Understanding Deep Image Representation by Inverting Them

Aravindh Mahendran Andrea Vedaldi
University of Oxford

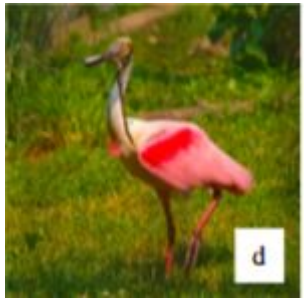
Presented by Xinmeng Li and Beibei Du
Mar 28, 2017

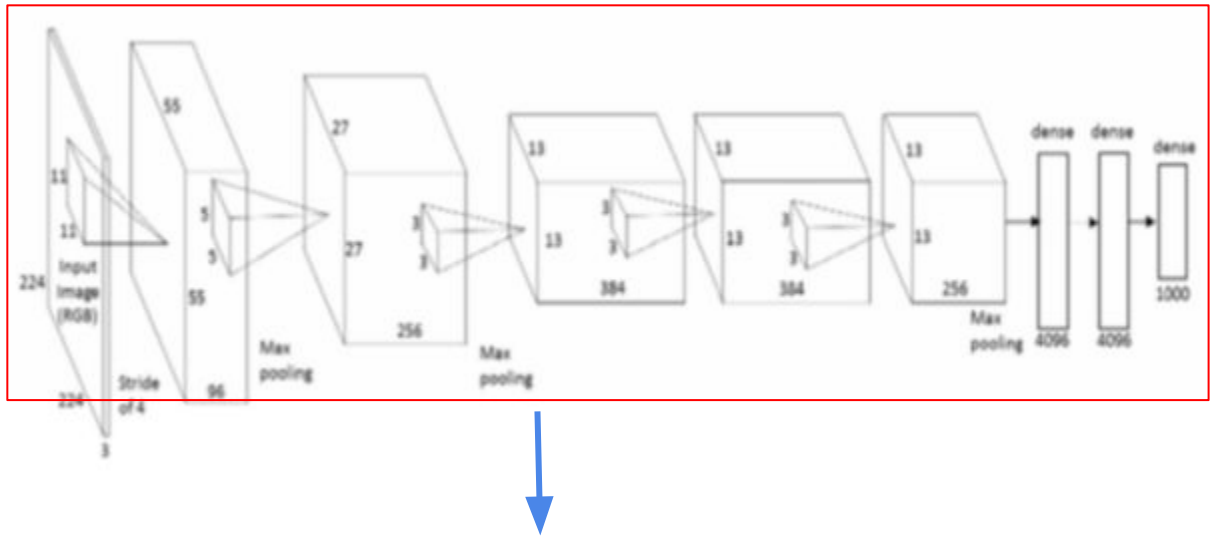
Outline

- **Introduction**
- **Contributions**
- **Related Works**
- **Inverting Method**
- **Representation**
- **Experiments**
- **Demo**

Outline

- **Introduction**
- **Contributions**
- **Related Works**
- **Inverting Method**
- **Representation**
- **Experiments**
- **Demo**

