Understanding Deep Image Representations by Inverting Them

Aravindh Mahendran
University of Oxford

Andrea Vedaldi
University of Oxford

Abstract

Image representations, from SIFT and Bag of Visual Words to Convolutional Neural Networks (CNNs), are a crucial component of almost any image understanding system. Nevertheless, our understanding of them remains limited. In this paper we conduct a direct analysis of the visual information contained in representations by asking the following question: given an encoding of an image, to which extent is it possible to reconstruct the image itself? To answer this question we contribute a general framework to invert representations. We show that this method can invert representations such as HOG and SIFT more accurately than recent alternatives while being applicable to CNNs too. We then use this technique to study the inverse of recent state-of-the-art CNN image representations for the first time. Among our findings, we show that several layers in CNNs retain photographically accurate information about the image, with different degrees of geometric and photometric invariance.

1. Introduction

Most image understanding and computer vision methods build on image representations such as textons [15], histogram of oriented gradients (SIFT [18] and HOG [4]), bag of visual words [3][25], sparse [35] and local coding [32], super vector coding [37], VLAD [9], Fisher Vectors [21], and, lately, deep neural networks, particularly of the convolutional variety [13, 23, 36]. However, despite the progress in the development of visual representations, their design is still driven empirically and a good understanding of their properties is lacking. While this is true of shallower handcrafted features, it is even more so for the latest generation of deep representations, where millions of parameters are learned from data.

In this paper we conduct a direct analysis of representations by characterising the image information that they retain (Fig. 1). We do so by modeling a representation as a function $\Phi(\mathbf{x})$ of the image $\mathbf{x}$ and then computing an approximated inverse ϕ^{-1}, *reconstructing* $\mathbf{x}$ *from the code* $\Phi(\mathbf{x})$. A common hypothesis is that representations collapse irrelevant differences in images (e.g. illumination or viewpoint), so that Φ should not be uniquely invertible. Hence, we pose this as a reconstruction problem and find a number of possible reconstructions rather than a single one. By doing so, we obtain insights into the invariances captured by the representation.

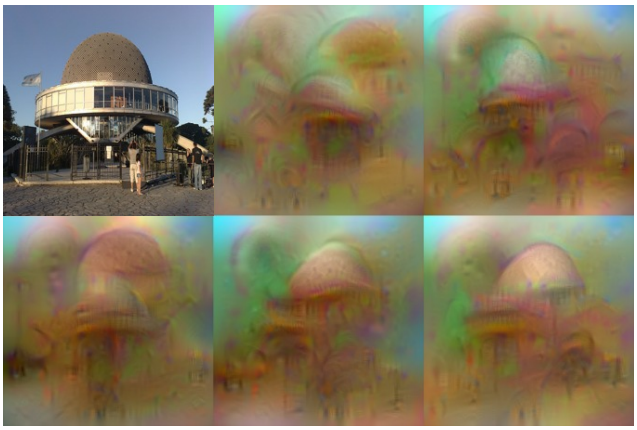

Figure 1. **What is encoded by a CNN?** The figure shows five possible reconstructions of the reference image obtained from the 1,000-dimensional code extracted at the penultimate layer of a reference CNN[13] (before the softmax is applied) trained on the ImageNet data. From the viewpoint of the model, all these images are practically equivalent. This image is best viewed in color/screen.

Our contributions are as follows. First, we propose a general method to invert representations, including SIFT, HOG, and CNNs (Sect. 2). Crucially, this method **uses only information from the image representation** and a generic natural image prior, starting from random noise as initial solution, and hence captures only the information contained in the representation itself. We discuss and evaluate different regularization penalties as natural image priors. Second, we show that, despite its simplicity and generality, this method recovers significantly better reconstructions from DSIFT and HOG compared to recent alternatives [31]. As we do so, we emphasise a number of subtle differences between these representations and their effect on invertibility. Third, we apply the inversion technique to the analysis of recent deep CNNs, exploring their invariance by sampling possible approximate reconstructions. We relate this to the depth of the representation, showing that the CNN gradually builds an increasing amount of invariance, layer after layer. Fourth, we study the locality of the information stored in

the representations by reconstructing images from selected groups of neurons, either spatially or by channel.

The rest of the paper is organised as follows. Sect. 2 introduces the inversion method, posing this as a regularised regression problem and proposing a number of image priors to aid the reconstruction. Sect. 3 introduces various representations: HOG and DSIFT as examples of shallow representations, and state-of-the-art CNNs as an example of deep representations. It also shows how HOG and DSIFT can be implemented as CNNs, simplifying the computation of their derivatives. Sect. 4 and 5 apply the inversion technique to the analysis of respectively shallow (HOG and DSIFT) and deep (CNNs) representations. Finally, Sect. 6 summarises our findings.

We use the matconvnet toolbox [30] for implementing convolutional neural networks.

Related work. There is a significant amount of work in understanding representations by means of visualisations. The works most related to ours are Weinzaepfel *et al*. [33] and Vondrick *et al*. [31] which invert sparse DSIFT and HOG features respectively. While our goal is similar to theirs, our method is substantially different from a technical viewpoint, being based on the direct solution of a regularised regression problem. The benefit is that our technique applies equally to shallow (SIFT, HOG) and deep (CNN) representations. Compared to existing inversion techniques for dense shallow representations [31], it is also shown to achieve superior results, both quantitatively and qualitatively.

