Inverting Visual Representations with Convolutional Networks

Alexey Dosovitskiy Thomas Brox
University of Freiburg
Freiburg im Breisgau, Germany
{dosovits,brox}@cs.uni-freiburg.de

Abstract

Feature representations, both hand-designed and learned ones, are often hard to analyze and interpret, even when they are extracted from visual data. We propose a new approach to study image representations by inverting them with an up-convolutional neural network. We apply the method to shallow representations (HOG, SIFT, LBP), as well as to deep networks. For shallow representations our approach provides significantly better reconstructions than existing methods, revealing that there is surprisingly rich information contained in these features. Inverting a deep network trained on ImageNet provides several insights into the properties of the feature representation learned by the network. Most strikingly, the colors and the rough contours of an image can be reconstructed from activations in higher network layers and even from the predicted class probabilities.

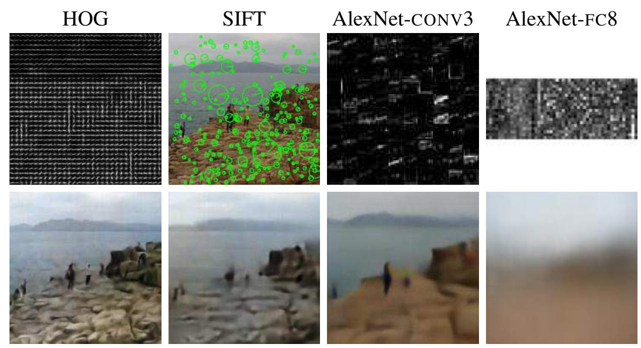

Figure 1: We train convolutional networks to reconstruct images from different feature representations. **Top row:** Input features. **Bottom row:** Reconstructed image. Reconstructions from HOG and SIFT are very realistic. Reconstructions from AlexNet preserve color and rough object positions even when reconstructing from higher layers.

1. Introduction

A feature representation useful for pattern recognition tasks is expected to concentrate on properties of the input image which are important for the task and ignore the irrelevant properties of the input image. For example, hand-designed descriptors such as HOG [3] or SIFT [17], explicitly discard the absolute brightness by only considering gradients, precise spatial information by binning the gradients and precise values of the gradients by normalizing the histograms. Convolutional neural networks (CNNs) trained in a supervised manner [14, 13] are expected to discard information irrelevant for the task they are solving [28, 19, 22].

In this paper we propose a new approach to analyze which information is preserved by a feature representation and which information is discarded. We train neural networks to invert feature representations in the following sense. Given a feature vector, the network is trained to predict the *expected pre-image*, that is, the (weighted) average of all natural images which could have produced the given feature vector. The content of this expected pre-image shows image properties which can be confidently inferred from the feature vector. The amount of blur corresponds to the level of invariance of the feature representation. We obtain further insights into the structure of the feature space, as we apply the networks to perturbed feature vectors, to interpolations between two feature vectors, or to random feature vectors.

We apply our inversion method to AlexNet [13], a convolutional network trained for classification on ImageNet, as well as to three widely used computer vision features: histogram of oriented gradients (HOG) [3, 7], scale invariant feature transform (SIFT) [17], and local binary patterns (LBP) [21]. The SIFT representation comes as a non-uniform, sparse set of oriented keypoints with their corresponding descriptors at various scales. This is an additional challenge for the inversion task. LBP features are not differentiable with respect to the input image. Thus, existing methods based on gradients of representations [19] could not be applied to them.

1.1. Related work

Our approach is related to a large body of work on inverting neural networks. These include works making use of backpropagation or sampling [15, 16, 18, 27, 9, 25] and, most similar to our approach, other neural networks [2]. However, only recent advances in neural network architectures allow us to invert a modern large convolutional network with another network.

Our approach is not to be confused with the DeconvNet [28], which propagates high level activations backward through a network to identify parts of the image responsible for the activation. In addition to the high-level feature activations, this reconstruction process uses extra information about maxima locations in intermediate max-pooling layers. This information has been shown to be crucial for the approach to work [22]. A visualization method similar to DeconvNet is by Springenberg et al. [22], yet it also makes use of intermediate layer activations.

Mahendran and Vedaldi [19] invert a differentiable image representation Φ using gradient descent. Given a feature vector Φ_0, they seek for an image x^* which minimizes a loss function – the squared Euclidean distance between Φ_0 and $\Phi(x)$ plus a regularizer enforcing a natural image prior. This method is fundamentally different from our approach in that it optimizes the difference between the feature vectors, not the image reconstruction error. Additionally, it includes a hand-designed natural image prior, while in our case the network implicitly learns such a prior. Technically, it involves optimization at test time, which requires computing the gradient of the feature representation and makes it relatively slow (the authors report 6s per image on a GPU). In contrast, the presented approach is only costly when training the inversion network. Reconstruction from a given feature vector just requires a single forward pass through the network, which takes roughly 5ms per image on a GPU. The method of [19] requires gradients of the feature representation, therefore it could not be directly applied to non-differentiable representations such as LBP, or recordings from a real brain [20].

