Laplacian Pyramids for Deep Feature Inversion

Aniket Singh
CVIT, IIIT-Hyderabad
Hyderabad, India
aniket.singh@research.iiit.ac.in

Anoop Namboodiri
CVIT, IIIT-Hyderabad
Hyderabad, India
anoop@iiit.ac.in

Abstract

Modern feature extraction pipelines, especially the ones using deep networks, involve an increasing variety of elements. With layered approaches heaping abstraction upon abstraction, it becomes difficult to understand what it is that these features are capturing. One appealing way of solving this puzzle is feature visualization, where features are mapped back to the image domain. Our work improves the generic approach of performing gradient descent (GD) in the image space to match a given set of features to achieve a visualization. Specifically, we note that coarse features of an image like blobs, outlines etc. are useful by themselves for classification purposes. We develop an inversion scheme based on this idea by recovering coarse features of the image before finer details. This is done by modeling the image as the composition of a Laplacian Pyramid. We show that by performing GD on the pyramid in a level-wise manner, we can recover meaningful images. Results are presented for inverting a shallow network: the densely calculated SIFT as well as a deep network: Krizehvsky et al.'s Imagenet CNN (Alexnet).

1. Introduction

Computer vision has seen a watershed moment with the advent of deep architectures. Feature extraction and classification pipelines have become more complex with an increasing variety in elements that compose them. With such advancements, a need is perceived for a broader, intuitive understanding of what these architectures learn. One paradigm within which we can ask such questions is feature visualization, where one tries to visualize an image equivalent of the features at any stage of the network.

Considering the feature extraction process as a vector valued function that acts on an image and produces a feature vector, the act of visualization is naturally posed as inverting this function *i.e.*, recovering a point in the domain (image) given a point in the range (feature space). For a feature representation, it is desirable to achieve invariances to nuisance factors such as pose and lighting as well as being efficient by representing large images as compact codes. These two requirements hint at the representation being a many-to-one function, where the domain is larger than the range. In this case, multiple inversions would be possible. However, among the large number of data points in this domain, recognizable images would occupy only a slim manifold. Hence the challenge for a visualization scheme is, given a query, find an inversion that falls on this image manifold and produce a feature representation that is close to the query.

Figure 1: Effect of coarse to fine inversion. Left to right: original image, inversion using a 2-level and a 6-level laplacian pyramid from the final convolution layer of Alex-net.

Consider the information about images captured by a classifier. We note that even blurred versions of images can help discount many improbable categories. Hence we hypothesize that an effective discriminative feature scheme would also leverage the coarse level information in an image. For deep networks, such information would persist as one ascends the layers. Our inversion scheme's modus operandi for localizing the image in the manifold is to discover coarse inversions before fine tuning them with more precise details. We show that this is effective in recovering recognizable inversions of features from the different layers of deep networks.

We first look at the related literature on feature visualization in section 2 followed by the motivation and the technical details of our visualization scheme in section 3. Section 4 provides analytical justification for the approach's effectiveness. Results of application of our scheme for inverting a dense SIFT feature extractor [9] and Alexnet [6] are pro-

978-1-4799-6100-9/15/$31.00 ©2015 IEEE

vided in Section 5 followed by concluding remarks in 6.

2. Related Work

Feature visualization has previously been explored for hand designed features in [14] for SIFT, [11] for GIST, and [2] for LBP. A general descriptor-agnostic scheme was presented in Hoggles [13] whose fast inversion capabilities were demonstrated on HoG features.

In the context of neural networks, directly visualizing the filter parameters of a neural network [7] has been a popular technique to understand what is learnt. Zeiler and Fergus [15] learn a deconvolutional network [16] as an inverter for Krizhevsky *et al.*'s Imagenet CNN [6] (Alexnet). The authors modified Alexnet's architecture to improve the sharpness of the visualization and observed that this also increased the network's classification performance. This hints at visualization playing a role in network design. A more recent work on learned inversion is by Dosovitskiy *et al.*[17], where an "up-convolutional" network is learnt to invert a deep network.

The present work builds on a scheme first discussed in [4], where Erhan *et al.*addressed the task of visualizing individual neurons by using gradient descent to find the input maximizing their activities. Simoyan *et al.* [12] used a similar gradient descent approach, but with an image norm regularization added to the objective, to visualize the classification layer of Alexnet. Mahendran *et al.* [10] added a total variation regularizer to the norm constraints and achieved high quality inversions from various layers of deep neural networks. By expressing SIFT and HOG as neural-network like architectures with differentiable elements, they showed gradient descent could be used to visualize shallow features as well.

