[M2, MVA]

Deep Learning

Maha ELBAYAD
maha.elbayad@student.ecp.fr

Final Assignment

February 11, 2016

Part 1. Personal research questions

Question 1.1 the top-5 error rate is the fraction of test images for which the correct label is not among the five labels considered most probable by the model. In the case of big image category datasets (ImageNet with over 22k categories) many synset (sub-categories) within the same high-level category are subtle and confusing even for human annotators.

Question 1.2 Data augmenation e.g rotating, translating, mirroring or illumination change artificially enlarges the training set and prevents over-fitting as it enhances the samples variance within the same class (in the context of classification problems). The resulting models are thus invariant to geometric and illumination transformations. The same goes for Data corruption e.g distorting and adding noise to the data helps the model generalization and improves its robustness to input noise.

Question 1.3 AdaGrad's optimization scheme gives frequently occurring features very low learning rates and infrequent features high learning rates. Hence it improves the convergence especially for sparse problems and it is likely to find more discriminative features and filters for the model. Concretely, it tunes the learning rate at each timestep regarding the history of the gradients for each feature dimension. The convergence is faster than with standard SGD but if the training data isn't variant enough, the monotonically decreasing learning rate of AdaGrad would stop learning too early.

Question 1.4 The authors incorporate the order of strokes into their framework by adopting a multi-channel model. The input sketch strokes are discretized into three sequential groups and the 6 compbinations are fed to the network. This way the CNN can learn the object's outline from the early strokes and give less weight to the detail strokes that come later. However, as most sketches start with some basic shapes (circles, ellipses, triagles) the first channel would mislead the learning.

Question 1.5 To deal with different sketching scales and abstraction levels the authors use an ensemble model with 5 CNN each trained on blurred images with variant levels. The penultimate layer (fully connected with 512 outputs) of all 5 networks are then concatenated and the Joint Bayesian method is used to learn a metric in the features space. The final classification is achieved with KNN matching on the learnt metric space. The simplest

decision-level fusion strategy would be averaging the 5 softmax scores which will treat the networks equally. We can also feed the concatenated feature vector to a classifier say kernel-SVM or LDA. A boltzmann machine might also be used to learn the optimal aggregated feature.

Part 2. Deep dreaming in MatConvNet

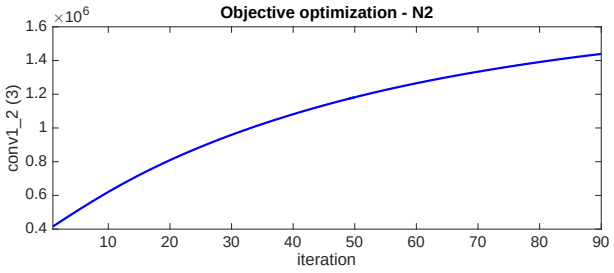
Question 2.1 Our objective function is $z = \|x^{(l+1)}\|_2^2 = \langle x^{(l+1)} | x^{(l+1)} \rangle$.
Thus, the gradient with regard to $x^{(l+1)}$ is:

$$\frac{dz}{dx^{(l+1)}} = \langle x^{'(l+1)} | x^{(l+1)} \rangle + \langle x^{(l+1)} | x^{'(l+1)} \rangle = 2x^{(l+1)}$$

Question 2.3 The last five layers {`fc6, relu6, fc7, fc8, prob`} are the fully connected layers which are encoded in matconvnet as convolution layers, plus a rectifier. `fc6` and `fc7` output 4096 neurons and the final `fc8` yields 1000 outputs (=|classes|) which are then treated with a softmax layer to get the probability of each class given the input as $\mathbb{P}(class = i|input) = \frac{\exp x_i}{\sum_j \exp x_j}$.

Question 2.4 We deep-dream on 3 different pictures with the vgg-16 at different layers and with a fixed step-size (:=1.5). Figures 1 to 5 illustrate the ascent on the norm of the target layer during 50 iterations of forward-backward updates.

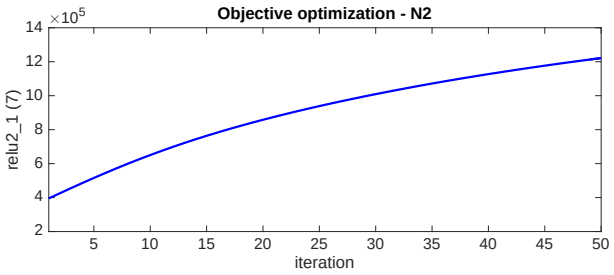
(a) Original (b) 20 iterations (c) 50 iterations (d) 70 iterations