An interesting conclusion of [31, 33] is that, while HOG and SIFT may not be exactly invertible, they capture a significant amount of information about the image. This is in apparent contradiction with the results of Tatu *et al*. [27] who show that it is possible to make any two images look nearly identical in SIFT space up to the injection of adversarial noise. A symmetric effect was demonstrated for CNNs by Szegedy *et al*. [26], where an imperceptible amount of adversarial noise suffices to change the predicted class of an image. The apparent inconsistency is easily resolved, however, as the methods of [26, 27] require the injection of high-pass structured noise which is very unlikely to occur in natural images.

Our work is also related to the DeConvNet method of Zeiler and Fergus [36], who backtrack the network computations to identify which image patches are responsible for certain neural activations. Simonyan *et al*. [24], however, demonstrated that DeConvNets can be interpreted as a sensitivity analysis of the network input/output relation. A consequence is that DeConvNets do not study the problem of representation inversion in the sense adopted here, which has significant methodological consequences; for example, DeConvNets require *auxiliary information* about the activations in several intermediate layers, while our inversion uses only the final image code. In other words, DeConvNets look at *how* certain network outputs are obtained, whereas we look for *what* information is preserved by the network output.

The problem of inverting representations, particularly CNN-based ones, is related to the problem of inverting neural networks, which received significant attention in the past. Algorithms similar to the back-propagation technique developed here were proposed by [14, 16, 19, 34], along with alternative optimisation strategies based on sampling. However, these methods did not use natural image priors as we do, nor were applied to the current generation of deep networks. Other works [10, 28] specialised on inverting networks in the context of dynamical systems and will not be discussed further here. Others [1] proposed to learn a second neural network to act as the inverse of the original one, but this is complicated by the fact that the inverse is usually not unique. Finally, auto-encoder architectures [8] train networks together with their inverses as a form of supervision; here we are interested instead in visualising feed-forward and discriminatively-trained CNNs now popular in computer vision.

2. Inverting representations

This section introduces our method to compute an approximate inverse of an image representation. This is formulated as the problem of finding an image whose representation best matches the one given [34]. Formally, given a representation function $\Phi : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^d$ and a representation $\Phi_0 = \Phi(\mathbf{x}_0)$ to be inverted, reconstruction finds the image $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$ that minimizes the objective:

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}\in\mathbb{R}^{H\times W\times C}} \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda\mathcal{R}(\mathbf{x}) \qquad (1)$$

where the loss ℓ compares the image representation $\Phi(\mathbf{x})$ to the target one Φ_0 and $\mathcal{R} : \mathbb{R}^{H\times W\times C} \to \mathbb{R}$ is a regulariser capturing a *natural image prior*.

Minimising (1) results in an image $\mathbf{x}^*$ that "resembles" $\mathbf{x}_0$ from the viewpoint of the representation. While there may be no unique solution to this problem, sampling the space of possible reconstructions can be used to characterise the space of images that the representation deems to be equivalent, revealing its invariances.

We next discusses the choice of loss and regularizer.

Loss function. There are many possible choices of the loss function ℓ. While we use the Euclidean distance:

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2, \qquad (2)$$

it is possible to change the nature of the loss entirely, for example to optimize selected neural responses. The latter was used in [5, 24] to generate images representative of given neurons.

Regularisers. Discriminatively-trained representations may discard a significant amount of low-level image statis-

tics as these are usually not interesting for high-level tasks. As this information is nonetheless useful for visualization, it can be partially recovered by restricting the inversion to the subset of natural images $\mathcal{X} \subset \mathbb{R}^{H\times W\times C}$. However, minimising over $\mathcal{X}$ requires addressing the challenge of modeling this set. As a proxy one can incorporate in the reconstruction an appropriate *image prior*. Here we experiment with two such priors. The first one is simply the α-norm $\mathcal{R}_\alpha(\mathbf{x}) = \|\mathbf{x}\|_\alpha^\alpha$, where $\mathbf{x}$ is the vectorised and mean-subtracted image. By choosing a relatively large exponent ($\alpha = 6$ is used in the experiments) the range of the image is encouraged to stay within a target interval instead of diverging.

A second richer regulariser is *total variation* (TV) $\mathcal{R}_{V^\beta}(\mathbf{x})$, encouraging images to consist of piece-wise constant patches. For continuous functions (or distributions) $f : \mathbb{R}^{H\times W} \supset \Omega \to \mathbb{R}$, the TV norm is given by:

$$\mathcal{R}_{V^\beta}(f) = \int_\Omega \left(\left(\frac{\partial f}{\partial u}(u,v)\right)^2 + \left(\frac{\partial f}{\partial v}(u,v)\right)^2 \right)^{\frac{\beta}{2}} du\, dv$$

where $\beta = 1$. Here images are discrete ($\mathbf{x} \in \mathbb{R}^{H\times W}$) and the TV norm is replaced by the finite-difference approximation:

$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\frac{\beta}{2}}.$$

It was observed empirically that the TV regularizer ($\beta = 1$) in the presence of subsampling, also caused by max pooling in CNNs, leads to "spikes" in the reconstruction. This is a known problem in TV-based image interpolation (see *e.g.* Fig. 3 in [2]) and is illustrated in Fig. 2.left when inverting a layer in a CNN. The "spikes" occur at the locations of the samples because: (1) the TV norm along any path between two samples depends only on the overall amount of intensity change (not on the sharpness of the changes) and (2) integrated on the 2D image, it is optimal to concentrate sharp changes around a boundary with a small perimeter. Hyper-Laplacian priors with $\beta < 1$ are often used as a better match of the gradient statistics of natural images [12], but they only exacerbate this issue. Instead, we trade-off the sharpness of the image with the removal of such artifacts by choosing $\beta > 1$ which, by penalising large gradients, distributes changes across regions rather than concentrating them at a point or curve. We refer to this as the V^β regularizer. As seen in Fig. 2 (right), the spikes are removed with $\beta = 2$ but the image is washed out as edges are penalized more than with $\beta = 1$.

