CrossMark

Visualizing Deep Convolutional Neural Networks Using Natural Pre-images

Aravindh Mahendran[1] · Andrea Vedaldi[1]

Abstract Image representations, from SIFT and bag of visual words to convolutional neural networks (CNNs) are a crucial component of almost all computer vision systems. However, our understanding of them remains limited. In this paper we study several landmark representations, both shallow and deep, by a number of complementary visualization techniques. These visualizations are based on the concept of "natural pre-image", namely a natural-looking image whose representation has some notable property. We study in particular three such visualizations: inversion, in which the aim is to reconstruct an image from its representation, activation maximization, in which we search for patterns that maximally stimulate a representation component, and caricaturization, in which the visual patterns that a representation detects in an image are exaggerated. We pose these as a regularized energy-minimization framework and demonstrate its generality and effectiveness. In particular, we show that this method can invert representations such as HOG more accurately than recent alternatives while being applicable to CNNs too. 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.

Keywords Visualization · Convolutional neural networks · Pre-image problem

✉ Aravindh Mahendran
  aravindh@robots.ox.ac.uk
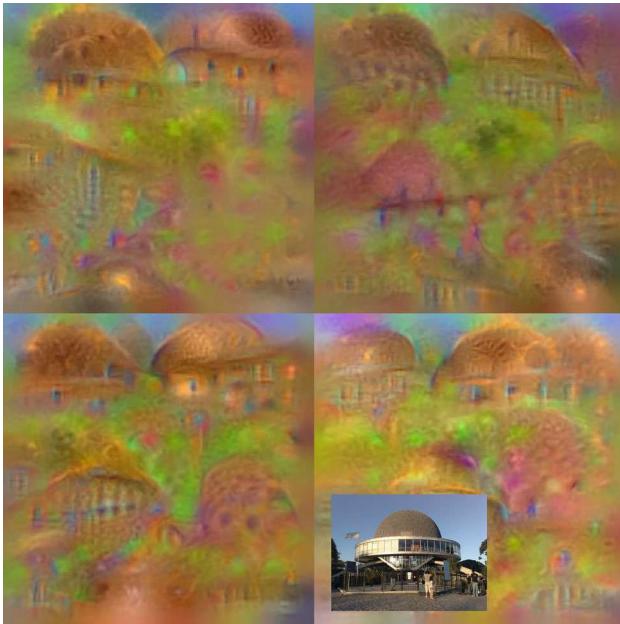
[1] University of Oxford, Oxford, UK

1 Introduction

Most image understanding and computer vision methods do not operate directly on images, but on suitable image representations. Notable examples of representations include textons (Leung and Malik 2001), histogram of oriented gradients (SIFT, Lowe 2004 and HOG, Dalal and Triggs 2005), bag of visual words (Csurka et al. 2004; Sivic and Zisserman 2003), sparse (Yang et al. 2010) and local coding (Wang et al. 2010), super vector coding (Zhou et al. 2010), VLAD (Jégou et al. 2010), Fisher Vectors (Perronnin and Dance 2006), and, lately, deep neural networks, particularly of the convolutional variety (Krizhevsky et al. 2012; Zeiler and Fergus 2014; Sermanet et al. 2014). While the performance of representations has been improving significantly in the past few years, their design remains eminently empirical. This is true for shallower hand-crafted features such as HOG or SIFT and even more so for the latest generation of deep representations, such as deep convolutional neural networks (CNNs), where millions of parameters are learned from data. A consequence of this complexity is that our understanding of such representations is limited.

In this paper, with the aim of obtaining a better understanding of representations, we develop a family of methods to investigate CNNs and other image features by means of visualizations. All these methods are based on the common idea of seeking natural-looking images whose representations are notable in some useful sense. We call these constructions *natural pre-images* and propose a unified formulation and algorithm to compute them (Sect. 3).

Within this framework, we explore three particular types of visualizations. In the first type, called **inversion** (Sect. 5), we compute the "inverse" of a representation (Fig. 1). We do so by modelling a representation as a function $\Phi_0 = \Phi(\mathbf{x}_0)$ of the image $\mathbf{x}_0$. Then, we attempt to recover the image from

Fig. 1 Four reconstructions of the bottom-right image obtained from the 1000 code extracted from the last fully connected layer of the VGG-M CNN (Chatfield et al. 2014) (Color figure online)

the information contained only in the code Φ_0. Notably, most representations Φ are *not* invertible functions; for example, a representation that is *invariant* to nuisance factors such as viewpoint and illumination removes this information from the image. Our aim is to characterize this loss of information by studying the equivalence class of images $\mathbf{x}^*$ that share the same representation $\Phi(\mathbf{x}^*) = \Phi_0$.

In **activation maximization** (Sect. 6), the second visualization type, we look for an image $\mathbf{x}^*$ that maximally excites a certain component $[\Phi(\mathbf{x})]_i$ of the representation. The resulting image is representative of the visual stimuli that are selected by that component and helps understand its "meaning" or function. This type of visualization is sometimes referred to as "deep dream" as it can be interpreted as the result of the representation "imagining" a concept.

In our third and last visualization type, which we refer to as **caricaturization** (Sect. 7), we modify an initial image $\mathbf{x}_0$ to exaggerate any pattern that excites the representation $\Phi(\mathbf{x}_0)$. Differently from activation maximization, this visualization method emphasizes the meaning of combinations of representation components that are active together.

Several of these ideas have been explored by us and others in prior work as detailed in Sect. 2. In particular, the idea of visualizing representations using pre-images has been investigated in connection with neural networks since at least the work of Linden and Kindermann (1989).

Our first contribution is to introduce the idea of a *natural pre-image* (Mahendran and Vedaldi 2015), i.e. to restrict reconstructions to the set of natural images. While this is difficult to achieve in practice, we explore different regularization

methods (Sect. 3.2) that can work as a proxy, including regularizers using the Total Variation (TV) norm of the image. We also explore an indirect regularization method, namely the application of random jitter to the reconstruction as suggested by Mordvintsev et al. (2015).

Our second contribution is to consolidate different visualization and representation types, including inversion, activation maximization, and caricaturization, in a common framework (Sect. 3). We propose a single algorithm applicable to a large variety of representations, from SIFT to very deep CNNs, using essentially a single set of parameters. The algorithm is based on optimizing an energy function using gradient descent and back-propagation through the representation architecture.

Our third contribution is to apply the three visualization types to the study of several different representations. First, we show that, despite its simplicity and generality, our method recovers significantly better reconstructions for shallow representations such as HOG compared to recent alternatives (Vondrick et al. 2013) (Sect. 5.1). In order to do so, we also rebuild the HOG and DSIFT representations as equivalent CNNs, simplifying the computation of their derivatives as required by our algorithm (Sect. 4.1). Second, we apply inversion (Sect. 5.2), activation maximization (Sect. 6), and caricaturization (Sect. 7) to the study of CNNs, treating each layer of a CNN as a different representation, and studying different state-of-the-art architectures, namely AlexNet, VGG-M, and VGG very deep (Sect. 4.2). As we do so, we emphasize a number of general properties of such representations, as well as differences between them. In particular, we study the effect of depth on representations, showing that CNNs gradually build increasing levels of invariance and complexity, layer after layer.

Our findings are summarized in Sect. 8. The code for the experiments in this paper and extended visualizations are available at http://www.robots.ox.ac.uk/~vgg/research/invrep/index.html. This code uses the open-source MatConvNet toolbox (Vedaldi and Lenc 2014) and publicly available copies of the models to allow for easy reproduction of the results.

This paper is a substantially extended version of Mahendran and Vedaldi (2015), which introduced the idea of natural pre-image, but was limited to visualization by inversion.

2 Related Work

With the development of modern visual representations, there has been an increasing interest in developing visualization methods to understand them. Most of the recent contributions (Mordvintsev et al. 2015; Yosinksi et al. 2015) build on the idea of *natural* pre-images introduced in Mahendran and Vedaldi (2015), extending or applying it in different ways. In

turn, this work is based on several prior contributions that have used pre-images to understand neural networks and classical computer vision representations such as HOG and SIFT. The rest of the section discusses these relationships in detail.

2.1 Natural Pre-images

Mahendran and Vedaldi (2015) note that not all pre-images are equally interesting in visualization; instead, more meaningful results can be obtained by restricting pre-images to the set of natural images. This is particularly true in the study of discriminative models such as CNNs that are essentially "unspecified" outside the domain of natural images used to train them. While capturing the concept of natural images in an algorithm is difficult in practice, Mahendran et al. proposed to use simple natural image priors as a proxy. They formulated this approach in a regularized energy minimization framework. Among these, the most important regularizer was the quadratic norm[1] of the reconstructed image (Sect. 3.2).

The visual quality of pre-images can be further improved by introducing complementary regularization methods. Google's "inceptionism" (Mordvintsev et al. 2015), for example, contributed the idea of regularization through jittering: they shift the pre-image randomly during optimization, resulting in sharper and more vivid reconstructions. The work of Yosinksi et al. (2015) used yet another regularizer: they applied Gaussian blurring and clipped pixels that have small values or that have a small effect on activating components in a CNN representation to zero.

2.2 Methods for Finding Pre-images

The use of pre-images to visualize representations has a long history. Simonyan et al. (2014) applied this idea to recent CNNs and optimized, starting from random noise and by means of back-propagation and gradient descent, the response of individual filters in the last layer of a deep convolutional neural network—an example of activation maximization. Related energy-minimization frameworks were adopted by Mahendran and Vedaldi (2015), Mordvintsev et al. (2015), and Yosinksi et al. (2015) to visualize recent CNNs. Prior to that, very similar methods were applied to early neural networks in Williams (1986), Linden and Kindermann (1989), Lee and Kil (1994), and Lu et al. (1999), using gradient descent or optimization strategies based on sampling.

Several pre-image methods alternative to energy minimization have been explored as well. Nguyen et al. (2015) used genetic programming to generate images that maximize the response of selected neurons in the very last layer of a modern CNN, corresponding to an image classifier. Vondrick et al. (2013) learned a regressor that, given a HOG-encoded image patch, reconstructs the input image. Weinzaepfel et al. (2011) reconstructed an image from SIFT features using a large vocabulary of patches to invert individual detections and blended the results using Laplace (harmonic) interpolation. Earlier works (Jensen et al. 1999; Várkonyi-Kóczy and Rövid 2005) focussed on inverting networks in the context of dynamical systems and will not be discussed further here.

The DeConvNet method of Zeiler and Fergus (2014) "transposes" CNNs to find which image patches are responsible for certain neural activations. While this transposition operation applied to CNNs is somewhat heuristic, Simonyan et al. (2014) suggested that it approximates the derivative of the CNN and that, thereby, DeConvNet is analogous to one step of the backpropagation algorithm used in their energy minimization framework. A significant difference from our work is that in DeConvNet the authors transfer the pattern of activations of max-pooling layers from the direct CNN evaluation to the transposed one, therefore copying rather than inferring this geometric information during reconstruction.

A related line of work (Dosovitskiy and Brox 2015; Bishop 1995) is to learn a second neural network to act as the inverse of the original one. This is difficult because the inverse is usually not unique. Therefore, these methods may regress an "average pre-image" conditioned on the target representation, which may not be as effective as sampling the pre-image if the goal is to characterize representation ambiguities. One advantage of these methods is that they can be significantly faster than energy minimization.

Finally, the vast family of auto-encoder architectures (Hinton and Salakhutdinov 2006) train networks together with their inverses as a form of auto-supervision; here we are interested instead in visualizing feed-forward and discriminatively-trained CNNs now popular in computer vision.

2.3 Types of Visualizations Using Pre-images

Pre-images can be used to generate a large variety of complementary visualizations, many of which have been applied to a variety of representations.

The idea of **inverting representations** in order to recover an image from its encoding was used to study SIFT in the work of Weinzaepfel et al. (2011), Local Binary Descriptors by d'Angelo et al. (2012), HOG in Vondrick et al. (2013) and bag of visual words descriptors in Kato and Harada (2014). Mahendran and Vedaldi (2015) looked at the inversion problem for HOG, SIFT, and recent CNNs; our method differs

---

[1] It is referred to as TV norm in Mahendran and Vedaldi (2015) but for $\beta = 2$ this is actually the quadratic norm.

significantly from the ones above as it addresses many different representations using the same energy minimization framework and optimization algorithm. In comparison to existing inversion techniques for dense shallow representations such as HOG (Vondrick et al. 2013), it is also shown to achieve superior results, both quantitatively and qualitatively.

Perhaps the first to apply **activation maximization** to recent CNNs such as AlexNet (Krizhevsky et al. 2012) was the work of Simonyan et al. (2014), where this technique was used to maximize the response of neural activations in the last layer of a deep CNN. Since these responses are learned to correspond to specific object classes, this produces versions of the object as conceptualized by the CNN, sometimes called "deep dreams". Recently, Mordvintsev et al. (2015) has generated similar visualizations for their inception network and Yosinksi et al. (2015) have applied activation maximization to visualize not only the last layers of a CNN, but also intermediate representation components. Related extensive component-specific visualizations were conducted in Zeiler and Fergus (2014), albeit in their DeConvNet framework. The idea dates back to at least (Erhan et al. 2009), which introduced activation maximization to visualize deep networks learned from the MNIST digit dataset.