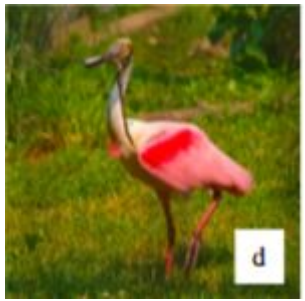


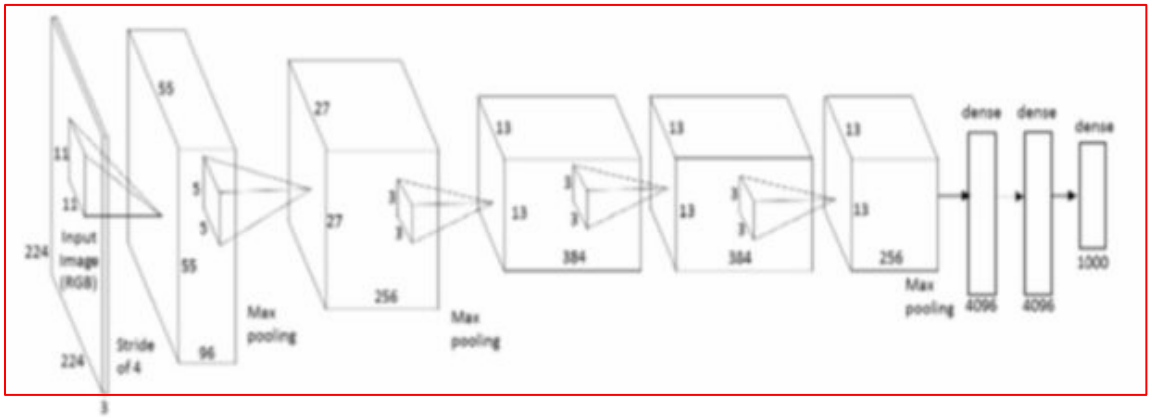


flamingo

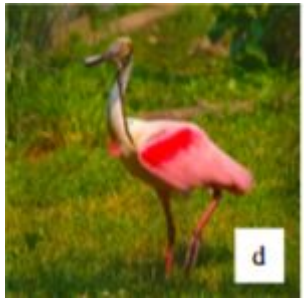
Image representations

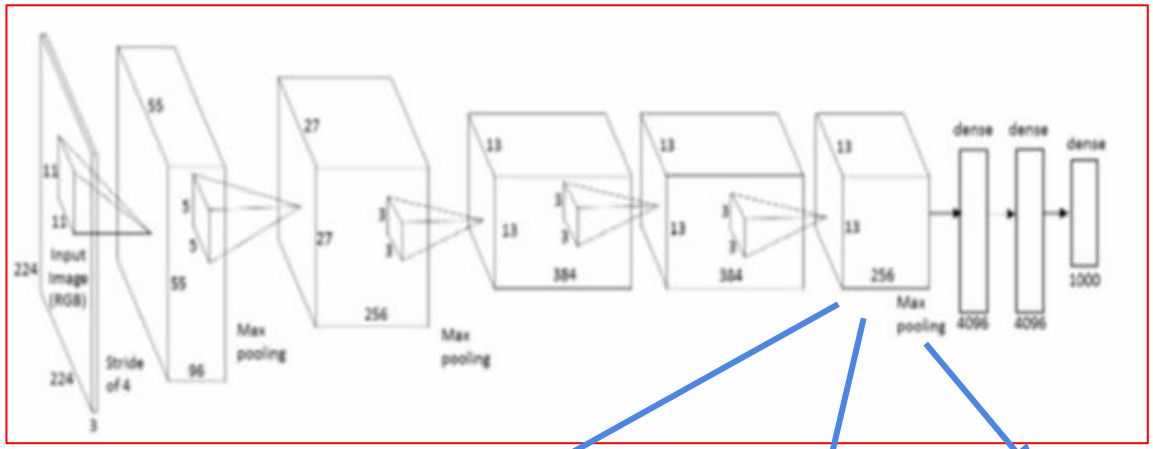
Introduction





flamingo

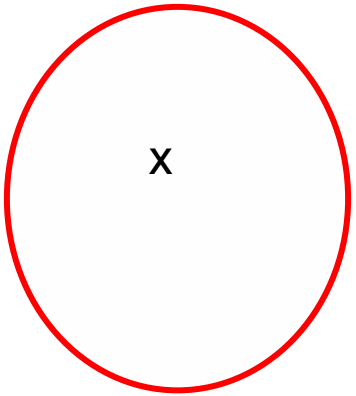



11
11
224
Input
Image
(RGB)
224
3
55
5
5
55
96
Stride
of 4
Max
pooling
27
27
256
Max
pooling
13
13
3
3
13
384
13
13
3
3
13
384
13
13
3
3
13
256
Max
pooling
dense
4096
dense
4096
dense
1000

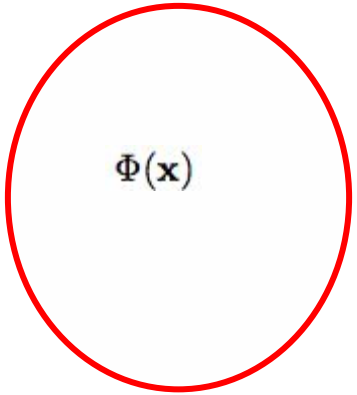
flamingo

Framework to do visualizations

Image space (pre-images)