When the target of the reconstruction is a colour image, both regularisers are summed for each colour channel.

Balancing the different terms. Balancing loss and regulariser(s) requires some attention. While an optimal tuning can be achieved by cross-validation, it is important to start

Figure 2. **Left:** Spikes in a inverse of norm1 features - detail shown. **Right:** Spikes removed by a V^β regularizer with $\beta = 2$.

from reasonable settings of the parameters. First, the loss is replaced by the normalized version $\|\Phi(\mathbf{x}) - \Phi_0\|_2^2 / \|\Phi_0\|_2^2$. This fixes its dynamic range, as after normalisation the loss near the optimum can be expected to be contained in the $[0, 1)$ interval, touching zero at the optimum. In order to make the dynamic range of the regulariser(s) comparable one can aim for a solution $\mathbf{x}^*$ which has roughly unitary Euclidean norm. While representations are largely insensitive to the scaling of the image range, this is not exactly true for the first few layers of CNNs, where biases are tuned to a "natural" working range. This can be addressed by considering the objective $\|\Phi(\sigma\mathbf{x}) - \Phi_0\|_2^2 / \|\Phi_0\|_2^2 + \mathcal{R}(\mathbf{x})$ where the scaling σ is the average Euclidean norm of natural images in a training set.

Second, the multiplier λ_α of the α-norm regularizer should be selected to encourage the reconstructed image $\sigma\mathbf{x}$ to be contained in a natural range $[-B, B]$ (*e.g.* in most CNN implementations $B = 128$). If most pixels in $\sigma\mathbf{x}$ have a magnitude similar to B, then $\mathcal{R}_\alpha(\mathbf{x}) \approx HWB^\alpha/\sigma^\alpha$, and $\lambda_\alpha \approx \sigma^\alpha/(HWB^\alpha)$. A similar argument suggests to pick the V^β-norm regulariser coefficient as $\lambda_{V^\beta} \approx \sigma^\beta/(HW(aB)^\beta)$, where a is a small fraction (*e.g.* $a = 1\%$) relating the dynamic range of the image to that of its gradient.

The final form of the objective function is

$$\|\Phi(\sigma\mathbf{x}) - \Phi_0\|_2^2 / \|\Phi_0\|_2^2 + \lambda_\alpha \mathcal{R}_\alpha(\mathbf{x}) + \lambda_{V^\beta} \mathcal{R}_{V^\beta}(\mathbf{x}) \quad (3)$$

It is in general non convex because of the nature of Φ. We next discuss how to optimize it.

2.1. Optimisation

Finding an optimizer of the objective (1) may seem a hopeless task as most representations Φ involve strong non-linearities; in particular, deep representations are a chain of *several non-linear layers*. Nevertheless, simple gradient descent (GD) procedures have been shown to be very effective in *learning* such models from data, which is arguably an even harder task. Hence, it is not unreasonable to use GD to solve (1) too. We extend GD to incorporate a few extensions that proved useful in learning deep networks [13], as discussed below.

Momentum. GD is extended to use *momentum*:

$$\mu_{t+1} \leftarrow m\mu_t - \eta_t \nabla E(\mathbf{x}), \qquad \mathbf{x}_{t+1} \leftarrow \mathbf{x}_t + \mu_\mathbf{t}$$

where $E(\mathbf{x}) = \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda\mathcal{R}(\mathbf{x})$ is the objective function. The vector μ_t is a weighed average of the last several gradients, with decaying factor $m = 0.9$. Learning proceeds a few hundred iterations with a fixed learning rate η_t and is reduced tenfold, until convergence.

Computing derivatives. Applying GD requires computing the derivatives of the loss function composed with the representation $\Phi(\mathbf{x})$. While the squared Euclidean loss is smooth, this is not the case for the representation. A key feature of CNNs is the ability of computing the derivatives of each computational layer, composing the latter in an overall derivative of the whole function using back-propagation. Our translation of HOG and DSIFT into CNN allows us to apply the same technique to these computer vision representations too.

3. Representations

This section describes the image representations studied in the paper: DSIFT (Dense-SIFT), HOG, and reference deep CNNs. Furthermore, it shows how to implement DSIFT and HOG in a standard CNN framework in order to compute their derivatives. Being able to compute derivatives is the only requirement imposed by the algorithm of Sect. 2.1. Implementing DSIFT and HOG in a standard CNN framework makes derivative computation convenient.

CNN-A: deep networks. As a reference deep network we consider the Caffe-Alex [11] model (CNN-A), which closely reproduces the network by Krizhevsky *et al*. [13]. This and many other similar networks alternate the following computational building blocks: linear convolution, ReLU gating, spatial max-pooling, and group normalisation. Each such block takes as input a d-dimensional image and produces as output a k-dimensional one. Blocks can additionally pad the image (with zeros for the convolutional blocks and with $-\infty$ for max pooling) or subsample the data. The last several layers are deemed "fully connected" as the support of the linear filters coincides with the size of the image; however, they are equivalent to filtering layers in all other respects. Table 2 details the structure of CNN-A.

