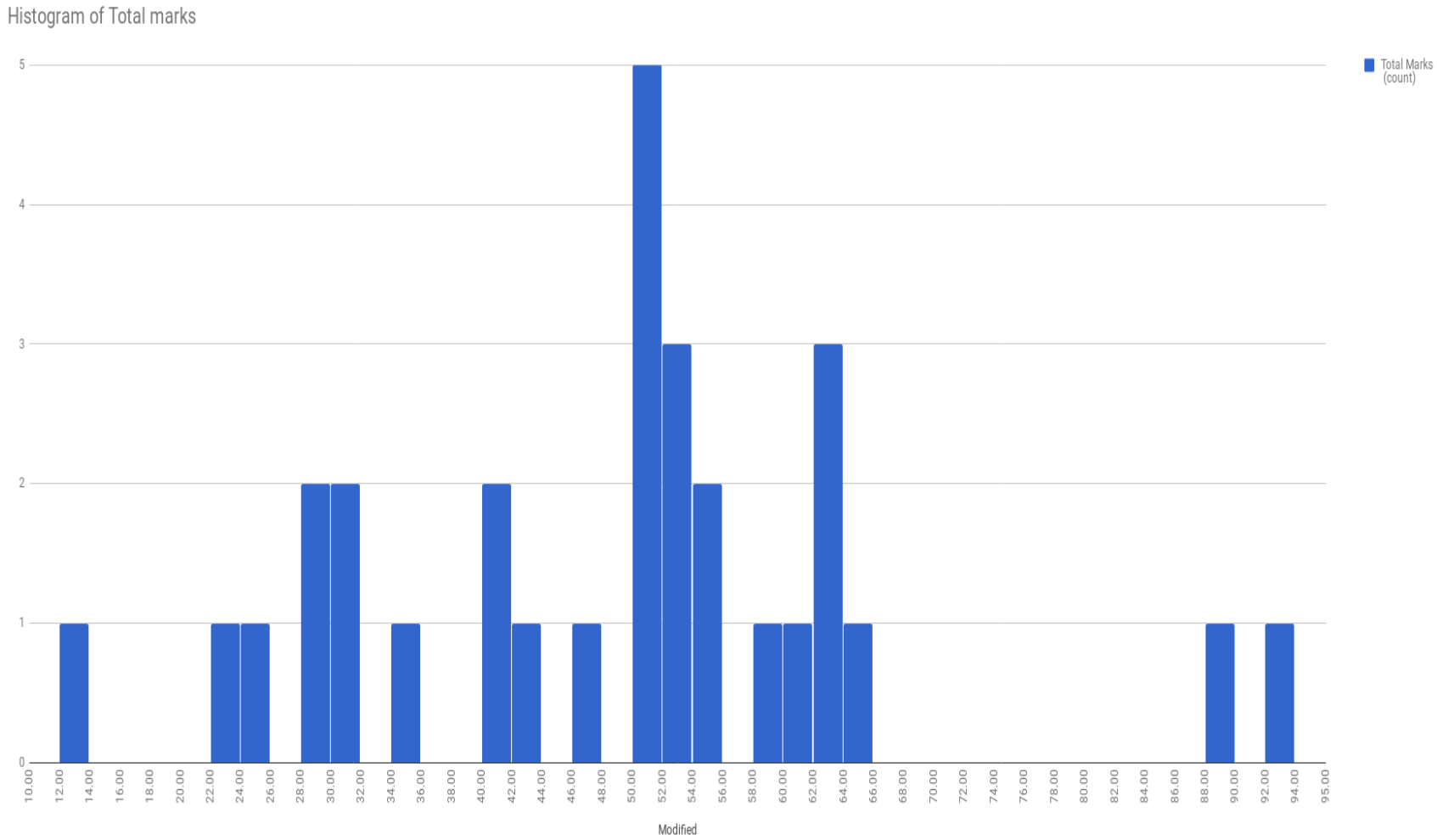
Deep Learning (for Computer Vision)

Arjun Jain

Mid-sem Performance



Question-wise breakup

Question	Response
Q1	Majority got it
Q2	Most were able to solve it
Q3	Few got the correct approach, and fewer got the second part right.
Q4	Though an easy question, maybe the question seemed daunting. Most didn't event attempt it , but we found that those who did attempt it were able to get it right
Q5	Most answers were incomplete and did not respect the matrix dimensions in the third part.
Q6	Leniently graded, although very vague or generic answers were penalized
Q7	Correctly solved by many students, well done!

Sources

A lot of the material has been shamelessly and gratefully collected from:

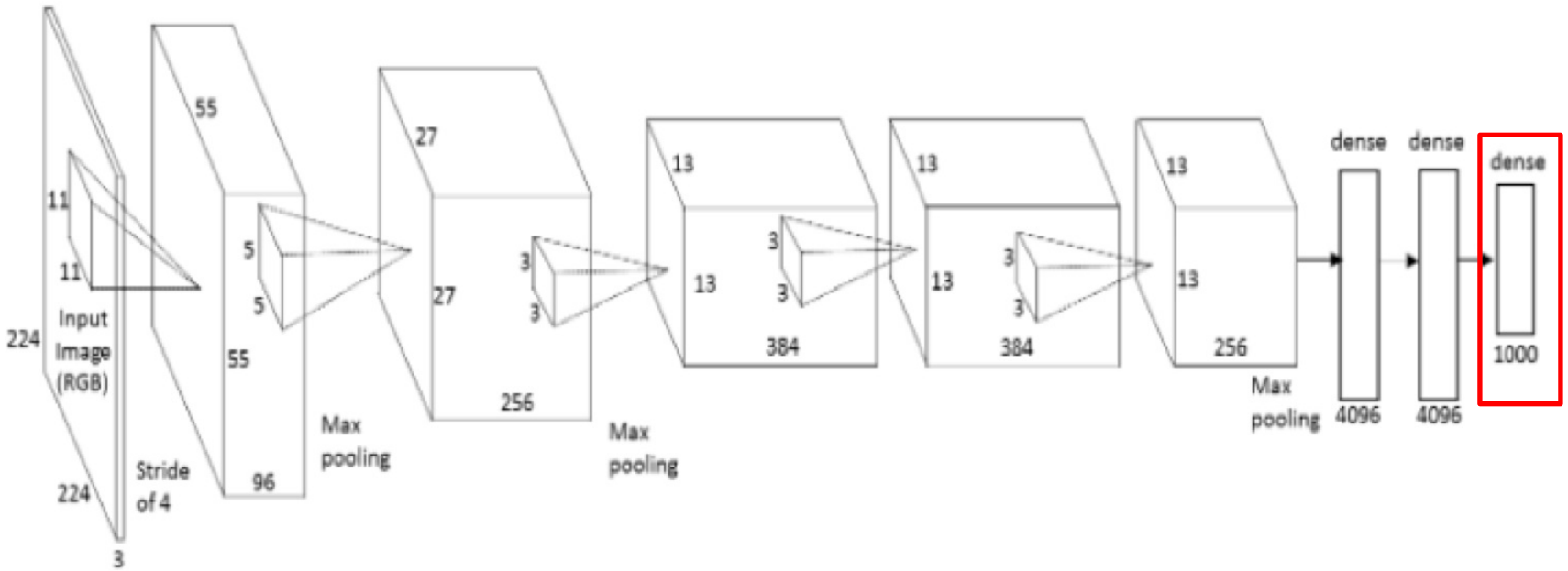
- http://cs231n.stanford.edu/

Recap:

- RNNs (how is the assignment coming along?)
- Visualizing kernels, t-SNE, occlusion experiments
- Deconv approaches, guided backprop

Visualizing and Understanding ConvNets

Optimization to Image

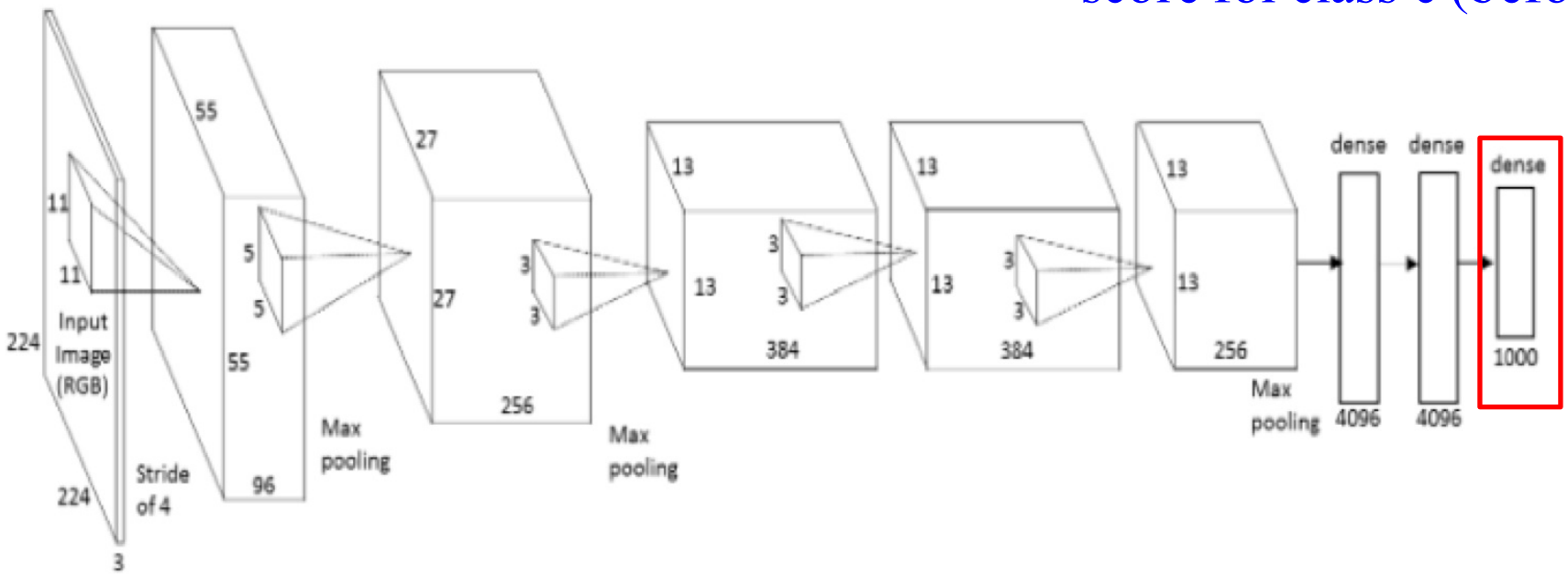


Q: can we find an image that maximizes some class score?

Optimization to Image

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

score for class c (before Softmax)

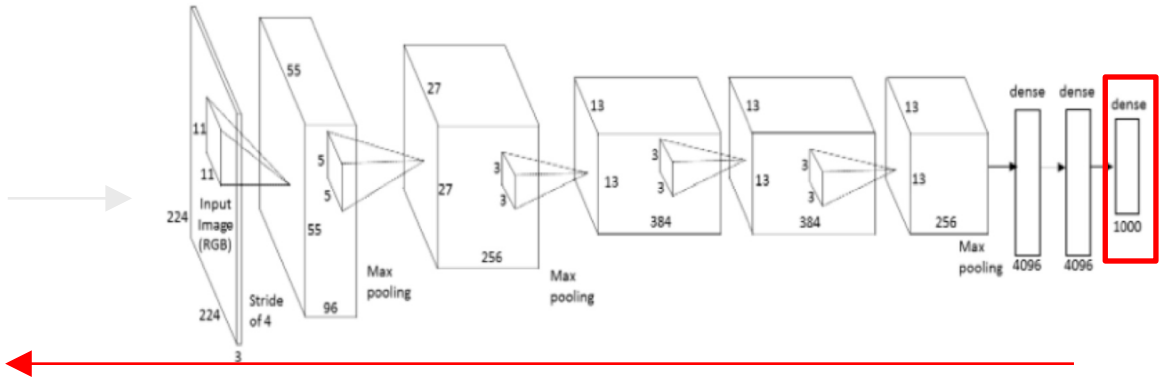


Q: can we find an image that maximizes some class score?