In contrast to the above approaches where regularization terms are tagged along with the inversion objective, we consider modelling the image at the input end. Our model allows us to recover visual details in a coarse to fine fashion, where larger edges and blobs are discovered first and this outline is subsequently filled in with finer details. This we accomplish by proposing an extension to the Laplacian pyramid framework[1], which we term as *switched Laplacian pyramids*. This image model is attached to the input of the network to be inverted to form what we call an *augmented network*. We describe these concepts in detail in the following section.

3. Our Approach: Modelling Images

We first define notation. For the current discussion assume images as vectors. The original image is I. The query feature for inversion is the i^{th} layer of the deep network acting on I, $f^i(I)$. On the recovery end, our image model is a mini network which connects 3 quantities of interest. The quantity that we actually perform optimization over is I_{pre} which is transformed into the image being recovered I_r. I_r is further linearly transformed into I_{in} which becomes the input to the deep network. We now detail the image model, and make clear the exact relation between these quantities. Our modelling comprises of 2 main elements:

3.1. Range Constraining

An image as presented during the training of a neural network has pixels which take on values within a finite range, say $[-B, B]$. Alexnet, for example, is trained on images in the $[-127, 128]$ range. We model this aspect in the image to be inverted I_r by expressing I_r as the pointwise sigmoid, $\sigma()$ of a real valued vector I_{pre}, and then bringing it into a range appropriate for the network:

$$I_r = 2B\sigma(I_{pre}) - B, I_{pre} \in [-\infty, \infty] \tag{1}$$

3.2. Switched Laplacian Pyramids

The Laplacian pyramid (LP) is a loss-less encoding-decoding scheme for images. Here an image is broken down into a low-pass version of itself and a series of bandpass images, which together form the levels of the pyramid. Denote a LP with K levels constructed from image I as $\mathbf{L}^K(I) = \{L_1^K(I), L_2^K(I), \ldots L_K^K(I)\}$ where $L_j^K(I)$ is its j^{th} level. Then, $L_K^K(I)$ is a low pass version of I, while the $L_K^j(I), j \in \{1, 2, \ldots K-1\}$ are the bandpass levels, where the center frequency of the band decreases with the index j. The process of re-composition of the original image from the pyramid is linear can be denoted by $R(\mathbf{L}^K(I))$ where R is a linear operator.

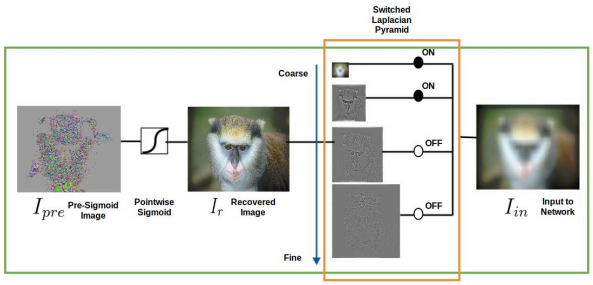
We propose a modification to the LP, which we term the *switched Laplacian Pyramid*, (sLP), where we introduce binary switch $s_j^K \in \{0, 1\}$ to be multiplied with the $L_j^K(I)$ before the re-composition operation. We denote the sLP by $\hat{\mathbf{L}}^K(I) = \{\hat{L}_1^K(I), \hat{L}_2^K(I), \ldots \hat{L}_K^K(I)\}$, where $\hat{L}_i^K(I) = s_i^K L_i^K(I)$. Reconstruction from the sLP is $R(\hat{\mathbf{L}}_K(I))$. At any stage of inversion, we will keep all levels in $\{j, \ldots K\}$ *on* while those in $\{1, \ldots, j-1\}$ are *off*, allowing only the coarsest $K - j + 1$ levels of the LP to be used for re-composition, while suppressing the $j - 1$ high frequency levels. Note the sLP leaves the process of decomposing an image into the pyramid unaltered, only modifying the re-composition.

3.3. Augmented Networks

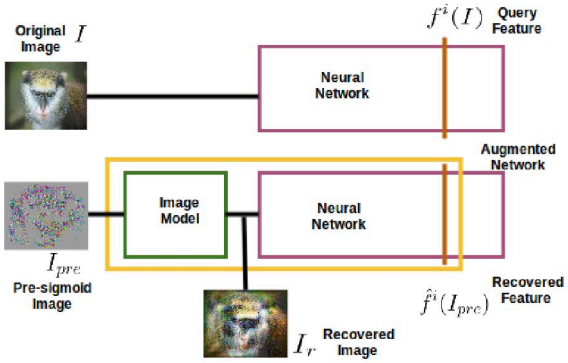
The input to the network, I_{in} is the result of applying the switched pyramid framework on the range constrained image I_r :