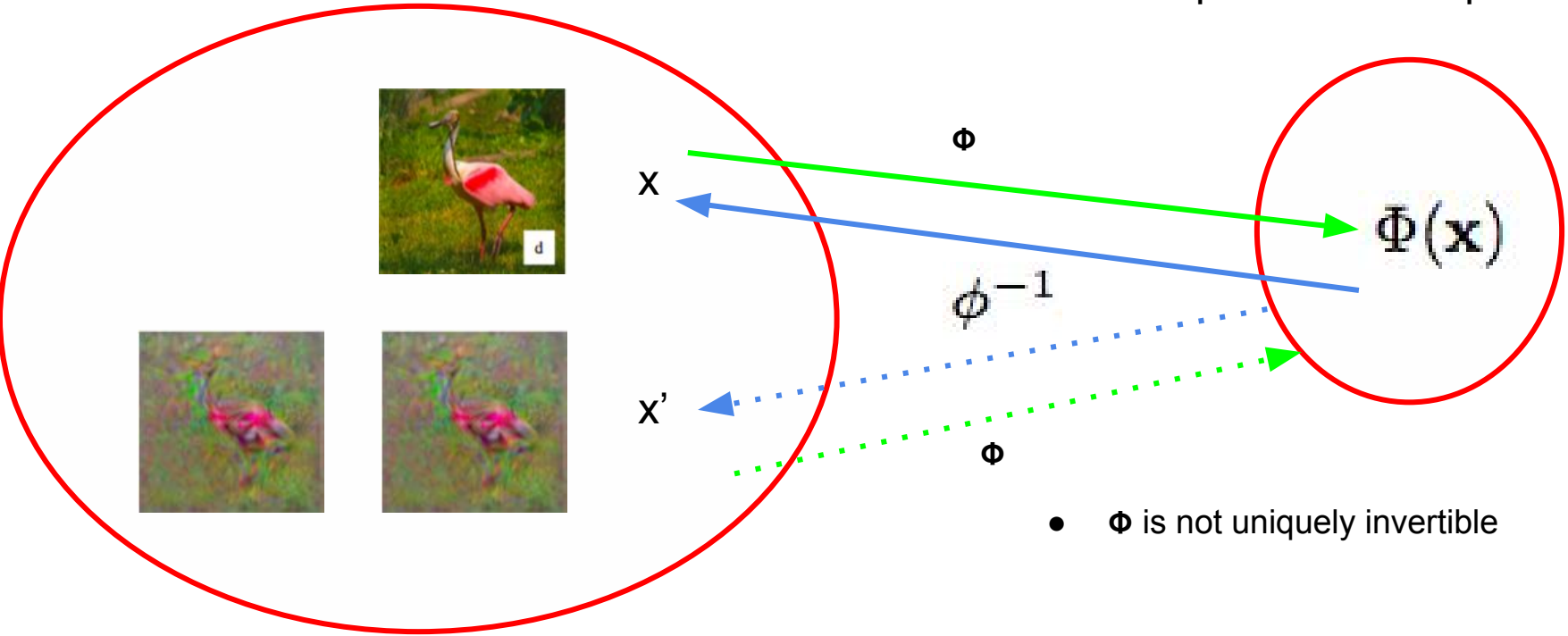
Representation space



Framework to invert representations

Image Space

Representation space



- Φ is not uniquely invertible

Outline

- Introduction
- **Contributions**
- Related Works
- Inverting Method
- Representation
- Experiments
- Demo

Contributions

- **Propose a general method to invert representations, including SIFT, HOG and CNNS**
- **Inverting representations method performs better for DSIFT and HOG compared to recent alternatives.**
- **Apply inversion technique to the analysis of deep CNNs, showing that CNN gradually builds an increasing amount of invariance, layer by layer**
- **Study locality of information stored in the representations by reconstructing image from selected groups of neurons, either spatially or by channel.**

Outline

- Introduction
- Contributions
- **Related Works**
- Inverting Method
- Representation
- Experiments
- Demo

Related Work #1 SIFT and Dense-SIFT

Reconstruct an image given keypoint of SIFT based on a huge database

Appearance descriptor Vi;
Coordinates of region's center Xi;
Scale Si;
Dominant gradient orientation Oi;
Region support information Ai.

Image from: P. Weinzaepfel, H. J´egou, and P. P´erez. Reconstructing an image from its local descriptors. In CVPR, 2011.

Related Work #1 Dense-SIFT

Figure 4. From left to right: the original picture and the reconstruction before and after completion of uncovered regions.

1. For each query appearance descriptor $\mathbf{v}_i$, search its nearest neighbor in the descriptor database

$$j^* = \arg \max_{j \in \{1 \cdots m\}} \|\mathbf{v}_i - \mathbf{v}_j\|_2, \qquad (1)$$

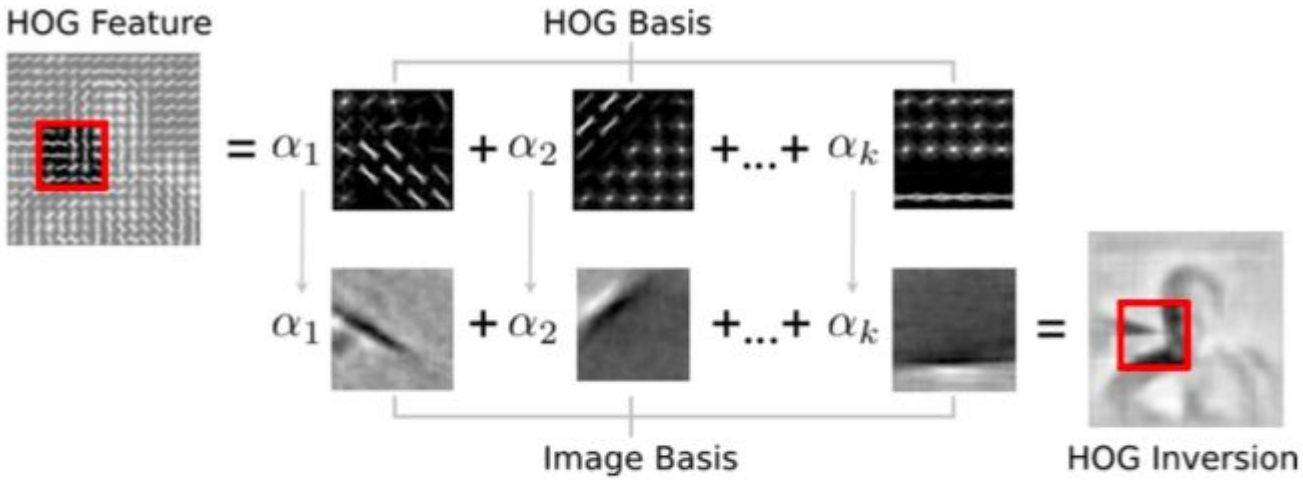
and recover the corresponding elliptic image patch

$$\mathbf{q}_j^* = \mathbf{I}_{k(j^*)}(S_{j^*}). \qquad (2)$$

2. Seamlessly stitch all patches $\mathbf{p}_i$, $i = 1 \cdots n$, together to obtain a partial reconstruction with support $S = \cup_{i=1}^{n} S_i \subset \Omega$ (see details below) .