Optimization to Image

1. feed in zeros.

zero image

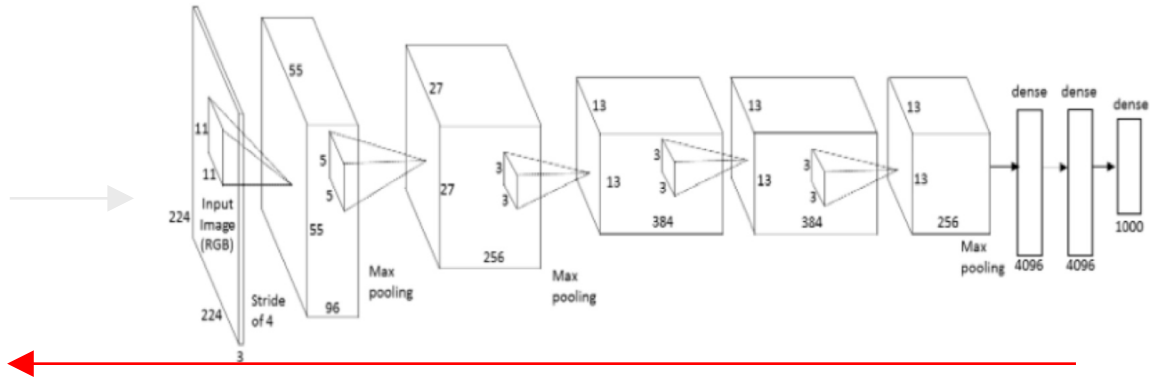


2. set the gradient of the scores vector to be [0,0,....1,....,0], then backprop to image

Optimization to Image

1. feed in zeros.

zero image



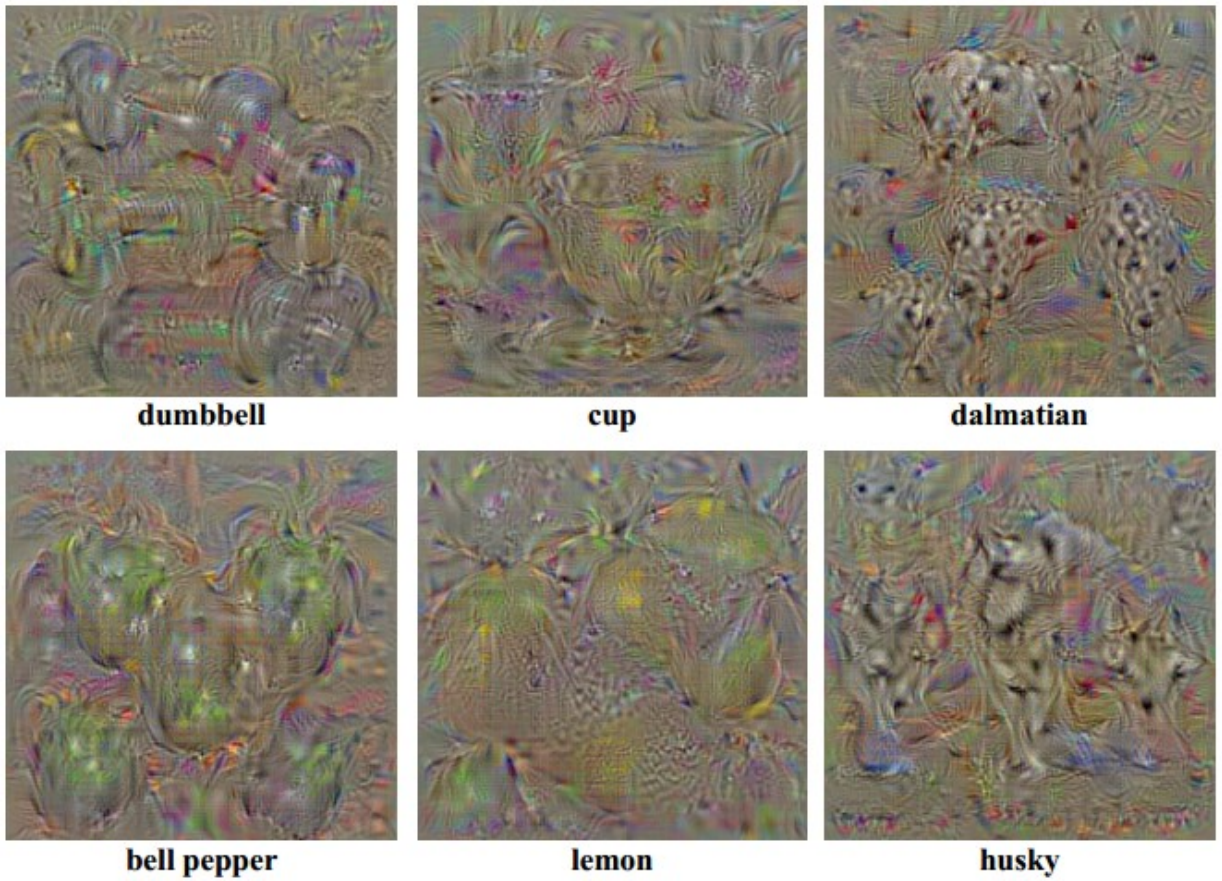
2. set the gradient of the scores vector to be [0,0,....1,....,0], then backprop to image
3. do a small "image update"
4. forward the image through the network.
5. go back to 2.

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

score for class c (before Softmax)

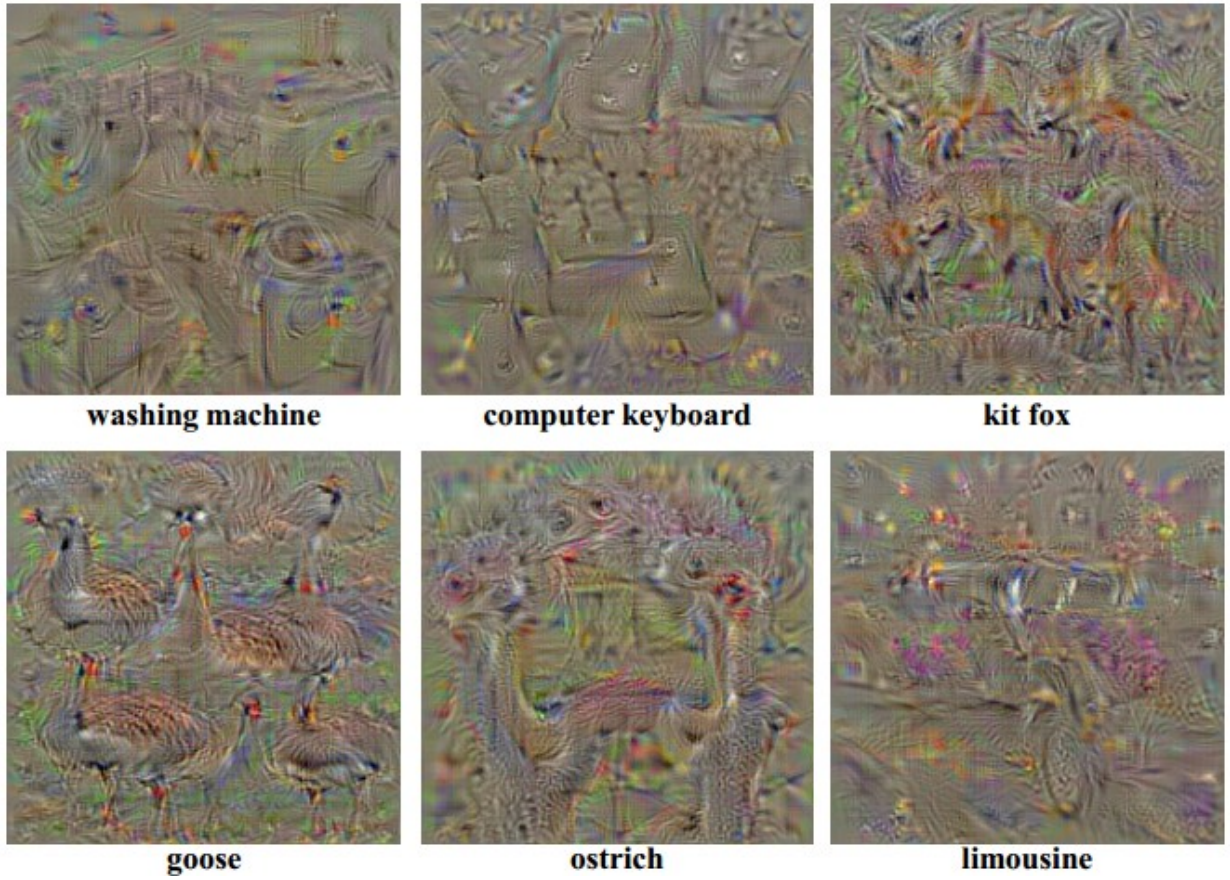
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014

1. Find images that maximize some class score:



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

1. Find images that maximize some class score:



[Understanding Neural Networks Through Deep Visualization, Yosinski et al. , 2015]

Proposed a different form of regularizing the image

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

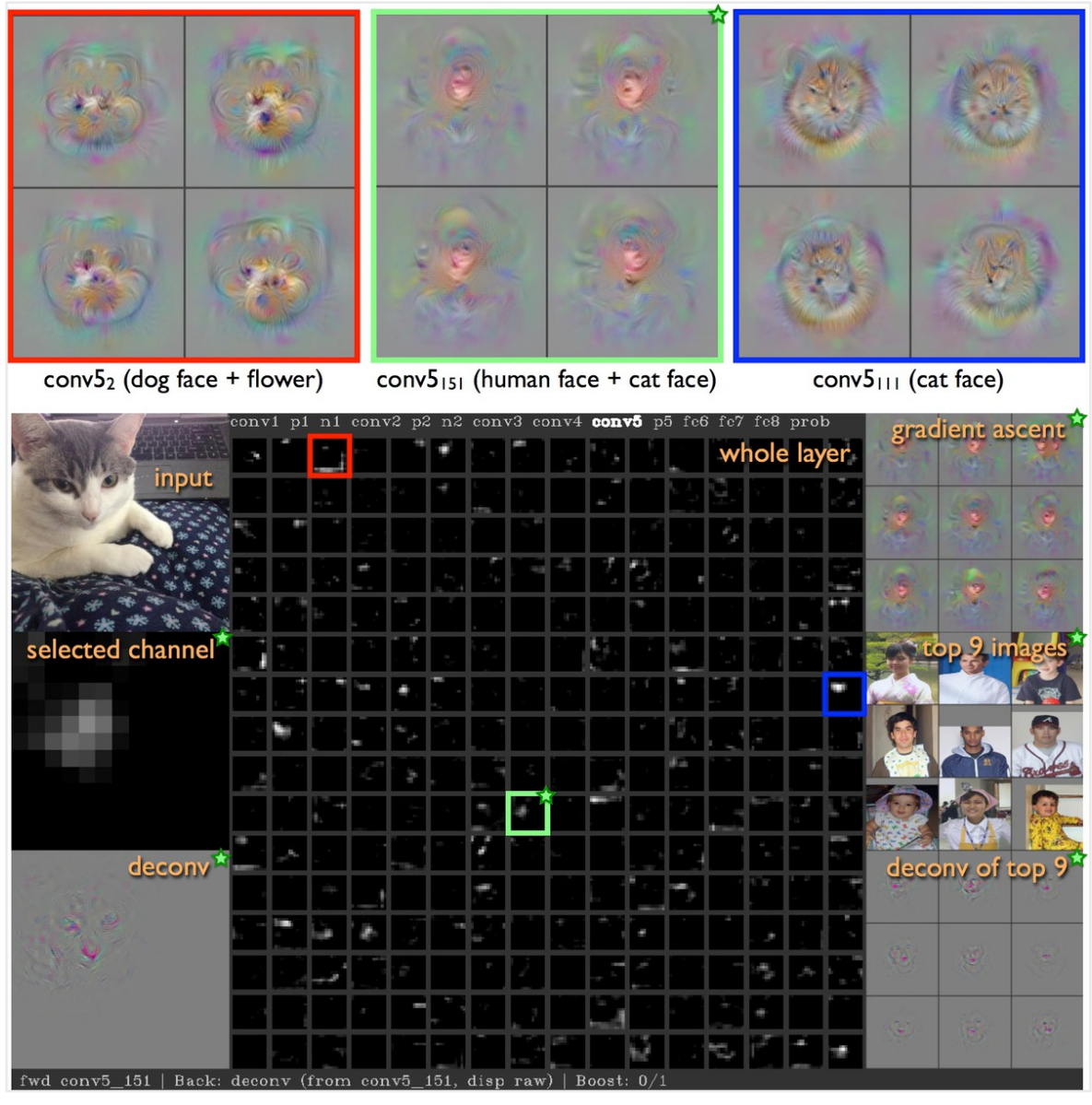
More explicit scheme:

Repeat:

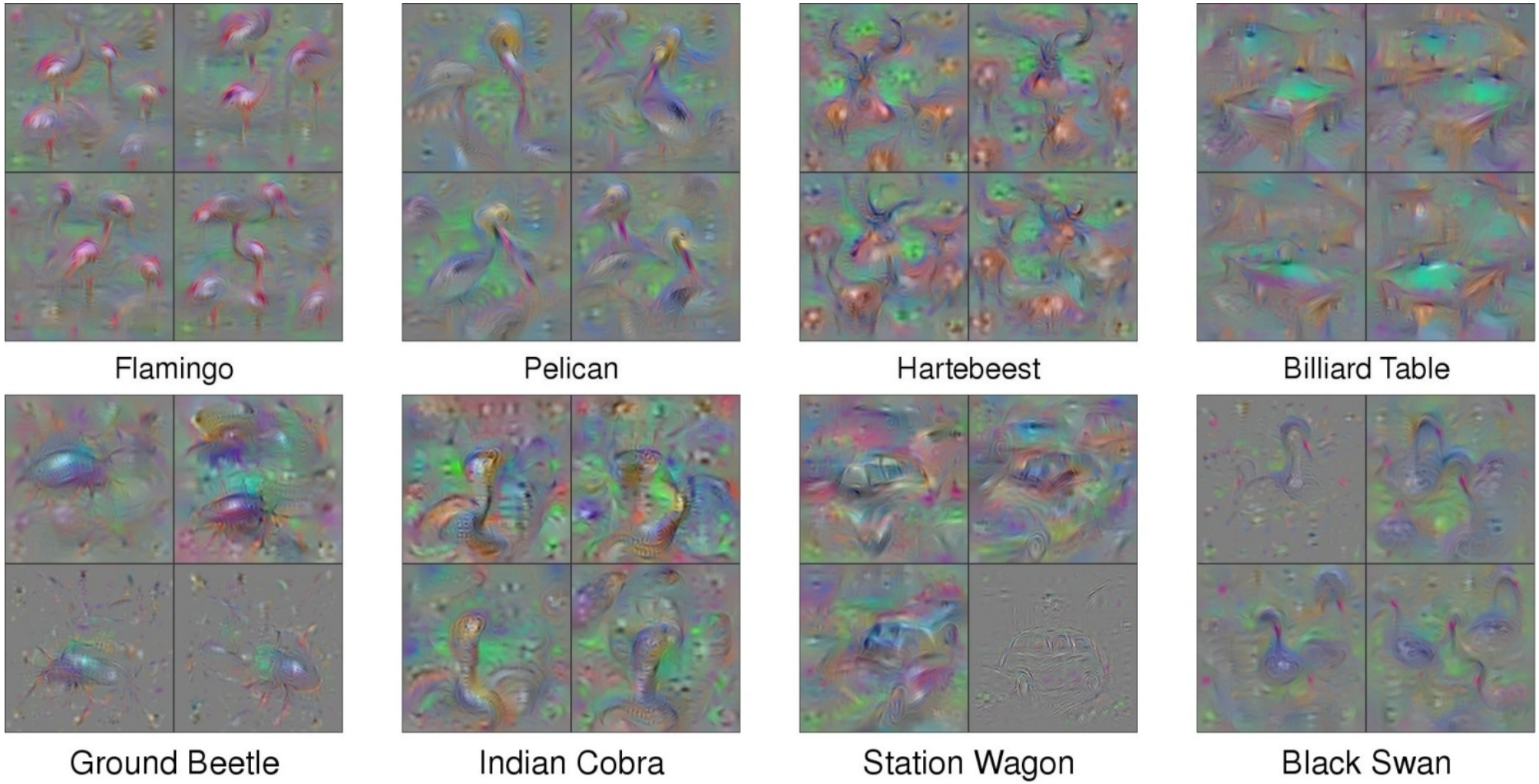
- Update the image **x** with gradient from some unit of interest
- Blur x a bit
- Take any pixel with small norm to zero (to encourage sparsity)

http://yosinski.com/deepvis

YouTube video
https://www.youtube.com/watch?v=AgkfIQ4IGaM
(4min)