The first version of **caricaturization** was explored in Simonyan et al. (2014) to maximize image features corresponding to a particular object class, although this was ultimately used to generate saliency maps rather than to generate an image. The authors of Mordvintsev et al. (2015) extensively explored caricaturization in their "inceptionism" research with two remarkable results. The first was to show which visual structures are captured at different levels in a deep CNN. The second was to show that CNNs can be used to generate aesthetically pleasing images.

In addition to these three broad visualization categories, there are several others which are more specific. In DeConvNet (Zeiler and Fergus 2014) for example, visualizations are obtained by activation maximization. They search in a large dataset for an image that causes a given representation component to activate maximally. The network is then evaluated feed-forward and the location of the max-pooling activations is recorded. Combined with the transposed "deconvolutional network", this information is used to generate crisp visualizations of the excited neural paths. However, this differs from both inversion and activation maximization in that it uses of information beyond that contained in the representation output itself.

2.4 Activation Statistics

In addition to inversion, activation maximization, and caricaturization, the pre-image method can be extended to several other visualization types with either the goal of understanding representations or of generating images for other purposes. Next, we discuss a few notable cases.

First, representations can be used as *statistics that describe a class of images*. This idea is rooted in the seminal work of Julesz (1981) that used the statistics of simple filters to describe **visual textures**. Julesz' ideas were framed probabilistically by Zhu et al. (1998) and their generation-by-sampling framework was later approximated by Portilla and Simoncelli (2000) as a pre-image problem which can be seen as a special case of the inversion method discussed here. More recently, Gatys et al. (2015b) showed that the results of Portilla and Simoncelli can be dramatically improved by replacing their wavelet-based statistics with the empirical correlation between deep feature channels in the convolutional layers of CNNs.

Gatys et al. further extended their work in Gatys et al. (2015a) with the idea of **style transfer**. Here a pre-image is found that simultaneously (1) reproduces the deep features of a reference "content image" (just like the inversion technique explored here) while at the same time (2) reproducing the correlation statistics of shallower features of a second reference "style image", treated as a source of texture information. This can be interpreted naturally in the framework discussed here as visualization by inversion where the natural image prior is implemented by "copying" the style of a visual texture. In comparison to the approach here, this generally results in more pleasing images. For understanding representations, such a technique can be used to encourage the generation of very different images that share a common deep feature representation, and that therefore may reveal interesting invariance properties of the representation.

Finally, another difference between the work of Gatys et al. (2015a, b) and the analysis in this paper is that they transfer information from several layers of the CNN simultaneously, whereas here we focus on individual layers, or even single feature components. Thus the two approaches are complementary. In their case, there is no need to add an explicit natural image prior as we do as this information is incorporated in the low-level CNN statistics that they import in style/texture transfer. As shown in the experiments, a naturalness prior is however important when the goal is to visualize deep features without biasing the reconstruction using this shallower information at the same time.

2.5 Fooling Representations

A line of research related to visualization by pre-images is that of "fooling representations". Here the goal is to generate images that a representation assigns to a particular category despite having distinctly incompatible semantics. Some of these methods look for *adversarial perturbations* of a source image. For instance, Tatu et al. (2011) show that it is possible to make any two images look nearly identical in SIFT space

up to the injection of adversarial noise in the data. The complementary effect was demonstrated for CNNs by Szegedy et al. (2014), where an imperceptible amount of adversarial noise was shown to change the predicted class of an image to any desired class. The latter observation was confirmed and extended by Nguyen et al. (2015). The instability of representations appear in contradiction with results in Weinzaepfel et al. (2011), Vondrick et al. (2013), and Mahendran and Vedaldi (2015). These show that HOG, SIFT, and early layers of CNNs are largely invertible. This apparent inconsistency may be resolved by noting that (Tatu et al. 2011; Szegedy et al. 2014; Nguyen et al. 2015) require the injection of adversarial noise which is very unlikely to occur in natural images. It is not unlikely that enforcing representation to be sufficiently regular would avoid the issue.

The work by Nguyen et al. (2015) proposes a second method to generate confounders. In this case, they use genetic programming to create, using a sequence of editing operations, an image that is classified as any desired class by the CNN, while not looking like an instance of any class. The CNN does not have a background class that could be used to reject such images; nonetheless the result is remarkable.

3 A Method for Finding the Pre-images of a Representation

This section introduces our method to find pre-images of an image representation. This method will then be applied to the inversion, activation maximization, and caricaturization problems. These are formulated as regularized energy minimization problems where the goal is to find a natural-looking image whose representation has a desired property (Williams 1986). Formally, given a representation function $\Phi : \mathbb{R}^{H \times W \times D} \to \mathbb{R}^d$ and a reference code $\Phi_0 \in \mathbb{R}^d$, we seek the image[2] $\mathbf{x} \in \mathbb{R}^{H \times W \times D}$ that minimizes the objective function:

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times D}}{\operatorname{argmin}} \ \mathcal{R}_\alpha(\mathbf{x}) + \mathcal{R}_{TV^\beta}(\mathbf{x}) + C\ell(\Phi(\mathbf{x}), \Phi_0) \quad (1)$$

The loss ℓ compares the image representation $\Phi(\mathbf{x})$ with the target value Φ_0, the two regularizer terms $\mathcal{R}_\alpha + \mathcal{R}_{TV^\beta} : \mathbb{R}^{H \times W \times D} \to \mathbb{R}_+$ capture a *natural image prior*, and the constant C trades off loss and regularizers.

The meaning of minimizing the objective function (1) depends on the choice of the loss and of the regularizer terms, as discussed below. While these terms contain several parameters, they are designed such that, in practice, all the parameters except C can be fixed for all visualization and representation types.

---

[2] In the following, the image $\mathbf{x}$ is assumed to have null mean, as required by most CNN implementations.

3.1 Loss Functions

Choosing different loss functions ℓ in Eq. (7) results in different visualizations. In *inversion*, ℓ is set to the Euclidean distance:

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

where $\Phi_0 = \Phi(\mathbf{x}_0)$ is the representation of a target image. Minimizing (1) results in an image $\mathbf{x}^*$ that "resembles" $\mathbf{x}_0$ from the viewpoint of the representation.

Sometimes it is interesting to restrict the reconstruction to a subset of the representation components. This is done by introducing a binary *mask M* of the same dimension as Φ_0 and by modifying Eq. (2) as follows:

$$\ell(\Phi(\mathbf{x}), \Phi_0; M) = \frac{\|(\Phi(\mathbf{x}) - \Phi_0) \odot M\|^2}{\|\Phi_0 \odot M\|^2}, \quad (3)$$

In *activation maximization* and *caricaturization*, $\Phi_0 \in \mathbb{R}^d_+$ is treated instead as a weight vector selecting which representation components should be maximally activated. This is obtained by considering the inner product:

$$\ell(\Phi(\mathbf{x}), \Phi_0) = -\frac{1}{Z}\langle \Phi(\mathbf{x}), \Phi_0 \rangle. \quad (4)$$

For example, if $\Phi_0 = \mathbf{e}_i$ is the indicator vector of the i-th component of the representation, minimizing Eq. (4) maximizes the component $[\Phi(\mathbf{x})]_i$. Alternatively, if Φ_0 is set to $\max\{\Phi(\mathbf{x}_0), 0\}$, the minimization of Eq. (1) will highlight components that are active in the representation $\Phi(\mathbf{x}_0)$ of a reference image $\mathbf{x}_0$, while ignoring the inactive components.

The choice of the normalization constant Z in activation maximization and caricaturization will be discussed later. Note also that, for the loss Eq. (4), there is no need to define a separate mask as this can be pre-multiplied into Φ_0.

3.2 Regularization

Discriminative representations discard a significant amount of low-level image information that is irrelevant to the target task (e.g. image classification). As this information is nonetheless useful for visualization, we propose to partially recover it by restricting the inversion to the subset of natural images $\mathcal{X} \subset \mathbb{R}^{H \times W \times D}$. This is motivated by the fact that, since representations are applied to natural images, there is comparatively little interest in understanding their behavior outside of this set. However, modeling the set of natural images is a significant challenge in its own right. As a proxy, we propose to regularize the reconstruction by using simple *image priors* implemented as regularizers in Eq. (1). We

Fig. 2 Input images used in the rest of the paper are shown above. From *left* to *right*: *row 1* spoonbill, gong, monkey; *row 2* building, red fox, abstract art (Color figure online)

experiment in particular with three such regularizers, discussed next.

3.2.1 Bounded Range

The first regularizer encourages the intensity of pixels to stay bounded. This is important for networks that include normalization layers, as in this case arbitrarily rescaling the image range has no effect on the network output. In activation maximization, it is even more important for networks that do *not* include normalization layers, as in this case increasing the image range increases neural activations by the same amount.

In Mahendran and Vedaldi (2015) this regularizer was implemented as a soft constraint using the penalty $\|\mathbf{x}\|_\alpha^\alpha$ for a large value of the exponent α. Here we modify it in several ways. First, for color images we make the term isotropic in RGB space by considering the norm

$$N_\alpha(\mathbf{x}) = \frac{1}{HWB^\alpha} \sum_{v=1}^{H} \sum_{u=1}^{W} \left(\sum_{k=1}^{D} \mathbf{x}(v, u, k)^2 \right)^{\frac{\alpha}{2}} \quad (5)$$

where v indexes the image rows, u the image columns, and k the color channels. By comparison, the norm used in Mahendran and Vedaldi (2015) is non-isotropic and might slightly bias the reconstruction of colors.

The term is normalized by the image area HW and by the scalar B. This scalar is set to the typical L^2 norm of the pixel RGB vector, such that $N_\alpha(\mathbf{x}) \approx 1$.

The soft constraint $N_\alpha(\mathbf{x})$ is combined with a hard constraint to limit the pixel intensity to be at most B_+:

$$R_\alpha(\mathbf{x}) = \begin{cases} N_\alpha(\mathbf{x}), & \forall v, u : \sqrt{\sum_k \mathbf{x}(v, u, k)^2} \leq B_+ \\ +\infty, & \text{otherwise.} \end{cases} \quad (6)$$

While the hard constraint may seem sufficient, in practice it was observed that without soft constraints, pixels tend to saturate in the reconstructions.

3.2.2 Bounded Variation

The second regularizer is the *total variation* (TV) $\mathcal{R}_{TV^\beta}(\mathbf{x})$ of the image, encouraging reconstructions to consist of piecewise constant patches. For a discrete image $\mathbf{x}$, the TV norm is approximated using finite differences as follows:

$$\mathcal{R}_{TV^\beta}(\mathbf{x}) = \frac{1}{HWV^\beta} \sum_{uvk} \Big((\mathbf{x}(v, u + 1, k) - \mathbf{x}(v, u, k))^2 + (\mathbf{x}(v + 1, u, k) - \mathbf{x}(v, u, k)))^2 \Big)^{\frac{\beta}{2}}$$
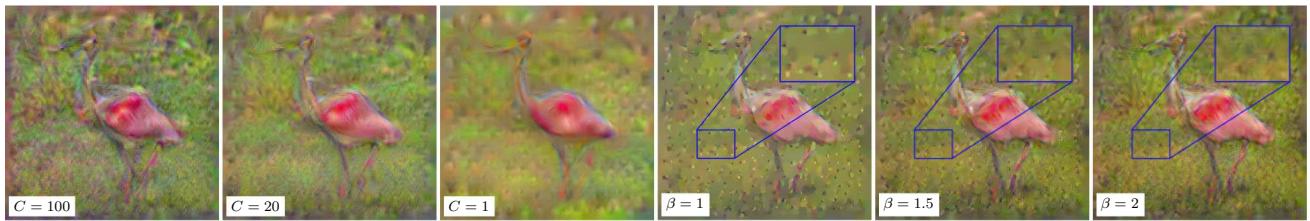
where $\beta = 1$. Here the constant V in the normalization coefficient is the typical value of the norm of the gradient in the image.

The standard TV regularizer, obtained for $\beta = 1$, was observed to introduce unwanted "spikes" in the reconstruction, as illustrated in Fig. 3 (right) when inverting a layer of a CNN. This is a known problem in TV-based image interpolation (see e.g.Fig. 3 in Chen et al. 2014). 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 (Krishnan and Fergus 2009), 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 penalizing large gradients, distributes changes across regions rather than concentrating them at a point or a curve. We refer to this as the TV^β regularizer. As seen in Fig. 3 (right), the spikes are removed for $\beta = 1.5, 2$ but the image is blurrier than for $\beta = 1$. At the same time, Fig. 3 (left) illustrates the importance of using the TV^β regularizer in obtaining clean reconstructions.