(e) Objective function

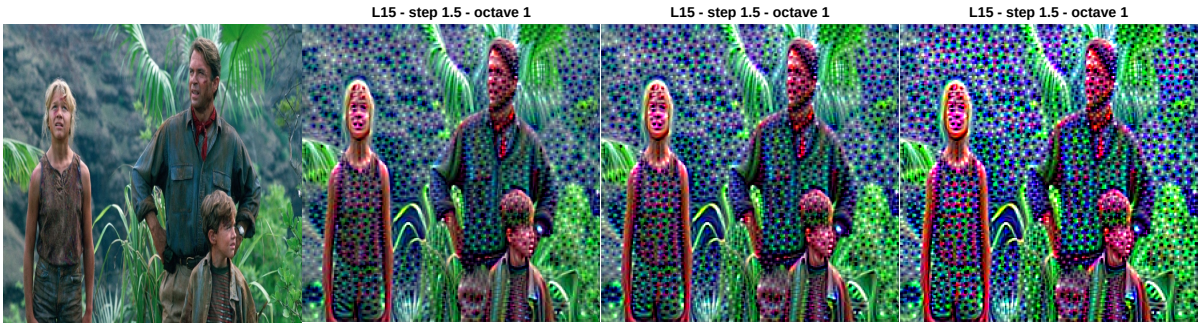
Figure 1: Image (1) - Layer 3 Step-size 1.5