CNN-DSIFT and CNN-HOG. This section shows how DSIFT [17, 20] and HOG [4] can be implemented as CNNs. This formalises the relation between CNNs and these standard representations. It also makes derivative computation for these representations simple; for the inversion algorithm of Sect. 2. The DSIFT and HOG implementations in the VLFeat library [29] are used as numerical references. These are equivalent to Lowe's [17] SIFT and the DPM V5 HOG [6, 7].

SIFT and HOG involve: computing and binning image gradients, pooling binned gradients into cell histograms, grouping cells into blocks, and normalising the blocks. Denote by $\mathbf{g}$ the gradient at a given pixel and consider binning this into one of K orientations (where $K = 8$ for SIFT and $K = 18$ for HOG). This can be obtained in two steps: directional filtering and gating. The k-th directional filter is $G_k = u_{1k}G_x + u_{2k}G_y$ where

$$\mathbf{u}_k = \begin{bmatrix} \cos\frac{2\pi k}{K} \\ \sin\frac{2\pi k}{K} \end{bmatrix}, \quad G_x = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad G_y = G_x^\top.$$

The output of a directional filter is the projection $\langle \mathbf{g}, \mathbf{u}_k \rangle$ of the gradient along direction $\mathbf{u}_k$. A suitable gating function implements binning into a histogram element h_k. DSIFT uses bilinear orientation binning, given by

$$h_k = \|\mathbf{g}\| \max\left\{0, 1 - \frac{K}{2\pi}\cos^{-1}\frac{\langle \mathbf{g}, \mathbf{u}_k \rangle}{\|\mathbf{g}\|}\right\},$$

whereas HOG (in the DPM V5 variant) uses hard assignments $h_k = \|\mathbf{g}\|\mathbf{1}\left[\langle \mathbf{g}, \mathbf{u}_k \rangle > \|\mathbf{g}\|\cos\pi/K\right]$. Filtering is a standard CNN operation but these binning functions are not. While their implementation is simple, an interesting alternative is the approximated bilinear binning:

$$\begin{aligned} h_k &\approx \|\mathbf{g}\| \max\left\{0, \frac{1}{1-a}\frac{\langle \mathbf{g}, \mathbf{u}_k \rangle}{\|\mathbf{g}\|} - \frac{a}{1-a}\right\} \\ &\propto \max\left\{0, \langle \mathbf{g}, \mathbf{u}_k \rangle - a\|\mathbf{g}\|\right\}, \quad a = \cos 2\pi/K. \end{aligned}$$

The norm-dependent offset $\|\mathbf{g}\|$ is still non-standard, but the ReLU operator is, which shows to which extent approximate binning can be achieved in typical CNNs.

The next step is to pool the binned gradients into cell histograms using bilinear spatial pooling, followed by extracting blocks of 2×2 (HOG) or 4×4 (SIFT) cells. Both such operations can be implemented by banks of linear filters. Cell blocks are then l^2 normalised, which is a special case of the standard local response normalisation layer. For HOG, blocks are further decomposed back into cells, which requires another filter bank. Finally, the descriptor values are clamped from above by applying $y = \min\{x, 0.2\}$ to each component, which can be reduced to a combination of linear and ReLU layers.

The conclusion is that approximations to DSIFT and HOG can be implemented with conventional CNN components plus the non-conventional gradient norm offset. However, all the filters involved are much sparser and simpler than the generic 3D filters in learned CNNs. Nonetheless, in the rest of the paper we will use exact CNN equivalents of DSIFT and HOG, using modified or additional CNN components as needed. [1] These CNNs are numerically indis-

[1] This requires addressing a few more subtleties. In DSIFT gradient contributions are usually weighted by a Gaussian centered at each descriptor (a 4×4 cell block); here we use the VLFeat approximation (`fast` option) of weighting cells rather than gradients, which can be incorporated in the block-forming filters. In UoCTTI HOG, cells contain both oriented and unoriented gradients (27 components in total) as well as 4 texture components. The latter are ignored for simplicity, while the unoriented gradients are obtained as average of the oriented ones in the block-forming filters.

descriptors method	HOG HOGgle	HOG our	HOGb our	DSIFT our
error (%)	66.20 ±13.7	28.10 ±7.9	10.67 ±5.2	10.89 ±7.5

Table 1. Average reconstruction error of different representation inversion methods, applied to HOG and DSIFT. HOGb denotes HOG with bilinear orientation assignments. The standard deviation shown is the standard deviation of the error and not the standard deviation of the mean error.

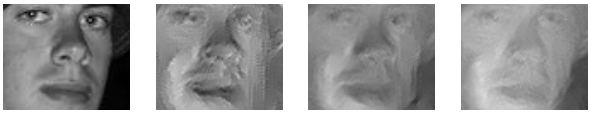

Figure 4. Effect of V^β regularization. The same inversion algorithm visualized in Fig. 3(d) is used with a smaller ($\lambda_{V^\beta} = 0.5$), comparable ($\lambda_{V^\beta} = 5.0$), and larger ($\lambda_{V^\beta} = 50$) regularisation coefficient.

tinguishable from the VLFeat reference implementations, but, true to their CNN nature, allow computing the feature derivatives as required by the algorithm of Sect. 2.

Next we apply the algorithm from Sect. 2 on **CNN-A**, **CNN-DSIFT** and **CNN-HOG** to analyze our method.