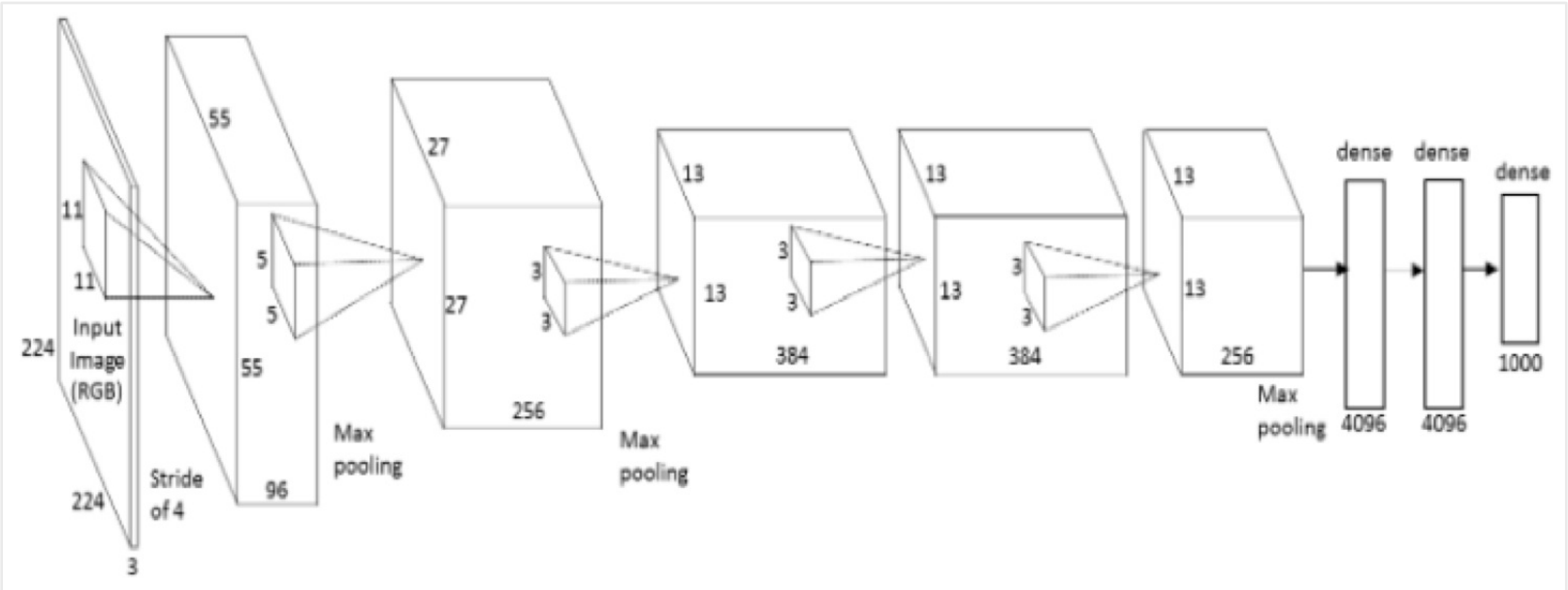
[Understanding Neural Networks Through Deep Visualization, Yosinski et al. , 2015]



http://yosinski.com/deepvis

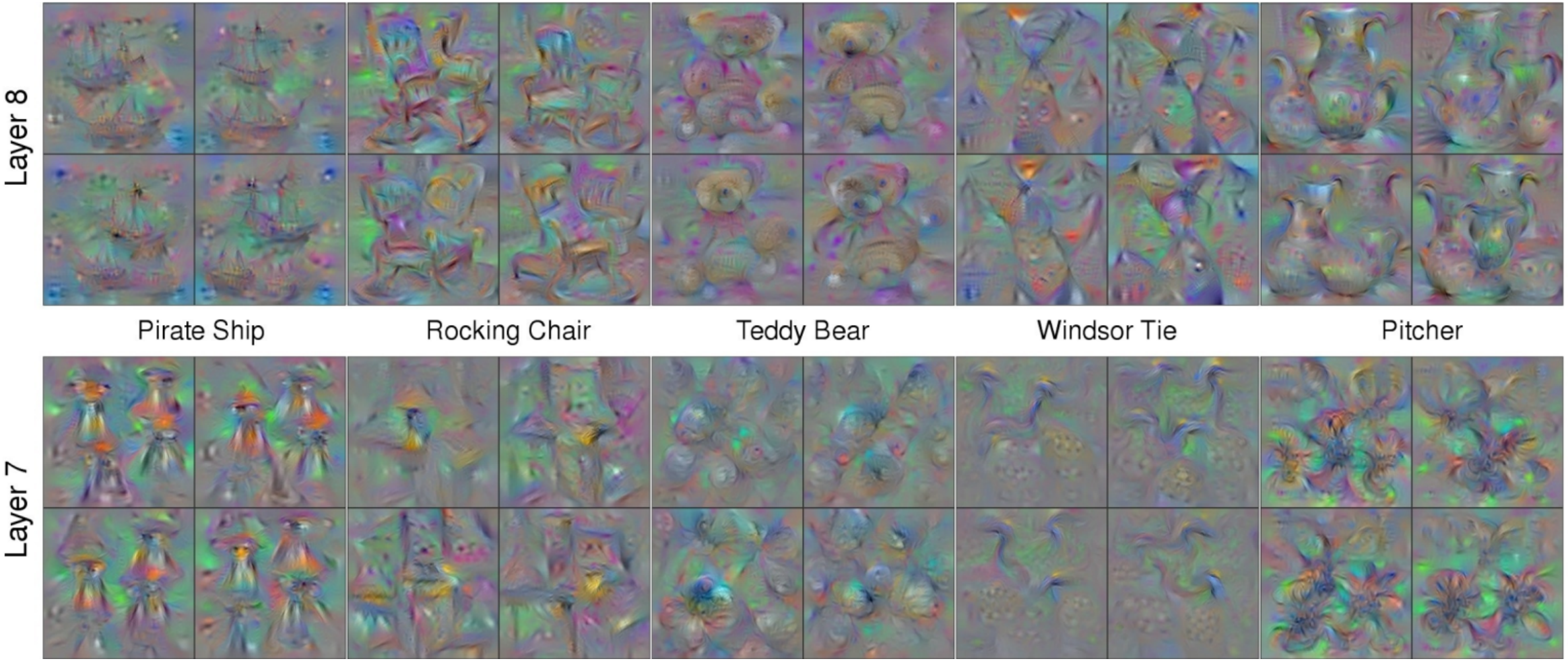
4 images = 4 different initializations

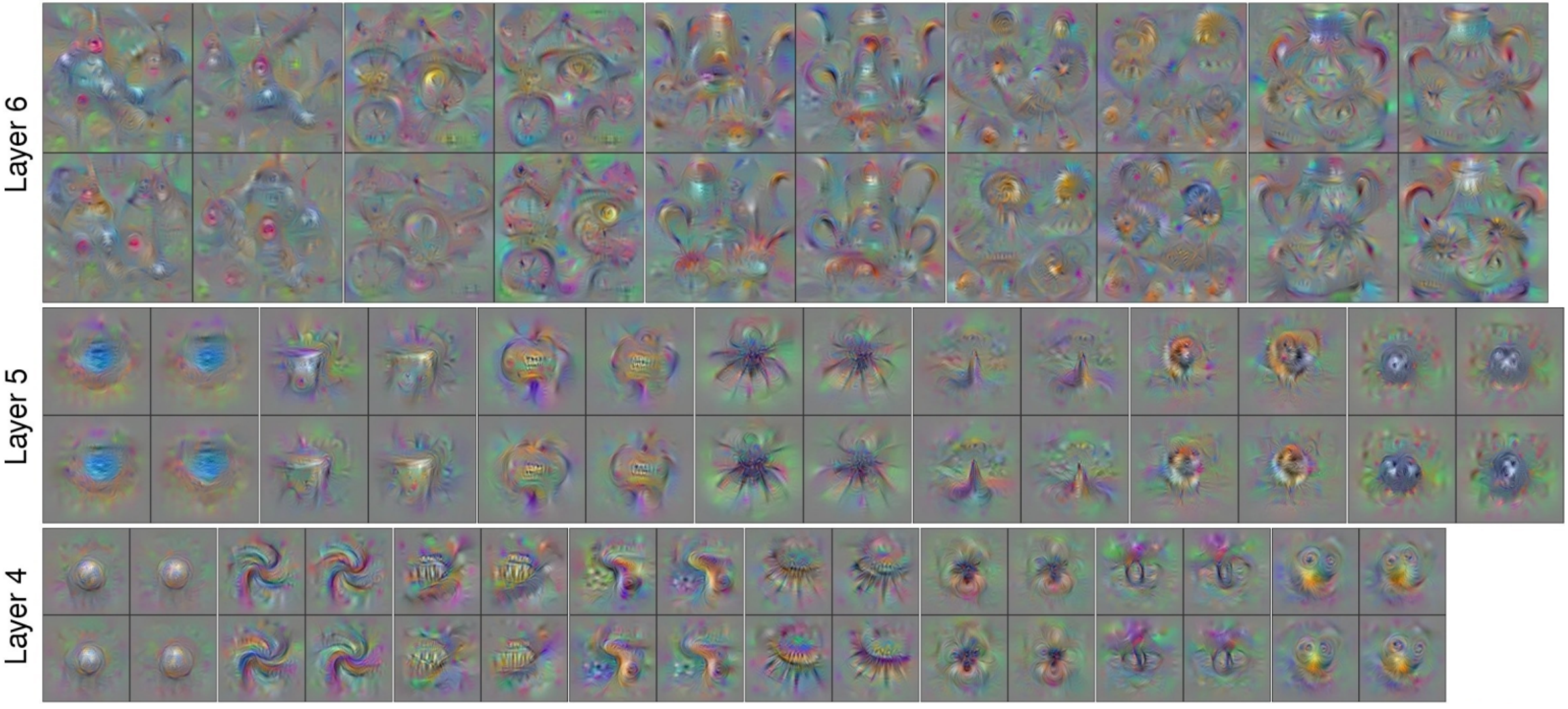
We can in fact do this for arbitrary neurons along the ConvNet

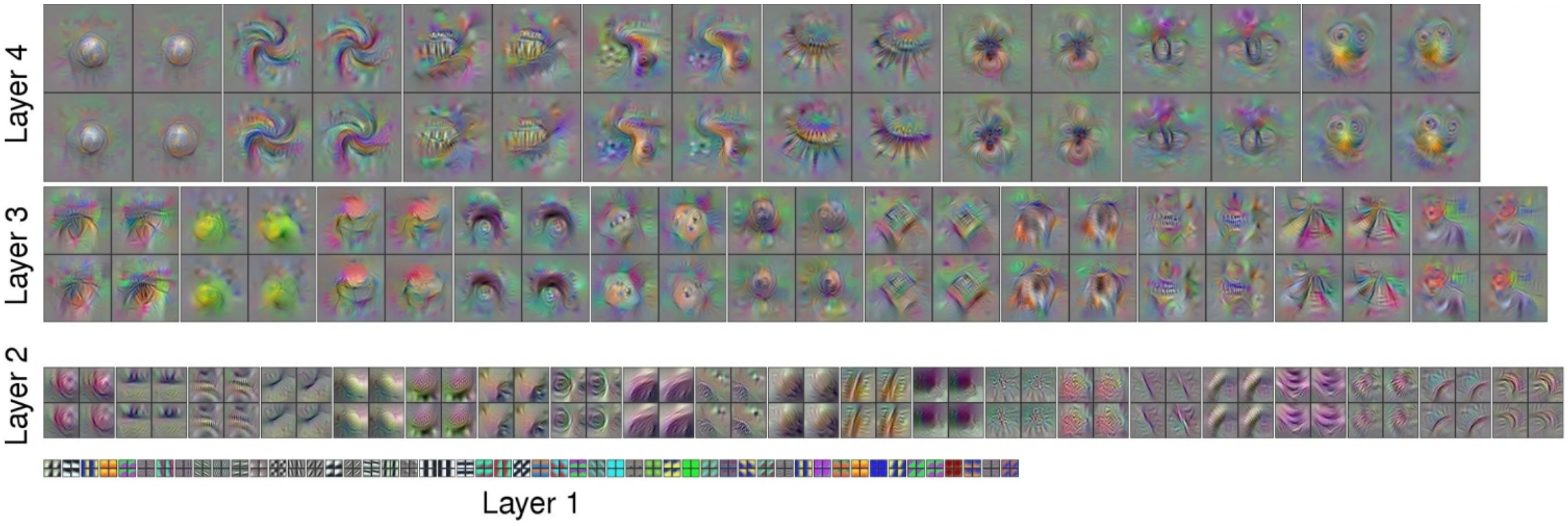


Repeat:

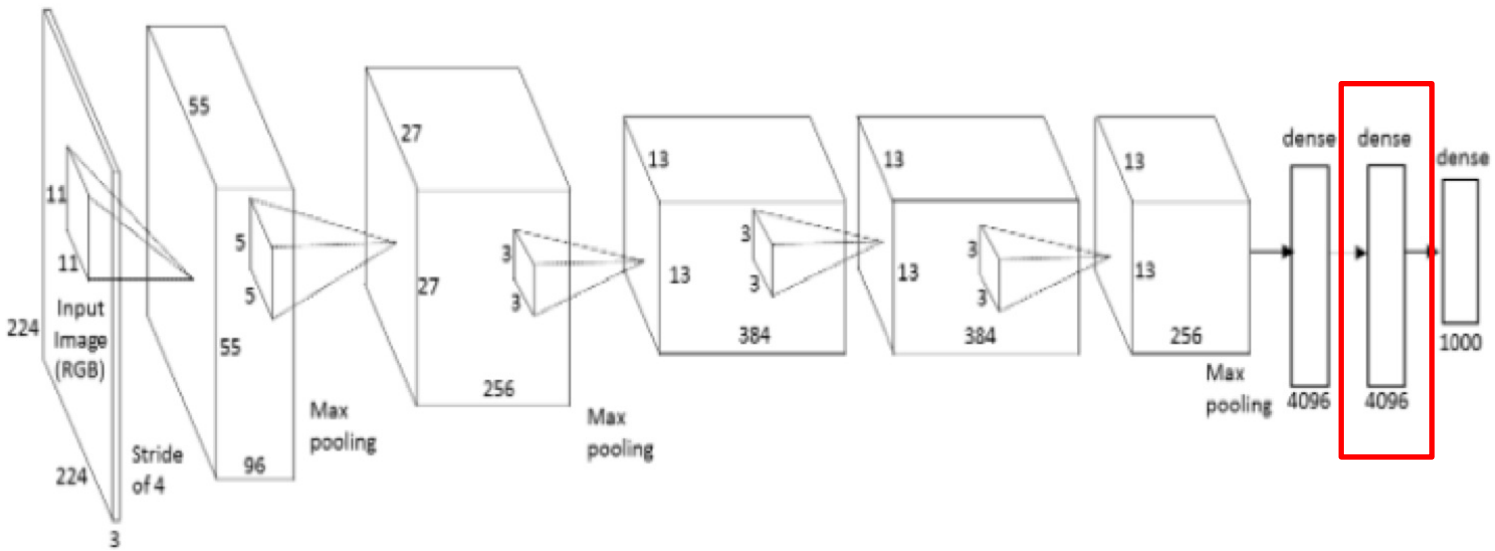
1. Forward an image
2. Set activations in layer of interest to all zero, except for a 1.0 for a neuron of interest
3. Backprop to image
4. Do an “image update”


Layer 8
Pirate Ship
Rocking Chair
Teddy Bear
Windsor Tie
Pitcher
Layer 7


Layer 6
Layer 5
Layer 4


Layer 4
Layer 3
Layer 2
Layer 1

Question: Given a CNN code, is it possible to reconstruct the original image?



Find an image such that:

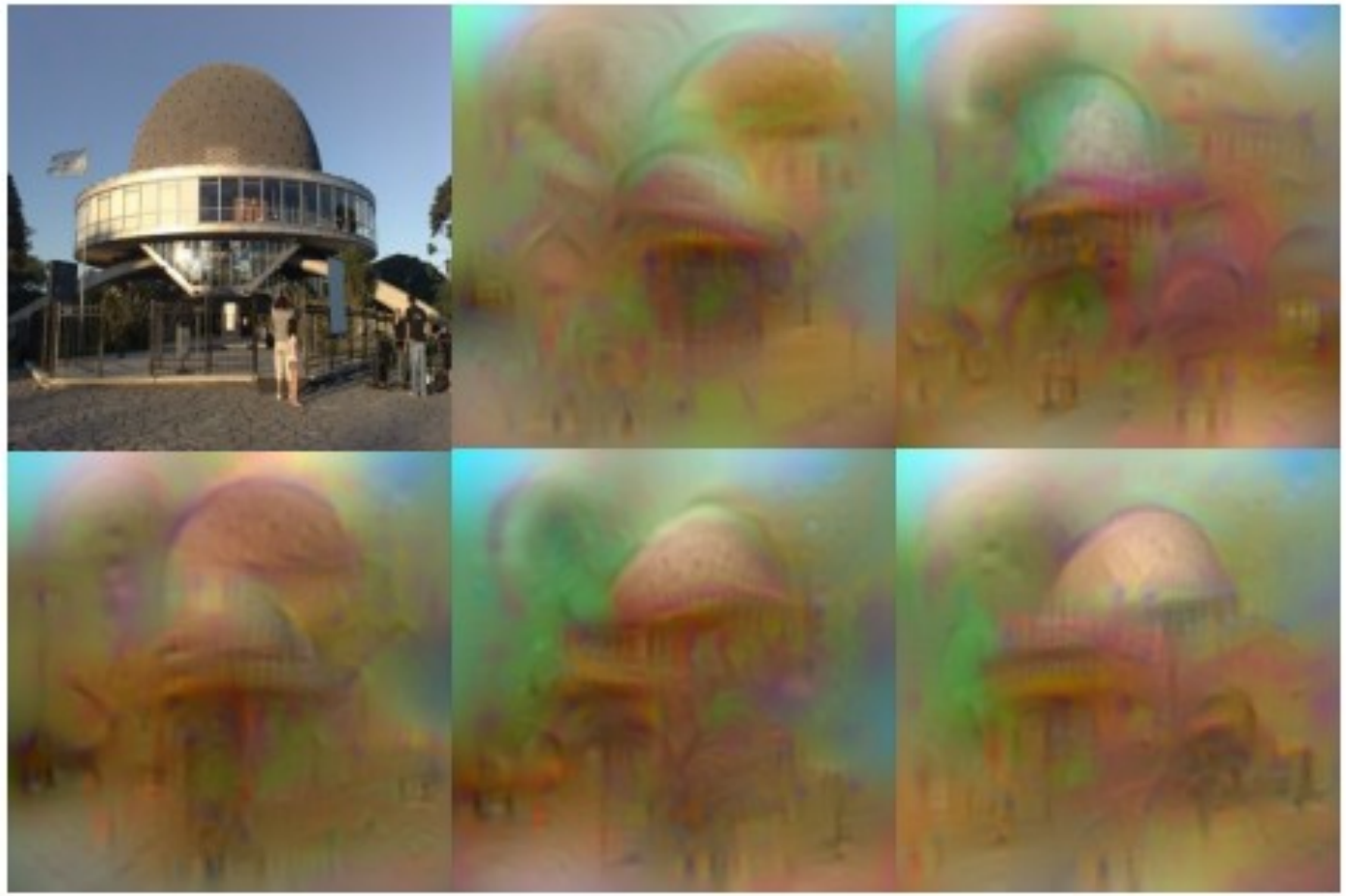
- Its code is similar to a given code
- It “looks natural” (image prior regularization)
- Multiple solutions but perhaps we can solve by sampling

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

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

Understanding Deep Image Representations by Inverting Them
[Mahendran and Vedaldi, 2014]

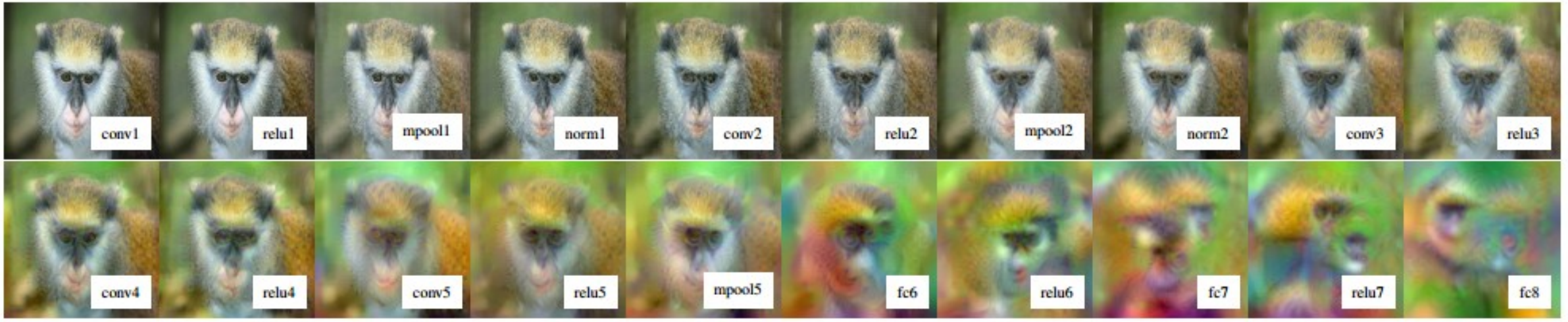
original image



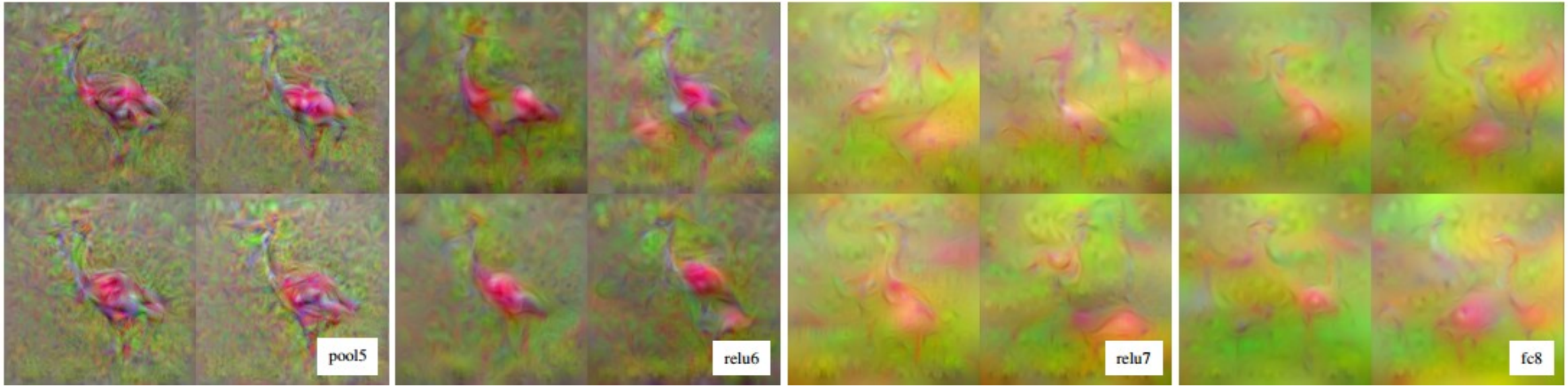
reconstructions from the 1000 log probabilities for ImageNet (ILSVRC) classes

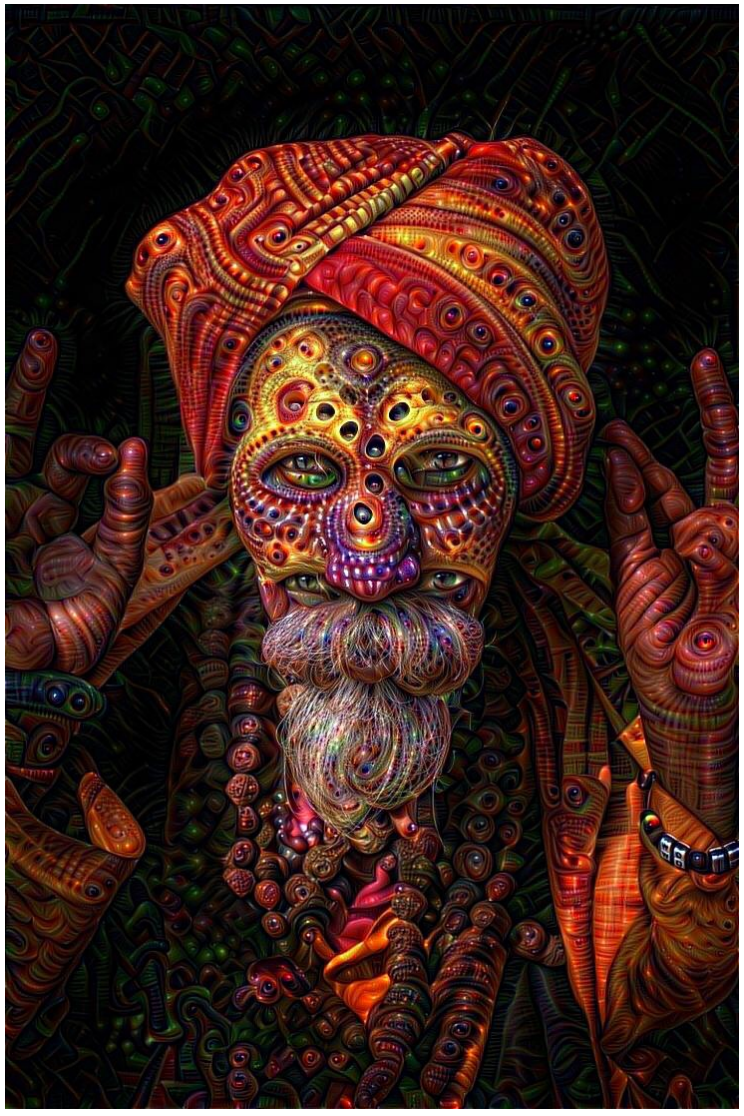
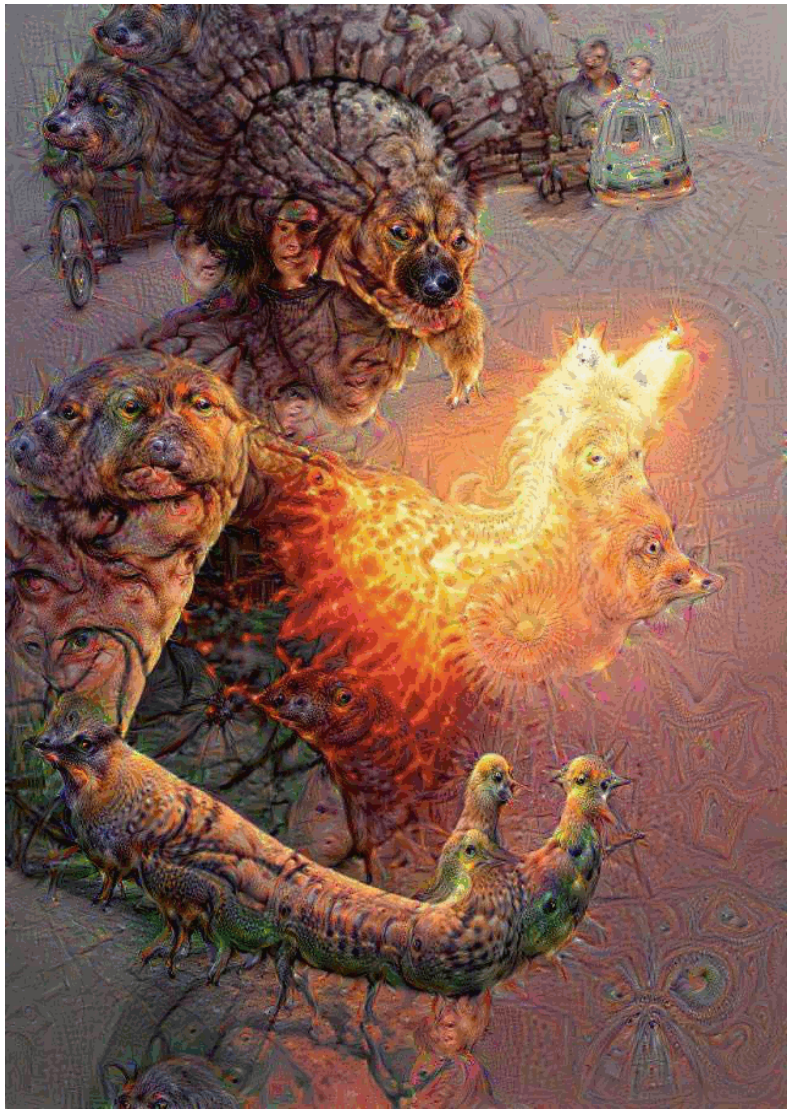
Reconstructions from the representation after last last pooling layer (immediately before the first Fully Connected layer)