(a) Original (b) 20 iterations (c) 35 iterations (d) 50 iterations

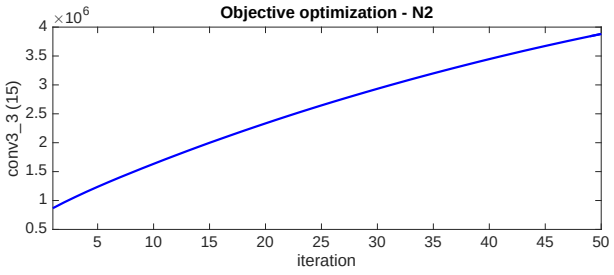


(e) Objective function

Figure 2: Image (1) - Layer 7 Step-size 1.5

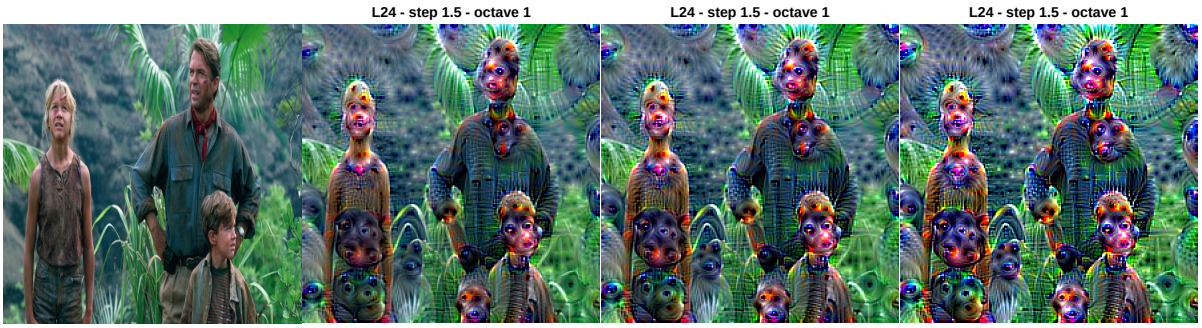


(a) Original (b) 20 iterations (c) 30 iterations (d) 50 iterations

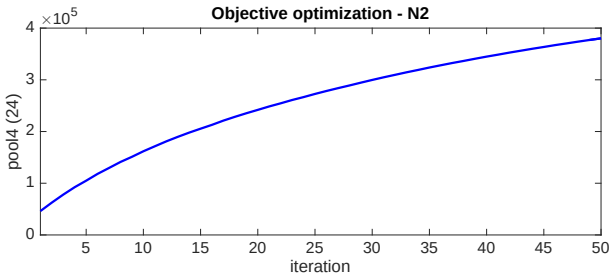


(e) Objective function

Figure 3: Image (1) - Layer 15 Step-size 1.5

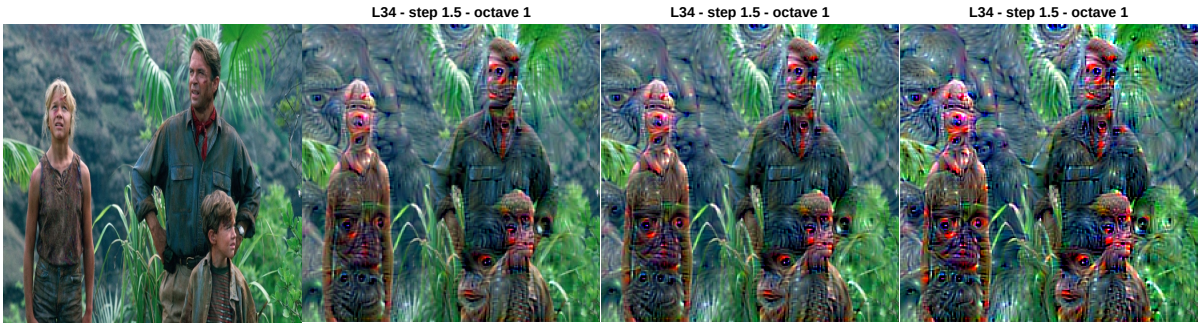


(a) Original (b) 20 iterations (c) 30 iterations (d) 50 iterations

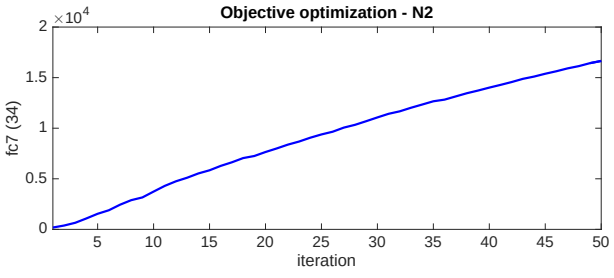


(e) Objective function

Figure 4: Image (1) - Layer 24 Step-size 1.5



(a) Original (b) 20 iterations (c) 30 iterations (d) 50 iterations



(e) Objective function

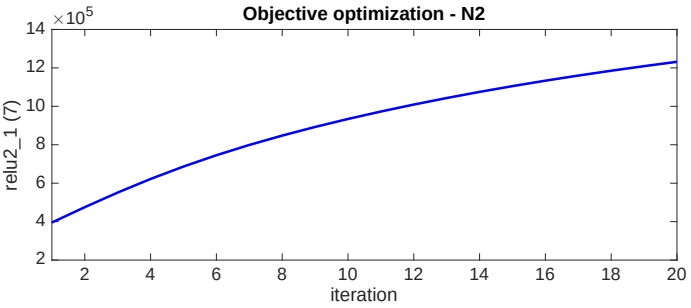
Figure 5: Image (1) - Layer 34 Step-size 1.5

We note that deepdreaming on the low level layers enhances the contrast and edges of the input image while deepdreaming on intermediate-level layers adds some patterns to the image without alternating the nature of the present objects while on the top-level layers the process tends to alter the image to match the categories the CNN was trained on, in this case 1000 imagenet synsets of mainly animals and plants.

When running the process for up to 50 iterations, the objective seems to converge and the output images are quite stabilized.
We can also tune the step-size, and generally with larger step the convergence is faster. However, since we normalize the gradient at each update, we're less sensible to the tuning of the step-size:

$$input := input + step\frac{g}{norm(g)}$$

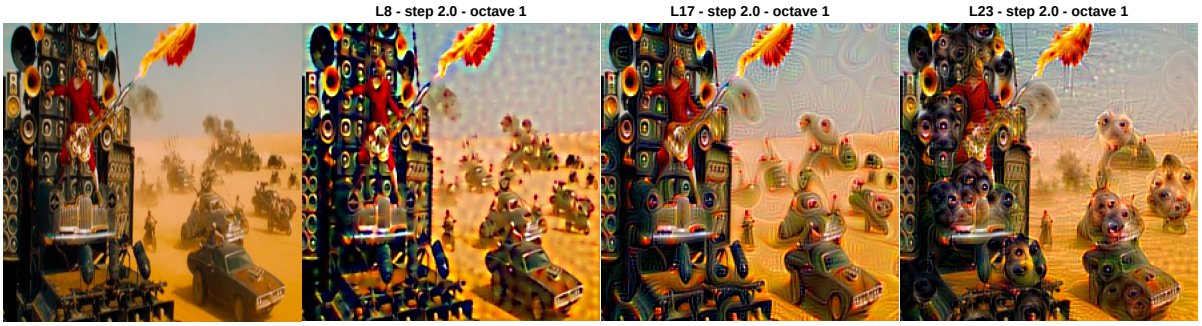
(a) step 1.5 - 20 iterations (b) step 2 - 20 iterations