There has been research on inverting various traditional computer vision representations: HOG and dense SIFT [24], keypoint-based SIFT [26], Local Binary Descriptors [4], Bag-of-Visual-Words [11]. All these methods are either tailored for inverting a specific feature representation or restricted to shallow representations, while our method can be applied to any feature representation.

2. Method

Denote by $(\mathbf{x}, \boldsymbol{\phi})$ random variables representing a natural image and its feature vector, and denote their joint probability distribution by $p(\mathbf{x}, \boldsymbol{\phi}) = p(\mathbf{x})p(\boldsymbol{\phi}|\mathbf{x})$. Here $p(\mathbf{x})$ is the distribution of natural images and $p(\boldsymbol{\phi}|\mathbf{x})$ is the distribution of feature vectors given an image. As a special case, $\boldsymbol{\phi}$ may be a deterministic function of $\mathbf{x}$. Ideally we would like to find $p(\mathbf{x}|\boldsymbol{\phi})$, but direct application of Bayes' theorem is not feasible. Therefore in this paper we resort to a point estimate $f(\boldsymbol{\phi})$ which minimizes the following mean squared error objective:

$$\mathbb{E}_{\mathbf{x},\boldsymbol{\phi}}\, ||\mathbf{x} - f(\boldsymbol{\phi})||^2 \quad (1)$$

The minimizer of this loss is the conditional expectation:

$$\hat{f}(\boldsymbol{\phi}_0) = \mathbb{E}_{\mathbf{x}}\, [\mathbf{x} \,|\, \boldsymbol{\phi} = \boldsymbol{\phi}_0], \quad (2)$$

that is, the *expected pre-image*.

Given a training set of images and their features $\{\mathbf{x}_i,\, \boldsymbol{\phi}_i\}$, we learn the weights $\mathbf{w}$ of an an up-convolutional network $f(\boldsymbol{\phi}, \mathbf{w})$ to minimize a Monte-Carlo estimate of the loss (1):

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \sum_i ||\mathbf{x}_i - f(\boldsymbol{\phi}_i, \mathbf{w})||_2^2. \quad (3)$$

This means that simply training the network to predict images from their feature vectors results in estimating the expected pre-image.

2.1. Feature representations to invert

Shallow features. We invert three traditional computer vision feature representations: histogram of oriented gradients (HOG), scale invariant feature transform (SIFT), and local binary patterns (LBP). We chose these features for a reason. There has been work on inverting HOG, so we can compare to existing approaches. LBP is interesting because it is not differentiable, and hence gradient-based methods cannot invert it. SIFT is a keypoint-based representation, so the network has to stitch different keypoints into a single smooth image.

For all three methods we use implementations from the *VLFeat* library [23] with the default settings. More precisely, we use the HOG version from Felzenszwalb et al. [7] with cell size 8, the version of SIFT which is very similar to the original implementation of Lowe [17] and the LBP version similar to Ojala et al. [21] with cell size 16. Before extracting the features we convert images to grayscale. More details can be found in the supplementary material.

AlexNet. We also invert the representation of the AlexNet network [13] trained on ImageNet, available at the *Caffe* [10] website.[1] It consists of 5 convolutional layers and 3 fully connected layers, with rectified linear units (ReLUs) after each layer, and local contrast normalization or max-pooling after some of them. Exact architecture is shown in the supplementary material. In what follows,

[1]More precisely, we used CaffeNet, which is almost identical to the original AlexNet.

when we say 'output of the layer', we mean the output of the last processing step of this layer. For example, the output of the first convolutional layer CONV1 would be the result after ReLU, pooling and normalization, and the output of the first fully connected layer FC6 is after ReLU. FC8 denotes the last layer, before the softmax.

2.2. Network architectures and training

An up-convolutional layer, also often referred to as 'deconvolutional', is a combination of upsampling and convolution [6]. We upsample a feature map by a factor 2 by replacing each value by a 2×2 block with the original value in the top left corner and all other entries equal to zero. Architecture of one of our up-convolutional networks is shown in Table 1. Architectures of other networks are shown in the supplementary material.

HOG and LBP. For an image of size $W \times H$, HOG and LBP features of an image form 3-dimensional arrays of sizes $\lceil W/8 \rceil \times \lceil H/8 \rceil \times 31$ and $\lceil W/16 \rceil \times \lceil H/16 \rceil \times 58$, respectively. We use similar CNN architectures for inverting both feature representations. The networks include a contracting part, which processes the input features through a series of convolutional layers with occasional stride of 2, resulting in a feature map 64 times smaller than the input image. Then the expanding part of the network again upsamples the feature map to the full image resolution by a series of up-convolutional layers. The contracting part allows the network to aggregate information over large regions of the input image. We found this is necessary to successfully estimate the absolute brightness.

