wise fashion via a sequential application of the chain rule ("back-propagation" [21]). In particular, when using ConvNets as feature extractors, the first phase consists in pre-training the network (*i.e.* obtaining θ^*) via SGD with back-propagation on a large labeled dataset like ImageNet [30]. Then, ConvNet features can be used for different tasks using forward passes on the pre-trained network. In the following, we describe how we can also use back-propagation *at test time* to transfer richer Fisher Vector-like representations based on the gradient of the loss with respect to the ConvNet parameters.

4 Gradient Features from Deep Nets

We now motivate the use of gradient features from deep nets by relating the likelihood ratio kernel in equation (3) to the ConvNet objective function in equation (4). We then explicit the gradient equations and relate our gradient features to the standard heuristic features derived from the outputs of intermediate layers. Finally, we explain how to efficiently compute the similarity between these high-dimensional representations.

4.1 Relating the likelihood ratio kernel and deep nets

The FK [17] and its extensions [12, 36] were proposed as generic frameworks to derive representations and kernels from generative models. As the standard ConvNet classification architecture does not define a generative model, such frameworks cannot be applied as-is. However, we can draw inspiration from the likelihood ratio kernel for that purpose. We start from equation (3) and note that it can be rewritten as the gradient of the log-likelihood of the ratio between posterior probabilities (assuming equal class priors), *i.e.*:

$$\varphi^{LR}_\theta(X) = \nabla_\theta \log \frac{p(X|c_1)}{p(X|c_2)} = \nabla_\theta \log \frac{p(c_1|X)}{p(c_2|X)}. \tag{6}$$

In the two-class problem of [36], we have $p(c_2|X) = 1 - p(c_1|X)$ and equation (6) gives:

$$\varphi^{LR}_\theta(X) = \frac{\varphi^{pos}_{\theta;1}(X)}{1 - p(c_1|X)} \tag{7}$$

where $\varphi^{pos}_{\theta;j} = \nabla_\theta \log p(c_j|X)$ is the gradient of the log-posterior for class c_j. We underline that the previous formula is general in the sense that it can be applied beyond generative models. To extend this representation beyond the two-class case, one may compute an embedding $\varphi^{pos}_{\theta;j}$ for each class j using the gradient of the corresponding log-posterior probability.

We can now observe the relation between the ConvNet objective E in equation (4) for an image I and label vector g with these gradient of log-posterior embeddings:

$$\nabla_\theta E(I,g;\theta) = -\sum_{j=1}^{P} g_j \nabla_\theta \log p(c_j|I) = -\sum_{j=1}^{P} g_j \varphi^{pos}_{\theta;j}(I) \tag{8}$$

Consequently, the gradient of the ConvNet objective can be interpreted as a sum of gradient embeddings $\varphi^{pos}_{\theta;j}(I)$, weighted by the labels g_j.

To use this gradient as an image representation, as is the case of the FK, there are two main challenges to be addressed. First, we do not have access to the value of the label g, which we need to compute the representation of a test image I according to equation (8). The simplest solution consists in using a constant uniform label vector $g = \bar{g} = [1/P,\ldots,1/P]$. Although $\bar{g}$ is non-informative, we experimentally validate the interest of this simple strategy. The second issue concerning the use of $\nabla_\theta E(I,\bar{g};\theta)$ as an image representation lies in the associated computational cost. Although scalable in the number of classes, this representation is very high-dimensional. The number of parameters θ in current deep architectures is indeed too large to be able to use the full gradient $\nabla_\theta E$ in practice. Therefore, we propose to use *only the partial derivatives with respect to the parameters of some fixed layers*, in the same spirit as what is currently done with layer-activation features. These partial derivatives can be computed and compared efficiently using the chain rule and a rank 1 decomposition, as shown in the following sections. Note also that this approach can be further combined with other existing techniques, including ones specialized for deep nets (*e.g.* model compression [4]) or for FV (*e.g.* product quantization [19]).