3. Complete remaining empty zone $\bar{S} = \Omega \setminus S$ by smooth interpolation, as shown in Figure 4 (see details below).

Related Work #1 HOG and HOGgles



Related Works #2

- Visualizing and understanding convolutional networks(DeConvNet).
  In ECCV, 2014.
  - Backtrack the network computations to identify which image patches are responsible for certain neural activations

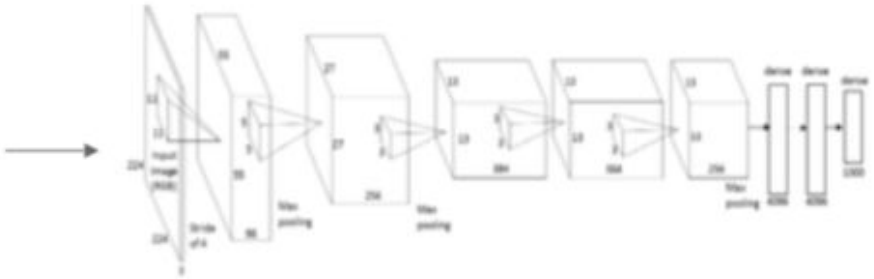
- Visualising image classification models and saliency maps.

  ICLR, 2014

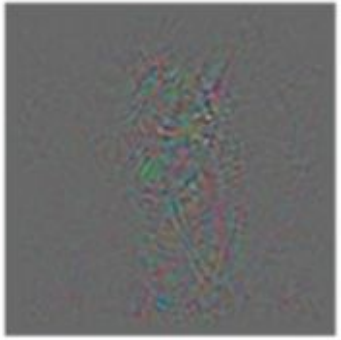
  - Reconstruction via maximizing class-neuron scores

- Visualizing and understanding convolutional networks(DeConvNet).

1. *Feed image into net*

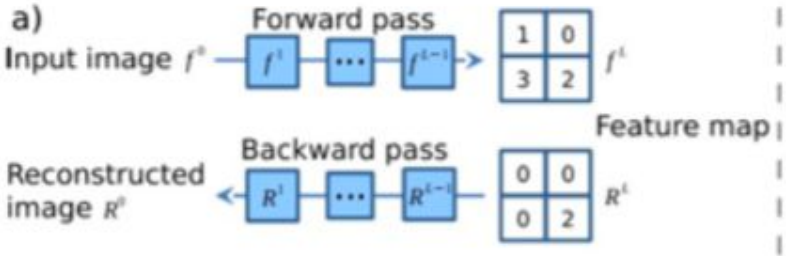


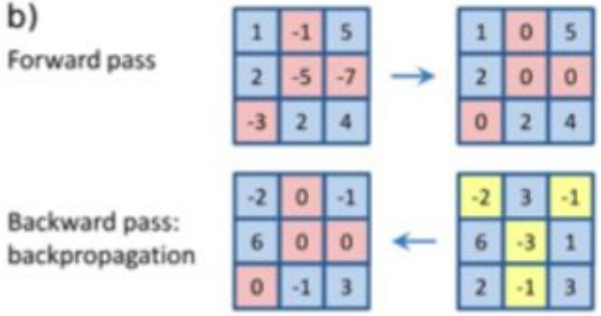
2. *Pick a layer, set the gradient there to be all zero except for one 1 for some neuron of interest*
3. *Backprop to image:*



"Guided backpropagation:" *only propagate positive gradients*

[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]
[Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al., 2014]
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]





c)

activation: $f_i^{l+1} = relu(f_i^l) = \max(f_i^l, 0)$

backpropagation: $R_i^l = (f_i^l > 0) \cdot R_i^{l+1}$, where $R_i^{l+1} = \frac{\partial f^{out}}{\partial f_i^{l+1}}$

guided backpropagation: $R_i^l = (f_i^l > 0) \cdot (R_i^{l+1} > 0) \cdot R_i^{l+1}$

* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

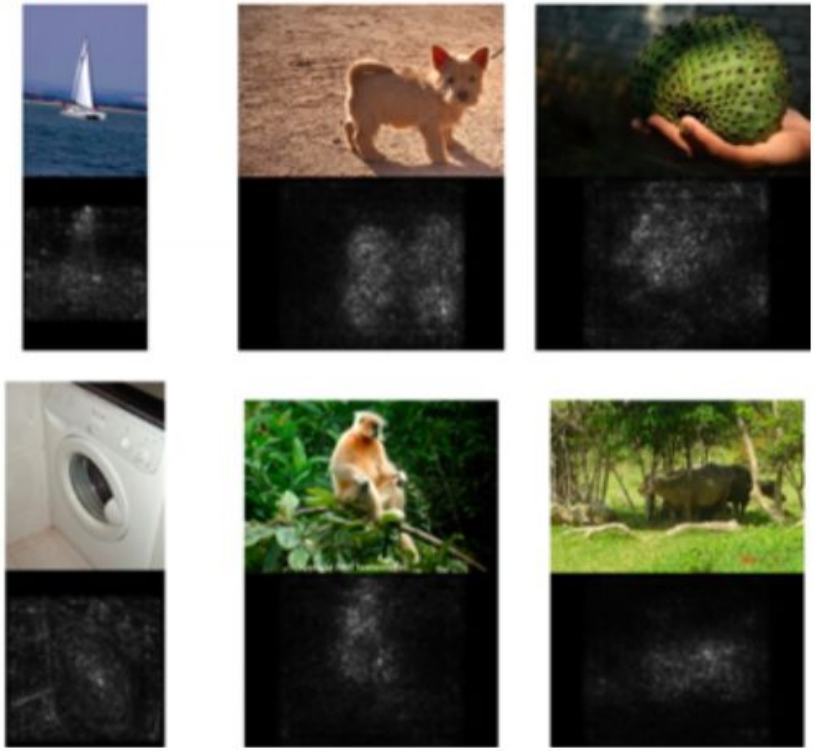
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014

2. Visualize the Data gradient:

(note that the gradient on data has three channels. Here they visualize M, s.t.:

$$M_{ij} = \max_c |w_{h(i,j,c)}|$$

(at each pixel take abs val, and max over channels)



* Original slides borrowed from Andrej Karpathy and Li Fei-Fei, Stanford cs231n

Outline

- Introduction
- Contributions
- Related Works
- **Inverting Method**
- Representation
- Experiments
- Demo

Inverting Method

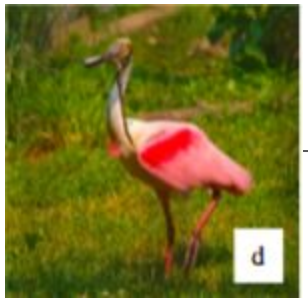
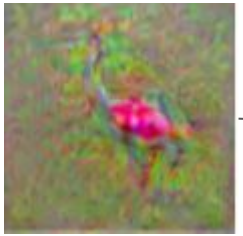


Image x_0



Reconstructed image x*

ϕ_0

Representation

$\phi(x^*)$

Inverting Method

- **To solve equation**

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

Loss: compare image representation and target one

R: regulariser to capture a natural image prior

- **Given:**
  - **Representation function** $\Phi : \mathbb{R}^{H \times W \times C} \to \mathbb{R}^d$
  - **Representation:** $\Phi_0 = \Phi(\mathbf{x}_0)$
- **Find: x* that minimizes the objective**

Inverting method - loss

Loss Function: Euclidean distance $\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$

Objective equation:

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

Code:

```
# Let's suppose we are using AlexNet and the representation is obtained in its 3rd fully connected (fc) layer
# h is the representation of image x given by the activations of the 3rd fc layer of the CNN.
h = cnn_fc3(x)
```

```
xr = argmin( lambda x: sum( ((cnn_fc3(x)-h)**2) ) )
```

```
xr = argmin( lambda x: sum( ((cnn_fc3(x)-h)**2) + regularizer(x) ) )
```

Code: http://cesarsalgado.com/ipython_notes_on_understanding_deep_image_representations_by_inverting_them/

Inverting method - regularisers

Regularisers: *α*-norm image prior

- ***α*-norm (*α* = 6) (X is vectorised and mean-subtracted image)** $\mathcal{R}_\alpha(\mathbf{x}) = \|\mathbf{x}\|_\alpha^\alpha$

```
def six_norm(x):
    return sum(x**6)
```

- **Example:**
  - **Assume image x with shape(H,W,C) H = 2, W = 2, C = 3(RGB)**

```
>>> x = np.random.randn(2,2,3)
>>> x
array([[[ 0.47451494,  0.29584752,  2.03439944],
        [-2.14093253,  0.68461236,  0.91091761]],

       [[ 1.06140708,  0.28991584, -0.30025526],
        [-0.34013064, -0.28461196,  0.84689234]]])
```

  - **Then *α*-norm will be:**

```
>>> np.sum(x**6)
169.68199780729023
```

Code: http://cesarsalgado.com/ipython_notes_on_understanding_deep_image_representations_by_inverting_them/

Inverting method - regularizes

Regularisers: TV-norm image prior

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

```
def total_variation(x):
    beta = 2
    result = 0.0
    # suppose x is properly zero padded.
    for i in xrange(x.shape[0]):
        for j in xrange(x.shape[1]):
            result += ( (x[i,j+1]-x[i,j])**2 + (x[i+1,j]-x[i,j])**2 )**(beta/2)
    return result
```

- **Example: total_variation(x) will be with shape(3,)**

```
>>> total(x)
array([ 7.18500788,  0.15117329,  6.71282397])
```

Inverting method - regularizes

Objective function: $\operatorname{argmin} \; \|\Phi(\mathbf{x}) - \Phi_0\|^2 + \lambda_\alpha \mathcal{R}_\alpha(\mathbf{x}) + \lambda_{V^\beta} \mathcal{R}_{V^\beta}(\mathbf{x})$