4. Experiments with shallow representations

This section evaluates the representation inversion method of Sect. 2 by applying it to HOG and DSIFT. The analysis includes both a qualitative (Fig. 3) and quantitative (Table 1) comparison with existing technique. The quantitative evaluation reports a normalized reconstruction error $\|\Phi(\mathbf{x}^*) - \Phi(\mathbf{x}_i)\|_2 / \mathrm{N}_\Phi$ averaged over 100 images $\mathbf{x}_i$ from the ILSVRC 2012 challenge [22] validation data (images 1 to 100). A normalization is essential to place the Euclidean distance in the context of the volume occupied by the features: if the features are close together, then even an Euclidean distance of 0.1 is very large, but if the features are spread out, then even an Euclidean distance of 10^5 may be very small. We use N_Φ to be the average pairwise euclidean distance between $\Phi(\mathbf{x}_i)$'s across the 100 test images.

We fix the parameters in equation 3 to $\lambda_\alpha = 2.16 \times 10^8$, $\lambda_{V^\beta} = 5$, and $\beta = 2$.

The closest alternative to our method is HOGgle, a technique introduced by Vondrick *et al.* [31] for the visualisation of HOG features. The HOGgle code is publicly available from the authors' website and is used throughout these experiments. Crucially, HOGgle is pre-trained to invert the UoCTTI implementation of HOG, which is numerically equivalent to CNN-HOG (Sect. 3), allowing for a direct comparison between algorithms.

Curiously, in UoCTTI HOG the l^2 normalisation factor is computed considering only the unoriented gradient components in a block, but applied to all, which requires modifying the normalization operator. Finally, when blocks are decomposed back to cells, they are averaged rather than stacked as in the original Dalal-Triggs HOG, which can be implemented in the block-decomposition filters.

Figure 5. Test images for qualitative results.

Compared to our method, HOGgle is fast (2-3s vs 60s on the same CPU) but not very accurate, as it is apparent both qualitatively (Fig. 3.c vs d) and quantitatively (66% vs 28% reconstruction error, see Table. 1). Interestingly, [31] propose a direct optimisation method similar to (1), but show that it does not perform better than HOGgle. This demonstrates the importance of the choice of regulariser and the ability of computing the derivative of the representation. The effect of the regularizer λ_{V^β} is further analysed in Fig. 4 (and later in Table 3): without this prior information, the reconstructions present a significant amount of discretization artifacts.

In terms of speed, an advantage of optimizing (1) is that it can be switched to use GPU code immediately given the underlying CNN framework; doing so results in a ten-fold speedup. Furthermore the CNN-based implementation of HOG and DSIFT wastes significant resources using generic filtering code despite the particular nature of the filters in these two representations. Hence we expect that an optimized implementation could be several times faster than this.

It is also apparent that different representations can be easier or harder to invert. In particular, modifying HOG to use bilinear gradient orientation assignments as SIFT (Sect. 3) significantly reduces the reconstruction error (from 28% down to 11%) and improves the reconstruction quality (Fig. 3.e). More impressive is DSIFT: it is quantitatively similar to HOG with bilinear orientations, but produces significantly more detailed images (Fig. 3.f). Since HOG uses a finer quantisation of the gradient compared to SIFT but otherwise the same cell size and sampling, this result can be imputed to the heavier block-normalisation of HOG that evidently discards more image information than SIFT.

5. Experiments with deep representations

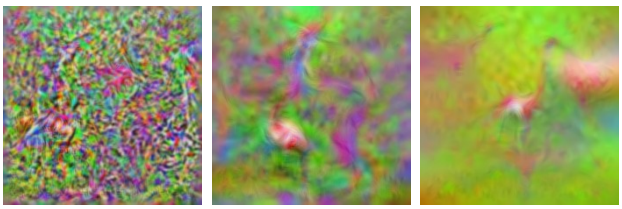

Figure 8. Effect of V^β regularization on CNNs. Inversions of the last layers of CNN-A for Fig. 5.d with a progressively larger regulariser λ_{V^β}. This image is best viewed in color/screen.

This section evaluates the inversion method applied to CNN-A described in Sect. 3. Compared to CNN-HOG

Figure 3. Reconstruction quality of different representation inversion methods, applied to HOG and DSIFT. HOGb denotes HOG with bilinear orientation assignments. This image is best viewed on screen.

layer	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
name	conv1	relu1	mpool1	norm1	conv2	relu2	mpool2	norm2	conv3	relu3	conv4	relu4	conv5	relu5	mpool5	fc6	relu6	fc7	relu7	fc8
type	cnv	relu	mpool	nrm	cnv	relu	mpool	nrm	cnv	relu	cnv	relu	cnv	relu	mpool	cnv	relu	cnv	relu	cnv
channels	96	96	96	96	256	256	256	256	384	384	384	384	256	256	256	4096	4096	4096	4096	1000
rec. field	11	11	19	19	51	51	67	67	99	99	131	131	163	163	195	355	355	355	355	355

Table 2. **CNN-A structure.** The table specifies the structure of CNN-A along with receptive field size of each neuron. The filters in layers from 16 to 20 operate as "fully connected": given the standard image input size of 227×227 pixels, their support covers the whole image. Note also that their receptive field is larger than 227 pixels, but can be contained in the image domain due to padding.

and CNN-DSIFT, this network is significantly larger and deeper. It seems therefore that the inversion problem should be considerably harder. Also, CNN-A is not handcrafted but learned from 1.2M images of the ImageNet ILSVRC 2012 data [22].