3.2.3 Jitter

The last regularizer, which is inspired by Mordvintsev et al. (2015), has an implicit form and consists of randomly shifting the input image before feeding it to the representation. Namely, we consider the optimization problem

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x} \in \mathbb{R}^{H \times W \times D}} \mathcal{R}_\alpha(\mathbf{x}) + \mathcal{R}_{TV^\beta}(\mathbf{x}) + CE_\tau[\ell(\Phi(\text{jitter}(\mathbf{x}; \tau)), \Phi_0)] \quad (7)$$

Fig. 3 *Left* Effect of the data term strength C in inverting a deep representation (the relu3 layer in AlexNet). Selecting a small value of C results in more regularized reconstructions, which is essential to obtain good results. *Right* Effect of the TV regularizer β exponent; note the spikes for $\beta = 1$ (zoomed in the *inset*). The input image in this case is the "spoonbill" image shown in Fig. 2 (Color figure online)

where $E[\cdot]$ denotes expectation and $\tau = (\tau_1, \tau_2)$ is a discrete random variable uniformly distributed in the set $\{0, \ldots, T - 1\}^2$, expressing a random horizontal and vertical translation of at most $T - 1$ pixels. The jitter$(\cdot)$ operator translates and crops $\mathbf{x}$ as follows:
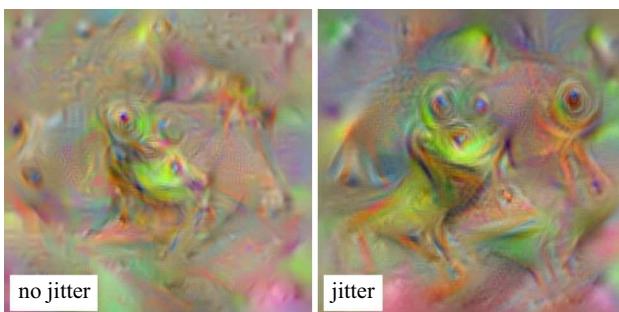
$$[\text{jitter}(\mathbf{x}; \tau)](v, u) = \mathbf{x}(v + \tau_2, u + \tau_1)$$

where $1 \leq v \leq H - T + 1$ and $1 \leq u \leq W - T + 1$. The expectation over τ is not computed explicitly; instead each iteration of SGD samples a new value of τ. Jittering counterbalances the very significant downsampling performed by the earlier layers of deep CNNs, interpolating between pixels in back-propagation. This generally results in crisper pre-images, particularly in the activation maximization problem (Fig. 4).

3.2.4 Texture and Style Regularizers

For completeness, we note that Eq. (1) can also be used to implement the texture synthesis and style transfer visualizations of Gatys et al. (2015a, b). One way to do so is to incorporate their texture/style term as an additional regularizer of the form

$$\mathcal{R}_{\text{tex}}(\mathbf{x}) = \sum_{l=1}^{L} w_l \, ||\psi \circ \Phi_l(\mathbf{x}) - \psi \circ \Phi_l(\mathbf{x}_{\text{tex}}))| \, ||_{\text{fro}}^2 \qquad (8)$$


Fig. 4 Effect of the jitter regularizer in activation maximization for the "tree frog" neuron in the fc8 layer in AlexNet. Jitter helps recover larger and crisper image structures (Color figure online)

Algorithm 1 Stochastic gradient descent for pre-image

Require: Given the objective function $E(\cdot)$ and the learning rate η_0
1: $G_0 \leftarrow 0, \mu_0 \leftarrow 0$
2: Initialize $\mathbf{x}_1$ to random noise
3: **for** $t = 1$ to T **do**
4:     $g_t \leftarrow \nabla E(\mathbf{x}_t)$ (using backprop)
5:     $G_t \leftarrow \rho G_{t-1} + g_t^2$ (component-wise)
6:     $\eta_t \leftarrow \dfrac{1}{\frac{1}{\eta_0} + \sqrt{G_t}}$ (component-wise)
7:     $\mu_t \leftarrow \rho \mu_{t-1} - \eta_t g_t$
8:     $\mathbf{x}_{t+1} \leftarrow \Pi_{B_+} (\mathbf{x}_t + \mu_t)$
9: **end for**

---

where $\mathbf{x}_{\text{tex}}$ is a reference image defining a texture or "visual style", $\Psi_l, l = 1, \ldots, L$ are increasingly deep layers in a CNN, $w_l \geq 0$ weights, and ψ is the *cross-channel correlation* operator

$$[\psi \circ \Phi_l(\mathbf{x})]_{cc'} = \sum_{uv} [\Phi_l(\mathbf{x})]_{uvc} [\Phi_l(\mathbf{x})]_{uvc'}$$

where $[\Phi_l(\mathbf{x})]_{uvc}$ denotes the cth feature channel activation at location (u, v).

The term $\mathcal{R}_{\text{tex}}(\mathbf{x})$ can be used as an objective function in its own right, yielding texture generation, or as a regularizer in the inversion problem, yielding style transfer.

3.3 Balancing the Loss and the Regularizers

One difficulty in implementing a successful image reconstruction algorithm using the formulation of Eq. (1) is to correctly balance the different terms. The loss functions and regularizers are designed in such a manner that, for reasonable reconstruction $\mathbf{x}$, they have comparable values (around unity). This normalization, though simple, makes a very significant difference. Without it we need to carefully tune parameters across different representation types. Unless otherwise noted, in the experiments we use the values, $C = 1$, $\alpha = 6$, $\beta = 2$, $B = 80$, $B_+ = 2B$, and $V = B/6.5$.

3.4 Optimization

Finding a minimizer of the objective (1) may seem difficult as most representations Φ are strongly non-linear; in particular, deep representations are a composition 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. In practice, a variant of GD was found to result in good reconstructions.

Algorithm The algorithm, whose pseudocode is given in Algorithm 1, is a variant of AdaGrad (Duchi et al. 2011). Like in AdaGrad, our algorithm automatically adapts the learning rate of individual components of the vector $\mathbf{x}_t$ by scaling it by the inverse of the accumulated squared gradient G_t. Similarly to AdaDelta (Zeiler 2012), however, it accumulates gradients only in a short temporal window, using the momentum coefficient $\rho = 0.9$. The gradient, scaled by the adaptive learning rate $\eta_t g_t$, is accumulated into a momentum vector μ_t with the same factor ρ. The momentum is then summed to the current reconstruction $\mathbf{x}_t$ and the result is projected back onto the feasible region $[-B_+, B_+]$.

Recently, Gatys et al. (2015a, b) have used the L-BFGS-B algorithm (Zhu et al. 1997) to optimize their texture/style loss (8). We found that L-BFGS-B is indeed better than (S)GD for their problem of texture generation, probably due to the particular nature of the term (8). However, preliminary experiments using L-BFGS-B for inversion did not show a significant benefit, so for simplicity we consider (S)GD-based algorithms in this paper.

The only parameters of Algorithm 1 are the initial learning rate η_0 and the number of iterations T. These are discussed next.

Learning rate η_0. This parameter can be heuristically set as follows. At the first iteration $G_0 \approx 0$ and $\eta_1 = 1/(1/\eta_0 + \sqrt{G_0}) \approx \eta_0$; the learning rate η_1 is approximately equal to the *initial learning rate* η_0. The value of the step size $\bar{\eta}_1$ that would minimize the term $R_\alpha(\mathbf{x})$ in a single iteration (ignoring momentum) is obtained by solving the equation $0 \approx \mathbf{x}_2 = \mathbf{x}_1 - \bar{\eta}_1 \nabla R_\alpha(\mathbf{x}_1)$. Assuming that all pixels in $\mathbf{x}_1$ have intensity equal to the parameter B introduced above, one then obtains the condition $0 = B - \bar{\eta}_1 \alpha/B$, so that $\bar{\eta}_1 = B^2/\alpha$. The initial learning rate η_0 is set to a hundredth of this value: $\eta_0 = 0.01\, \bar{\eta}_1 = 0.01\, B^2/\alpha$.

Number of iterations T. Algorithm 1 is run for $T = 300$ iterations. When jittering is used as a regularizer, we found it beneficial to eventually disable it and run the algorithm for a further 50 iterations, after reducing the learning rate tenfold. This fine tuning does not change the results qualitatively, but for inversion it slightly improves the reconstruction error;

thus it is not applied in caricaturization and activation maximization.

The cost of running Algorithm 1 is dominated by the cost of computing the derivative of the representation function, usually by back-propagation in a deep neural network. By comparison, the cost of computing the derivative of the regularizers and the cost of the gradient update are negligible. This also means that the algorithm runs faster for shallower representations and slower for deeper ones; on a CPU, it may in practice take only a few seconds to visualize shallow layers in a deep network and a few minutes for deep ones. GPUs can accelerate the algorithm by an order of magnitude or more. Another simple speedup is to stop the algorithm earlier; here using 300-350 iterations is a conservative choice that works for all representation types and visualizations we tested.

4 Representations

In this section, the image representations studied in the paper—dense SIFT, HOG, and several reference deep CNNs, are described. It is also shown how DSIFT and HOG can be implemented in a standard CNN framework, which simplifies the computation of their derivatives as required by the algorithm of Sect. 3.4.

4.1 Classical Representations

The *histograms of oriented gradients* are probably the best known family of "classical" computer vision features popularized by Lowe in Lowe (1999) with the SIFT descriptor. Here we consider two densely-sampled versions (Nowak et al. 2006), namely DSIFT (Dense SIFT) and HOG (Dalal and Triggs 2005). In the remainder of this section these two representations are reformulated as CNNs. This clarifies the relationship between SIFT, HOG, and CNNs in general and helps implement them in standard CNN toolboxes for experimentation. The DSIFT and HOG implementations in the VLFeat library (Vedaldi 2007) are used as numerical references. These are equivalent to Lowe's (1999) SIFT and the DPM V5 HOG (Felzenszwalb 2010; Girshick et al. 2010).

SIFT and HOG involve: computing and binning image gradients, pooling binned gradients into cell histograms, grouping cells into blocks, and normalizing the blocks. Let us denote by $\mathbf{g}$ the image 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 non-linear activation. The kth 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 $\mathbf{u}_k$. This is combined with a non-linear activation function to assign gradients to histogram elements h_k. DSIFT uses bilinear orientation assignment, 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 assignment $h_k = \|\mathbf{g}\| \mathbf{1} \left[\langle \mathbf{g}, \mathbf{u}_k \rangle > \|\mathbf{g}\| \cos \pi/K \right]$. Filtering is a standard CNN operation but these activation functions are not. While their implementation is simple, an interesting alternative is to approximate bilinear orientation assignment by using the activation function:

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

This activation function is the standard ReLU operator modified to account for the norm-dependent offset $a\|\mathbf{g}\|$. While the latter term is still non-standard, this indicates that a close approximation of binning can be achieved in standard CNN architectures.

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 operations can be implemented by banks of linear filters. Cell blocks are then l^2 normalized, which is a special case of the standard local response normalization 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.[3] These CNNs are numerically indistinguishable from the VLFeat reference implementations, but, true to their CNN nature, allow computing the feature derivatives as required by the algorithm of Sect. 3.4.

4.2 Deep Convolutional Neural Networks

The first CNN model considered in this paper is **AlexNet**. Due to its popularity, we use the implementation that ships

---

[3] This requires addressing a few more subtleties. Please see files `dsift_net.m` and `hog_net.m` for details.

with the Caffe framework (Jia 2013), which closely reproduces the original network by Krizhevsky et al. (2012). Occasionally, we also consider the **CaffeNet**, a network similar to AlexNet that also comes with Caffe. This and many other similar networks alternate the following computational building blocks: linear convolution, ReLU, spatial max-pooling, and local response normalization. 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 1 (left) details the structure of AlexNet.

The second network is the **VGG-M** model from Chatfield et al. (2014). The structure of VGG-M (Table 1—middle) is very similar to AlexNet, with the following differences: it includes a significantly larger number of filters in the different layers, filters at the beginning of the network are smaller, and filter strides (subsampling) is reduced. While the network is slower than AlexNet, it also performs much better on the ImageNet ILSVRC 2012 data.

The last network is the **VGG-VD-16** model from Simonyan et al. (2014). VGG-VD-16 is also similar to AlexNet, but with more substantial changes compared to VGG-M (Table 1—right). Filters are very narrow (3×3) and very densely sampled. There are no normalization layers. Most importantly, the network contains many more intermediate convolutional layers. The resulting model is very slow, but very powerful.

All pre-trained models are implemented in the MatConvNet framework and are publicly available at http://www.vlfeat.org/matconvnet/pretrained.

5 Visualization by Inversion

The experiments in this section apply the visualization by inversion method to both classical (Sect. 5.1) and CNN (Sect. 5.2) representations. As detailed in Sect. 3.1, for inversion, the objective function (1) is set up to minimize the L^2 distance (2) between the representation $\Phi(\mathbf{x})$ of the reconstructed image $\mathbf{x}$ and the representation $\Phi_0 = \Phi(\mathbf{x}_0)$ of a reference image $\mathbf{x}_0$.

Importantly, the optimization starts by initializing the reconstructed image to random i.i.d. noise such that *the only information available to the algorithm is the code Φ_0*. When started from different random initializations, the algorithm is expected to produce different reconstructions. This is partly due to the local nature of the optimization, but more fundamentally to the fact that representations are designed to be invariant to nuisance factors. Hence, images with irrelevant

Table 1 CNN architectures

AlexNet			VGG-M			VGG-VD-16		
Name	Size	Stride	Name	Size	Stride	Name	Size	Stride
conv1	11	4	conv1	7	2	conv1_1	3	1
relu1	11	4	relu1	7	2	relu1_1	3	1
						conv1_2	5	1
						relu1_2	5	1
norm1	11	4	norm1	7	2			
pool1	19	8	pool1	11	4	pool1	6	2
conv2	51	8	conv2	27	8	conv2_1	10	2
relu2	51	8	relu2	27	8	relu2_1	10	2
						conv2_2	14	2
						relu2_2	14	2
norm2	51	8	norm2	27	8			
pool2	67	16	pool2	43	16	pool2	16	4
conv3	99	16	conv3	75	16	conv3_1	24	4
relu3	99	16	relu3	75	16	relu3_1	24	4
						conv3_2	32	4
						relu3_2	32	4
						conv3_3	40	4
						relu3_3	40	4
						pool3	44	8
conv4	131	16	conv4	107	16	conv4_1	60	8
relu4	131	16	relu4	107	16	relu4_1	60	8
						conv4_2	76	8
						relu4_2	76	8
						conv4_3	92	8
						relu4_3	92	8
						pool4	100	16
conv5	163	16	conv5	139	16	conv5_1	132	16
relu5	163	16	relu5	139	16	relu5_1	132	16
						conv5_2	164	16
						relu5_2	164	16
						conv5_3	196	16
						relu5_3	196	16
pool5	195	32	pool5	171	32	pool5	212	32
fc6	355	32	fc6	331	32	fc6	404	32
relu6	355	32	relu6	331	32	relu6	404	32
fc7	355	32	fc7	331	32	fc7	404	32
relu7	355	32	relu7	331	32	relu7	404	32
fc8	355	32	fc8	331	32	fc8	404	32
prob	355	32	prob	331	32	prob	404	32

Structure of the AlexNet, VGG-M and VGG-VD-16 CNNs, including the layer names, the receptive field sizes (size), and the strides (stride) between feature samples, both in pixels. Note that, due to down-sampling and padding, the receptive field size can be larger than the size of the input image

differences should have the same representation and should be considered equally good reconstruction targets. In fact, it is by observing the differences between such reconstructions that we can obtain insights into the nature of the representation invariances.