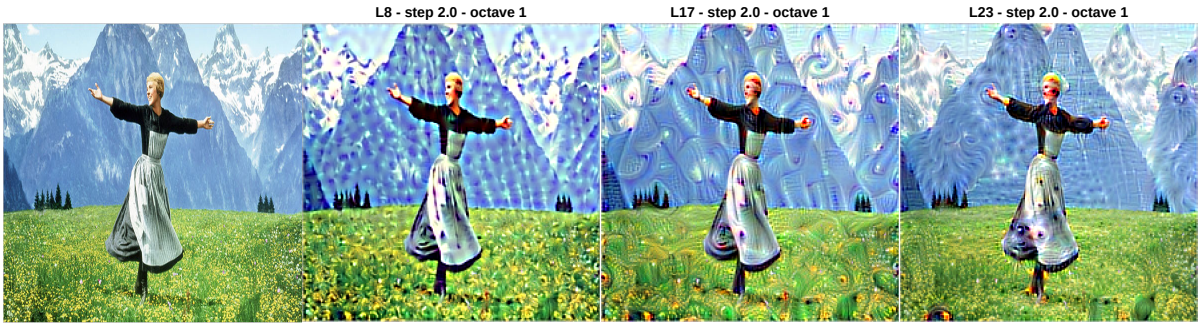
(c) step 4 - 20 iterations

Figure 6

Other input images at different layers:



(a) original (b) Layer 8 (c) Layer 17 (d) Layer 23



(e) original (f) Layer 8 (g) Layer 17 (h) Layer 23

Figure 7

Question 2.5 When giving the network an image with different size, the convolution layers still work fine except when they're intended to be fully connected layers as a matching of dimensions give the required output but when

the inout size is different the matching doesn't hold.

Question 2.6 Instead of deepdreaming on the whole input image we process zoomed crops of the image. the crops are either picked randomly or in order following a grid.

Figure 8: Layer 23 - Multi-channel crops on a 5x5 grid with zooming scale=2

Figure 9: Layer 17 - Multi-channel crops on a 3x3 grid with zooming scale=2

We can also process the whole image at increasing scales as in the deep dreaming ipython notebook:

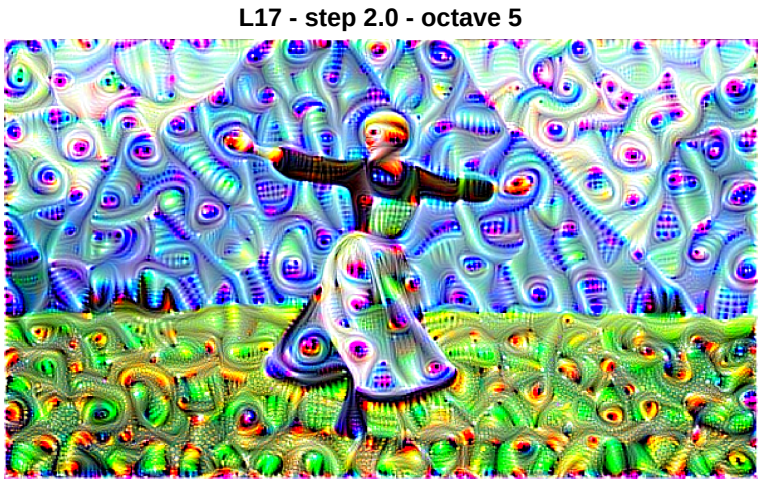
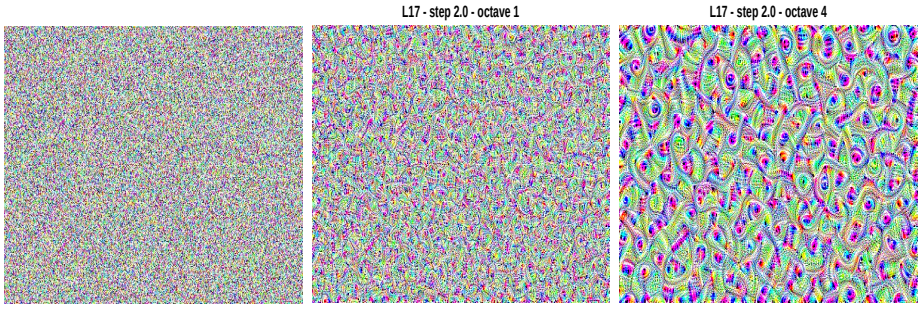


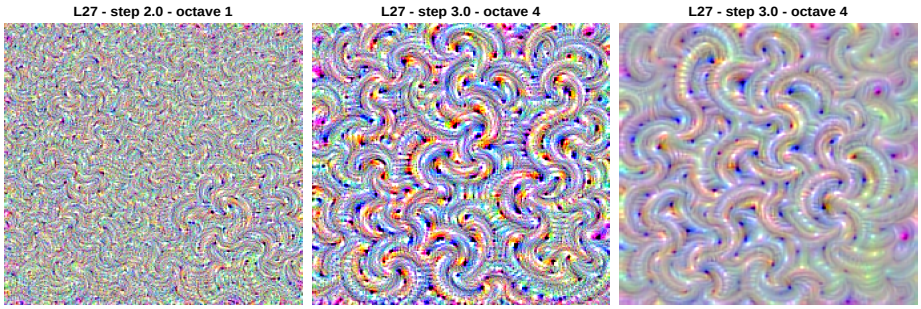
Figure 10: Layer 17 - 5 octaves with zooming scale=1.2