The algorithm of Sect. 2.1 is used to invert the code obtained from each individual CNN layer for 100 ILSVRC validation images (these were not used to train the CNN-A model [13]). Similar to Sect. 4, the normalized inversion error is computed and reported in Table 3. The experiment is repeated by fixing λ_α to a fixed value of 2.16×10^8 and gradually increasing λ_{V^β} ten-folds, starting from a relatively small value $\lambda_1 = 0.5$. The ImageNet ILSVRC mean image is added back to the reconstruction before visualisation as this is subtracted when training the network. Somewhat surprisingly, the quantitative results show that CNNs are, in fact, not much harder to invert than HOG. The error rarely exceeds 20%, which is comparable to the accuracy of HOG (Sect. 4). The last layer is in particular easy to invert with an average error of 8.5%.

We choose the regularizer coefficients for each representation/layer based on a quantitative and qualitative study of the reconstruction. We pick $\lambda_1 = 0.5$ for layers 1-6, $\lambda_2 = 5.0$ for layers 7-12 and $\lambda_3 = 50$ for layers 13-20. The error value corresponding to these parameters is marked in bold face in table 3. Increasing λ_{V^β} causes a deterioration for the first layers, but for the latter layers it helps recover a more visually interpretable reconstruction. Though this parameter can be tuned by cross validation on the normalized reconstruction error, a selection based on qualitative analysis is preferred because the method should yield images that are visually meaningful.

Qualitatively, Fig. 6 illustrates the reconstruction for a test image from each layer of CNN-A. The progression is remarkable. The first few layers are essentially an invertible code of the image. All the convolutional layers maintain a photographically faithful representation of the image, although with increasing fuzziness. The 4,096-dimensional fully connected layers are perhaps more interesting, as they invert back to a *composition of parts similar but not identical to the ones found in the original image*. Going from relu7 to fc8 reduces the dimensionality further to just 1,000; nevertheless some of these visual elements can still be identified. Similar effects can be observed in the reconstructions in Fig. 7. This figure includes also the reconstruction of an abstract pattern, which is not included in any of the ImageNet classes; still, all CNN codes capture distinctive visual features of the original pattern, clearly indicating that even very deep layers capture visual information.

Next, Fig. 7 examines the invariance captured by the CNN model by considering multiple reconstructions out of each deep layer. A careful examination of these images re-

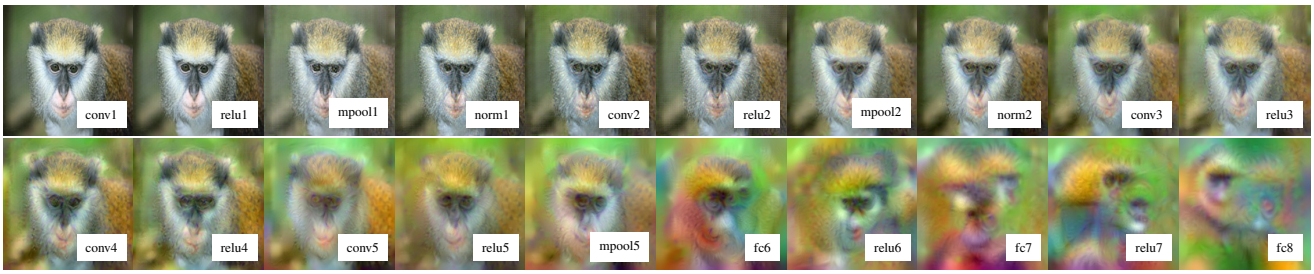

Figure 6. **CNN reconstruction.** Reconstruction of the image of Fig. 5.a from each layer of CNN-A. To generate these results, the regularization coefficient for each layer is chosen to match the highlighted rows in table 3. This figure is best viewed in color/screen.

λ_{V^β}	1 conv1	2 relu1	3 pool1	4 norm1	5 conv2	6 relu2	7 pool2	8 norm2	9 conv3	10 relu3	11 conv4	12 relu4	13 conv5	14 relu5	15 pool5	16 fc6	17 relu6	18 fc7	19 relu7	20 fc8
λ_1	**10.0** ±5.0	**11.3** ±5.5	**21.9** ±9.2	**20.3** ±5.0	**12.4** ±3.1	**12.9** ±5.3	15.5 ±4.7	15.9 ±4.6	14.5 ±4.7	16.5 ±5.3	14.9 ±3.8	13.8 ±3.8	12.6 ±2.8	15.6 ±5.1	16.6 ±4.6	12.4 ±3.5	15.8 ±4.5	12.8 ±6.4	10.5 ±1.9	5.3 ±1.1
λ_2	20.2 ±9.3	22.4 ±10.3	30.3 ±13.6	28.2 ±7.6	20.0 ±4.9	17.4 ±5.0	**18.2** ±5.5	**18.4** ±5.0	**14.4** ±3.6	**15.1** ±3.3	**13.3** ±2.6	**14.0** ±2.8	15.4 ±2.7	13.9 ±3.2	15.5 ±3.5	14.2 ±3.7	13.7 ±3.1	15.4 ±10.3	10.8 ±1.6	5.9 ±0.9
λ_3	40.8 ±17.0	45.2 ±18.7	54.1 ±22.7	48.1 ±11.8	39.7 ±9.1	32.8 ±7.7	32.7 ±8.0	32.4 ±7.0	25.6 ±5.6	26.9 ±5.2	23.3 ±4.1	23.9 ±4.6	**25.7** ±4.3	**20.1** ±4.3	**19.0** ±4.3	**18.6** ±4.9	**18.7** ±3.8	**17.1** ±3.4	**15.5** ±2.1	**8.5** ±1.3

Table 3. **Inversion error for CNN-A.** Average inversion percentage error (normalized) for all the layers of CNN-A and various amounts of V^β regularisation: $\lambda_1 = 0.5$, $\lambda_2 = 10\lambda_1$ and $\lambda_3 = 100\lambda_1$. In bold face are the error values corresponding to the regularizer that works best both qualitatively and quantitatively. The deviations specified in this table are the standard deviations of the errors and not the standard deviations of the mean error value.