Sparse SIFT. Running the SIFT detector and descriptor on an image gives a set of N keypoints, where the i-th keypoint is described by its coordinates (x_i, y_i), scale s_i, orientation α_i, and a feature descriptor $\mathbf{f}_i$ of dimensionality D. In order to apply a convolutional network, we arrange the keypoints on a grid. We split the image into cells of size $d \times d$ (we used $d = 4$ in our experiments), this yields $\lceil W/d \rceil \times \lceil H/d \rceil$ cells. In the rare cases when there are several keypoints in a cell, we randomly select one. We then assign a vector to each of the cells: a zero vector to a cell without a keypoint and a vector $(\mathbf{f}_i, x_i \mod d, y_i \mod d, \sin \alpha_i, \cos \alpha_i, \log s_i)$ to a cell with a keypoint. This results in a feature map $\mathbf{F}$ of size $\lceil W/d \rceil \times \lceil H/d \rceil \times (D+5)$. Then we apply a CNN to $\mathbf{F}$, as described above.

AlexNet. To reconstruct from each layer of AlexNet we trained a separate network. We used two basic architectures: one for reconstructing from convolutional layers and one for reconstructing from fully connected layers. The network for reconstructing from fully connected layers contains three fully connected layers and 5 up-convolutional layers, as shown in Table 1. The network for reconstructing from convolutional layers consists of three convolutional and several up-convolutional layers (the exact number depends on the layer to reconstruct from). Filters in all (up-)convolutional layers have 5×5 spatial size. After each layer we apply leaky ReLU nonlinearity with slope 0.2, that is, $r(x) = x$ if $x \geqslant 0$ and $r(x) = 0.2 \cdot x$ if $x < 0$.

Layer	Input	InSize	K	S	OutSize
fc1	AlexNet-FC8	1000	–	–	4096
fc2	fc1	4096	–	–	4096
fc3	fc2	4096	–	–	4096
reshape	fc3	4096	–	–	$4 \times 4 \times 256$
upconv1	reshape	$4 \times 4 \times 256$	5	2	$8 \times 8 \times 256$
upconv2	upconv1	$8 \times 8 \times 256$	5	2	$16 \times 16 \times 128$
upconv3	upconv2	$16 \times 16 \times 128$	5	2	$32 \times 32 \times 64$
upconv4	upconv3	$32 \times 32 \times 64$	5	2	$64 \times 64 \times 32$
upconv5	upconv4	$64 \times 64 \times 32$	5	2	$128 \times 128 \times 3$

Table 1: Network for reconstructing from AlexNet FC8 features. K stands for kernel size, S for stride.

Training details. We trained networks using a modified version of Caffe [10]. As training data we used the ImageNet [5] training set. In some cases we predicted downsampled images to speed up computations. We used the Adam [12] optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and mini-batch size 64. For most networks we found an initial learning rate $\lambda = 0.001$ to work well. We gradually decreased the learning rate towards the end of training. The duration of training depended on the network: from 15 epochs (passes through the dataset) for shallower networks to 60 epochs for deeper ones.

Quantitative evaluation. As a quantitative measure of performance we used the average normalized reconstruction error, that is the mean of $||x_i - f(\Phi(x_i))||_2/N$, where x_i is an example from the test set, f is the function implemented by the inversion network and N is a normalization coefficient equal to the average Euclidean distance between images in the test set. The test set we used for quantitative and qualitative evaluations is a subset of the ImageNet validation set.

3. Experiments: shallow representations

Figures 1 and 3 show reconstructions of several images from the ImageNet validation set. Normalized reconstruction error of different approaches is shown in Table 2. Clearly, our method significantly outperforms existing approaches. This is to be expected, since our method explicitly aims to minimize the reconstruction error.

Hoggles [24]	HOG^{-1} [19]	HOG our	SIFT our	LBP our
0.61	0.63	0.24	0.28	0.38

Table 2: Normalized error of different methods when reconstructing from HOG.

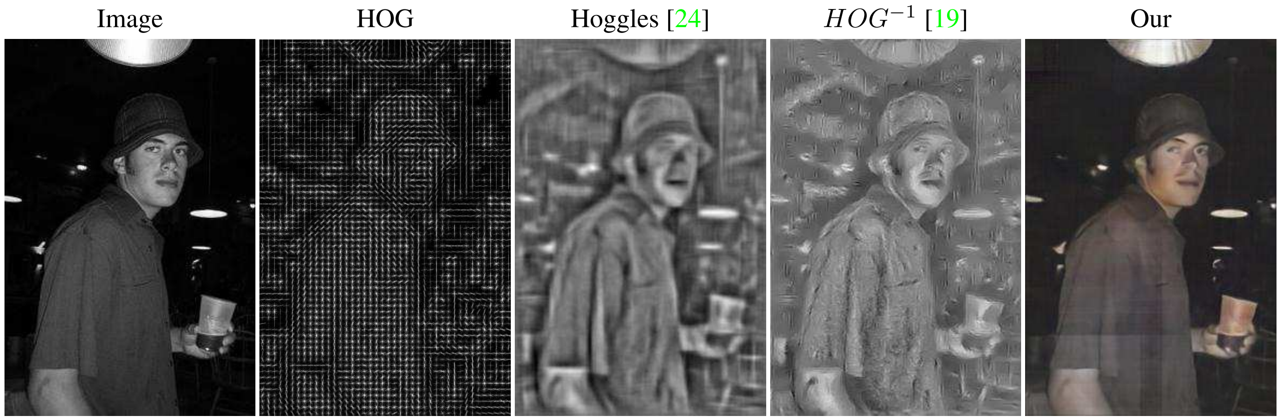

Figure 2: Reconstructing an image from its HOG descriptors with different methods.