Due to their intuitive nature, it is not immediately obvious how visualizations should be assessed quantitatively. Here we do so from multiple angles. The first is to test whether the algorithm successfully attains its goal of reconstructing an image $\mathbf{x}$ that has the desired representation $\Phi(\mathbf{x}) = \Phi_0$.

In Sect. 5.1 and Sect. 5.2 this is tested in terms of the relative reconstruction error of Eq. (2). Furthermore, in Sect. 5.2 it is also verified for CNNs whether the reconstructed and original representations have the same "meaning", in the sense that they are mapped to the same class label. Note that such tests assess the reconstruction quality in *feature space* rather than in *image space*. This is an important point: as noted above, we are not interested in recovering an image $\mathbf{x}$ which is perceptually similar to the reference image $\mathbf{x}_0$; rather, in order to study the invariances of the representation Φ, we would like to recover an image $\mathbf{x}$ that differs from $\mathbf{x}_0$ but has the same representation. Measuring the difference in feature space is therefore appropriate.

Finally, the effect of regularization is assessed empirically, via human assessment, to check whether the proposed notion of naturalness does in fact improve the interpretability of the visualizations.

5.1 Inverting Classical Representations: SIFT and HOG

In this section the visualization by inversion method is applied to the HOG and DSIFT representations.

5.1.1 Implementation Details

The parameter C in Eq. (1), trading off regularization and feature reconstruction fidelity, is set to 100 unless noted otherwise. Jitter is not used and the other parameters are set as stated in Sect. 3.3. HOG and DSIFT cell sizes are set to 8 pixels.
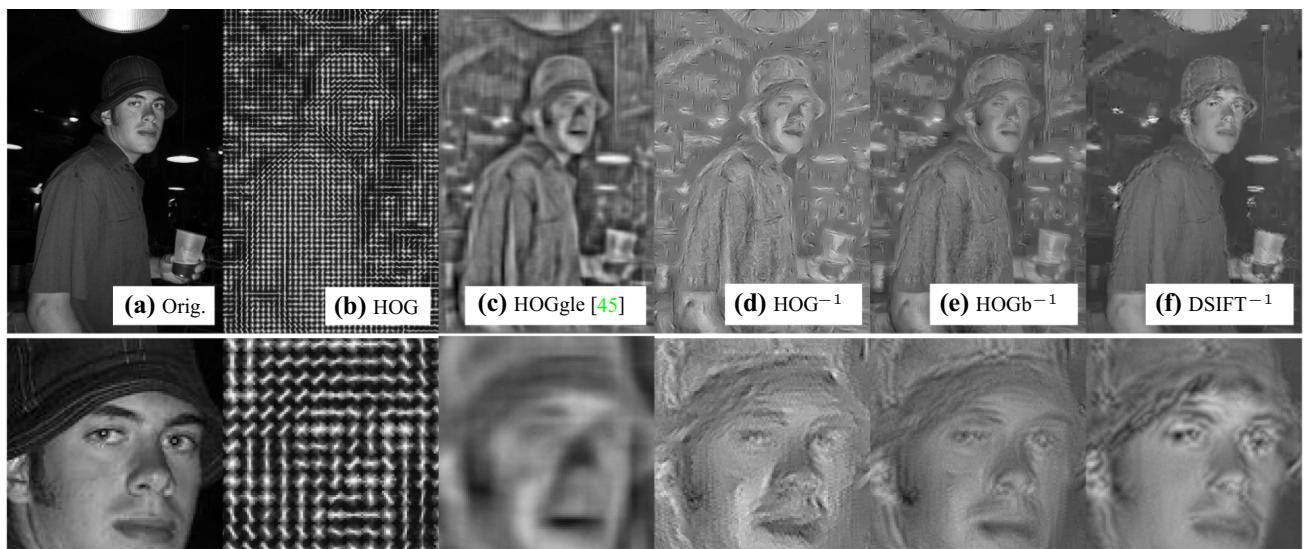
5.1.2 Reconstruction Quality

Based on the discussion above, the reconstruction quality is assessed by reporting the normalized reconstruction error (3), averaged over the first 100 images in the ILSVRC 2012 challenge validation images (Russakovsky et al. 2015). The closest alternative to our inversion method is HOGgle, a technique introduced by Vondrick et al. (2013) for visualizing HOG features. The HOGgle code is publicly available from the authors' website and is used throughout these experiments. HOGgle is pre-trained to invert the UoCTTI variant of HOG, which is numerically equivalent to the CNN-HOG network of Sect. 4, which allows us to compare algorithms directly.

Compared to our method, HOGgle is faster (2–3 vs. 60 s on the same CPU) but not as accurate, as is apparent both qualitatively (Fig. 5c vs. d) and quantitatively (60.1 vs. 36.6 % reconstruction error, see Table 2). Notably, Vondrick et al. did propose a direct optimization method similar to (1), but found that it did not perform better than HOGgle. This demonstrates the importance of the choice of regularizer and of the ability to compute the derivative of the representation analytically in order to implement optimization effectively. 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 speed-up.

5.1.3 Representation Comparison

Different representations are easier or harder to invert. For example, modifying HOG to use bilinear gradient orientation assignments as in SIFT (Sect. 4) significantly reduces

Fig. 5 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

Table 2 Average reconstruction error of different representation inversion methods, applied to HOG and DSIFT

Descriptors method	HOG HOGgle	HOG our	HOGb our	DSIFT our
Error (%)	60.1 ± 2.3	36.6 ± 3.4	11.6 ± 0.9	9.4 ± 1.7

HOGb denotes HOG with bilinear orientation assignments. The error bars show the 95 % confidence interval for the mean

the reconstruction error (from 36.6 % down to 11.5 %) and improves the reconstruction quality (Fig. 5e). More remarkable are the reconstructions obtained by inverting DSIFT: they are quantitatively similar to HOG with bilinear orientation assignment, but produce significantly more detailed images (Fig. 5f). Since HOG uses a finer quantization of the gradient compared to SIFT but otherwise the same cell size and sampling, this result can be imputed to the stronger normalization in HOG that evidently discards more visual information than in SIFT.

5.2 Inverting CNNs

In this section the visualization by inversion method is applied to representative CNNs: AlexNet, VGG-M, and VGG-VD-16.

5.2.1 Implementation Details

The jitter amount T is set to the integer closest to a quarter of the *stride* of the representation; the stride is the step in the receptive field of representation components when stepping through spatial locations. Its value is given in Table 1. The other parameters are set as stated in Sect. 3.3.

The parameter C in Eq. (1) is set to one of 1, 20, 100 or 300. Based on the analysis below, unless otherwise specified, visualizations use the following values: For AlexNet and VGG-M, we choose $C = 300$ up to relu3, $C = 100$ up to relu4, $C = 20$ up to relu5, and $C = 1$ for the remaining layers. For VGG-VD we use $C = 300$ up to conv4_3, $C = 100$ for conv5_1, $C = 20$ up to conv5_3, and $C = 1$ onwards.

5.2.2 Reconstruction Quality
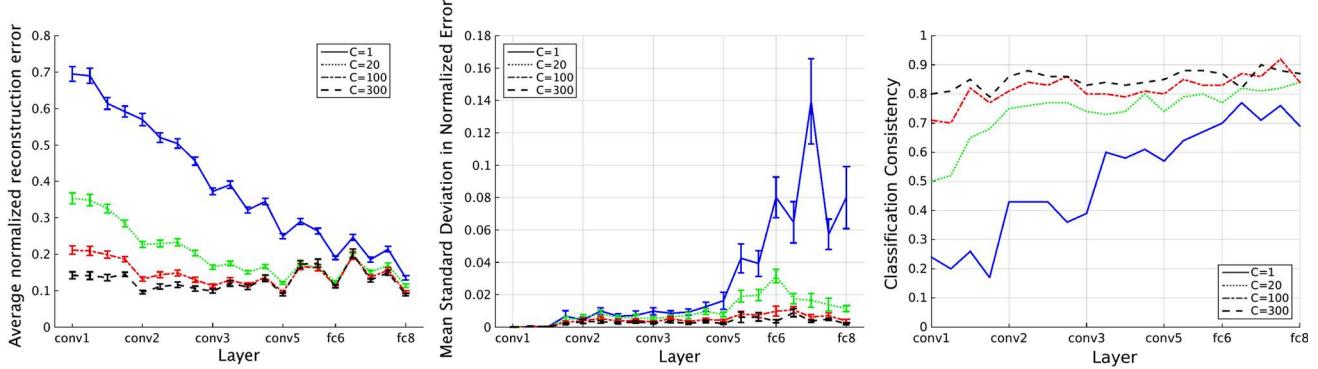
The reconstruction accuracy is assessed in three ways: reconstruction error, consistency with respect to different random initializations, and classification consistency.

Reconstruction error Similar to Sect. 5.1, the reconstruction error (3) is averaged over the first 100 images in the ILSVRC 2012 challenge validation images (Russakovsky et al. 2015) (these images were not used to train the CNNs). The experiment is repeated for all the layers of AlexNet and for different values of the parameter C to assess its effect. The resulting average errors are reported in Fig. 6 (left panel).

CNNs such as AlexNet are significantly larger and deeper than the CNN implementations of HOG and DSIFT. Therefore, it seems that the inversion problem should be considerably harder for them. Instead, comparing the results in Fig. 6 to the ones in Table 2 indicates that CNNs are, in fact, not much more difficult to invert than HOG. In particular, for a sufficiently large value of C, the reconstruction error can be maintained in the range 10–20 %, including for the deepest layers. Therefore, the non-linearities in the CNN seem to be rather benign, which could explain why SGD can learn these models successfully. Using a stronger regularization (small C) significantly deteriorates the quality of the reconstructions from earlier layers of the network. At the same time, it has little to no effect on the reconstruction quality of deeper layers. Since, as verified below, a strong regularization significantly improves the interpretability of resulting pre-images, it should be used for these layers.

Consistency of multiple pre-images As explained earlier, different random initializations are expected to result in different reconstructions. However, this diversity should reflect genuine representation ambiguities and invariances rather than the inability of the local optimization method to escape bad local optima. To verify that this is the case, the standard deviation of the reconstruction error (3) is computed from 24 different reconstructions obtained from the same reference image and 24 different initializations. The experiment is repeated using as reference the first 24 images in the ILSVRC 2012 validation dataset and the average standard deviation of the reconstruction errors is reported in Fig. 6 (middle panel). This figure shows that, for the values of C except $C = 1$, all pre-images have very similar reconstruction errors, with standard deviation of around 0.02 or less. Thus in all but the very deep layers all pre-images can be treated as equally good from the viewpoint of reconstruction error. In the next paragraph, we show that even for very deep layers, pre-images are substantially equivalent from the viewpoint of classification consistency.

Classification consistency One question that may arise is whether a reconstruction error of 20 %, or even 10 %, is sufficiently small to validate the visualizations. To answer this question, Fig. 6 (right panel) reports the *classification consistency* of the reconstructions. Here "classification consistency" is the fraction of reconstructed pre-images $\mathbf{x}$ that the CNN associates with the same class label as the refer-

Fig. 6 Quality of CNN inversions. The plots report three reconstruction quality indices for the CNN inversion method applied to different layers of AlexNet and for different reconstruction fidelity strengths C. The *left plot* shows the average reconstruction error averaged using 100 ImageNet ILSVRC validation images as reference images (the *error bars* show the 95 % confidence interval for the mean). The *middle plot* reports the standard deviation of the reconstruction error obtained from 24 different random initializations per reference image, and then averaged over 24 such images. The *right plot* reports the percentage of reconstructions that are associated by the CNN to the same class as their reference image (Color figure online)
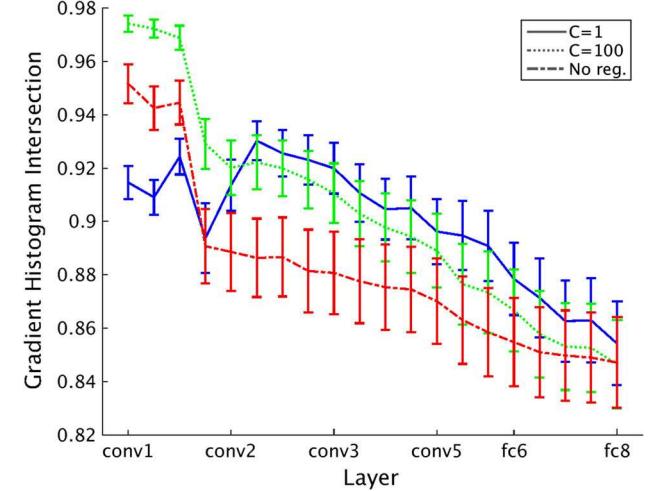
ence image x_0. This value, which would be equal to 1 for perfect reconstructions, measures whether imperfections in the reconstruction process are small enough to not affect the "meaning" of the image from the viewpoint of the CNN.