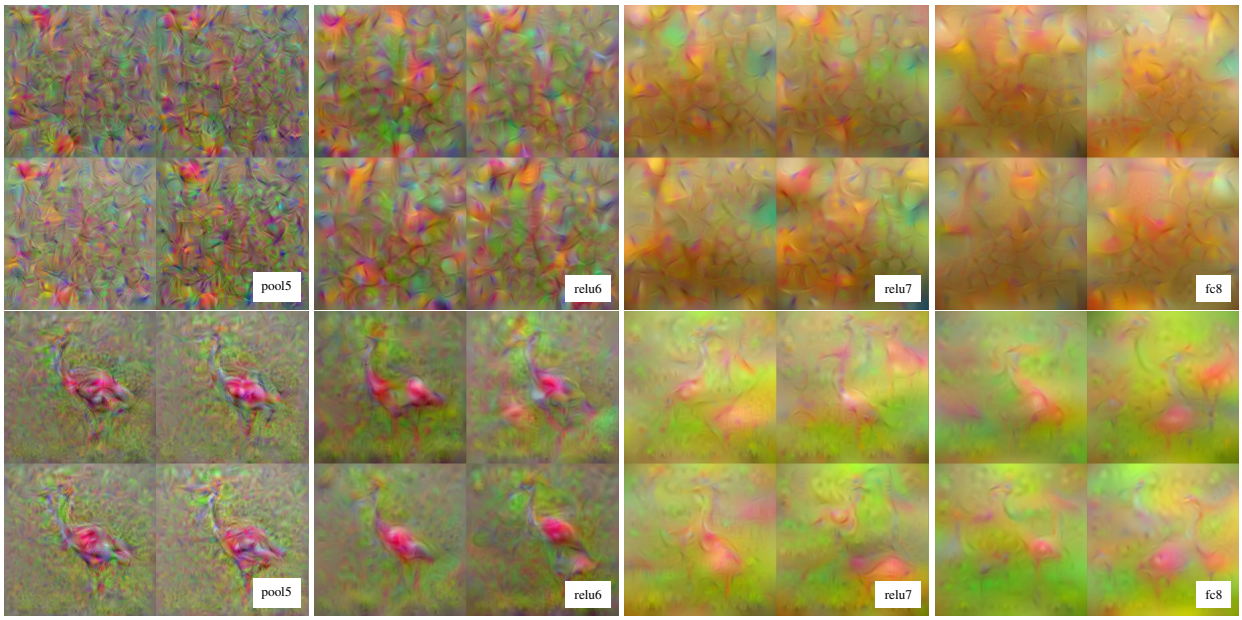

Figure 7. **CNN invariances.** Multiple reconstructions of the images of Fig. 5.c–d from different deep codes obtained from CNN-A. This figure is best seen in colour/screen.

veals that the codes capture progressively larger deformations of the object. In the "flamingo" reconstruction, in particular, relu7 and fc8 invert back to multiple copies of the object/parts at different positions and scales.

Note that all these and the original images are nearly indistinguishable from the viewpoint of the CNN model; it is therefore interesting to note the lack of detail in the deepest reconstructions, showing that the network captures just a sketch of the objects, which evidently suffices for classification. Considerably lowering the regulariser parameter still yields very accurate inversions, but this time with barely any resemblance to a natural image. This confirms that CNNs have strong non-natural confounders.

We now examine reconstructions obtained from subset of neural responses in different CNN layers. Fig. 9 explores the *locality* of the codes by reconstructing a central 5×5 patch of features in each layer. The regulariser encourages portions of the image that do not contribute to the neural

Figure 9. **CNN receptive field.** Reconstructions of the image of Fig. 5.a from the central 5×5 neuron fields at different depths of CNN-A. The white box marks the field of view of the 5×5 neuron field. The field of view is the entire image for conv5 and relu5.