$$I_{in} = R(\hat{\mathbf{L}}^K(I_r)) \tag{2}$$

$$= R(\hat{\mathbf{L}}^K(2B\sigma(I_{pre}) - B)) \tag{3}$$



(a) Image Model



(b) Inversion Setup

Figure 2: Left shows the image modelling using switched laplacian pyramids and the range constraints. Right shows the the augmented network (enclosed by yellow-orange box) formed by the concatenation of the neural network and the image model, and the setup used for inversion. Highlighted are the query, $f(I)$, and the recovered, $\hat{f}(I_{pre})$, features (red-brown). Also shown are the recovered image I_r as an intermediate representation within the augmented network, while the input of the augmented net and the quantity being optimized is I_{pre}

If we compose the function for the i^{th} layer of the deep network, $f^i(I_{in})$, with equation 3 , we get the end-to-end function, which we'll denote by $\hat{f}^i(I_{pre})$, which represents feature extraction from I_{pre}. We term $\hat{f}^i(I_{pre})$ as the *augmented network*.

3.4. Coarse-to-Fine Inversion

The process of inversion is to find the I^*_{pre} whose feature $\hat{f}^i(I^*_{pre})$ is closest to the query feature $f^i(I)$, according to an error function $E(I, I_{pre})$:

$$E(I, I_pre) = \|\hat{f}^i(I_{pre}) - f^i(I)\|_2 \quad (4)$$

$$I^*_{pre} = \arg\min_{I_{pre}} E(I, I_{pre}) \quad (5)$$

This is optimized by performing stochastic gradient descent (SGD) over I^*_p with 4 as the objective. The switching capabilities of the sLP serve to regularize this process and produce high quality inversions. We start with just s^K_K being on which means only the coarsest level of the pyramid is available for the computation of I_{in}. After running SGD till convergence we introduce the next coarsest level by turning on s^K_{K-1}. This process continues with further levels being introduced into the calculation of I_{in}, such that at any point of the inversion all switches $s^K_{j:K}$ from the j^{th} to the coarsest K are ON. In the final phase all the levels are being utilized, and due to the loss-less nature of LP encoding-decoding, $I_{in} = I_r$ at this point.

In this manner the switching framework is utilized for the discovery of coarse level description of the recovered image before exploring its finer details.

When performing SGD for inversion a large learning rate may cause spilling over of the recovered image into levels finer than those whose switches are on. This is the same phenomenon where scaling up the pixel intensities of an image increases the high frequency components in its spectrum. To adjust for this effect, we introduce a *compactness term* $D(I_{in}, I_r)$ to the inversion objective, which encourages I_r to remain within the sLP levels that are currently on:

$$D(I_{in}, I_r) = \|I_r - I_{in}\|_2 \quad (6)$$

The relative importance of the compactness term for inversion is controlled by the *compactness coefficient* λ_D. The resultant objective $Z(I, I_{pre})$ used for inversion is :

$$Z(I, I_{pre}) = E(I, I_{pre}) + \lambda_D D(I_{in}, I_r) \quad (7)$$

Note that by the final phase of inversion where all pyramid levels are ON, and $I_{in} = I_r$, this compactness term disappears.

4. Analysis

We now investigate the effect of the switched-pyramid approach on the optimization. For ease of analysis we make two simplifications to the model: (1) We ignore the upsampling and downsampling operations involved in the Laplacian pyramid. By doing so, we deal with pyramids where all the levels have the same size. Size preserving blurring is used, and reconstruction from the Laplacian pyramid involves straightforward addition of all the levels. (2) We consider the optimization only till I_r , and not I_{pre}. Since they are separated by just a pointwise function, our arguments will remain unaffected.

Denote the Toeplitz matrix for blurring/interpolating kernel by B. A Gaussian pyramid with K levels, $\mathbf{G}^K$ =

$\{G_1^K, G_2^K \dots G_K^K\}$, is the series of blurred images obtained by recursively multiplying I_r by B, where the j^{th} blurred image is G_j^K calculated as (B^j represents B to the power j):

$$G_j^K = B^{j-1} I_r \tag{8}$$

A Laplacian pyramid $\mathbf{L}^K$ with K levels is associated with this Gaussian pyramid. The j^{th} level of $\mathbf{L}^K$ denoted by L_j^K is calculated as:

$$L_j^K = \begin{cases} G_K^K = B^{j-1} I_r & \text{if } j = K \\ G_j^K - G_{j+1}^K = (B^{j-1} - B^j) I_r & \text{if } j < K \end{cases} \tag{9}$$