As it may be expected, classification consistency results show a trend similar to the reconstruction error, where better reconstructions are obtained for small amounts of regularization for the shallower layers, whereas deep layers can afford much stronger regularization. It is interesting to note that even visually odd inversions of deep layers such as those shown in Fig. 1 or 9 are classification consistent with their reference image, demonstrating the high degree of invariance in such layers. Finally, we note that, by choosing the correct amount of regularization, the classification consistency of the reconstructions can be kept above 0.8 in most cases, validating the visualizations.

5.2.3 Naturalness

One of our contributions is the idea that imposing even simple naturalness priors on the reconstructed images improves their interpretability. Since interpretability is a purely subjective attribute, in this section we conduct a small human study to validate this idea. Before that, however, we check whether regularization works as expected and produces images that are statistically closer to natural ones.

Natural image statistics Natural images are well known to have certain statistical regularities; for example, the magnitude of image gradients have an exponential distribution (Huang and Mumford 1999). Here we check whether regularized pre-images are more "natural" by comparing them to natural images using such statistics. To do so, we compute the histogram of gradient magnitudes for the 100 ImageNet


Fig. 7 Mean histogram intersection similarity for the gradient statistics of the reference image x_0 and the reconstructed pre-image x for different layers of AlexNet and values of the parameter C (only a few such values are reported to reduce clutter). *Error bars* show the 95 % confidence interval of the mean histogram intersection similarity (Color figure online)

ILSVRC reference images used above and for their AlexNet inversions, for different values of the parameter C. Then the original and reconstructed histograms are compared using histogram intersection and their similarity is reported in Fig. 7. As before, a small amount of regularization is clearly preferable for shallow layers, and a stronger amount is clearly better for intermediate ones. However, the difference is not all that significant for the deepest layers, which are therefore best analyzed in terms of their interpretability.
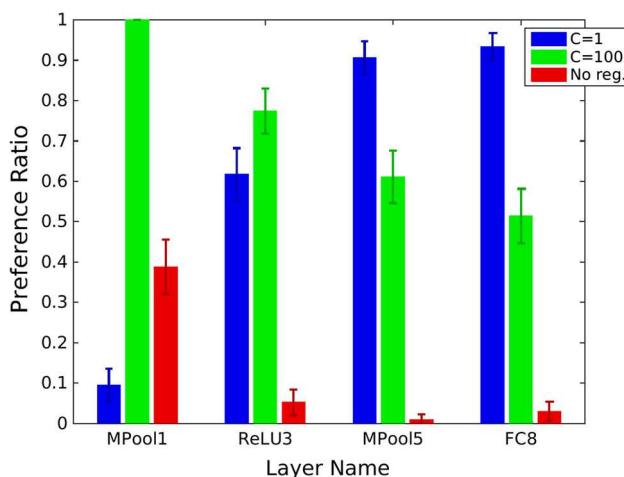
Interpretability In this experiment, pre-images were obtained using the first 25 ILSVRC 2012 validation images as reference. Inversions were obtained from the mpool1, relu3,

mpool5, and fc8 layers of AlexNet for three regularizations strengths: (a) no regularization ($C = \infty$), (b) weak regularization ($C = 100$), and (c) strong regularization ($C = 1$). Thus we have three pre-images per layer per reference image. In a user study, each subject was shown two randomly picked regularization settings for a layer and reference image. Each subject was asked to select the image that was more interpretable ("whose content is more easily understood"). We conducted this study with 13 human subjects who were not familiar with this line of research, or not familiar with computer vision at all. The first five votes from each subject were discarded to allow them to become familiar with the task. The ordering of images and layers was randomized independently for each subject. Uniform random sampling between the three regularization strengths ensures that no regularization strength dominates the screen or even one side of the screen.

Figure 8 shows the fraction of time a certain regularization strength was found to produce more interpretable results for a given AlexNet layer. Based on these results, at least a small amount of regularization is always preferable for interpretability. Furthermore, strong regularization is highly desirable for very deep layers.

5.2.4 Inversion of Different Layers

Having established the legitimacy of the inversions, next we study qualitatively the reconstructions obtained from different layers of the three CNNs for a test image ("red fox"). In particular, Fig. 9 shows the reconstructions obtained from each layer of AlexNet and Figs. 10 and 11 do the same for all the linear (convolutional and fully connected) layers of VGG-M and VGG-VD.


Fig. 8 Fraction of times a certain regularization parameter value C was found to be more interpretable than one of the other two by humans looking at pre-images form different layers of AlexNet (Color figure online)
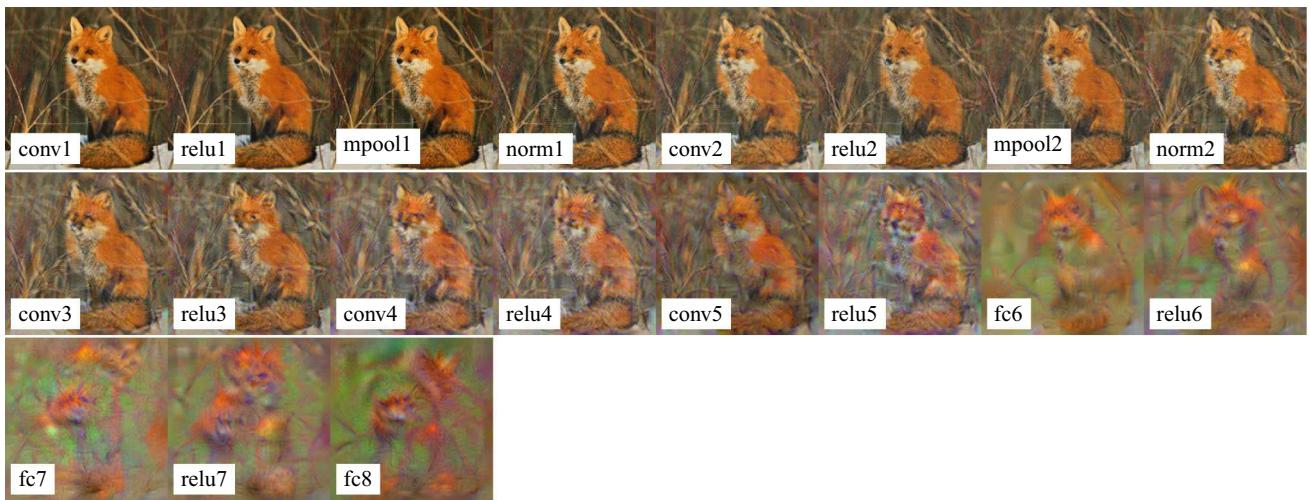
The progression is remarkable. The first few layers of all the networks compute a code of the image that is nearly exactly invertible. All the layers prior to the fully-connected ones preserve instance-specific details of the image, although with increasing fuzziness. The 4096-dimensional fully connected layers discard more geometric as well as instance-specific information as they invert back to a *composition of parts, which are similar but not identical to the ones found in the original image*. Unexpectedly, even the very last layer, fc8, whose 1000 components are in principle category predictors, still appears to preserve some instance-specific details of the image.

Comparing different architectures, VGG-M reconstructions are sharper and more detailed than the ones obtained from AlexNet, as it may be expected due to the denser and higher dimensional filters used here (compare for example conv4 in Figs. 9 and 10). VGG-VD emphasizes these differences more. First, abstractions are achieved much more gradually in this architecture: conv5_1, conv5_2 and conv5_3 reconstructions resemble the reconstructions from conv5 in AlexNet and VGG-M, despite the fact that there are three times more intermediate layers in VGG-VD. Nevertheless, fine details are preserved more accurately in the deep layers in this architecture (compare for example, the nose and eyes of the fox in conv5 in VGG-M and conv5_1 – conv5_3 in VGG-VD).
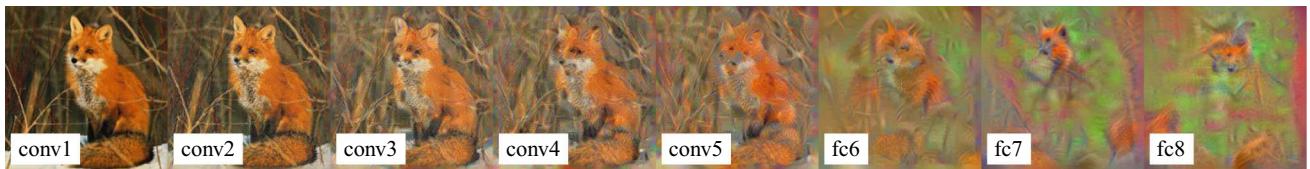
Another difference we noted here, as well as in Fig. 18, is that reconstructions from deep VGG-VD layers are often more zoomed in compared to other networks (see for example the "abstract art" and "monkey" reconstructions from fc7 in Fig. 12 and, for activation maximization, in Fig. 18). The preference of VGG-VD for large, detailed object occurrences may be explained by its better ability to represent fine-grained object details, such as textures.

5.2.5 Reconstruction Ambiguity and Invariances
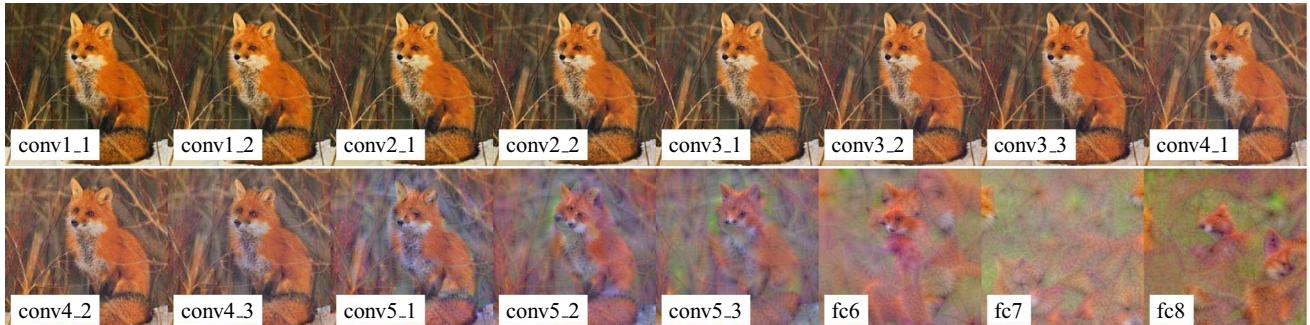
Figure 12 examines the invariances captured by the VGG-VD codes by comparing multiple reconstructions obtained from several deep layers. A careful examination of these images reveals that the codes capture progressively larger deformations of the object. In the "spoonbill" image, for example, conv5_2 reconstructions show slightly different body poses, evident from the different leg configurations. In the "abstract art" test image, a close examination of the pre-images reveals that, while the texture statistics are preserved well, the instance-specific details are in fact completely different in each image: the location of the vertexes, the number and orientation of the edges, and the color of the patches are not the same at the same locations in different images. This case is also remarkable as the training data for VGG-VD, i.e. ImageNet ILSVRC, does not contain any such pattern suggesting that these codes are indeed rather generic. Inversions

Fig. 9 AlexNet inversions (all layers) from the representation of the "red fox" image obtained from each layer of AlexNet (Color figure online)