Regulariser defined as sum of both subterms

Code:

```
def regularizer(x):
    mult1 = 2.6*1e8
    mult2 = 5
    return mult1*six_norm(x) + mult2*total_variation(x)
```

Example: regularizer result will be with shape(3,)

```
>>> regularizer(x)
(3,)
array([  4.41173195e+10,   4.41173194e+10,   4.41173195e+10])
```

Inverting method - loss and regulariser

- **Balancing loss and regularizer**

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

  - **Normalized Loss**
  - ***σ*: average Euclidean norm of natural images in a training set**
  - **B = 128 *σ*x ∈[-B,B]**
  - $\mathcal{R}_\alpha(\mathbf{x}) \approx HWB^\alpha/\sigma^\alpha \quad \lambda_\alpha \approx \sigma^\alpha/(HWB^\alpha)$
  - $\lambda_{V^\beta} \approx \sigma^\beta/(HW(aB)^\beta)$ **(2.16 * 10^8)**

Inverting method - Optimisation

- **Gradient Descent(GD)**
- **GD extensions**
  - **Momentum**

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

m = 0.9

Weighted avg

Learning rate

$$E(\mathbf{x}) = \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

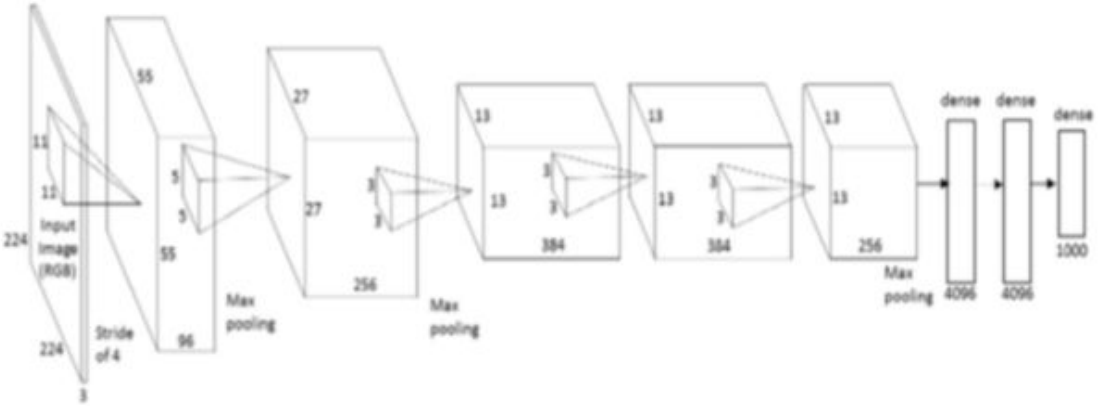
  - **Computing derivatives**
    - **CNNs : backpropagation**
    - **CNN-HOG and CNN-DSIFT: same as CNNs**

Outline

- Introduction
- Contributions
- Related Works
- Inverting Method
- **Representation**
- Experiments
- Demo

Representations - CNN-A(Caffe-Alex) deep networks

- **CNN-A structure**



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

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

Representations - CNN-DSIFT and CNN-HOG

How DSIFT and HOG implemented in CNNs:

- **Step1: Computing and binning image gradients**
- **Step2: Pooling binned gradients into cell histograms**
- **Step 3: Grouping cells into blocks (CNN layers)**
- **Step 4: Normalising the blocks (CNN layers)**

Outline

- Introduction
- Contributions
- Related Works
- Inverting Method
- Representation
- **Experiments**
- Demo

Experiments with shallow representation (HOG & SIFT)

- **Quantitative Analysis**

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

- **Error:** $\|\Phi(\mathbf{x}^*) - \Phi(\mathbf{x}_i)\|_2 / \mathrm{N}_\Phi$
- **Qualitative Analysis**

Experiments with shallow representation (HOG & SIFT)

Comparison HOG and HOGle

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

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

Experiments with shallow representation (HOG & SIFT)

Comparison HOG , DSIFT(quantative)

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

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

Experiments with deep representation (CNN-A)

- **Train Data: 1.2M images of the ImageNet ILSVRC 2012**
- **Validation Data: 100 ILSVRC validation images**
- $\lambda_\alpha = 2.16 \times 10^8$
- **Increasing** λ_{V^β} **ten-folds, starting from 0.5**

Experiments with deep representation (CNN-A)

Quantitative Analysis

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

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

Experiments with deep representation (CNN-A)

- **Qualitatively Analysis:**
  - **reconstruction for a image from each layer of CNN-A**