4.2 Gradient derivation

One remarkable property of ConvNets and other feed-forward architectures is that they are differentiable through all their layers. In the case of ConvNets, it is easy to show that the gradients of the loss with respect to the weights of the fully-connected layers are:

$$\frac{\partial E}{\partial W_k} = x_{k-1}\left[\frac{\partial E}{\partial y_k}\right]^T. \tag{9}$$

To compute the partial derivatives of the loss with respect to the output parameters needed in Equation (9), one can apply the chain rule. In the case of fully-connected layers and ReLU non-linearities, this leads to the following recursive definition and base case,

$$\frac{\partial E}{\partial y_k} = \left[W_{k+1}\frac{\partial E}{\partial y_{k+1}}\right] \circ \mathbb{I}_{[y_k>0]}, \qquad \frac{\partial E}{\partial y_L} = \bar{g} - \sigma(y_L), \tag{10}$$

where $\mathbb{I}_{[y>0]}$ is an indicator vector, set to one at the positions where $y > 0$ and to zero otherwise, $\circ$ is the Hadamard or element-wise product, $\bar{g}$ is a supplied vector of labels with which to compute the loss, and σ is the SoftMax function. From the previous section, we use $\bar{g} = [1/P,...,1/P]$, *i.e.* we assume that all classes have equal probabilities. It is worth noticing how $\frac{\partial E}{\partial y_L}$ is simply a shifted version of the output probabilities, while the derivatives

w.r.t. y_k with $k < L$ are linear transformations of these shifted probabilities, as the Hadamard product can be rewritten as a matrix multiplication.

4.3 Computing similarities between gradients

Using the gradients in equation (9) as features is problematic in practice due to their high-dimensional nature with current deep architectures. In the case of AlexNet [20], $\frac{\partial E}{\partial W_8}$ is around 4 million floating point values, while $\frac{\partial E}{\partial W_7}$ and $\frac{\partial E}{\partial W_6}$ are each around 16 and 36 million floats. Thus, explicitly computing the dot-product between the gradients is impractical. Instead, we propose to take advantage of the unique structure of our gradients (rank-1 matrices, *cf*. Eq. (9)) by using the trace kernel, defined for two matrices A and B as:

$$K_{tr}(A,B) = Tr(A^T B) \tag{11}$$

For rank-1 matrices, the trace can be decomposed as the product of two kernels. If we let $A = au^T$, $A \in \mathbb{R}^{d\times D}$, and $B = bv^T$, $B \in \mathbb{R}^{d\times D}$, with $a, b \in \mathbb{R}^d$ and $u, v \in \mathbb{R}^D$, then:

$$K_{tr}(A,B) = Tr(au^T(bv^T)^T) = Tr(au^T vb^T) = Tr(b^T au^T v) = (b^T a)\cdot(u^T v).$$

Therefore, for two images A and B, we can compute the similarity between gradients in a low-dimensional space without explicitly computing the gradients *w.r.t.* the weights W_k:

$$K_{tr}\left(\frac{\partial E}{\partial W_k}(A,\bar{g};\theta), \frac{\partial E}{\partial W_k}(B,\bar{g};\theta)\right) = (x^A_{k-1})^T x^B_{k-1} \cdot \left[\frac{\partial E}{\partial y_k}(A,\bar{g};\theta)\right]^T \frac{\partial E}{\partial y_k}(B,\bar{g};\theta) \tag{12}$$

The left part of this equation indicates that the forward activations of the two inputs should be similar. This is the standard measure of similarity which is used between images when described by the outputs of the intermediate layers of ConvNets. However, this similarity is multiplicatively weighted by the similarity between the back-propagated quantities. This indicates that, to obtain a high value with the proposed kernel, both the target forward activations *and* the back-propagated quantities of the images need to be similar.

Normalization. The ℓ_2-normalization of the activation features consistently leads to superior results [6]. In our experiments we ℓ_2-normalize our forward and backward features independently. This is consistent with normalizing the gradient matrix using a Frobenius norm, since $||au^T||_F = ||a||_2||u||_2$.