We attach a binary switch s_j^K to each Laplacian layer. The input to the network I_{in} calculated from the switched-pyramid is:

$$I_{in} = s_1^K I_r + \sum_1^{K-1} (s_{j+1}^K - s_j^K) B^j I_r \tag{10}$$

In our inversion framework, at any time only switches including and coarser than a particular level, say j, are *ON*. In such a setting, I_{in} simply becomes the j^{th} level of $\mathbf{G}^K$. Let our loss function be $E(I, I_r)$. We wish to compare the gradients employed by SGD when using the sLP versus those in its absence. In the latter case $I_{in} = I_r$, and the gradient is $\nabla_{I_{in}} E(I, I_{in})$. When using sLP the gradient becomes:

$$\nabla_{I_r} E(I, I_r) = (B^{j-1})^T \nabla_{I_{in}} E(I, I_{in}) \tag{11}$$

The transpose of Toeplitz is Toeplitz, and $(B^{j-1})^T = (B^T)^{j-1}$. So, the action of $(B^T)^{j-1}$ may be seen as repeatedly blurring by B^T. From an optimization point of view, blurring the gradients induces gradient 'sharing' causing neighbouring pixels intensities to move in the same way, and acts as a smoothness regularizer. As figure 1 shows, however, even with this implicit regularization, it is important to start at a coarse enough level to get good quality inversions.

5. Experiments and Results

The reader should take care that levels refer to Laplacian pyramid levels, and layers refer to neural network layers. The first network we invert is Alexnet. We used the implementation due to [3]. We use a Laplacian pyramid with 6 levels.

Visualizations from all the layers are shown in 3. There is a clear difference between the nature of inversions of layers where spatial information is retained and the fully connected layers. The inversions from the former set are highly similar to the original image even as one ascends the neural network, albeit becoming increasingly rough. One phenomenon that is discernable is localized displacements of parts such as eyes. In the case of the fully connected layers, the inversions take on a "bag of parts" aspect, with relative location of the parts such as eyes,lips being lost in the inversion. As compared [10], the present scheme shows greater ability to recover the coarse color of the original image: the large brown and grey regions on the monkey's fur are also present in the inversions. For quantifying the inversion, we follow a scheme similar to [10]: We take the mean normalized reconstruction error, $\|\hat{f^i}(I_p) - f^i(I)\|_2 / N_{f^i}$ over 100 images from Imagenet validation set. Here N_{f^i} is the average pairwise euclidean distance between $f^i(I)$ of 100 images from the validation set. The results are reported in table 1. Our scheme performs favourably in comparison to [10], in part due to the fact that the final phase of inversion is unregularized. Note however both [10] and the current method were evaluated on a small random set of images, so the numerical results are not directly comparable. We used the compactness coefficient as 0.001 for 1-8, 0.01 for layers 9-12, 0.1 for layers 13-16 and 1 for layers 17-24.

We also apply the inversion scheme to the SIFT feature representation. We followed the implementation according to Jia et al. in [5]: This calculates a 128 dimensional feature vector from 16x16 patches. This patchwise SIFT was wrapped up in a Network in Network [8] framework to obtain the SIFT for the full image. Patches were taken at strides of 8 on grayscale images of size 128×128, which results in a $15 \times 15 \times 128$ feature representation. Figure 4 shows inversion results for dense sift. We implemented a color SIFT descriptor by simply applying Grayscale SIFT to the R,G,B channels individually and concatenating the 3 vectors to give a 384 dimensional feature. Inversions for this are also shown in 4. The reconstructions for SIFT are visually very similar to the original image. High quality inversions is a common trend for shallow features, be it SIFT or the earlier layers of the deep network. Table 2 shows the reconstruction errors for SIFT. We set the compactness coefficient as 0. for the experiments.

SIFT	Grey	Color
Error	4.5%	3.2%

Table 2: Reconstruction errors for inverting SIFT descriptors

6. Conclusions And Future Work

We presented a new method for improving quality of feature inversions. This scheme is based on coarse-to-fine inversion by using Laplacian pyramids for representing images. We demonstrated the method's potential by inverting a deep network: Krizhevsky et. al.'s Imagenet CNN , and a shallow network: dense SIFT. We provided some theoretical arguments for the scheme's effectiveness. We will