Question 2.7 Rather than starting from a ntural image we can deepdream from a white noise. We experiment with different settings for zooming and blurring:



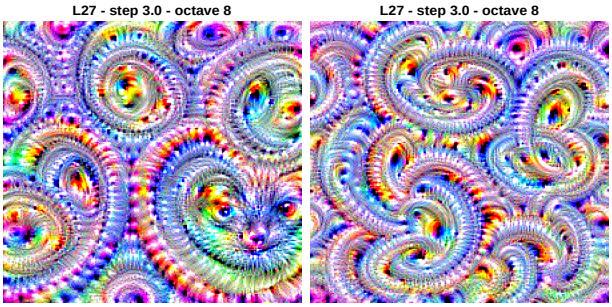
(a) original (b) L17 - straightforward (c) L17 - zooming 4 octaves

Figure 11: Layer 17 - pool3



(a) straightforward (b) zooming 4 octaves (c) zooming 4 octaves blurring $\sigma = .5$

Figure 12: Layer 27 - conv5_2



(a) zooming 8 octaves (b) zooming 8 octaves jittering$\{-8, 8\}$

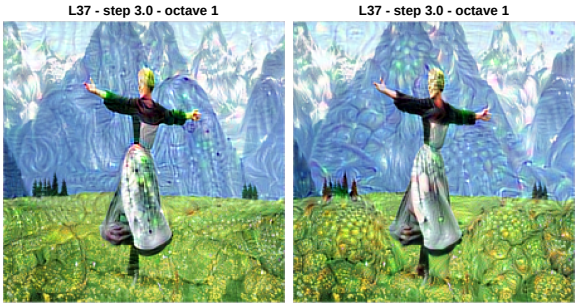
Figure 13: Layer 27 - conv5_2

Part 3. Personal experiments

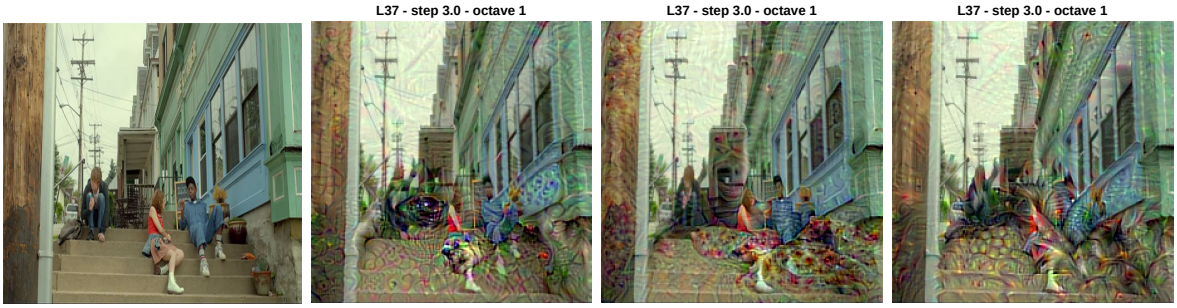
3.1 Experiment 2

3.1.1 Single neurone

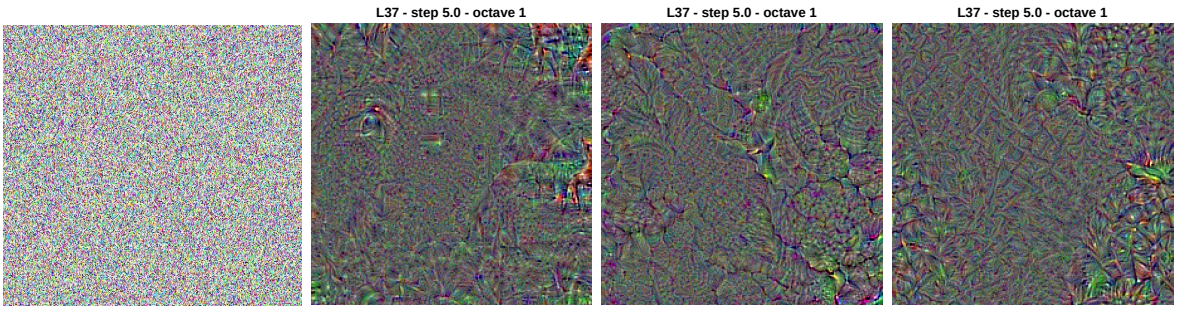
Instead of maximizing the N2 of the target layer we change the objective function to the activation of a subset of neurons from the target layer. We select a single class in the probabilities layer (37) to sort of visualize the DNN's perception of this category, especially when starting from random noise (with $\lambda = .03$ regularization). Although we expected a much more explicit perception of the class we end up most of the time with some characteristic patterns of the class even if the objective function is at its maximum (i.e 1).