Colorization. As mentioned above, we compute the features based on grayscale images, but the task of the networks is to reconstruct the color images. The features do not contain any color information, so to predict colors the network has to analyze the content of the image and make use of a natural image prior it learned during training. It does successfully learn to do so, as can be seen in Figures 1 and 3. Quite often the colors are predicted correctly, especially for sky, sea, grass, trees. In other cases, the network cannot predict the color (for example, people in the top row of Figure 3) and leaves some areas gray. Occasionally the network predicts the wrong color, such as in the bottom row of Figure 3.

HOG. Figure 2 shows an example image, its HOG representation, the results of inversion with existing methods [24, 19] and with our approach. Most interestingly, the network is able to reconstruct the overall brightness of the image very well, for example the dark regions are reconstructed dark. This is quite surprising, since the HOG descriptors are normalized and should not contain information about absolute brightness.

Normalization is always performed with a smoothing 'epsilon', so one might imagine that some information about the brightness is present even in the normalized features. We checked that the network does not make use of this information: multiplying the input image by 10 or 0.1 hardly changes the reconstruction. Therefore, we hypothesize that the network reconstructs the overall brightness by 1) analyzing the distribution of the HOG features (if in a cell there is similar amount of gradient in all directions, it is probably noise; if there is one dominating gradient, it must actually be in the image), 2) accumulating gradients over space: if there is much black-to-white gradient in one direction, then probably the brightness in that direction goes from dark to bright and 3) using semantic information.

SIFT. Figure 4 shows an image, the detected SIFT keypoints and the resulting reconstruction. There are roughly

Figure 3: Inversion of shallow image representations. Note how in the first row the color of grass and trees is predicted correctly in all cases, although it is not contained in the features.

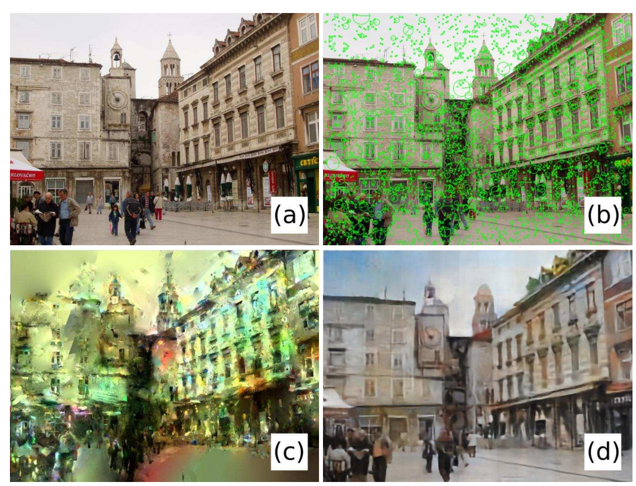

Figure 4: Reconstructing an image from SIFT descriptors with different methods. **(a)** an image, **(b)** SIFT keypoints, **(c)** reconstruction of [26], **(d)** our reconstruction.

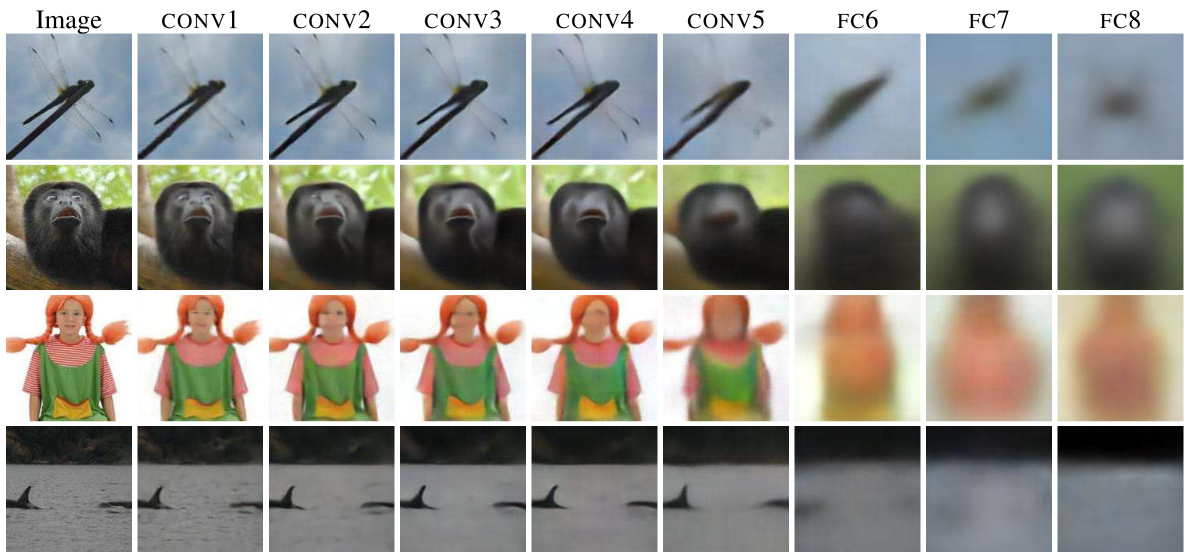

Figure 5: Reconstructions from different layers of AlexNet.