Reconstructions from intermediate layers



Multiple reconstructions. Images in quadrants all "look" the same to the CNN (same code)



DeepDream https://github.com/google/deepdream

```lua
require 'nn'
require 'cunn'
require 'cudnn'
require 'image'


local cuda = true
torch.setdefaulttensortype('torch.FloatTensor')
net = torch.load('./GoogLeNet_v2.t7'):cuda()

local Normalization = {mean = 118.380948/255, std = 61.896913/255}

function reduceNet(full_net,end_layer)
    local net = nn.Sequential()
    for l=1,end_layer do
        net:add(full_net:get(l))
    end
    print(net)
    return net
end

function make_step(net, img, clip,step_size, jitter)..

function deepdream(net, base_img, iter_n, octave_n, octave_scale, end_layer, clip, visualize)..

img = image.load('./sky1024px.jpg')
x = deepdream(net,img,20)
image.display(x)
```

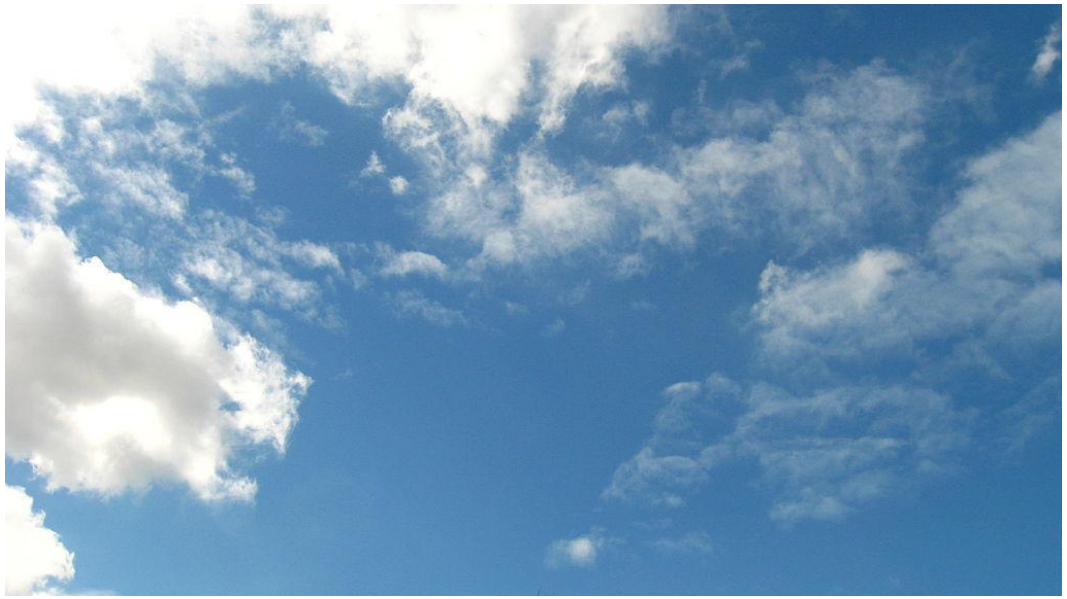
https://github.com/eladhoffer/DeepDream.torch

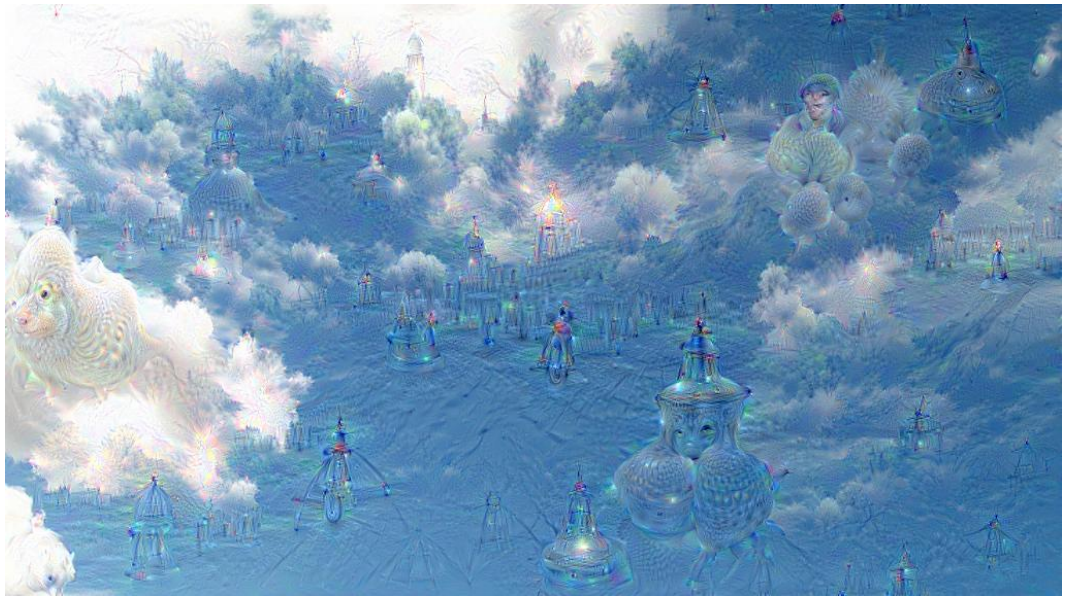
```
112 function make_step(net, img, clip,step_size, jitter)
113     local step_size = step_size or 0.01
114     local jitter = jitter or 32
115     local clip = clip
116     if clip == nil then clip = true end
117
118     local ox = 0--2*jitter - math.random(jitter)
119     local oy = 0--2*jitter - math.random(jitter)
120     img = image.translate(img,ox,oy) -- apply jitter shift
121     local dst, g
122     dst = net:forward(img)
123     g = net:updateGradInput(img,dst)
124     -- apply normalized ascent step to the input image
125     img:add(g:mul(step_size/torch.abs(g):mean()))
126
127     img = image.translate(img,-ox,-oy) -- apply jitter shift
128     if clip then
129         bias = Normalization.mean/Normalization.std
130         img:clamp(-bias,1/Normalization.std-bias)
131     end
132     return img
133 end
```

DeepDream: set dx = x :)

https://github.com/eladhoffer/DeepDream.torch

inception_4c/output

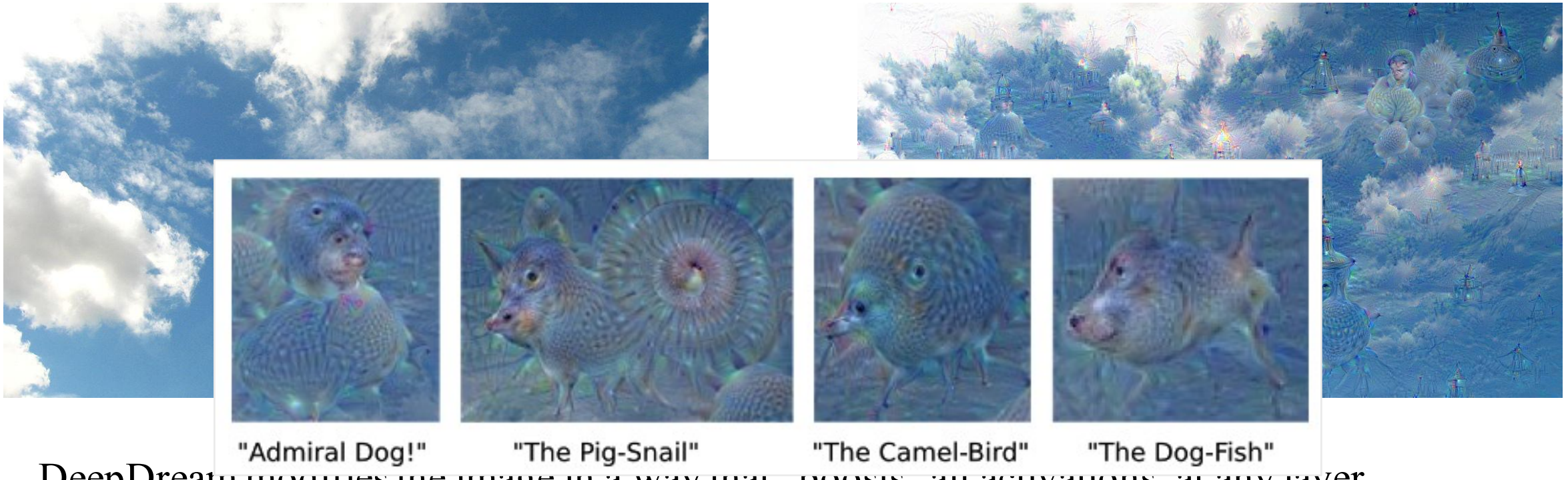




DeepDream modifies the image in a way that “boosts” all activations, at any layer

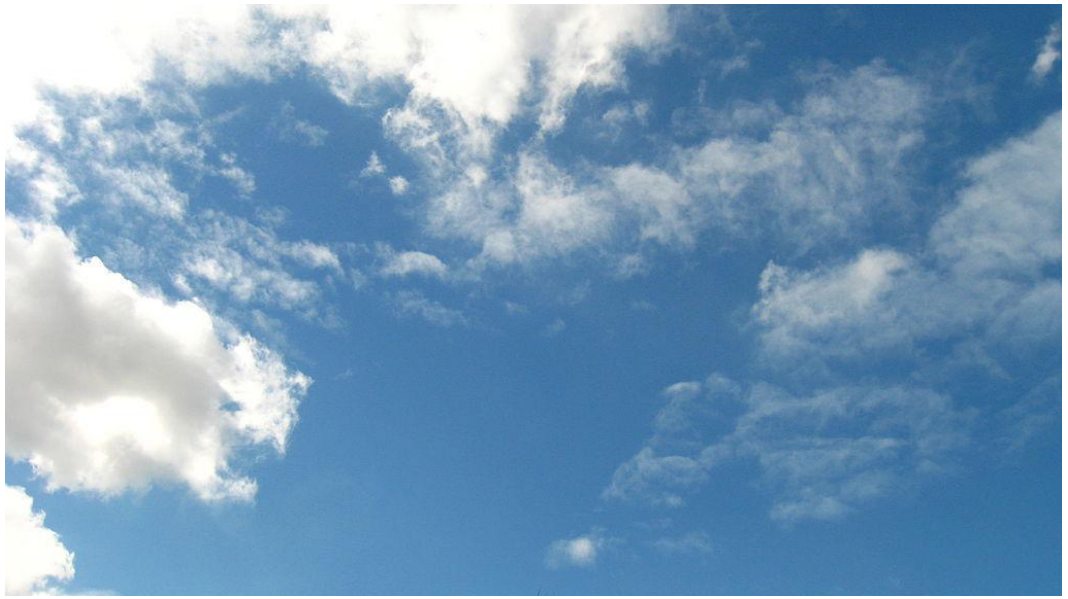
this creates a feedback loop: e.g. any slightly detected dog face will be made more and more dog like over time