(a) Neurone 944 (cucumber) (b) Neurone 954 (pineapple)



(c) Original (d) Neurone 255 (pug) (e) Neurone 964 (pizza) (f) Neurone 954 (pineapple)

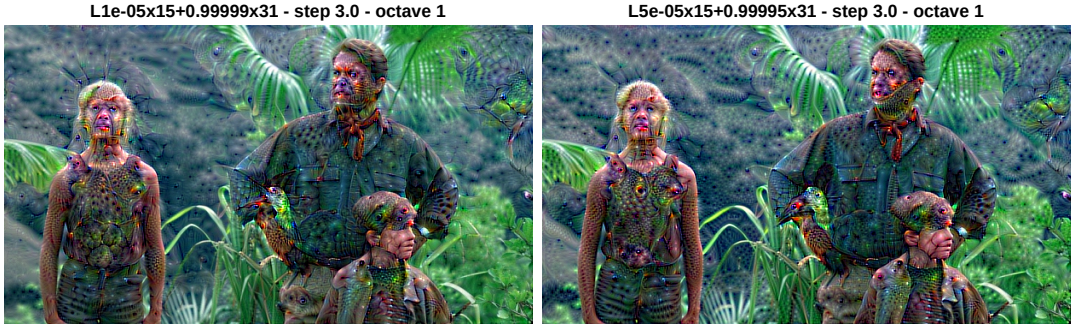


(g) Original (h) Neurone 275 (dhole) (i) Neurone 939 (cauliflower) (j) Neurone 954 (pineapple)

Figure 14: Single neurone activation (with regularization)

3.1.2 Multiple layers

We can deep-dream on multiple layers by cumulating the gradients in the backpropagation step. Here are some results for combining 2 layers with different weight for each. We note that the result is a combination of simple deep-dreaming on respective layers and that the weights allow us to tune down the activation values of layers that tend to overshodw the rest.



(a) 1e-5 × layer 15 + 0.99999 × layer 31 (b) 5e-5 × layer 15 + 0.99995 × layer 31

(c) 1e-4 × layer 15 + 0.9999 × layer 31

(d) 0.03 × layer 8 + 0.97 × layer 17

(e) 0.3 × layer 8 + 0.7 × layer 17

(f) 0.5 × layer 8 + 0.5 × layer 17

(g) 0.01× layer 8 + 0.99 × layer 23

(h) 0.03 × layer 8 + 0.97 × layer 23

(i) 0.1 × layer 8 + 0.9 × layer 23

(j) 0.5 × layer 8 + 0.5 × layer 23

Figure 15: Multiple layers deep-dreaming

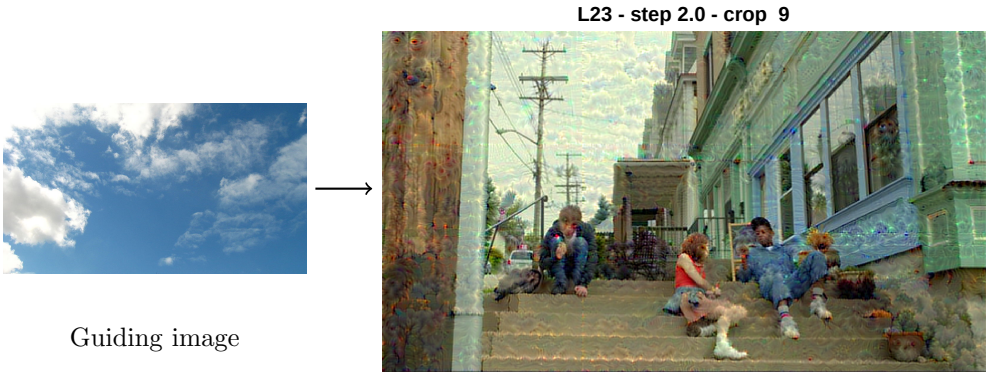
3.1.3 Controlling dreams

We maximize the dot-products between activations of input image, and their best matching correspondences from the guide image.