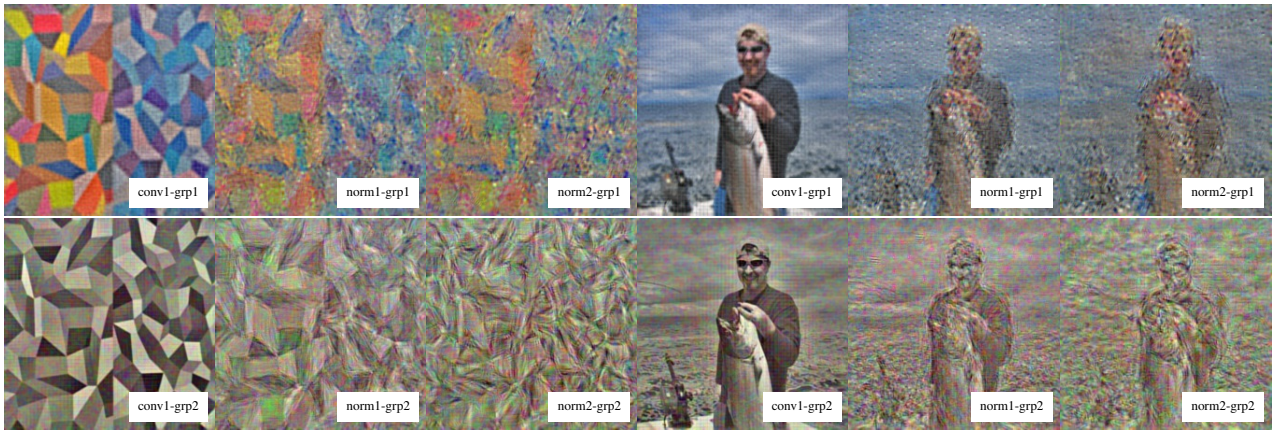

Figure 10. **CNN neural streams.** Reconstructions of the images of Fig. 5.c-b from either of the two neural streams of CNN-A. This figure is best seen in colour/screen.

responses to be switched off. The locality of the features is obvious in the figure; what is less obvious is that the effective receptive field of the neurons is in some cases significantly smaller than the theoretical one - shown as a white box in the image.

Finally, Fig. 10 reconstructs images from a subset of feature channels. CNN-A contains in fact two subsets of feature channels which are independent for the first several layers (up to norm2) [13]. Reconstructing from each subset individually, clearly shows that one group is tuned towards low-frequency colour information whereas the second one is tuned to towards high-frequency luminance components. Remarkably, this behaviour emerges naturally in the learned network without any mechanism directly encouraging this pattern.

6. Summary

This paper proposed an optimisation method to invert shallow and deep representations based on optimizing an objective function with gradient descent. Compared to alternatives, a key difference is the use of image priors such as the V^β norm that can recover the low-level image statistics removed by the representation. This tool performs better

Figure 11. **Diversity in the CNN model.** mpool5 reconstructions show that the network retains rich information even at such deep levels. This figure is best viewed in color/screen (zoom in).

than alternative reconstruction methods for HOG. Applied to CNNs, the visualisations shed light on the information represented at each layer. In particular, it is clear that a progressively more invariant and abstract notion of the image content is formed in the network.

In the future, we shall experiment with more expressive natural image priors and analyze the effect of network hyper-parameters on the reconstructions. We shall extract subsets of neurons that encode object parts and try to establish sub-networks that capture different details of the image.

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[2] Y. Chen, R. Ranftl, and T. Pock. A bi-level view of inpainting-based image compression. In *Proc. of Computer Vision Winter Workshop*, 2014.

[3] G. Csurka, C. R. Dance, L. Dan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Proc. ECCV Workshop on Stat. Learn. in Comp. Vision*, 2004.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[5] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical Report 1341, University of Montreal, 2009.

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[7] R. B. Girshick, P. F. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5. http://people.cs.uchicago.edu/~rbg/latent-release5/.

[8] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 2006.

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

[10] C. A. Jensen, R. D. Reed, R. J. Marks, M. El-Sharkawi, J.-B. Jung, R. Miyamoto, G. Anderson, and C. Eggen. Inversion of feedforward neural networks: algorithms and applications. *Proc. of the IEEE*, 87(9), 1999.

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

[12] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *NIPS*, 2009.

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

[14] S. Lee and R. M. Kil. Inverse mapping of continuous functions using local and global information. *IEEE Trans. on Neural Networks*, 5(3), 1994.

[15] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43(1), 2001.

[16] A. Linden and J. Kindermann. Inversion of multilayer nets. In *Proc. Int. Conf. on Neural Networks*, 1989.

[17] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.

[18] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2(60):91–110, 2004.

[19] B.-L. Lu, H. Kita, and Y. Nishikawa. Inverting feedforward neural networks using linear and nonlinear programming. *IEEE Trans. on Neural Networks*, 10(6), 1999.

[20] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *ECCV*, 2006.

[21] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorizaton. In *CVPR*, 2006.

[22] 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, 2014.

[23] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *CoRR*, volume abs/1312.6229, 2014.

[24] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proc. ICLR*, 2014.

[25] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[26] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[27] A. Tatu, F. Lauze, M. Nielsen, and B. Kimia. Exploring the representation capabilities of the HOG descriptor. In *ICCV Workshop*, 2011.

[28] A. R. Várkonyi-Kóczy and A. Rövid. Observer based iterative neural network model inversion. In *IEEE Int. Conf. on Fuzzy Systems*, 2005.

[29] A. Vedaldi. An open implementation of the SIFT detector and descriptor. Technical Report 070012, UCLA CSD, 2007.

[30] A. Vedaldi and K. Lenc. MatConvNet: CNNs for MATLAB. http://www.vlfeat.org/matconvnet/, 2014.

[31] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. HOGgles: Visualizing object detection features. In *ICCV*, 2013.

[32] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. *CVPR*, 2010.

[33] P. Weinzaepfel, H. Jégou, and P. Pérez. Reconstructing an image from its local descriptors. In *CVPR*, 2011.

[34] R. J. Williams. Inverting a connectionist network mapping by back-propagation of error. In *Proc. CogSci*, 1986.

[35] J. Yang, K. Yu, and T. Huang. Supervised translation-invariant sparse coding. In *CVPR*, 2010.

[36] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[37] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010.