**Fig. 10** VGG-M inversions (selected layers) (Color figure online)

Fig. 11 VGG-VD-16 inversions (selected layers) (Color figure online)
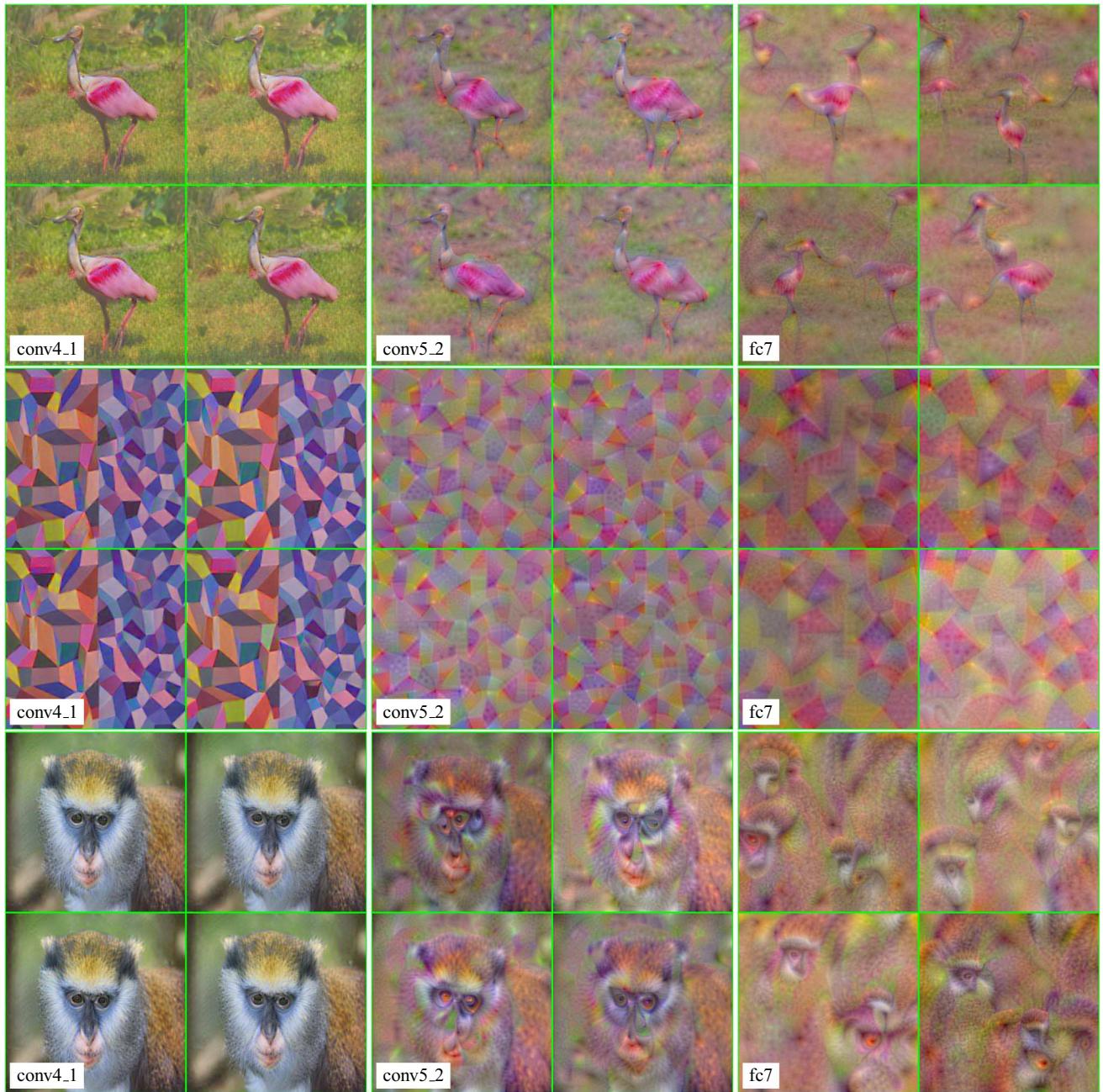
from fc7 result in multiple copies of the object/parts at different positions and scales for the "spoonbill" and "monkey" cases. For the "monkey" and "abstract art" cases, inversions from fc7 appear to result in slightly magnified versions of the pattern: for instance, the reconstructed monkey's eye is about 20 % larger than in the original image; and the reconstructed patches in "abstract art" are about 70 % larger than in the original image. The preference for reconstructing larger object scales seems to be typical of VGG-VD (see also Fig. 18).

Note that all these reconstructions and the original images are very similar from the viewpoint of the CNN representation; we conclude in particular that the deepest layers find the original images and a number of scrambled parts to be equivalent. This may be considered as another type of nat-

ural confounder for CNNs alternative to those discussed in Nguyen et al. (2015).

5.2.6 Reconstruction Biases

It is interesting to note that some of the inverted images have large green regions (for example see Fig. 11 fc6 to fc8). This property is likely to be intrinsic to the networks and not induced, for example, by the choice of natural image prior, the effect of which is demonstrated in Fig. 3 and Sect. 3.2.3. The prior only encourages smoothness as it is equivalent (for $\beta = 2$) to penalizing high-frequency components of the reconstructed image. Importantly, the prior is applied to all color channels equally. When gradually removing the

Fig. 12 For three test images, "spoonbill", "abstract art", and "monkey", we generate four different reconstructions from layers conv4_1, conv5_2, and fc7 in VGG-VD (Color figure online)
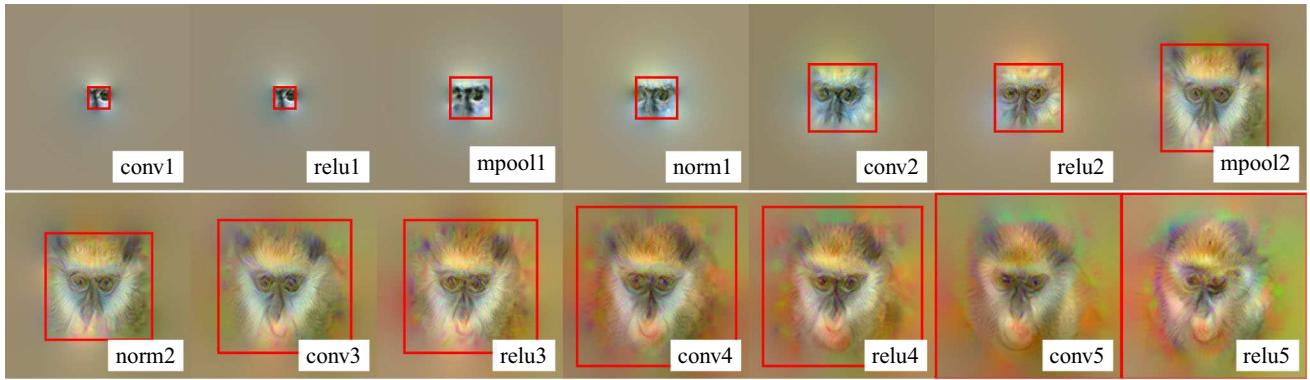
prior, random high-frequency components dominate and it is harder to discern a human-interpretable signal.

5.2.7 Inversion from Selected Representation Components
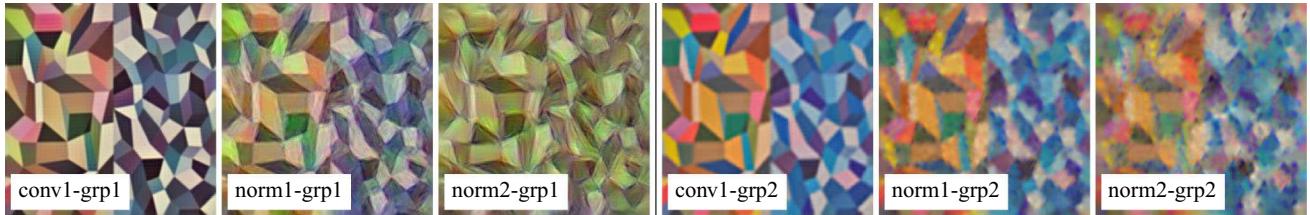
It is also possible to examine reconstructions obtained from subsets of neural responses in different CNN layers. Figure 13 explores the *locality* of the codes by reconstructing a central 5×5 patch of features in each layer. The regularizer

encourages portions of the image that do not contribute to the neural 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 red box in the image.

Finally, Fig. 14 reconstructs images from two different subsets of feature channels for CaffeRef. These subsets are induced by the fact that the first several layers (up to norm2)

Fig. 13 Reconstructions of the "monkey" image from a central 5×5 window of feature responses in the convolutional layers of CaffeRef. The *red box* marks the overall receptive field of the 5×5 window (Color figure online)


Fig. 14 CNN neural streams. Reconstructions of the "abstract" test image from either of the two neural streams in CaffeRef (Color figure online)

of the CaffeRef architecture are trained to have blocks of independent filters (Krizhevsky et al. 2012). Reconstructing individually from each subset clearly shows that one group is tuned towards color information, whereas the second one is tuned towards sharper edges and luminance components. Remarkably, this behavior emerges spontaneously in the learned network.

6 Visualization by Activation Maximization

In this section the activation maximization method is applied to classical and CNN representations.

6.1 Classical Representations

For classical representations, we use activation maximization to visualize HOG templates. Let $\Phi_{\text{HOG}}(\mathbf{x})$ denote the HOG descriptor of a gray scale image $\mathbf{x} \in \mathbb{R}^{H \times W}$; a *HOG template* is a vector $\mathbf{w}$, usually learned by a latent SVM, that defines a scoring function $\Phi(\mathbf{x}) = \langle \mathbf{w}, \Phi_{\text{HOG}}(\mathbf{x}) \rangle$ for a particular object category. The function $\Phi(\mathbf{x})$ can be interpreted as a CNN consisting of the HOG CNN $\Phi_{\text{HOG}}(\mathbf{x})$ followed by a linear projection layer of parameter $\mathbf{w}$. The output of $\Phi(\mathbf{x})$ is a scalar, expressing the confidence that the image $\mathbf{x}$ contains the target object class.

In order to visualize the template $\mathbf{w}$ using activation maximization, the loss function $-\langle \Phi(\mathbf{x}), \Phi_0 \rangle / Z$ of Eq. (4) is plugged in the objective function (1). Since in this case $\Phi(\mathbf{x})$ is a scalar function, the reference vector Φ_0 is also a scalar, which is set to 1. The normalization constant Z is set to

$$Z = M\rho^2 \tag{9}$$

where $\rho = \max\{H, W\}$ and M is an estimate of range of $\Phi(\mathbf{x})$, obtained as $M = \langle |\mathbf{w}|, \Phi_{\text{HOG}}(\mathbf{x}) \rangle$ where $|\mathbf{w}|$ is the element wise absolute value of $\mathbf{w}$ and $\mathbf{x}$ is set to a white noise sample. The method is used to visualize the DPM (v5) models (Girshick et al. 2010) trained on the VOC2010 (Everingham et al. 2010) dataset.
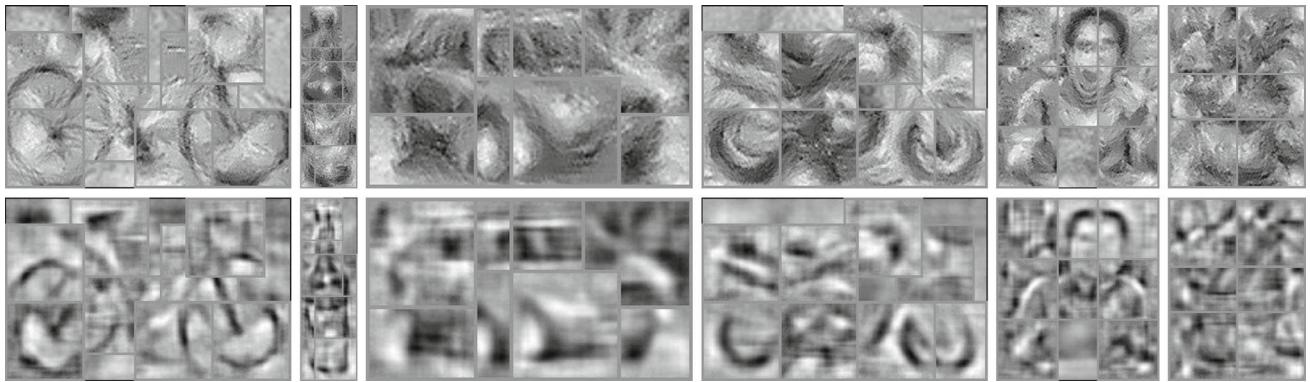
These visualizations are compared to the ones obtained in the analogous experiment by Vondrick et al. (2013, Fig. 14) using HOGgle. An important difference is that HOGgle does not perform activation maximization, but rather inversion and returns an approximate pre-image $\Phi_{\text{HOG}}^{-1}(\mathbf{w}_+)$, where $\mathbf{w}_+ = \max\{0, \mathbf{w}\}$ is the rectified template. Strictly speaking, inversion is not applicable here because the template $\mathbf{w}$ is *not* a HOG descriptor. In particular, $\mathbf{w}$ contains negative components which HOG descriptors do not contain. Even after rectification, there is usually no image such that $\Phi_{\text{HOG}}(\mathbf{x}) = \mathbf{w}_+$. By contrast, activation maximization is principled because it works on top of the detector scoring function; in this manner it can correctly reflect the effect of both positive and negative components in $\mathbf{w}$.

Fig. 15 Visualization of DPMv5 HOG models using activation maximization (*top*) and HOGgle (*bottom*). Each model comprises a "root" filter overlaid with several part filters. From *left* to *right*: Bicycle (Component 2), Bottle (5), Car (4), Motorbike (2), Person (1), Potted Plant (4)

The visualizations of five DPMs using activation maximization and HOGgle are shown in Fig. 15. Note that the DPMs are hierarchical, and consist of a root object template and several higher-resolution part templates. For simplicity, each part is processed independently, but activation maximization could be applied to the composition of all the parts to remove the seams between them.

Compared to HOGgle, activation maximization reconstructs finer details, as can be noted for parts such as the bicycle wheel and bottle top. On the other hand, HOGgle reconstructions contain stronger and straighter edges (see for example the car roof). The latter may be a result of HOGgle using a restricted dictionary computed from natural images whereas our approach uses a more generic smoothness prior.