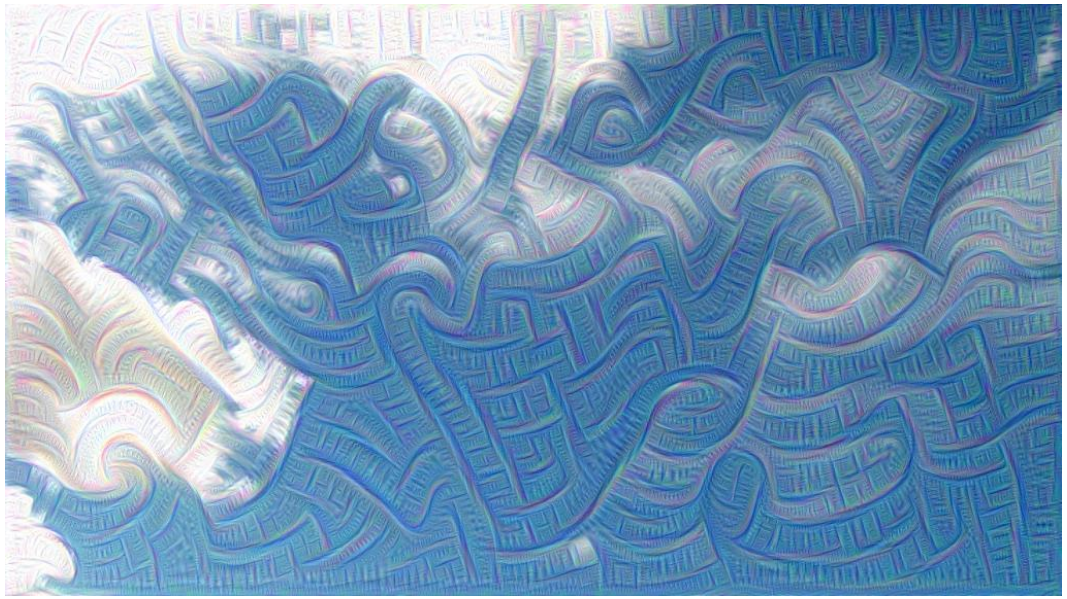
inception_4c/output



DeepDream modifies the image in a way that "boosts" all activations, at any layer

inception_3b/5x5_reduce





DeepDream modifies the image in a way that “boosts” all activations, at any layer

Bonus videos

Deep Dream (Grocery **Trip**)

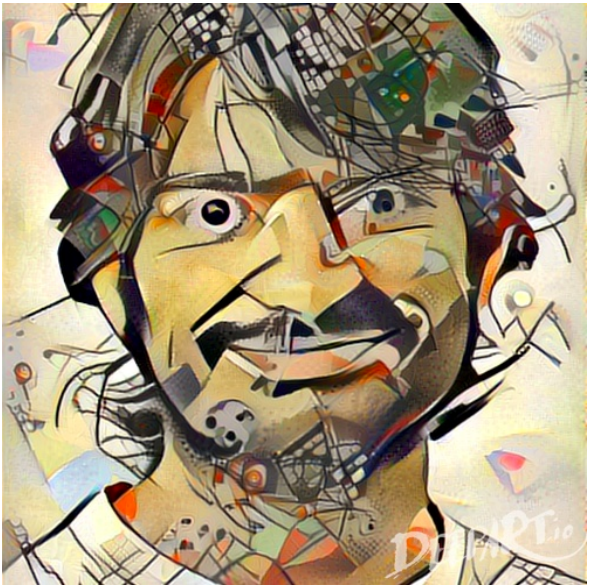
https://www.youtube.com/watch?v=DgPaCWJL7XI

NeuralStyle

[A Neural Algorithm of Artistic Style by Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, 2015]

good implementation in Torch:

https://github.com/jcjohnson/neural-style

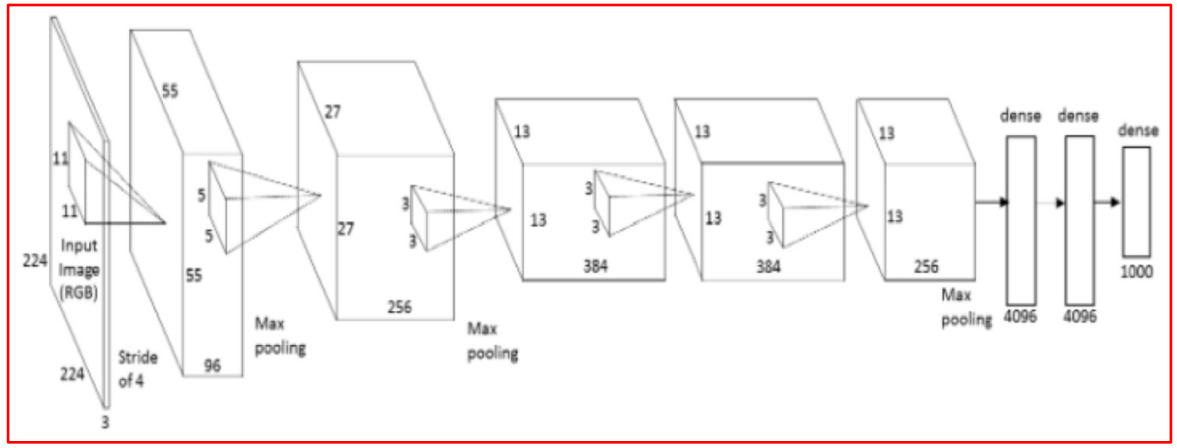







make your own easily on deepart.io

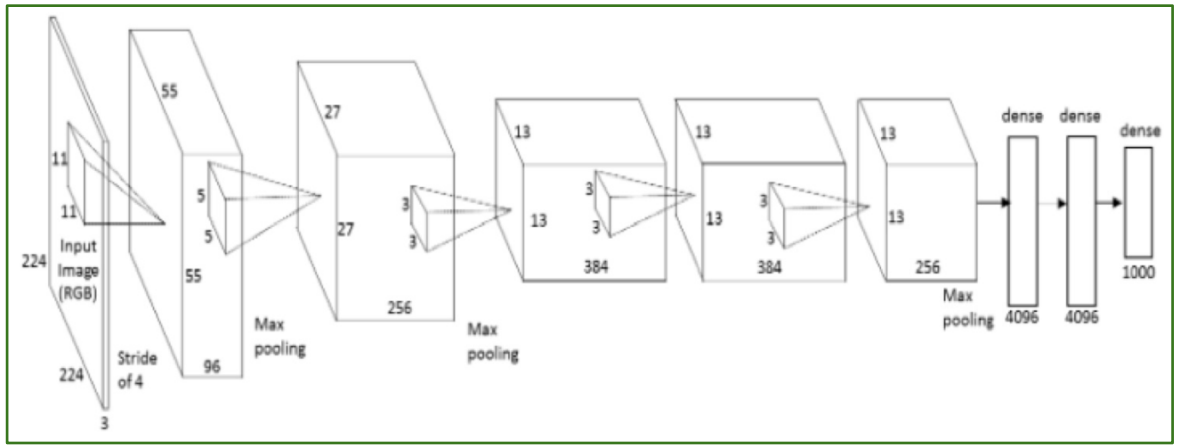
Step 1: Extract **content targets** (ConvNet activations of all layers for the given content image)



content activations

e.g.
at CONV5 layer we would have a [14x14x512] array of target activations

Step 2: Extract **style targets** (Gram matrices of ConvNet activations of all layers for the given style image)



style gram matrices

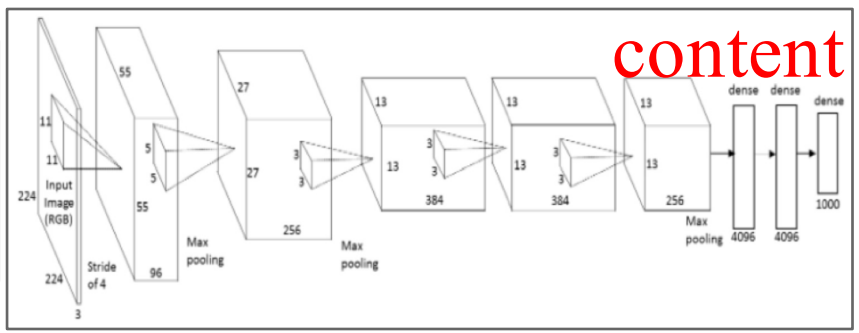
$$G = V^{\mathrm{T}} V$$

e.g.
at CONV1 layer (with [224x224x64] activations) would give a [64x64] Gram matrix of all pairwise activation covariances (summed across spatial locations)

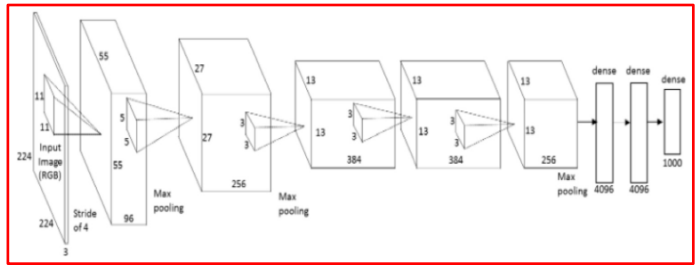
Step 3: Optimize over image to have:

- The **content** of the content image (activations match content)
- The **style** of the style image (Gram matrices of activations match style)

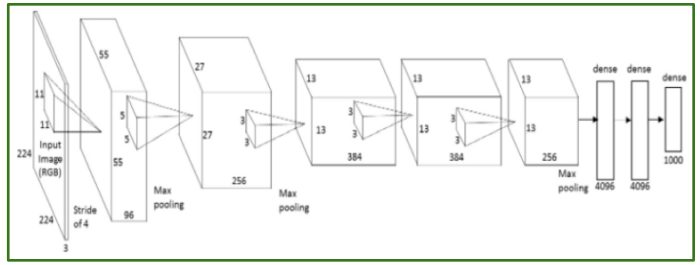
$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$



match content



match style

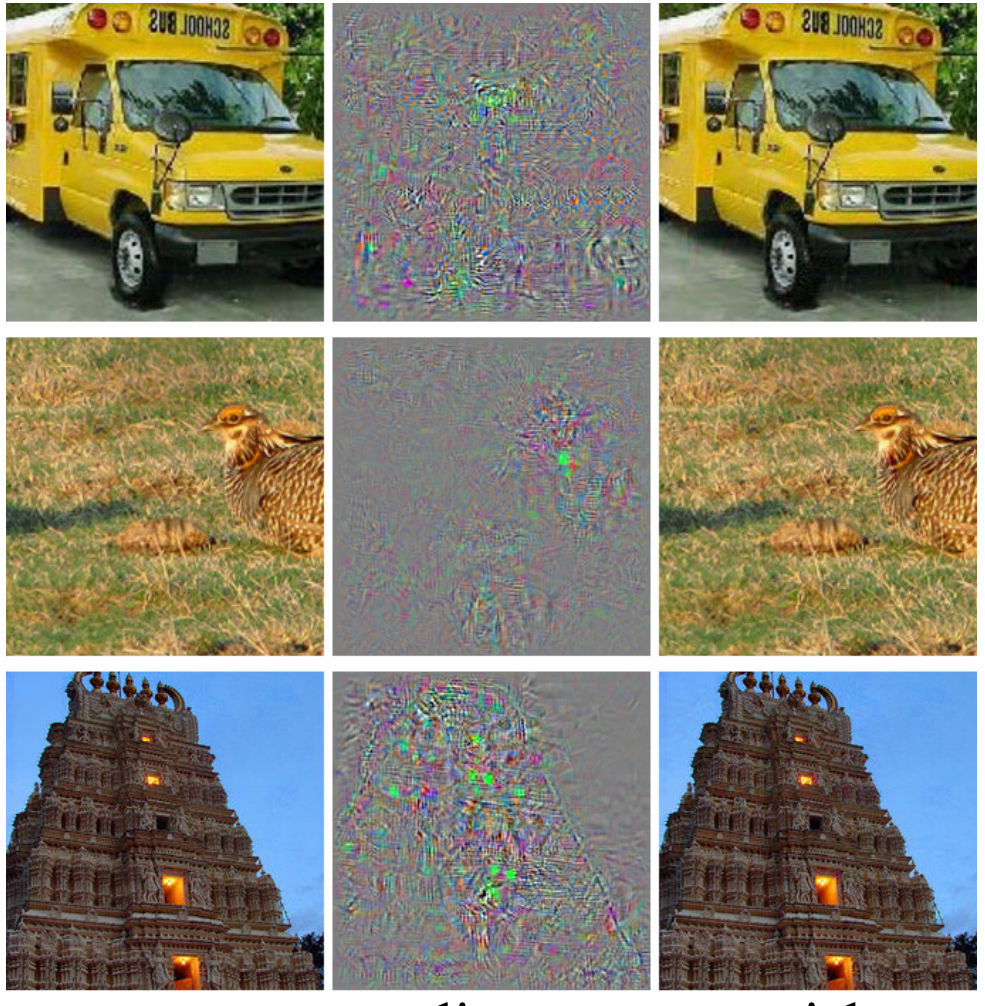


We can pose an optimization over the input image to maximize any class score.
That seems useful.