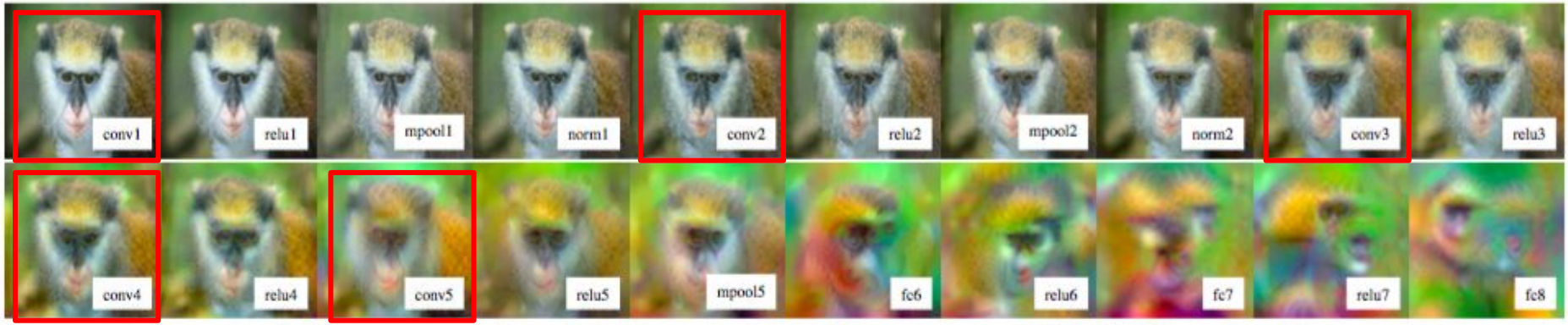


Figure 6. **CNN reconstruction.** Reconstruction of the image of Fig. 5.a from each layer of CNN-A. To generate these results, the regularization coefficient for each layer is chosen to match the highlighted rows in table 3. This figure is best viewed in color/screen.

Experiments with deep representation (CNN-A)

- **Qualitatively Analysis:**
  - **reconstruction for a image from each layer of CNN-A**

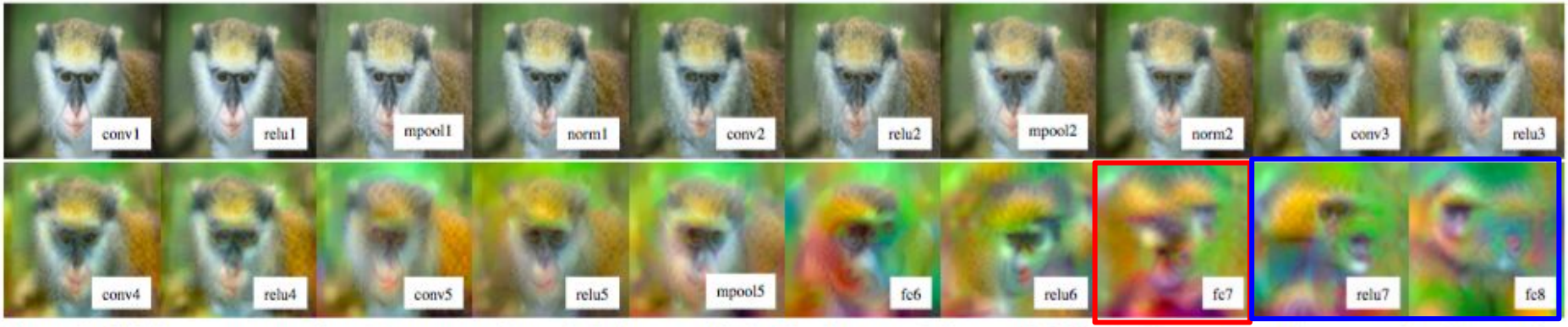


Figure 6. **CNN reconstruction.** Reconstruction of the image of Fig. 5.a from each layer of CNN-A. To generate these results, the regularization coefficient for each layer is chosen to match the highlighted rows in table 3. This figure is best viewed in color/screen.

Experiments with deep representation (CNN-A)

- **Reconstructions obtained from subset of representation in different CNN layers**