6.2 CNN Representations
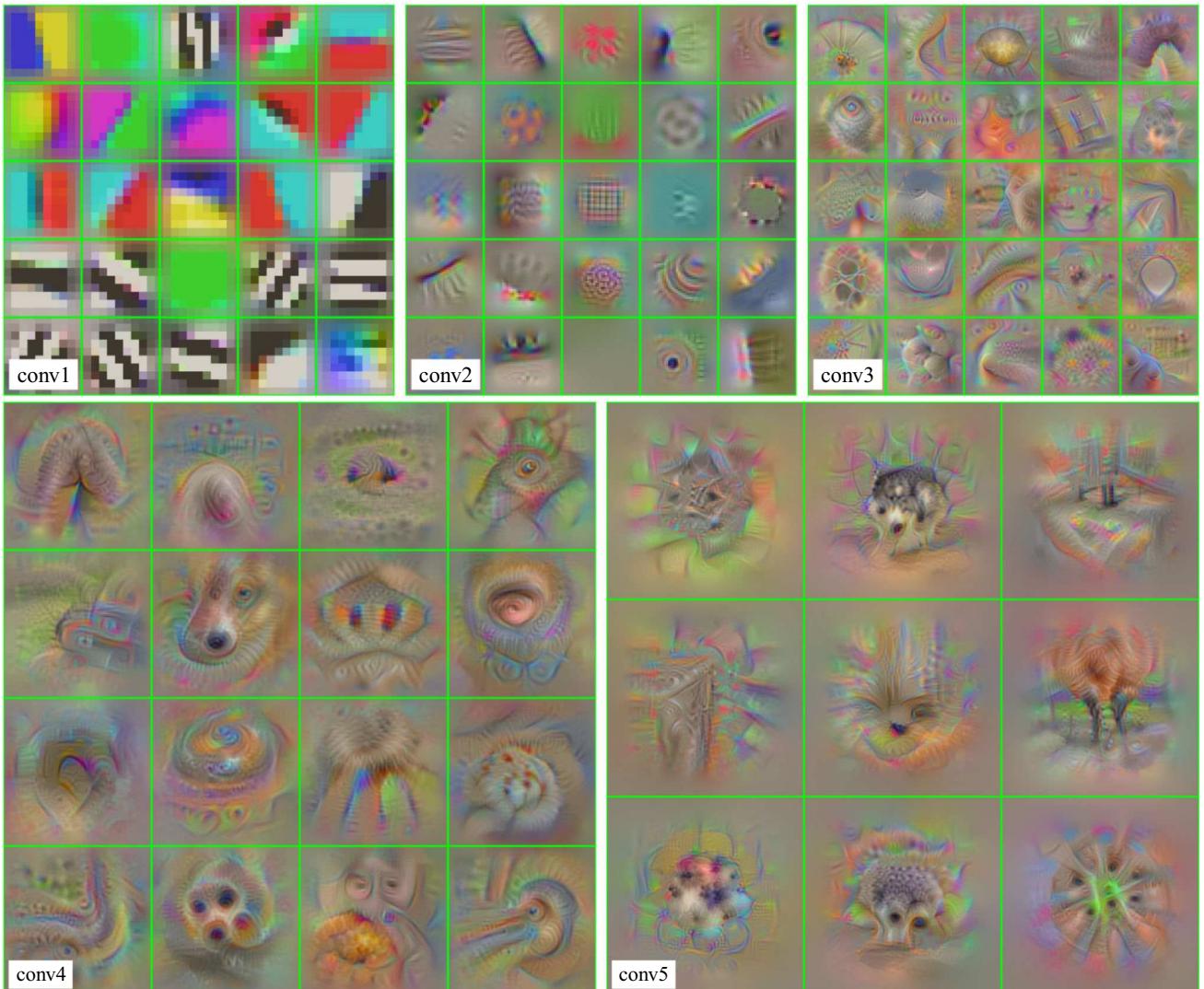
Next, the activation maximization method is applied to the study of deep CNNs. As before, the inner-product loss Eq. (4) is used in the objective (1), but this time the reference vector Φ_0 is set to the one-hot indicator vector of the representation component being visualized. It was not possible to find an architecture independent normalization constant Z in Eq. (4) as different CNNs have very different ranges of neuron output. Instead Z is calculated in the same way as Eq. (9) where M is the maximum value achieved by the representation component in the ImageNet ILSVRC 2012 validation data and ρ the size of the component receptive field, as reported in Table 1. As in Sect. 5.2 the jitter amount T in (Sect. 3.2.3) is set to a fourth of the stride of the feature and all the other parameters are set as described in Sect. 3.3 (including $C = 1$).

Figure 16 shows the visual patterns obtained by maximally activating the few components in the convolutional layers of VGG-M. Similarly to Zeiler and Fergus (2014) and Yosinksi et al. (2015), the complexity of the patterns increases substantially with depth. The first convolutional layer conv1 captures colored edges and blobs, but the complexity of the patterns generated by conv3, conv4 and conv5 is remarkable. While some of these pattern do evoke objects or object parts, it remains difficult to associate to them a clear semantic interpretation (differently from Yosinksi et al. 2015 we prefer to avoid hand-picking semantically interpretable filters). This is not entirely surprising given that the representation is distributed and activations may need to be combined to form a meaning. Experiments from AlexNet yielded entirely analogous if a little blurrier results.

Figure 17 shows the patterns obtained from VGG-VD. The complexity of the patterns build up more gradually than for VGG-M and AlexNet. Qualitatively, the complexity of the stimuli in conv5 in AlexNet and VGG-M seems to be comparable to conv4_3 and conv5_1 in VGG-VD. conv5_2 and conv5_3 do appear to be significantly more complex, however. A second observation is that the reconstructed colors tend to be much more saturated, probably due to the lack of normalization layers in the architecture. Thirdly, we note that reconstructions contain significantly more fine-grained details, and in particular tiny blob-like structures, which probably activate very strongly the first very small filters in the network.

Figure 18 repeats the experiment from Simonyan et al. (2014), more recently reprised by Yosinksi et al. (2015) and Mordvintsev et al. (2015), and maximizes the component of fc8 that correspond to a given class prediction. Four classes are considered: two similar animals ("black swan" and "goose"), a different one ("tree frog"), and an inanimate object ("cheeseburger"). We note that in all cases it is easy to identify several parts or even instances of the target object class. However, reconstructions are fragmented and scrambled, indicating that the representations are highly invariant to occlusions and pose changes. Secondly, reconstructions from VGG-M are considerably sharper than the ones obtained from AlexNet, as could be expected. Thirdly, VGG-VD-16 differs significantly from the other two architectures. Colors are more "washed out", which we impute to the lack of normalization in the architecture as for Fig. 17.

Fig. 16 Activation maximization of the first filters of each convolutional layer in VGG-M (Color figure online)

Reconstructions tends to focus on much larger objects; for example, the network clearly captures the feather pattern of the bird as well as the rough skin of the frog.

Finally, Fig. 19 shows multiple pre-images of a few representation components obtained by starting activation maximization from different random initializations. This is analogous to Fig. 12 and is meant to probe the invariances in the representation. The variation across the four pre-images is a mix of geometric and style transformations. For example, the second neuron appears to represent four different variants of a wheel of a vehicle.

7 Visualization by Caricaturization

Our last visualization is inspired by Google's Inceptionism (Mordvintsev et al. 2015). It is similar to activation maxi-

mization (Sect. 6) and in fact uses the same formulation for Eq. (1) with the inner-product loss Eq. (4). However, there are two key differences. First, the target mask is now set to
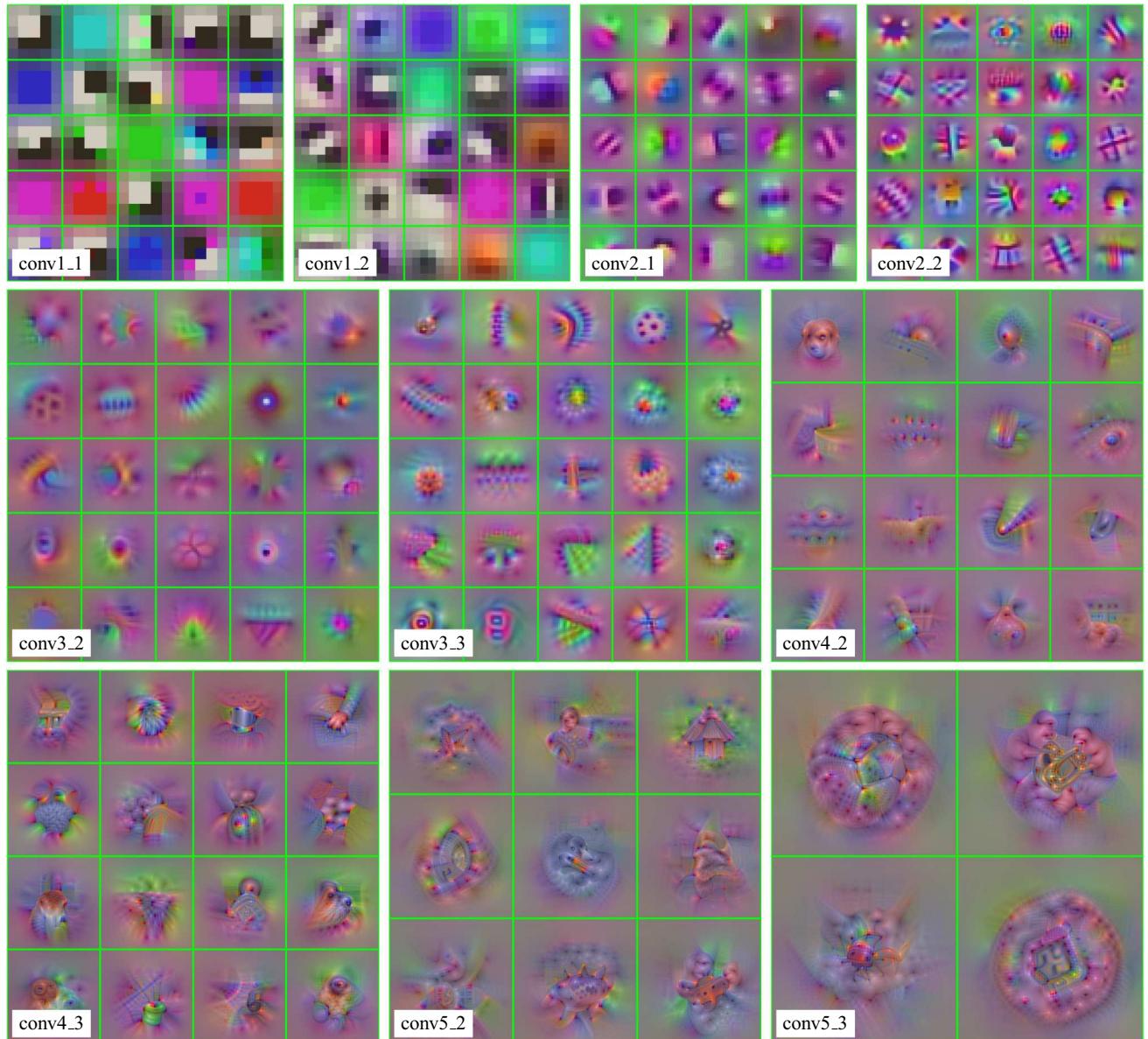
$$\Phi_0 = \max\{0, \Phi(\mathbf{x}_0)\}$$

where $\mathbf{x}_0$ is a reference image and the normalization factor Z is set to $\|\Phi_0\|^2$. Second, the optimization is started from the image $\mathbf{x}_0$ itself.

The idea of this visualization is to exaggerate any pattern in $\mathbf{x}_0$ that is active in the representation $\Phi(\mathbf{x}_0)$, hence creating a "caricature" of the image according to this model. Furthermore, differently from activation maximization, this visualization works with combinations of multiple activations instead of individual ones.

Figure 20 shows the caricatures of the "red-fox" image obtained from the different layers of VGG-M. Applied to the

Fig. 17 Activation maximization of the first filters for each convolutional layer in VGG-VD-16 (Color figure online)
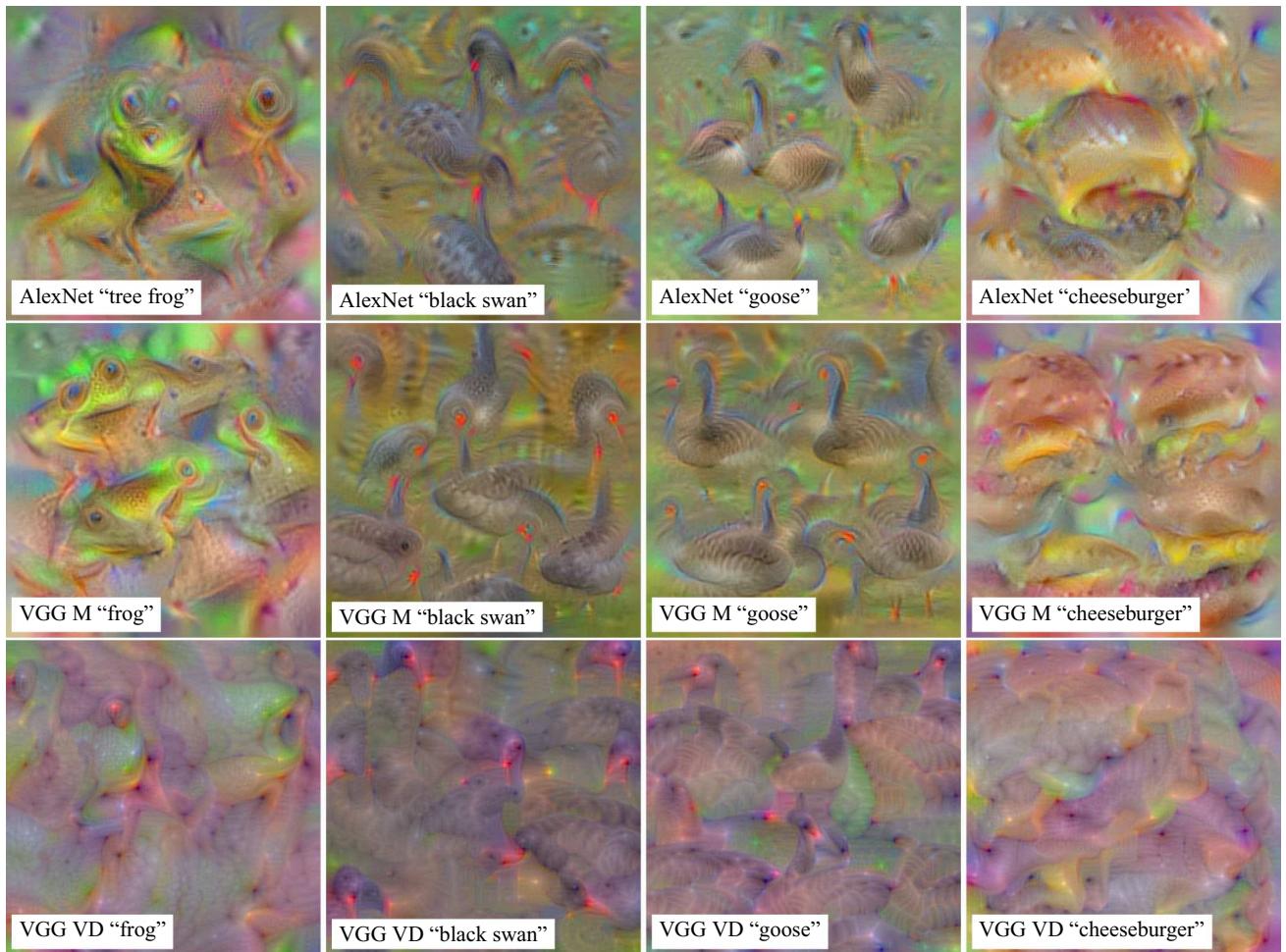
first block of layers, the procedure simply saturates the color. conv2 appears to be tuned to long, linear structures, conv4 to round ones, and conv5 to the head (part) of the fox. The fully connected layers generate mixtures of fox heads, including hallucinating several in the background, as already noted in Mordvintsev et al. (2015). Figure 21 shows the caricatures obtained from selected layers of VGG-M and VGG-VD, with similar results.