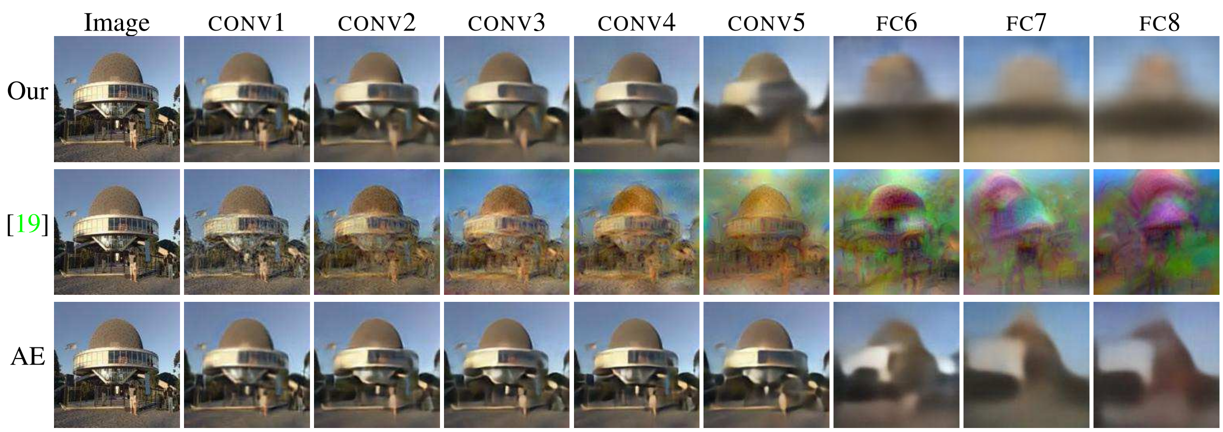

Figure 6: Reconstructions from layers of AlexNet with our method (top), [19] (middle), and autoencoders (bottom).

3000 keypoints detected in this image. Although made from a sparse set of keypoints, the reconstruction looks very natural, just a little blurry. To achieve such a clear reconstruction the network has to properly rotate and scale the descriptors and then stitch them together. Obviously it successfully learns to do this.

For reference we also show a result of another existing method [26] for reconstructing images from sparse SIFT descriptors. The results are not directly comparable: while we use the SIFT detector providing circular keypoints, Weinzaepfel et al. [26] use the Harris affine keypoint detector which yields elliptic keypoints, and the number and the locations of the keypoints may be different from our case. However, the rough number of keypoints is the same, so a qualitative comparison is still valid.

4. Experiments: AlexNet

We applied our inversion method to different layers of AlexNet and performed several additional experiments to better understand the feature representations. More results are shown in the supplementary material.

4.1. Reconstructions from different layers

Figure 5 shows reconstructions from various layers of AlexNet. When using features from convolutional layers, the reconstructed images look very similar to the input, but lose fine details as we progress to higher layers. There is an obvious drop in reconstruction quality when going from CONV5 to FC6. However, the reconstructions from higher convolutional layers and even fully connected layers preserve color and the approximate object location very well. Reconstructions from FC7 and FC8 still look similar to the input images, but blurry. This means that high level features

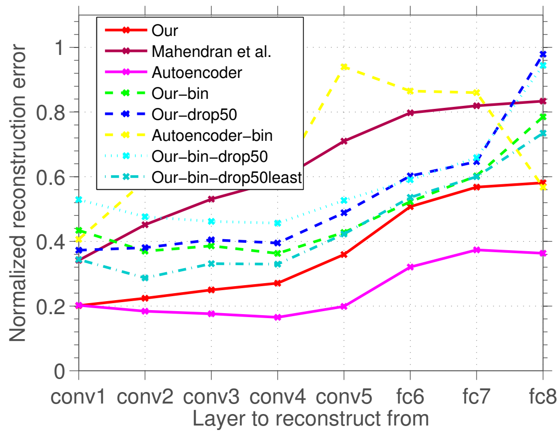

Figure 7: Average normalized reconstruction error depending on the network layer.