(a) Maximizing the N2

Guiding image

(b) Maximizing the dot product

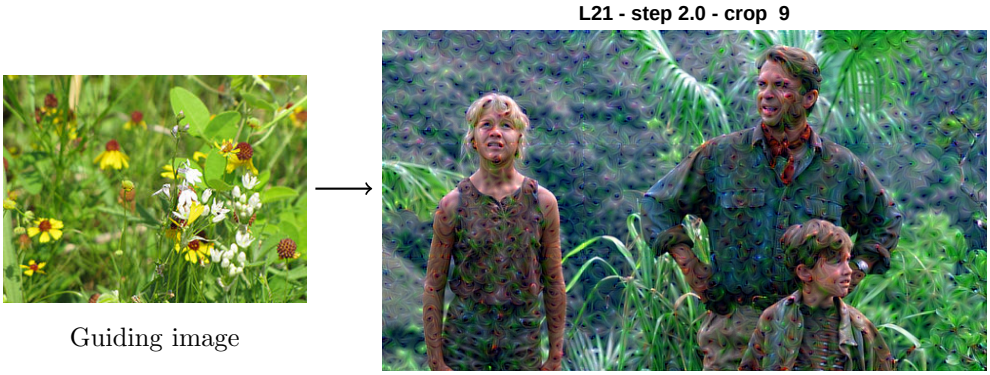


Guiding image

(c) Maximizing the dot product

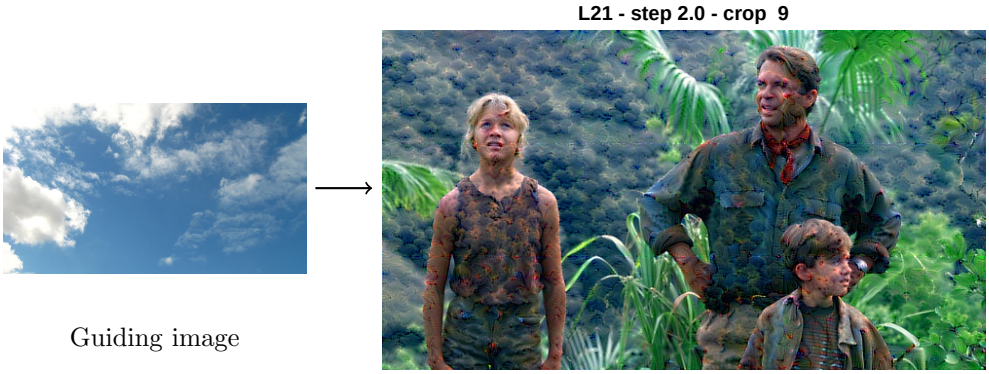
Figure 16: Layer 23 - relu4_3

(a) Maximizing the N2



Guiding image

(b) Maximizing the dot product



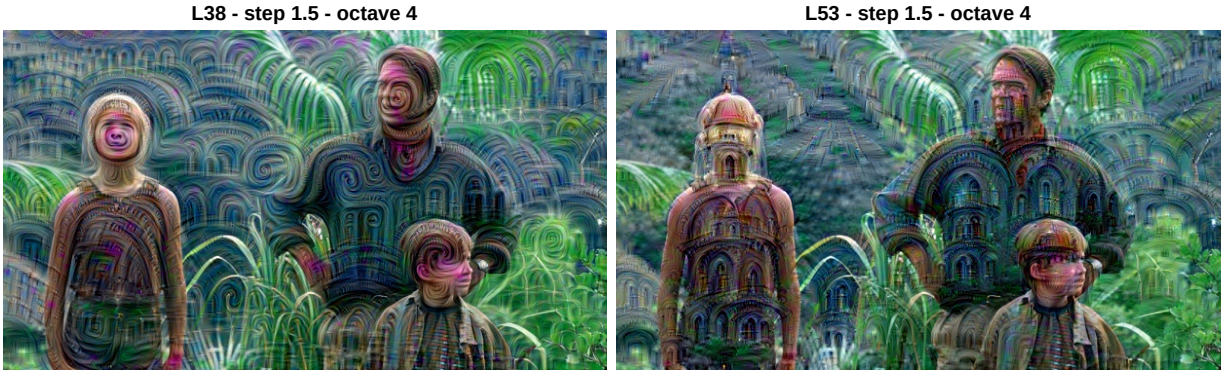
Guiding image

(c) Maximizing the dot product

Figure 17: Layer 21 - relu4_2

3.2 Experiment 3

We import the caffe model `googlenet_places205` trained on the MIT places database and use it to deepdream on some of its layers.