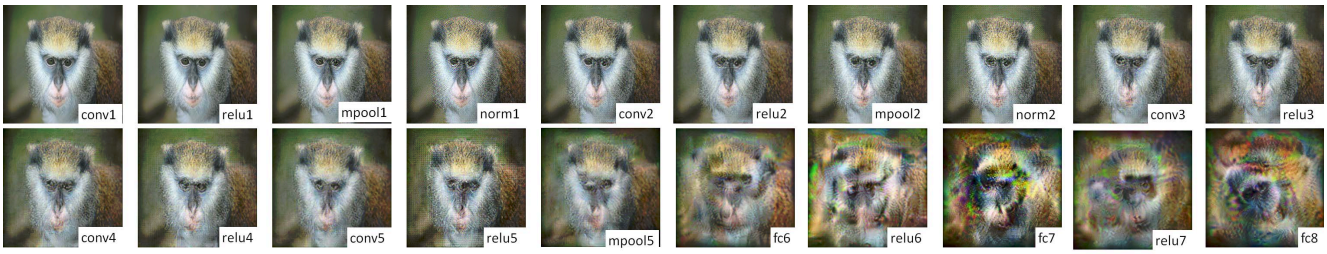


Figure 3: Inversions for Alexnet

Layer	conv1	relu1	mpool1	norm1	conv2	relu2	mpool2	norm2	conv3	relu3	conv4	relu4	conv5	relu5	mpool5	fc6	relu6	fc7	relu7	fc8
Error (Ours)	**6.1%**	**5.7%**	**12.1%**	**5.2%**	**6.4%**	**3.6%**	**11.1%**	**7.4%**	**8.1%**	**8.1%**	**7.7%**	**8.5%**	**11.1%**	**12.1%**	**13.3%**	**7.4%**	**9.1%**	**11.1%**	**6.1%**	**5.5%**
Error ([10])	10.0%	11.3%	21.9%	20.3%	12.4%	12.9%	18.2%	18.4%	14.4%	15.1%	13.3%	14.0%	25.7%	20.1%	19.0%	18.6%	18.7%	17.1%	15.5%	8.5%

Table 1: Layer wise reconstruction errors for inverting Alexnet

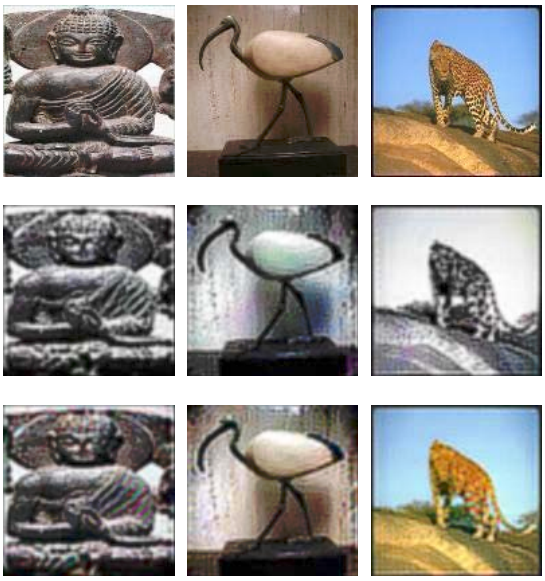


Figure 4: Top: Original Images, Middle: Inversions from Grayscale SIFT, Bottom: Inversions from Color SIFT

look to improve the scheme's performance, especially for inverting the fully connected layers of Neural Networks.

References

[1] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *Communications, IEEE Transactions on*, 1983. 2

[2] E. d'Angelo, A. Alahi, and P. Vandergheynst. Beyond bits: Reconstructing images from local binary descriptors. In *ICPR*. IEEE, 2012. 2

[3] W. Ding, R. Wang, F. Mao, and G. Taylor. Theano-based large-scale visual recognition with multiple gpus. *arXiv preprint arXiv:1412.2302*, 2014. 4

[4] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *Dept. IRO, Université de Montréal, Tech. Rep*, 2009. 2

[5] Y. Jia, C. Huang, and T. Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *CVPR*. IEEE, 2012. 4

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1, 2

[7] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*. ACM, 2009. 2

[8] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013. 4

[9] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 1

[10] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. *arXiv preprint arXiv:1412.0035*, 2014. 2, 4, 5

[11] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001. 2

[12] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 2

[13] C. Vondrick, A. Khosla, T. Malisiewicz, and A. Torralba. Hoggles: Visualizing object detection features. In *ICCV*. IEEE, 2013. 2

[14] P. Weinzaepfel, H. Jégou, and P. Pérez. Reconstructing an image from its local descriptors. In *CVPR*. IEEE, 2011. 2

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

[16] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *CVPR*. IEEE, 2010. 2

[17] A. Dosovitskiy,, T. Brox. Inverting Convolutional Networks with Convolutional Networks. In *arXiv preprint arXiv:1506.02753*. 2015. 2