5 Experimental results

5.1 Datasets and evaluation protocols

We evaluate our approach to transfer features from pretrained models on two standard classification benchmarks, Pascal VOC 2007 and Pascal VOC 2012 [11]. These datasets contain 9,963 and 22,531 annotated images, respectively. Each image is annotated with one or more labels corresponding to 20 object categories. The datasets include partitions for training, validating, and testing, and the accuracy is measured in terms of per class mean average precision (mAP). The test annotations of VOC 2012 are not public, but an evaluation server with a limited number of submissions per week is available. Therefore, we use the validation set for the first part of our analysis on the VOC 2012 dataset, and evaluate on the test set only for the final experiments. We conduct all VOC 2007 experiments on the full dataset.

Table 1: Left: Results on Pascal VOC 2007 and VOC 2012 with AlexNet (A) and VGG16 (V). Results on VOC 2012 are on the validation set. Right: Comparison with other ConvNet results (mAP in %).

Features	VOC2007 (A)	VOC2007 (V)	VOC2012 (A)	VOC2012 (V)
x_5 (Pool5)	71.0	86.7	66.1	81.4
x_6 (FC6)	77.1	89.3	72.6	84.4
x_7 (FC7)	79.4	89.4	74.9	84.6
y_8 (FC8)	79.1	88.3	74.3	84.1
x_8 (Prob)	76.2	86.0	71.9	81.3
$x_5; x_6$	76.4	89.2	71.6	84.0
$\frac{\partial E}{\partial W_6} = x_5 \left[\frac{\partial E}{\partial y_6}\right]^T$	80.2	89.3	75.1	84.6
$x_6; x_7$	79.1	89.5	74.3	84.6
$\frac{\partial E}{\partial W_7} = x_6 \left[\frac{\partial E}{\partial y_7}\right]^T$	**80.9**	**90.0**	**76.3**	**85.2**
$x_7; y_8$	79.7	89.2	75.3	84.6
$\frac{\partial E}{\partial W_8} = x_7 \left[\frac{\partial E}{\partial y_8}\right]^T$	79.7	88.2	75.0	83.4

		VOC'07	VOC'12
Proposed - AlexNet	[20]	80.9	76.5
Proposed - VGG16	[33]	**90.0**	85.3
DeCAF	[9] from [6]	73.4	-
Razavian *et al.*	[29]	77.2	-
Oquab *et al.*	[23]	77.7	78.7
Zeiler *et al.*	[42]	-	79.0
Chatfield *et al.*	[6]	82.4	83.2
He *et al.*	[16]	80.1	-
Wei *et al.*	[41]	81.5	81.7
Simonyan *et al.*	[33]	89.7	**89.3**

5.2 Implementation details

We tested our approach on two different deep ConvNets: AlexNet [20] and VGG16 [33]. VGG16 is a much deeper architecture than AlexNet, with many more convolutional layers, leading to superior performance, but also to a much slower training and feature extraction. We used the pre-trained networks that are publicly available[1]. Both networks were pre-trained on the ILSVRC2012 subset of ImageNet, which is disjoint from the Pascal VOC datasets, and therefore suitable for our evaluation of feature transfer.

To extract descriptors from the Pascal images, we first resize the images so that the shortest size has 227 pixels (224 on the VGG16 case), and then take the central square crop, without distorting the aspect ratio. We found this cropping technique to work well in practice. For simplicity, we do no data augmentation. The feature extraction is performed on a customized version of the *caffe* library[2], modified to expose the back-propagation features. This allows us to extract forward and backward features of the training and testing images. At testing time we use a tempered version of SoftMax, $\sigma(y,\tau) = exp(y/\tau)/\sum_i exp(y_i/\tau)$, with $\tau = 2$, to produce softer probability distributions for backpropagation. As discussed in section 4.1, we use non-informative uniform labels for the backward pass to extract the gradient features. All forward and backward features are then ℓ_2-normalized.

To perform classification, we use the SVM implementation of *scikit-learn* [23][3]. The cost parameter C of the solver was set to the default value of 1, which worked well in practice.