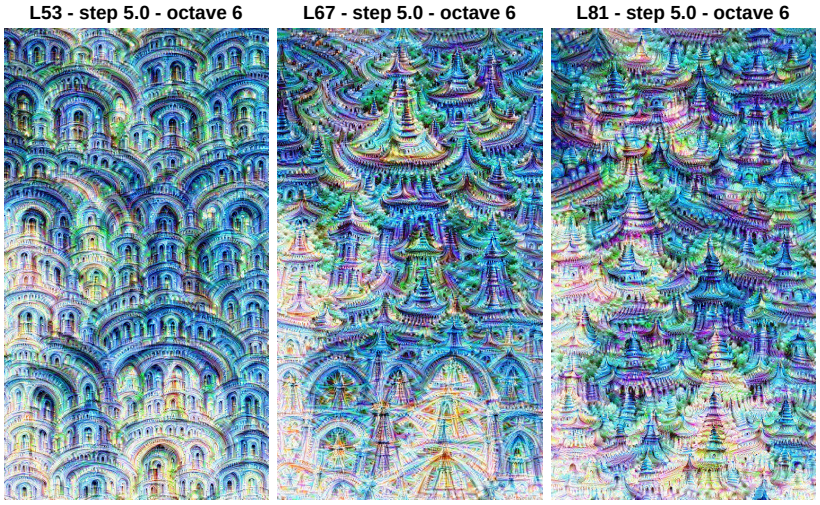


(a) Layer 38 - inception_3b_output (b) Layer 53 - inception_4a_output

(c) Layer 67 - inception_4b_output (d) Layer 81 - inception_4c_output

Figure 18: Googlenet places205

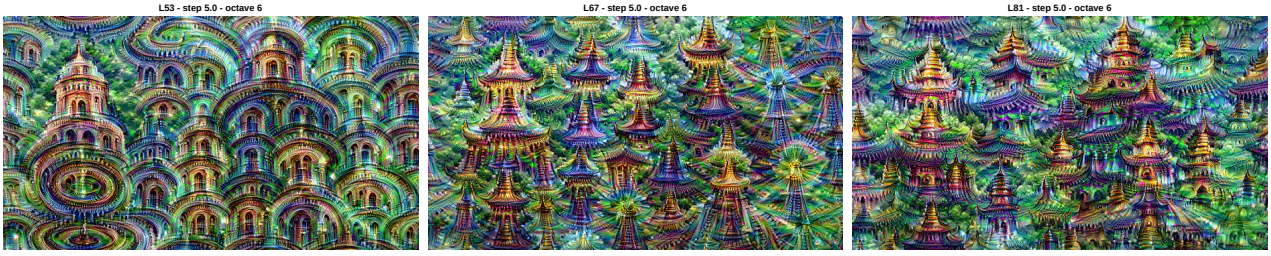
After iterative zooming we can add space like patterns to the sky image (figure 16c)



(a) layer 53 - inception_4a_output (b) layer 67 - inception_4b_output (c) layer 81 - inception_4c_output

Figure 19: 6 zoomings at scale=1.5 step=5 iterations=20

We apply the same process to any natural image (e.g figure 2a) with a large step to completely alter the image



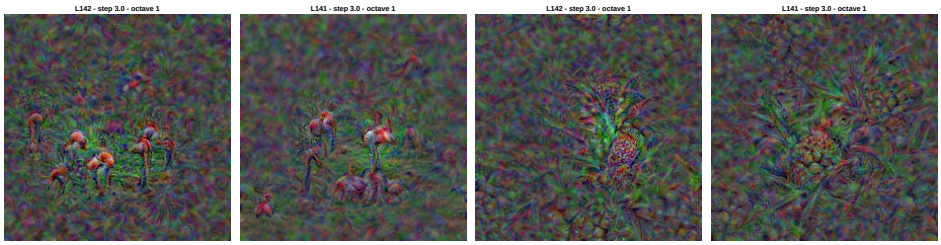
(a) layer 53 - inception_4a_output (b) layer 67 - inception_4b_output (c) layer 81 - inception_4c_output

Figure 20: 6 zoomings at scale=1.2 step=5 iterations=20

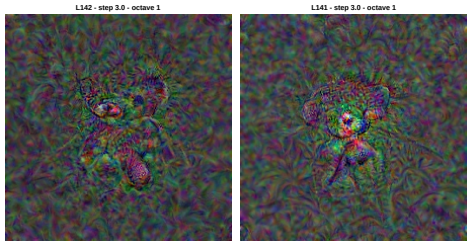
3.3 Experiment 4

We implement some of the suggested methods in [2] [1] to regularize the optimization of the DNN's activation especially when starting from random noise as the main objective of the regularization is to incorporate natural-image priors in the optimization process.

- The Gaussian blurring of width σ that we apply every τ iterations.
- The L2 regularization of the input image by maximizing $(\text{activation}(x) - \theta\|x\|_2^2)$ where x is our input image. Practically we multiply the image after the ascent by the scalar $1 - \theta$.
- Clipping pixels with small norm: we set to zero pixels with norm (across the channels) inferior to $\gamma\times$ the mean norm of all pixels.



(a) Flamingo - layer 142 (b) Flamingo - layer 141 (c) Pineapple - layer 142 (d) Pineapple - layer 141



(e) Teddy bear - layer 142 (f) Teddy bear - layer 141

Figure 21: Regularization: $\theta = 0.003$, $\tau = 8$, $\sigma = 0.7$, $\gamma = 0.0001$, $step = 3$, $iterations = 50$ - model : bvlc_googlenet

References

[1] 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] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.