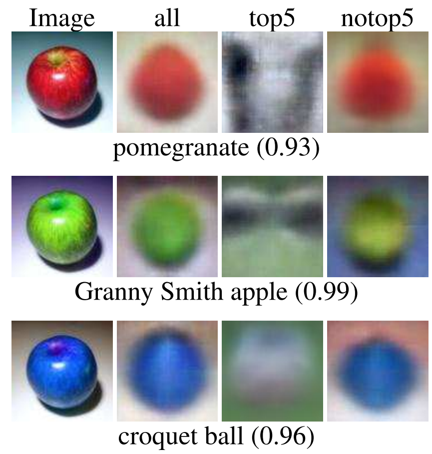

Figure 8: The effect of color on classification and reconstruction from layer FC8. Left to right: input image, reconstruction from FC8, reconstruction from 5 largest activations in FC8, reconstruction from all FC8 activations except the 5 largest ones. Below each row the network prediction and its confidence are shown.

are much less invariant to color and pose than one might expect: in principle fully connected layers need not preserve any information about colors and locations of objects in the input image. This is somewhat in contrast with the results of [19], as shown in Figure 6. While their reconstructions are sharper, the color and position are completely lost in reconstructions from higher layers.

For quantitative evaluation before computing the error we up-sample reconstructions to input image size with bilinear interpolation. Error curves shown in Figure 7 support the conclusions made above. When reconstructing from FC6, the error is roughly twice as large as from CONV5. Even when reconstructing from FC8, the error is fairly low because the network manages to get the color and the rough placement of large objects in images right. For lower layers, the reconstruction error of [19] is still much higher than of our method, even though visually the images look somewhat sharper. The reason is that in their reconstructions the color and the precise placement of small details do not perfectly match the input image, which results in a large overall error.

4.2. Autoencoder training

Our inversion network can be interpreted as the decoder of the representation encoded by AlexNet. The difference to an autoencoder is that the encoder part stays fixed and only the decoder is optimized. For comparison we also trained autoencoders with the same architecture as our reconstruction nets, i.e., we also allowed the training to fine-tune the parameters of the AlexNet part. This provides an upper bound on the quality of reconstructions we might expect from the inversion networks (with fixed AlexNet).

As shown in Figure 7, autoencoder training yields much lower reconstruction errors when reconstructing from higher layers. Also the qualitative results in Figure 6 show much better reconstructions with autoencoders. Even from CONV5 features, the input image can be reconstructed almost perfectly. When reconstructing from fully connected layers, the autoencoder results get blurred, too, due to the compressed representation, but by far not as much as with the fixed AlexNet weights. The gap between the autoencoder training and the training with fixed AlexNet gives an estimate of the amount of image information lost due to the training objective of the AlexNet, which is not based on reconstruction quality.

An interesting observation with autoencoders is that the reconstruction error is quite high even when reconstructing from CONV1 features, and the best reconstructions were actually obtained from CONV4. Our explanation is that the convolution with stride 4 and consequent max-pooling in CONV1 loses much information about the image. To decrease the reconstruction error, it is beneficial for the network to slightly blur the image instead of guessing the details. When reconstructing from deeper layers, deeper networks can learn a better prior resulting in slightly sharper images and slightly lower reconstruction error. For even deeper layers, the representation gets too compressed and the error increases again. We observed (not shown in the paper) that without stride 4 in the first layer, the reconstruction error of autoencoders got much lower.

4.3. Case study: Colored apple

We performed a simple experiment illustrating how the color information influences classification and how it is preserved in the high level features. We took an image of a red apple (Figure 8 top left) from Flickr and modified its

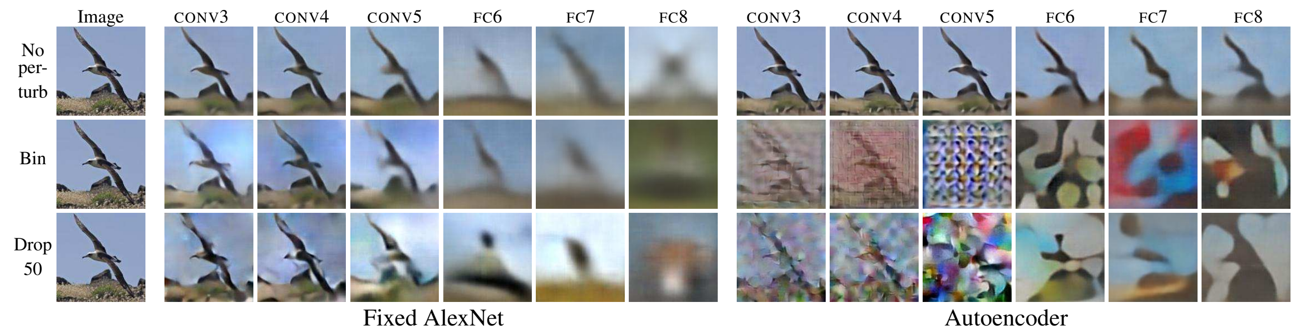

Figure 9: Reconstructions from different layers of AlexNet with disturbed features.

hue to make it green or blue. Then we extracted AlexNet FC8 features of the resulting images. Remind that FC8 is the last layer of the network, so the FC8 features, after application of softmax, give the network's prediction of class probabilities. The largest activation, hence, corresponds to the network's prediction of the image class. To check how class-dependent the results of inversion are, we passed three versions of each feature vector through the inversion network: 1) just the vector itself, 2) all activations except the 5 largest ones set to zero, 3) the 5 largest activations set to zero.