5.3 Results and discussion

Table 1 summarizes the results, and compares our approach with the state of the art and different baselines. We extract and compare several features for each dataset and network architecture: (i) individual forward activation features, from Pool5 up to the probability layer; (ii) concatenation of forward activation features, *e.g.* Pool5+FC6, FC6+FC7, FC7+FC8; (iii) our

[1] https://github.com/BVLC/caffe/wiki/Model-Zoo
[2] http://caffe.berkeleyvision.org
[3] http://scikit-learn.org/

proposed gradient features: $\frac{\partial E}{\partial W_6}$, $\frac{\partial E}{\partial W_7}$, and $\frac{\partial E}{\partial W_8}$. The similarity between ℓ_2-normalized forward activation features is measured with the dot-product, while the similarity between gradient representations is measured using the trace kernel. We highlight the following points.

Forward activations. In all cases, FC7 is the best performing individual layer on both VOC2007 and VOC2012, independently of the network. This is consistent with previous findings. Also consistent is the fact that the probability layer performs badly in this case. More surprisingly, concatenating forward layers does not seem to bring any noticeable accuracy improvements in any setup.

Gradient representations. We compare the gradient representations with the concatenation of forward activations, since they are very related and share part of the features. On the deeper layers (6 and 7) the gradient representations outperform the individual features as well as the concatenation both for AlexNet and VGG16 on both datasets. For AlexNet, the improvements are quite significant: $+3.8\%$ and $+3.5\%$ absolute improvement for the gradients with respect to W_6 on VOC2007 and VOC2012, and $+1.8\%$ and $+2\%$ for W_7. The improvements for VGG16 are more modest but still noticeable: $+0.1\%$ and $+0.6\%$ for the gradients with respect to W_6 and $+0.5\%$ and $+0.6\%$ for the gradients with respect to W_7. Larger relative improvements on less discriminative networks such as AlexNet seem to suggest that the more complex gradient representation can, to some extent, compensate for the lack of discriminative power of the network, but that one obtains diminishing returns as the power of the network increases. Once one reaches the top of the network ($FC8$), the gradient representations perform worse and these improvements diminish or disappear completely. This is expected, as the derivative with respect to W_8 depends heavily on the output of the probability layer, which is known to saturate. However, for the derivatives with respect to W_6 and W_7, more information is involved, leading to superior results.

Comparison with other works. Our best results are compared with the state-of-the-art on PASCAL VOC2007 and VOC2012 in Table 1. We can see that we obtain competitive performance on both datasets. We note however that our results with VGG16 are somewhat inferior to those reported in [33] with a similar model. We believe this might be explained by the more costly feature extraction strategy employed by Simonyan and Zisserman which involves aggregating image descriptors at multiple scales.

Per-class results. We report per-class results for Pascal VOC2007 on Table 2 and for VOC2012 on Table 3. We compare the best forward features (individual FC7) with the best gradient representation ($\frac{\partial E}{\partial W_7}$). The results on VOC2007 are on the test set. For VOC2012, we report results both on validation and on test. We observe how, on both networks and datasets, the results are consistently better even when the improvements are not large. For

Table 2: Results on Pascal VOC2007 with AlexNet and VGG16. Comparison between the standard forward activation features and the proposed gradient features.

Features	**mean**	*aeroplane*	*bicycle*	*bird*	*boat*	*bottle*	*bus*	*car*	*cat*	*chair*	*cow*	*diningtable*	*dog*	*horse*	*motorbike*	*person*	*pottedplant*	*sheep*	*sofa*	*train*	*tvmonitor*
AlexNet																					
x_7 (FC7)	79.4	95.4	88.6	92.6	87.3	42.1	80.1	90.5	89.6	59.9	68.2	74.1	85.3	89.8	85.6	95.3	58.1	**78.9**	57.9	94.7	**74.4**
$\frac{\partial E}{\partial W_7}$	**80.9**	**96.6**	**89.2**	**93.8**	**89.5**	**44.9**	**81.0**	**91.9**	**89.9**	**61.2**	**70.4**	**78.5**	**86.2**	**91.4**	**87.4**	**95.7**	**60.5**	78.8	**62.5**	**95.2**	73.5
VGG16																					
x_7 (FC7)	89.3	99.2	95.9	**99.1**	96.9	**63.8**	92.8	95.1	98.1	70.4	87.8	84.3	97.0	97.2	93.5	97.3	68.6	92.2	73.3	98.7	**85.5**
$\frac{\partial E}{\partial W_7}$	**90.0**	**99.6**	**97.2**	98.8	**97.0**	63.3	**93.8**	**95.6**	**98.4**	**71.1**	**89.4**	**85.3**	**97.7**	**97.7**	**95.6**	**97.5**	**70.3**	**92.7**	**76.2**	**98.8**	84.2

Table 3: Results on Pascal VOC2012 with AlexNet and VGG16. Comparison between the standard forward activation features and the proposed gradient features.