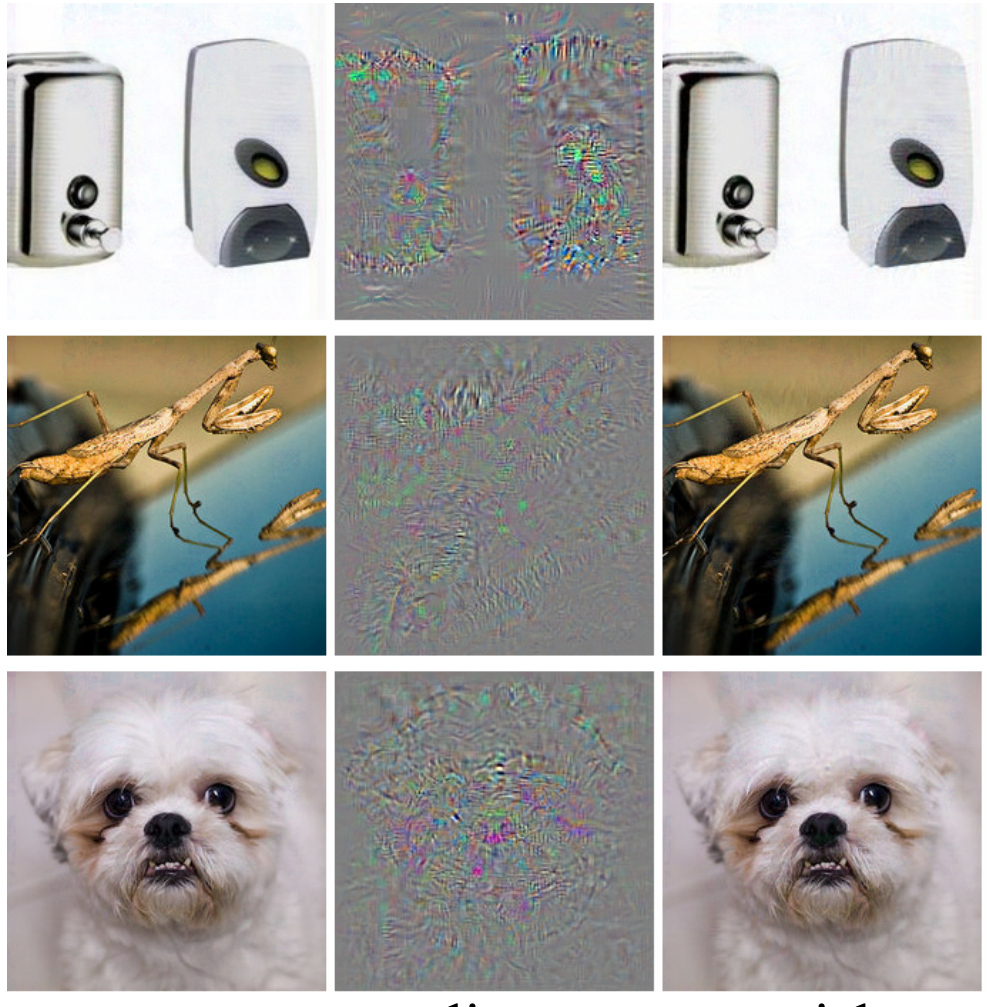
Question: Can we use this to “fool” ConvNets?

spoiler alert: yeah

[Intriguing properties of neural networks, Szegedy et al., 2013]



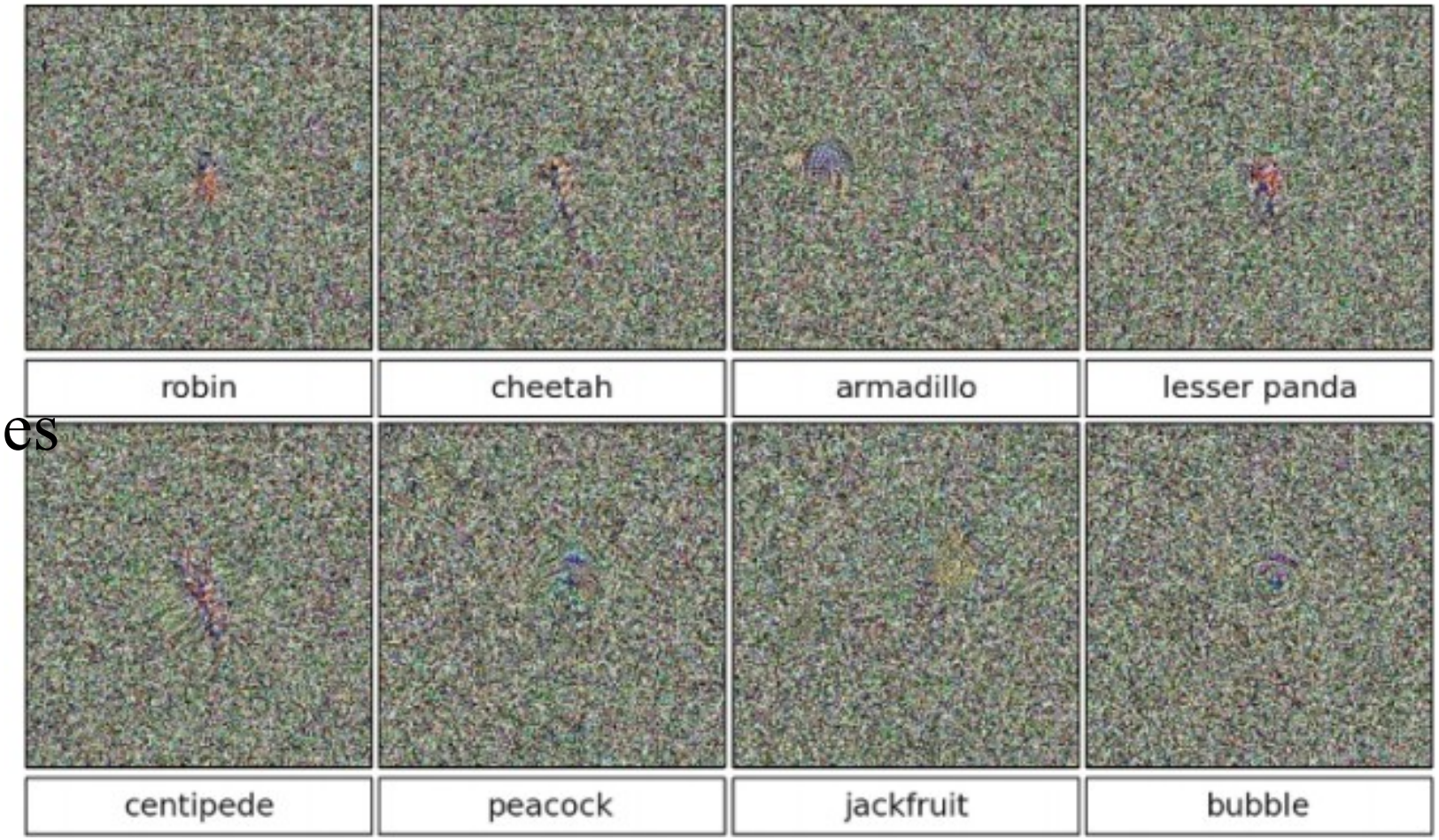
correct +distort ostrich



correct +distort ostrich

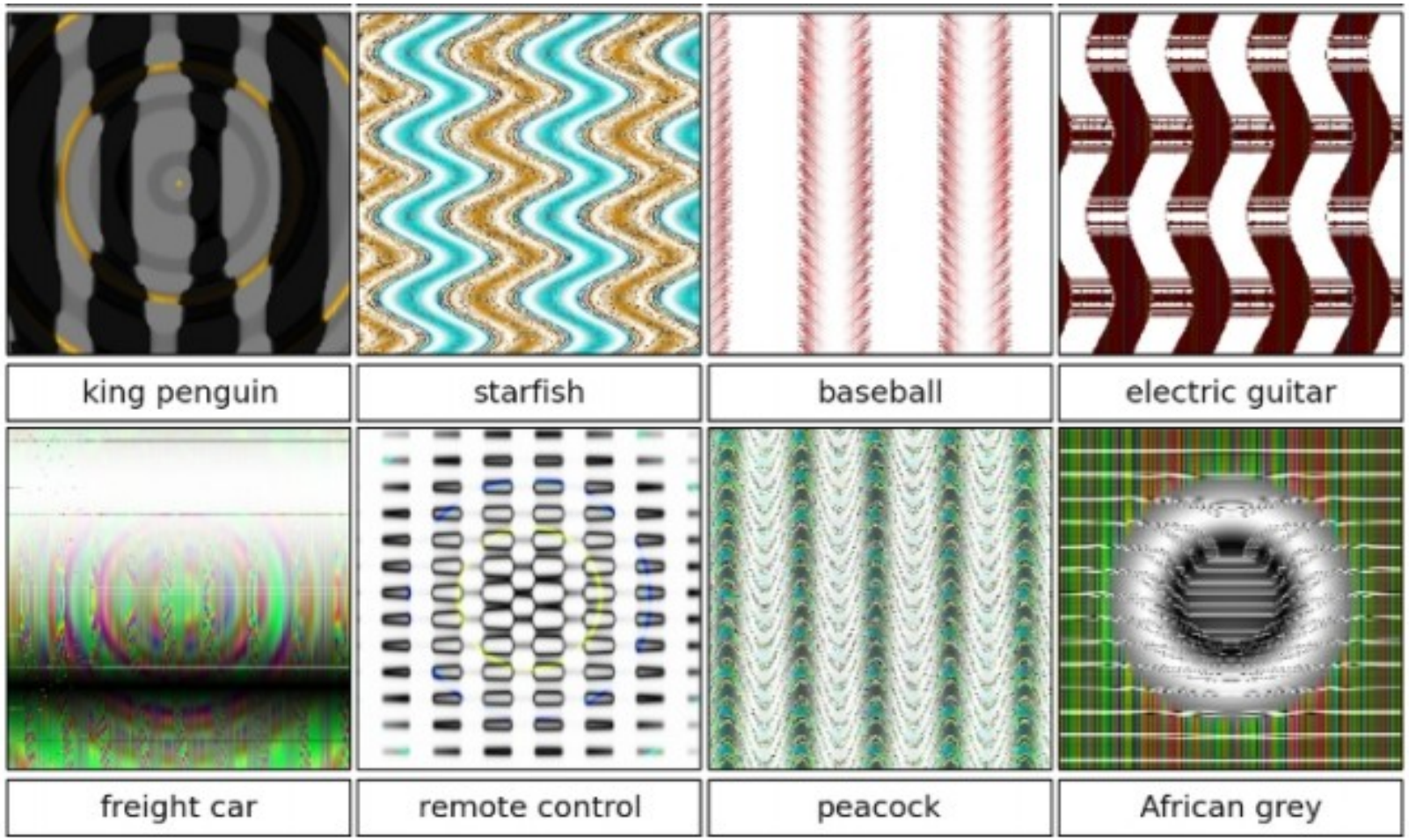
[Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images Nguyen, Yosinski, Clune, 2014]

>99.6% confidences



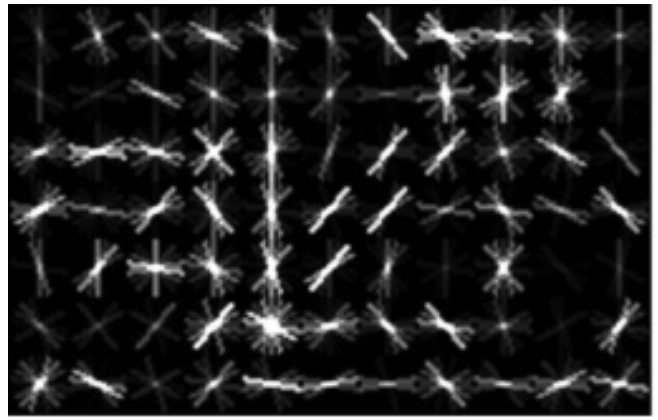
[Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images Nguyen, Yosinski, Clune, 2014]

>99.6% confidences



These kinds of results were around even before ConvNets…

[Exploring the Representation Capabilities of the HOG Descriptor, Tatu et al., 2011]



Identical HOG represention

Backpropping to the image is powerful

It can be used for:

- **Understanding** (e.g. visualize optimal stimuli for arbitrary neurons)
- **Segmenting** objects in the image (kind of)
- **Inverting** codes and introducing privacy concerns
- **Fun** (NeuralStyle/DeepDream)
- **Confusion and chaos** (Adversarial examples)

Dimension Reduction Using CNN (Siamese and Triplet Networks)

Dimension Reduction Using CNN

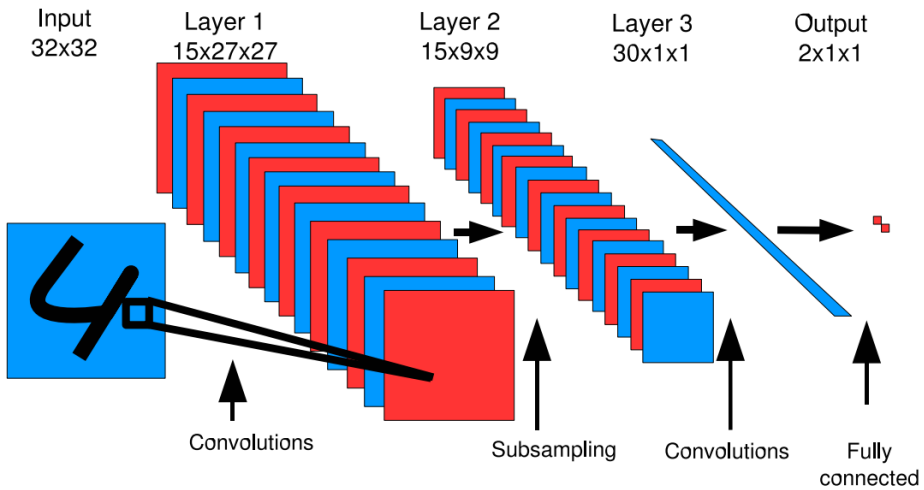
- Traditional dimension reduction algorithms like PCA do not capture semantic similarity

Properties of a learnt embedding:

- Semantic similarity between two points can be measured by simple Euclidean distance between them.
- Should be invariant to deformations and transformation.
- Should be applicable to new samples which are not part of the training.

Dimension Reduction Using CNN

- Use a Siamese Network with a contrastive loss.