8 Summary
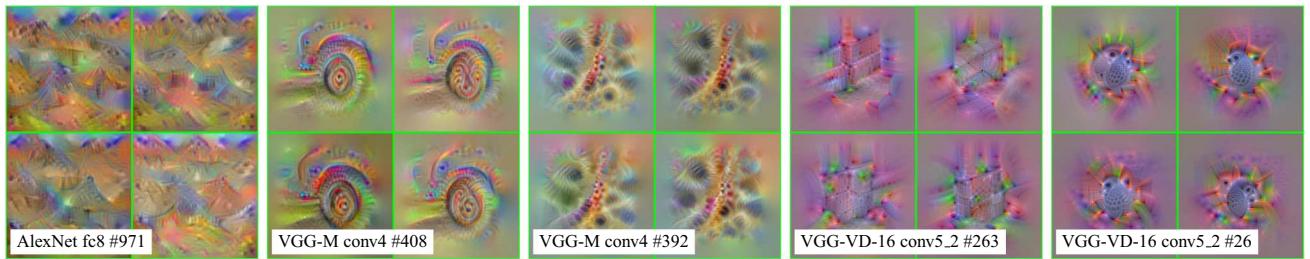
There is a growing interest in methods that can help us understand computer vision representations, and in partic-ular representations learned automatically from data such as those constructed by deep CNNs. Recently, several authors have proposed complementary visualization techniques to do so. In this manuscript we have extended our previous work on inverting representations using natural pre-images to a uni-fied framework that encompasses several visualization types. We have then experimented with three such visualizations (inversion, activation maximization, and caricaturization), and used those to probe and compare standard classical rep-resentations, and CNNs.

The robustness of our visualization method has been assessed quantitatively in the case of the inversion problem by comparing the output of our approach to earlier feature inversion techniques. The most important results, however,

Fig. 18 Activation maximization for the second to last layer of AlexNet, VGG-M, VGG-VD-16 for the classes "frog", "black swan", "goose", and "vending machine". The second to last layer codes directly for different classes, before softmax normalization (Color figure online)
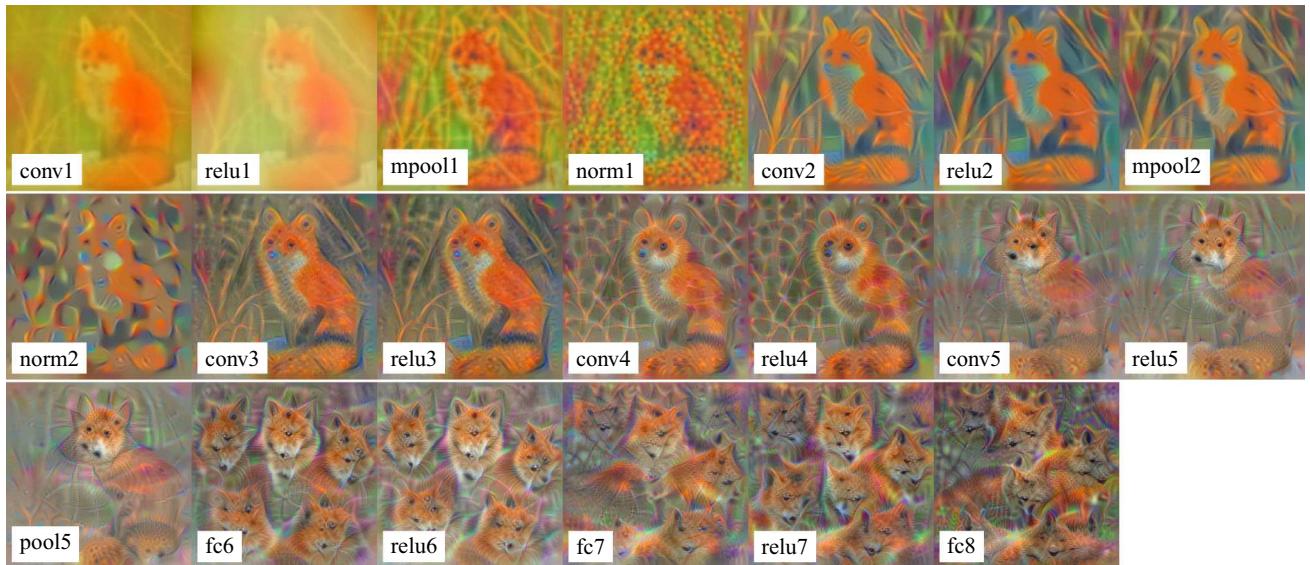

Fig. 19 Activation maximization with four different initializations (Color figure online)

emerged from an analysis of the visualizations obtained from deep CNNs; some of these are: the fact that photometrically accurate information is preserved deep down in CNNs, that even very deep layers contain instance-specific information about objects, that intermediate convolutional layers capture local invariances to pose and fully connected layers to large variations in the object layouts, that individual CNN components code for complex but, for the most part, not semantically-obvious patterns, and that different CNN layers appear to capture different types of structures in images, from lines and curves to parts.

We believe that these visualization methods can be used as direct diagnostic tools to further research in CNNs. For example, an interesting problem is to look for semantically-meaningful activation patterns in deep CNN layers (given that individual responses are often not semantic); inversion, or variants of activation maximization, can be used to validate such activation patterns by means of visualizations.

Fig. 20 Caricatures of the "red fox" image obtained from the different layers in VGG-M (Color figure online)


Fig. 21 Caricatures of a number of test images obtained from the different layers of VGG-M (conv3, conv4, conv5, fc6, fc8) and VGG-VD (conv5_3 and fc8) (Color figure online)

References

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Clarendon Press.

Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. In Proceedings of BMVC.

Chen, Y., Ranftl, R., & Pock, T. (2014). A bi-level view of inpainting-based image compression. In *Proceedings of computer vision winter workshop*.

Csurka, G., Dance, C. R., Dan, L., Willamowski, J., & Bray, C. (2004). Visual categorization with bags of keypoints. In *Proceedings of ECCV workshop on statistical learning in computer vision*.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*.

d'Angelo, E., Alahi, A., & Vandergheynst, P. (2012). Beyond bits: Reconstructing images from local binary descriptors. In *ICPR* (pp. 935–938).

Dosovitskiy, A., & Brox, T. (2015). Inverting convolutional networks with convolutional networks. *CoRR*. arXiv:1506.02753.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learninig Research*, *12*, 2121–2159.

Erhan, D., Bengio, Y., Courville, A., & Vincent, P. (2009). *Visualizing higher-layer features of a deep network*. Technical report (Vol. 1341). Montreal: University of Montreal.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The PASCAL visual object classes challenge 2010 (VOC2010) results. http://host.robots.ox.ac.uk/pascal/VOC/voc2010/results/index.html. Accessed 10 Apr 2016.

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

Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). A neural algorithm of artistic style. *CoRR*. arXiv:1508.06576.

Gatys, L. A., Ecker, A. S., & Bethge, M. (2015). Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks. In *NIPS proceedings*.

Girshick, R. B., Felzenszwalb, P. F., & McAllester, D. (2010). *Discriminatively trained deformable part models, release 5*. http://people.cs.uchicago.edu/~rbg/latent-release5/. Accessed 10 April 2016.

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

Huang, W. J., & Mumford, D. (1999). Statistics of natural images and models. In *Proceedings of CVPR*.

Jégou, H., Douze, M., Schmid, C., & Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *CVPR*.

Jensen, C. A., Reed, R. D., Marks, R. J., El-Sharkawi, M., Jung, J. B., Miyamoto, R., et al. (1999). Inversion of feedforward neural networks: Algorithms and applications. *Proceedings of the IEEE*, *87*, 9.

Jia, Y. (2013). Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/. Accessed 14 Nov 2014.

Julesz, B. (1981). Textons, the elements of texture perception, and their interactions. *Nature*, *290*(5802), 91–97.

Kato, H., & Harada, T. (2014). Image reconstruction from bag-of-visual-words. In *CVPR*.

Krishnan, D., & Fergus, R. (2009). Fast image deconvolution using hyper-laplacian priors. In *NIPS*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*.

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

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

Linden, A., & Kindermann, J. (1989). Inversion of multilayer nets. In *Proceedings of international conference on neural networks*.

Lowe, D. G. (1999). Object recognition from local scale-invariant features. In *ICCV*.

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

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

Mahendran, A., & Vedaldi, A. (2015). Understanding deep image representations by inverting them. In *Proceedings of CVPR*.

Mordvintsev, A., Olah, C., & Tyka, M. (2015). *Inceptionism: Going deeper into neural networks*. http://googleresearch.blogspot.co.uk/2015/06/inceptionism-going-deeper-into-neural.html. Accessed 17 Oct 2015.

Nguyen, A., Yosinski, J., & Clune, J. (2015). Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of CVPR*.

Nowak, E., Jurie, F., & Triggs, B. (2006). Sampling strategies for bag-of-features image classification. In *ECCV*.

Perronnin, F., & Dance, C. (2006). Fisher kernels on visual vocabularies for image categorizaton. In *CVPR*.

Portilla, J., & Simoncelli, E. P. (2000). A parametric texture model based on joint statistics of complex wavelet coefficients. *IJCV*, *40*, 49–70.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet large scale visual recognition challenge. *IJCV*, *115*(3), 211–252. doi:10.1007/s11263-015-0816-y.

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *CoRR*. arXiv:1312.6229.

Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In *Proceedings of ICLR*.

Sivic, J., & Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *ICCV*

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., et al. (2014). Intriguing properties of neural networks. In *Proceedings of ICLR*.

Tatu, A., Lauze, F., Nielsen, M., & Kimia, B. (2011). Exploring the representation capabilities of the HOG descriptor. In *ICCV workshop*.

Várkonyi-Kóczy, A. R., & Rövid, A. (2005). Observer based iterative neural network model inversion. In *IEEE international conference on fuzzy systems*.

Vedaldi, A. (2007). *An open implementation of the SIFT detector and descriptor*. Technical report 070012. Los Angeles: UCLA CSD.

Vedaldi, A., & Lenc, K. (2014). MatConvNet: CNNs for MATLAB. http://www.vlfeat.org/matconvnet/. Accessed 17 Oct 2015.

Vondrick, C., Khosla, A., Malisiewicz, T., & Torralba, A. (2013). HOGgles: Visualizing object detection features. In *ICCV*.

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., & Gong, Y. (2010). Locality-constrained linear coding for image classification. In *CVPR*.

Weinzaepfel, P., Jégou, H., & Pérez, P. (2011). Reconstructing an image from its local descriptors. In *CVPR*.

Williams, R. J. (1986). Inverting a connectionist network mapping by back-propagation of error. In *Proceedings of CogSci*.

Yang, J., Yu, K., & Huang, T. (2010). Supervised translation-invariant sparse coding. In *CVPR*.

Yosinksi, J., Clune, J., Nguyen, A., Fuchs, T., & Lipson, H. (2015). Understanding neural networks through deep visualization. In *Proceedings of ICML workshop*.

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. *CoRR*. arXiv:1212.5701.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *ECCV*.

Zhou, X., Yu, K., Zhang, T., & Huang, T. S. (2010). Image classification using super-vector coding of local image descriptors. In *ECCV*.

Zhu, C., Byrd, R. H., Lu, P., & Nocedal, J. (1997). Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, *23*(4), 550–560.

Zhu, S. C., Wu, Y., & Mumford, D. (1998). Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *IJCV*, *27*(2), 4.