This leads to several conclusions. First, color clearly can be very important for classification, so the feature representation of the network has to be sensitive to it, at least in some cases. Second, the color of the image can be precisely reconstructed even from FC8 or, equivalently, from the predicted class probabilities. Third, the reconstruction quality does not depend much on the top predictions of the network but rather on the small probabilities of all other classes. This is consistent with the 'dark knowledge' idea of [8]: small probabilities of non-predicted classes carry more information than the prediction itself. More examples of this are shown in the supplementary material.

4.4. Robustness of the feature representation

We have shown that high level feature maps preserve rich information about the image. How is this information represented in the feature vector? It is difficult to answer this question precisely, but we can gain some insight by perturbing the feature representations in certain ways and observing images reconstructed from these perturbed features. If perturbing the features in a certain way does not change the reconstruction much, then the perturbed property is not important. For example, if setting a non-zero feature to zero does not change the reconstruction, then this feature does not carry information useful for the reconstruction.

We applied binarization and dropout. To binarize the feature vector, we kept the signs of all entries and set their absolute values to a fixed number, selected such that the Euclidean norm of the vector remained unchanged (we tried several other strategies, and this one led to the best result). For all layers except FC8, feature vector entries are nonnegative, hence, binarization just sets all non-zero entries to a fixed positive value. To perform dropout, we randomly set 50% of the feature vector entries to zero and then normalize the vector to keep its Euclidean norm unchanged (again, we found this normalization to work best). Qualitative results of these perturbations of features in different layers of AlexNet are shown in Figure 9. Quantitative results are shown in Figure 7. Surprisingly, dropout leads to larger decrease in reconstruction accuracy than binarization, even in the layers where it had been applied during training. In layers FC7 and especially FC6, binarization hardly changes the reconstruction quality at all. Although it is known that binarized ConvNet features perform well in classification [1], it comes as a surprise that for reconstructing the input image the exact values of the features are not important. In FC6 virtually all information about the image is contained in the binary code given by the pattern of non-zero activations. Figures 7 and 9 show that this binary code only emerges when training with the classification objective and dropout, while autoencoders are very sensitive to perturbations in the features.

To test the robustness of this binary code, we applied binarization and dropout together. We tried dropping out 50% random activations or 50% least non-zero activations and then binarizing. Dropping out the 50% least activations reduces the error much less than dropping out 50% random activations and is even better than not applying any dropout for most layers. However, layers FC6 and FC7 are the most interesting ones: here dropping out 50% random activations decreases the performance substantially, while dropping out 50% least activations only results in a small decrease. Possibly the exact values of the features in FC6 and FC7 do not affect the reconstruction much, but they estimate the importance of different features.

4.5. Interpolation and random feature vectors

Another way to analyze the feature representation is by traversing the feature manifold and by observing the corre-

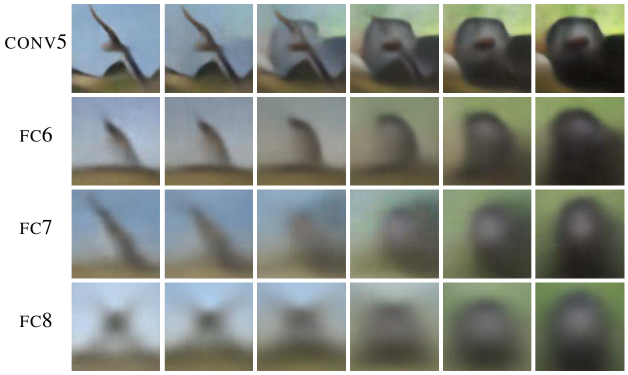

Figure 10: Interpolation between the features of two images.

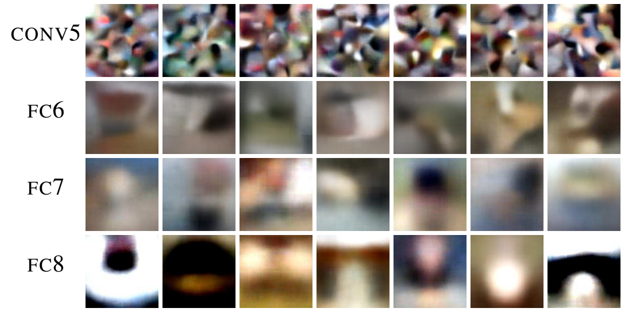

Figure 11: Images generated from random feature vectors of top layers of AlexNet.

sponding images generated by the reconstruction networks. We have seen the reconstructions from feature vectors of actual images, but what if a feature vector was not generated from a natural image? In Figure 10 we show reconstructions obtained with our networks when interpolating between feature vectors of two images. It is interesting to see that interpolating CONV5 features leads to a simple overlay of images, but the behavior of interpolations when reconstructing from FC6 is very different: images smoothly morph into each other. More examples, together with the results for autoencoders, are shown in the supplementary material.