Features	**mean**	*aeroplane*	*bicycle*	*bird*	*boat*	*bottle*	*bus*	*car*	*cat*	*chair*	*cow*	*diningtable*	*dog*	*horse*	*motorbike*	*person*	*pottedplant*	*sheep*	*sofa*	*train*	*tvmonitor*
AlexNet (evaluated on the validation set)																					
x_7 (FC7)	74.9	92.9	75.4	88.7	81.7	48.0	89.0	70.3	88.0	62.3	63.6	57.8	83.5	78.0	82.9	92.9	49.1	74.8	50.5	90.2	78.7
$\frac{\partial E}{\partial W_7}$	**76.3**	**94.3**	**77.4**	**89.5**	**82.2**	**50.8**	**90.2**	**72.4**	**89.3**	**64.8**	**63.9**	**60.3**	**84.0**	**79.6**	**84.0**	**93.2**	**50.6**	**76.7**	**52.6**	**91.8**	**79.2**
AlexNet (evaluated on the test set)																					
x_7 (FC7)	75.0	93.8	75.0	86.4	82.2	48.2	82.5	73.8	87.6	63.8	63.5	69.3	85.7	80.3	84.1	92.3	47.4	72.2	51.8	88.1	72.5
$\frac{\partial E}{\partial W_7}$	**76.5**	**95.0**	**76.6**	**87.7**	**82.9**	**52.5**	**83.4**	**75.6**	**88.6**	**65.3**	**65.4**	**69.8**	**86.5**	**82.1**	**85.1**	**93.0**	**48.2**	**74.5**	**57.0**	**88.4**	**73.0**
VGG16 (evaluated on the validation set)																					
x_7 (FC7)	84.6	98.2	88.3	94.6	90.5	66.0	93.6	80.5	96.4	**73.9**	81.3	70.2	93.0	91.3	91.3	95.1	**56.3**	87.7	64.2	**95.8**	84.5
$\frac{\partial E}{\partial W_7}$	**85.2**	**98.6**	**89.4**	**94.7**	**91.5**	**67.2**	**94.0**	**80.9**	**96.8**	73.7	**83.7**	**71.9**	**93.4**	**91.6**	**91.5**	**95.4**	56.0	**88.3**	**65.2**	95.5	**85.2**
VGG16 (evaluated on the test set)																					
x_7 (FC7)	85.0	97.8	85.2	**92.3**	91.1	64.5	**89.7**	82.2	95.4	74.1	**84.7**	**81.1**	94.1	93.5	91.9	95.0	**57.9**	86.0	67.8	**95.2**	**81.5**
$\frac{\partial E}{\partial W_7}$	**85.3**	**98.0**	**86.0**	91.7	**91.3**	**65.7**	89.6	**82.4**	**95.5**	**74.5**	84.2	80.7	**94.3**	**93.7**	**92.2**	**95.4**	57.7	**87.2**	**69.2**	**95.2**	81.4

AlexNet, the gradient representation has the best performance on 18 out of the 20 classes on VOC2007, and on all classes for VOC2012. For VGG, the gradient representation is the best one on 17 out of the 20 classes both on VOC2007 and VOC2012 (validation). The differences between validation and test on VOC2012 are minimal.

6 Conclusions

In this paper we show a link between ConvNets as feature extractors and Fisher Vector encodings. We have introduced a gradient-based representation for features extracted with ConvNets inspired by the Fisher Kernel framework. This representation takes advantage of the high-quality features learned by ConvNets on an end-to-end supervised manner, and of the discriminative power of gradient-based representations. We also presented an approach to compute similarities between gradients in an efficient manner without computing explicitly the high-dimensional gradient representations. We show that this similarity can be seen as a weighed version of the forward feature similarities that takes into account not only the features themselves, but also information back-propagated from the ConvNet objective. We tested our approach on the Pascal VOC2007 and VOC2012 benchmarks using two different popular deep architectures, showing consistent improvements over using only the individual forward activation features or their combination as it is standard practice.