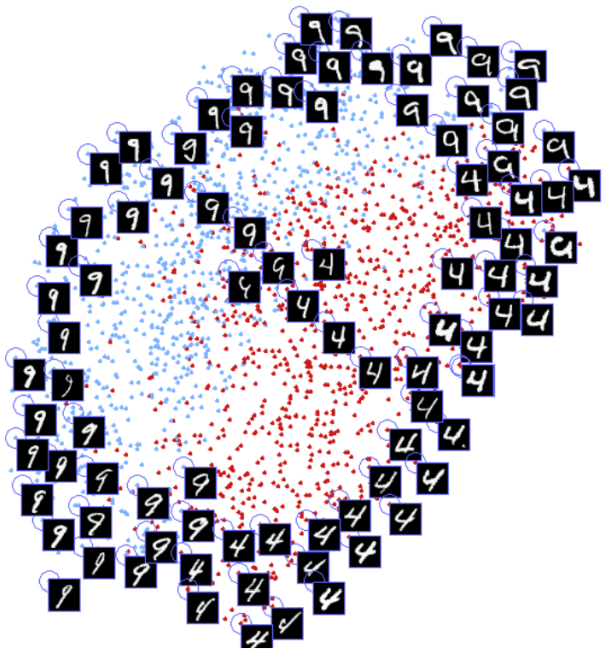


$$L(W, Y, \vec{X}_1, \vec{X}_2) =$$
$$(1 - Y)\frac{1}{2}(D_W)^2 + (Y)\frac{1}{2}\{max(0, m - D_W)\}^2$$

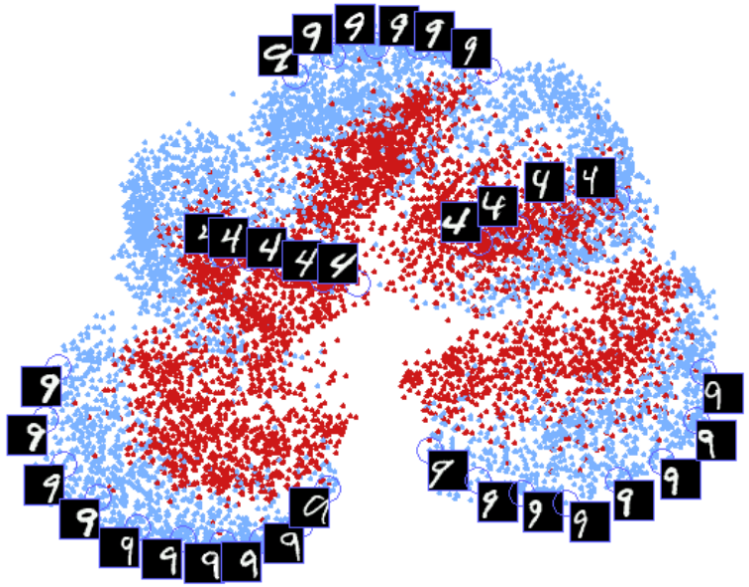
- Here $\overrightarrow{X_1}$, and $\overrightarrow{X_2}$ is a pair of input data points. $Y = 0$ for matching pair and 1 otherwise.
- D_w is the L2 distance between $\overrightarrow{X_1}$ and $\overrightarrow{X_2}$ in the reduced dimensioned space. In the image the network map an image of a digit having dimension of 32 x 32 into a 2 dimension space.
- m is the desired margin by which two negative examples should be far apart.

Image Courtesy: Hadsell et al. CVPR 2006

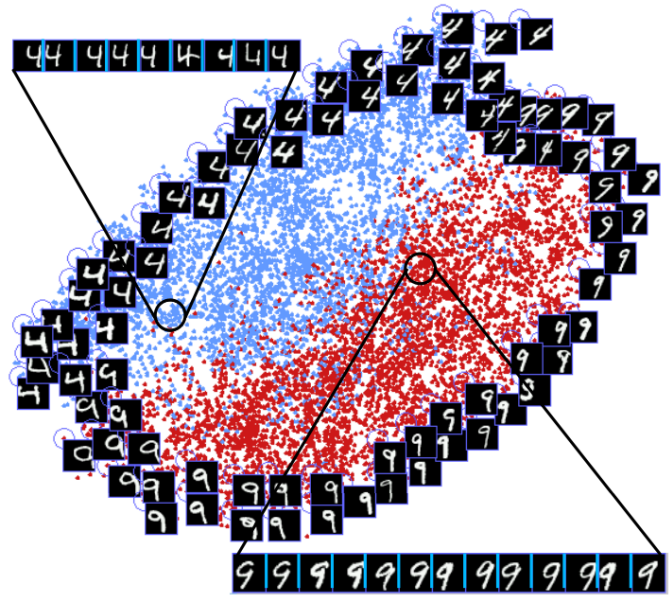
Embedding in 2D



Embedding of original test data



Embedding of perturbed test data

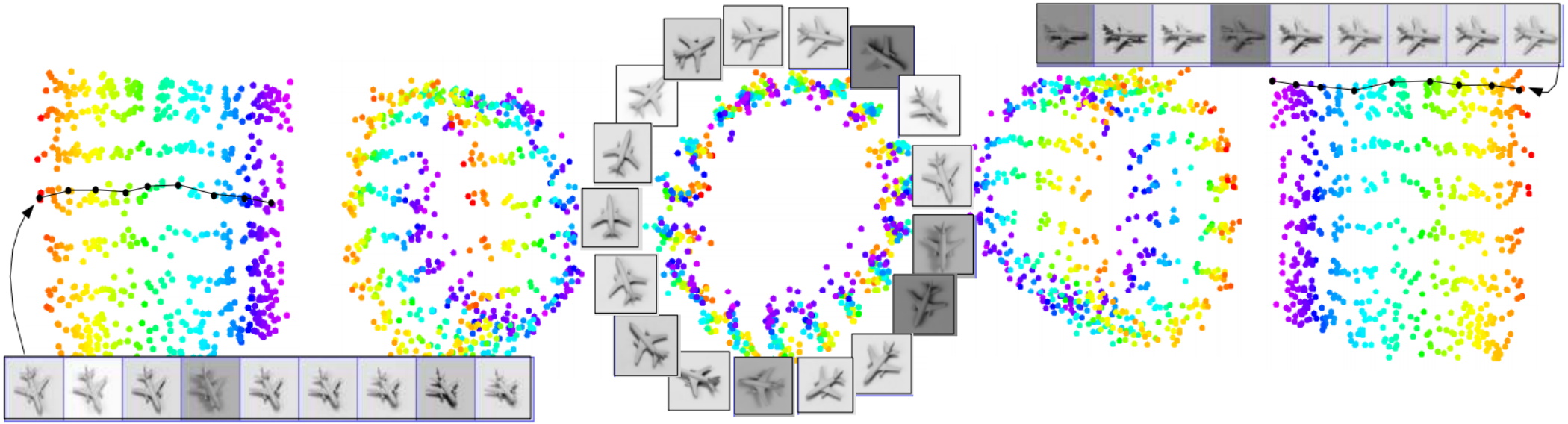


Embedding of perturb test data with perturb training

- Perturbations were horizontal translation of –6, -3, +3, +6 pixels. MNIST dataset is used

Image Courtesy: Hadsell et al. CVPR 2006

Embedding in 3D



- Mapping is roughly in form of a cylinder.
- Changes in azimuthal angles varies along the circumference, while variation in elevation is mapped along the height of the cylinder.
- Invariant to illumination.

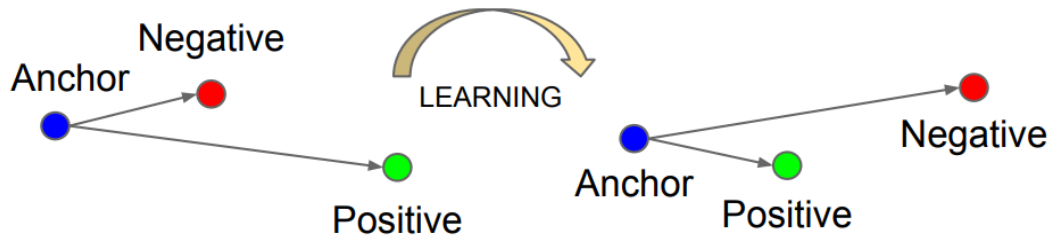
Image Courtesy: Hadsell et al. CVPR 2006

Better Loss Functions - Triplet

- Sometimes contrastive loss too restrictive
- $\{\mathbf{X_a}, \mathbf{X_p}, \mathbf{X_n}\}$ forms a triplet where $\mathbf{X_a}$ (anchor) is semantically similar to $\mathbf{X_p}$ (positive) and dissimilar to $\mathbf{X_n}$ (negative).

$$L(\mathbf{X_a}, \mathbf{X_p}, \mathbf{X_n}, f) = \max\left(0,\ \|f(\mathbf{X_a}) - f(\mathbf{X_p})\|_2 - \|f(\mathbf{X_a}) - f(\mathbf{X_n})\|_2 + \alpha\right)$$

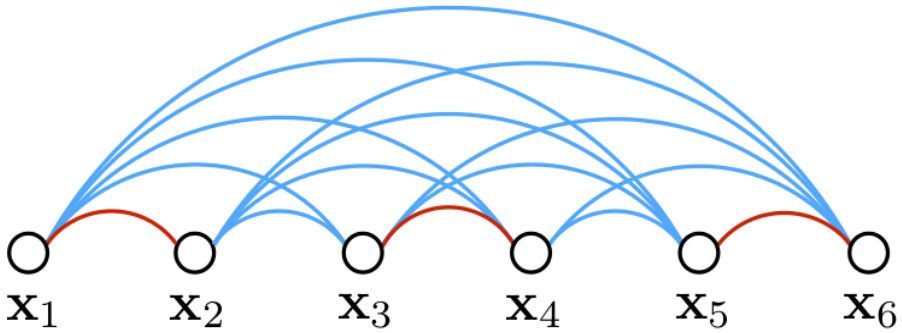
- L() is loss , f() is the mapping function, α is the margin.



- Given an anchor $\mathbf{X_a}$, distance of all ($\mathbf{X_a}$, $\mathbf{X_p}$) are below all ($\mathbf{X_a}$, $\mathbf{X_n}$) by a margin α
- Learns an ranking among similar instances.

Better Loss Functions – N pairs

- N-pair loss (lifted feature embedding)



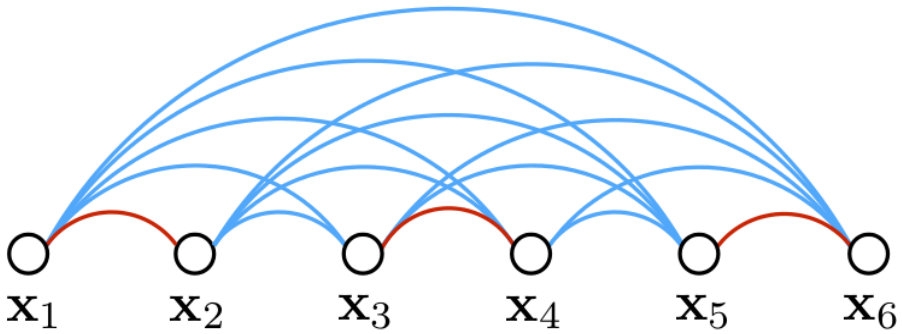
Brown edges connect matching pairs while blue edges connect non-matching pairs.

$$J_{i,j} = \max\left(0,\ \log\left(\sum_{(i,k)\in N} exp(\alpha - D_{i,k}) + \sum_{(j,l)\in N} exp(\alpha - D_{j,l})\right) + D_{i,j}\right)^2$$

- $J_{i,j}$ is the loss for a given matching pair (i, j). N is the set of pairs in the batch. $D_{m,k}$ is the distance between pair (m, k).

Better Loss Functions – N pairs

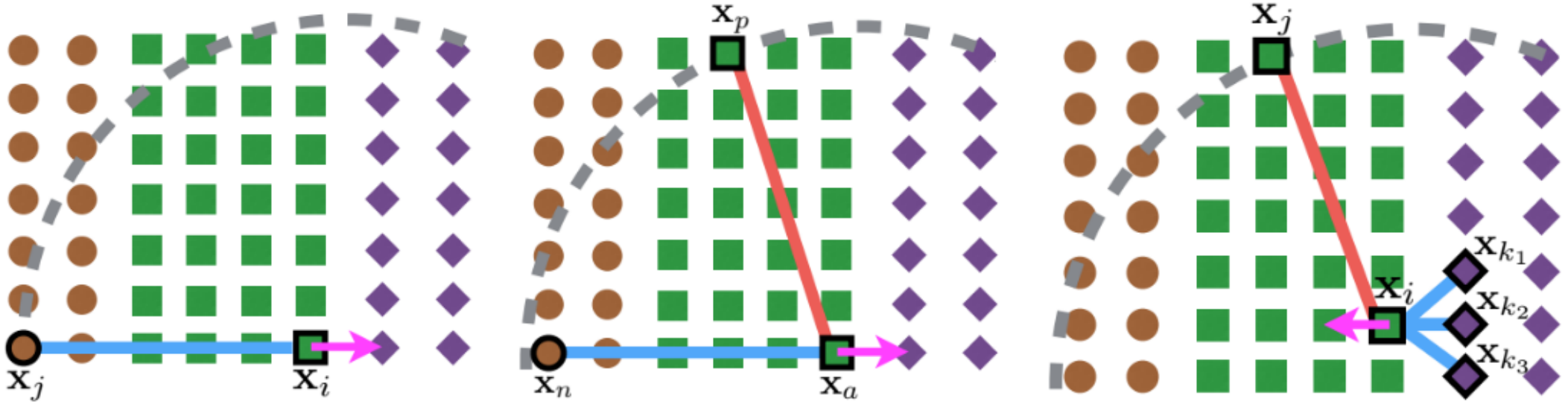
- N-pair loss (lifted feature embedding)



Brown edges connect matching pairs while blue edges connect non-matching pairs.

- In triplet version, for matching pair (*i, j*), say i = 3 and j = 4, a *k* in (1, 2, 5, 6) is chosen such that $D_{i,k}$ or $D_{j,k}$ is the minimum among all negatives of *i* or *j*.
- (*i, j, k*) then forms a triplet and the usual triplet loss is applied with margin α.

Comparing Loss Functions



(a) Contrastive embedding (b) Triplet embedding (c) Lifted structured similarity

- Face recognition (triplets)
- Image patch correspondence

Thank you!