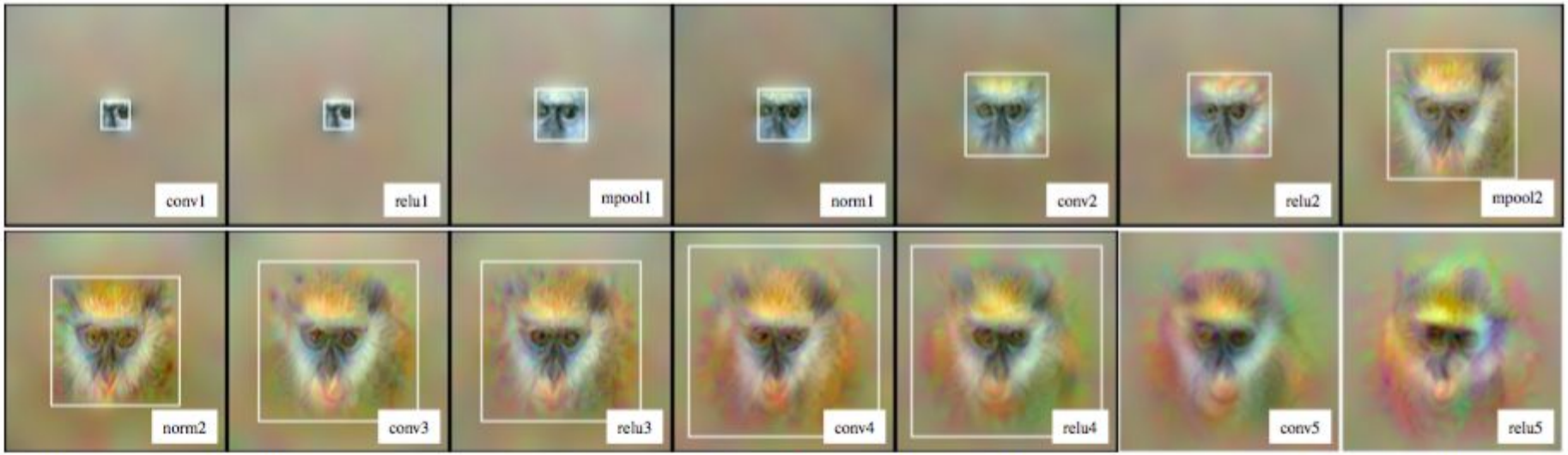


Figure 9. **CNN receptive field.** Reconstructions of the image of Fig. 5.a from the central 5×5 neuron fields at different depths of CNN-A. The white box marks the field of view of the 5×5 neuron field. The field of view is the entire image for conv5 and relu5.

Weakness and Future Work

- **Weakness**
  - **Generate an image with the feature representation that similar to the given representation, not the image similar to the given image**
  - **Error detect for comparing the original image representation and the inverse image representation**
  - **Optimization implements at test time, requires computed gradient of feature representation, makes it relatively slow**
  - **Implementation on only AlexNet, but not other state-of-art CNN**
  - **“Spikes” generated by using regularizer**

- **Future Work**
  - **Implementation on other state-of-art CNN**
  - **Experiment with more expressive natural image priors**
  - **Extract subsets of neurons that encode object parts and try to establish sub-networks that capture different details of the image**

Recent work

Inverting visual representations with convolutional networks

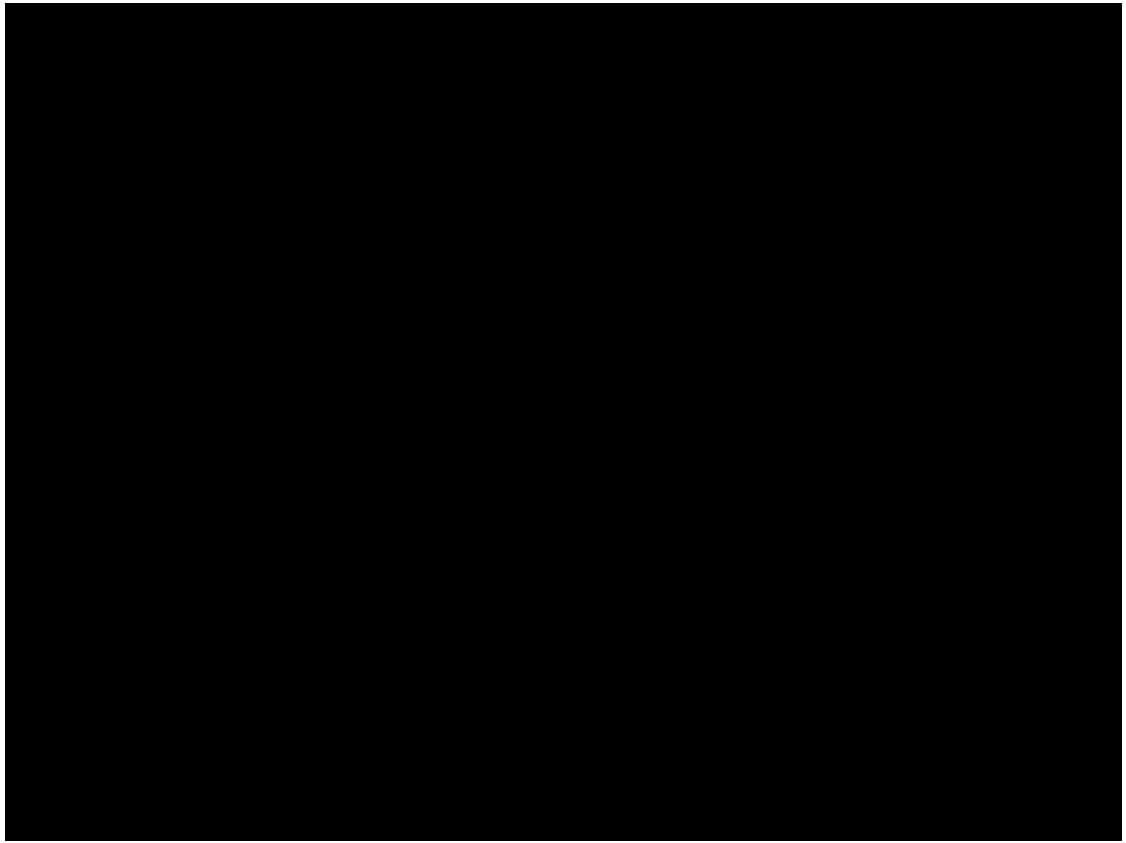
Dosovitskiy, Alexey, and Thomas Brox

Visualizing Deep Convolutional Neural Networks Using Natural Pre-images

Aravindh Mahendran, Andrea Vedald

http://link.springer.com/article/10.1007/s11263-016-0911-8

Demo



Original video:
http://techtalks.tv/talks/understanding-deep-image-representations-by-inverting-them/61629/