Another analysis method is by sampling feature vectors randomly. Our networks were trained to reconstruct images given their feature representations, but the distribution of the feature vectors is unknown. Hence, there is no simple principled way to sample from our model. However, by assuming independence of the features (a very strong and wrong assumption!), we can approximate the distribution of each dimension of the feature vector separately. To this end we simply computed a histogram of each feature over a set of 4096 images and sampled from those. We ensured that the sparsity of the random samples is the same as that of the actual feature vectors. This procedure led to low contrast images, perhaps because by independently sampling each dimension we did not introduce interactions between the features. Multiplying the feature vectors by a constant factor $\alpha = 2$ increases the contrast without affecting other properties of the generated images.

Random samples obtained this way from four top layers of AlexNet are shown in Figure 11. No pre-selection was performed. While samples from CONV5 look much like abstract art, the samples from fully convolutional layers are much more realistic. This shows that the networks learn a natural image prior that allows them to produce somewhat realistically looking images from random feature vectors. We found that a much simpler sampling procedure of fitting a single shifted truncated Gaussian to all feature dimensions produces qualitatively very similar images. These are shown in the supplementary material together with images generated from autoencoders, which look much less like natural images.

5. Conclusions

We have proposed to invert image representations with up-convolutional networks and have shown that this yields more or less accurate reconstructions of the original images, depending on the level of invariance of the feature representation. The networks implicitly learn natural image priors which allow the retrieval of information that is obviously lost in the feature representation, such as color or brightness in HOG or SIFT. The method is very fast at test time and does not require the gradient of the feature representation to be inverted. Therefore, it can be applied to virtually any image representation.

Application of our method to the representations learned by the AlexNet convolutional network leads do several conclusions: 1) Features from all layers of the network, including the final FC8 layer, preserve the precise colors and the rough position of objects in the image; 2) In higher layers, almost all information about the input image is contained in the pattern of non-zero activations, not their precise values; 3) In the layer FC8, most information about the input image is contained in small probabilities of those classes that are not in top-5 network predictions.

Acknowledgements

We acknowledge funding by the ERC Starting Grant VideoLearn (279401). We are grateful to Aravindh Mahendran for sharing with us the reconstructions achieved with the method of Mahendran and Vedaldi [19]. We thank Jost Tobias Springenberg for comments.

References

[1] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, 2014. 7

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford Uni. Press, New York, USA, 1995. 2

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893, 2005. 1

[4] E. d'Angelo, L. Jacques, A. Alahi, and P. Vandergheynst. From bits to images: Inversion of local binary descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(5):874–887, 2014. 2

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 3

[6] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, 2015. 3

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 32(9):1627–1645, 2010. 1, 2

[8] G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015. 7

[9] C. Jensen, R. Reed, R. Marks, M. El-Sharkawi, J.-B. Jung, R. Miyamoto, G. Anderson, and C. Eggen. Inversion of feedforward neural networks: Algorithms and applications. In *Proc. IEEE*, pages 1536–1549, 1999. 2

[10] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014. 2, 3

[11] H. Kato and T. Harada. Image reconstruction from bag-of-visual-words. In *CVPR*, June 2014. 2

[12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 3

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012. 1, 2

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

[15] S. Lee and R. M. Kil. Inverse mapping of continuous functions using local and global information. *IEEE Transactions on Neural Networks*, 5(3):409–423, 1994. 2

[16] A. Linden and J. Kindermann. Inversion of multilayer nets. In *Proc. Int. Conf. on Neural Networks*, 1989. 2

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. 1, 2

[18] B. Lu, H. Kita, and Y. Nishikawa. Inverting feedforward neural networks using linear and nonlinear programming. *IEEE Transactions on Neural Networks*, 10(6):1271–1290, 1999. 2

[19] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015. 1, 2, 3, 4, 5, 6, 8

[20] S. Nishimoto, A. Vu, T. Naselaris, Y. Benjamini, B. Yu, and J. Gallant. Reconstructing visual experiences from brain activity evoked by natural movies. *Current Biology*, 21(19):1641–1646, 2011. 2

[21] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *TPAMI*, 24(7):971–987, 2002. 1, 2

[22] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR Workshop Track*, 2015. 1, 2

[23] A. Vedaldi and B. Fulkerson. Vlfeat: an open and portable library of computer vision algorithms. In *International Conference on Multimedia*, pages 1469–1472, 2010. 2

[24] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. Hoggles: Visualizing object detection features. *ICCV*, 2013. 2, 3, 4

[25] A. R. Vrkonyi-Kczy. Observer-based iterative fuzzy and neural network model inversion for measurement and control applications. In I. J. Rudas, J. C. Fodor, and J. Kacprzyk, editors, *Towards Intelligent Engineering and Information Technology*, volume 243 of *Studies in Computational Intelligence*, pages 681–702. Springer, 2009. 2

[26] P. Weinzaepfel, H. Jegou, and P. Prez. Reconstructing an image from its local descriptors. In *CVPR*. IEEE Computer Society, 2011. 2, 4, 5

[27] R. J. Williams. Inverting a connectionist network mapping by back-propagation of error. In *Eighth Annual Conference of the Cognitive Society*, pages 859–865, 1986. 2

[28] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 1, 2