References

[1] A Babenko, A Slesarev, A Chigorin, and V Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014.

[2] A Bergamo, L Torresani, and A Fitzgibbon. PiCoDeS: Learning a compact code for novel-category recognition. In *NIPS*, 2011.

[3] L. Bottou. Online algorithms and stochastic approximations. In David Saad, editor, *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998. URL http://leon.bottou.org/papers/bottou-98x.

[4] C. Bucilua, R. Caruana, and A. Niculescu-Mizil. Model compression. In *SIGKDD*, 2006.

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

[6] K. Chatfield, K. Simonyan, A. vedaldi, and A. Zisserman. Return of the devil in the details: deving deep into convolutional nets. In *BMVC*, 2014.

[7] G. Csurka, C. Dance, L Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. *ECCV SLCV workshop*, 2004.

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

[9] 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. In *ICML*, 2014.

[10] M Douze, A Ramisa, and C Schmid. Combining attributes and fisher vectors for efficient image retrieval. In *CVPR*, 2011.

[11] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal visual object classes (VOC) challenge. *IJCV*, 2010.

[12] S. Fine, J. Navrátil, and R. Gopinath. A hybrid GMM/SVM approach to speaker identification. In *ICASSP*, 2001.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[14] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014.

[15] A Gordo, JA Rodríguez-Serrano, F Perronnin, and E Valveny. Leveraging category-level labels for instance-level image retrieval. In *CVPR*, 2012.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[17] T. Jaakkola and D. Haussler. Exploting generative models in discriminative classifiers. In *NIPS*, 1998.

[18] H. Jégou, M Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[19] H. Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE TPAMI*, 2011.

[20] A Krizhevsky, I Sutskever, and G Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012.

[21] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. *NIPS*, 1989.

[22] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.

[23] F. Pedregosa et al. Scikit-learn: Machine learning in Python. *JMLR*, 2011.

[24] X Peng, L Wang, Y Qiao, and Q Peng. Boosting VLAD with supervised dictionary learning and high-order statistics. In *ECCV*, 2014.

[25] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked Fisher vectors. In *ECCV*, 2014.

[26] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. *CVPR*, 2007.

[27] F. Perronnin and D. Larlus. Fisher vectors meet neural networks: A hybrid classification architecture. In *CVPR*, 2015.

[28] F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. *ECCV*, 2010.

[29] AS Razavian, H Azizpour, J Sullivan, and S Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *CVPR Deep Vision Workshop*, 2014.

[30] O Russakovsky et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

[31] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 2013.

[32] 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.

[33] K Simonyan and A Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015.

[34] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep fisher networks for large-scale image classification. In *NIPS*, 2013.

[35] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. 2003.

[36] N. Smith and M. Gales. Speech recognition using SVMs. In *NIPS*, 2001.

[37] N. Smith and M. Gales. Using SVMs to classify variable length speech patterns. Technical report, Cambridge University, 2002.

[38] V. Sydorov, M. Sakurada, and C. Lampert. Deep Fisher kernels – End to end learning of the Fisher kernel GMM parameters. In *CVPR*, 2014.

[39] L Torresani, M Szummer, and A Fitzgibbon. Efficient object category recognition using classemes. In *ECCV*, 2010.

[40] G Wang, D Hoiem, and D Forsyth. Learning image similarity from Flickr groups using stochastic intersection kernel machines. In *ICCV*, 2009.

[41] Y Wei, W Xia, J Huang, B Ni, J Dong, Y Zhao, and S Yan. CNN: single-label to multi-label. arXiv, 2014.

[42] J Weston, S Bengio, and N Usunier. Large scale image annotation: Learning to rank with joint word-image embeddings. *ECML*, 2010.

[43] J Yosinski, J Clune, Y Bengio, and H Lipson. How transferable are features in deep neural networks ? In *NIPS*, 2014.

[44] MD. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*. 2014.