CS291K – Advanced Data Mining

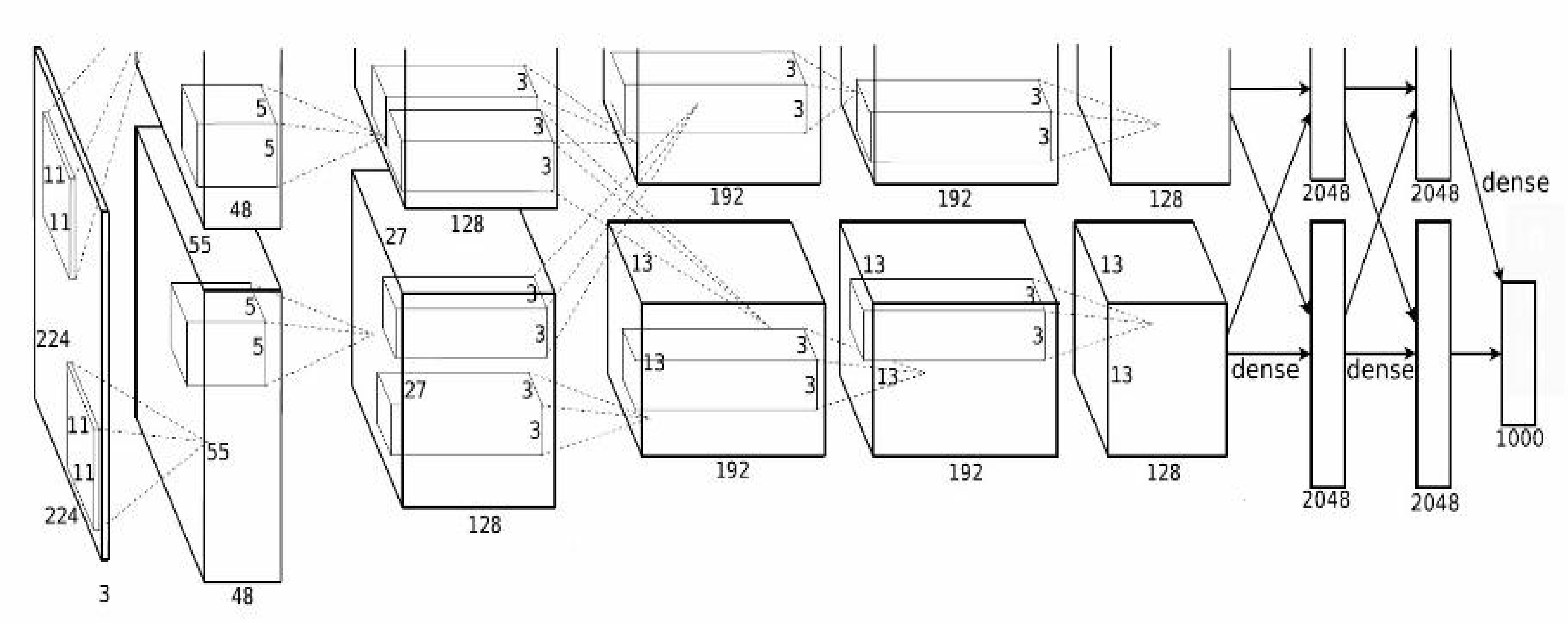
Instructor: Xifeng Yan
Computer Science
University of California at Santa Barbara

Understanding Convolutional Neural Networks

Lecturer: Fangqiu Han
Computer Science
University of California at Santa Barbara

- The slides are made from:
  - **Stanford course 'CS231n: Convolutional Neural Networks for Visual Recognition', Fei-Fei Li and Andrej Karpathy**
  - Coursera online course, '**Neural Networks for Machine Learning',** Geoffrey Hinton
  - Deep Learning – ICML 2013 Tutorial, Yann LeCun

ImageNet Classification with Deep CNN



Input layer 5 conv layers 3 full connection layers

Details/Retrospectives

- Popularize ReLU
- Used Norm layers (not common anymore)
- Heavy data augmentation
- Dropout 0.5 (last fc layers)
- Batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

ImageNet Classification with Deep CNN

- ImageNet Dateset:
  - Over15 million labeled high-resolution images
  - Roughly 22,000 categories
  - Roughly 1000 images in each category

- LSVRC:
  - ImageNet Large Scale Visual Recognition Competition
  - Subset of ImageNet with 1000 categories
  - Roughly 1000 images in each category

Results


mite
mite
black widow
cockroach
tick
starfish
container ship
container ship
lifeboat
amphibian
fireboat
drilling platform
motor scooter
motor scooter
go-kart
moped
bumper car
golfcart
leopard
leopard
jaguar
cheetah
snow leopard
Egyptian cat
grille
convertible
grille
pickup
beach wagon
fire engine
mushroom
agaric
mushroom
jelly fungus
gill fungus
dead-man's-fingers
cherry
dalmatian
grape
elderberry
ffordshire bullterrier
currant
Madagascar cat
squirrel monkey
spider monkey
titi
indri
howler monkey

Understanding ConvNets

- Visualize patches that maximally activate neurons
- Visualize the weights
- Visualize the representation space (e.g. with t-SNE)
- Occlusion experiments
- Deconv approaches (single backward pass)
- Optimization over image approaches (optimization)
- Deep Dream
- How to “fool” ConvNets?

Visualize patches that maximally activate neurons

one-stream AlexNet

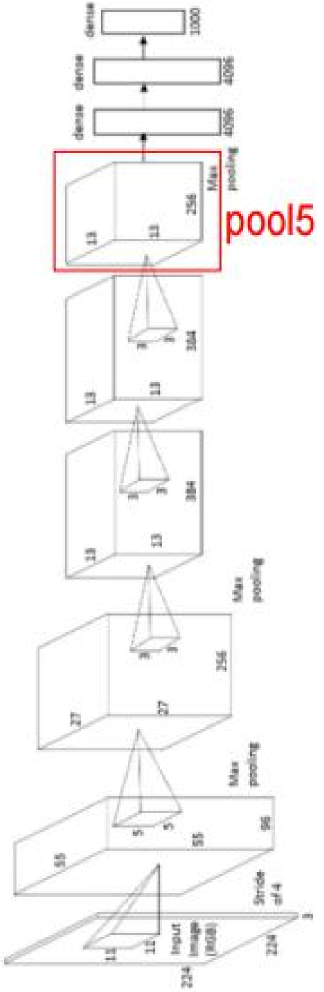
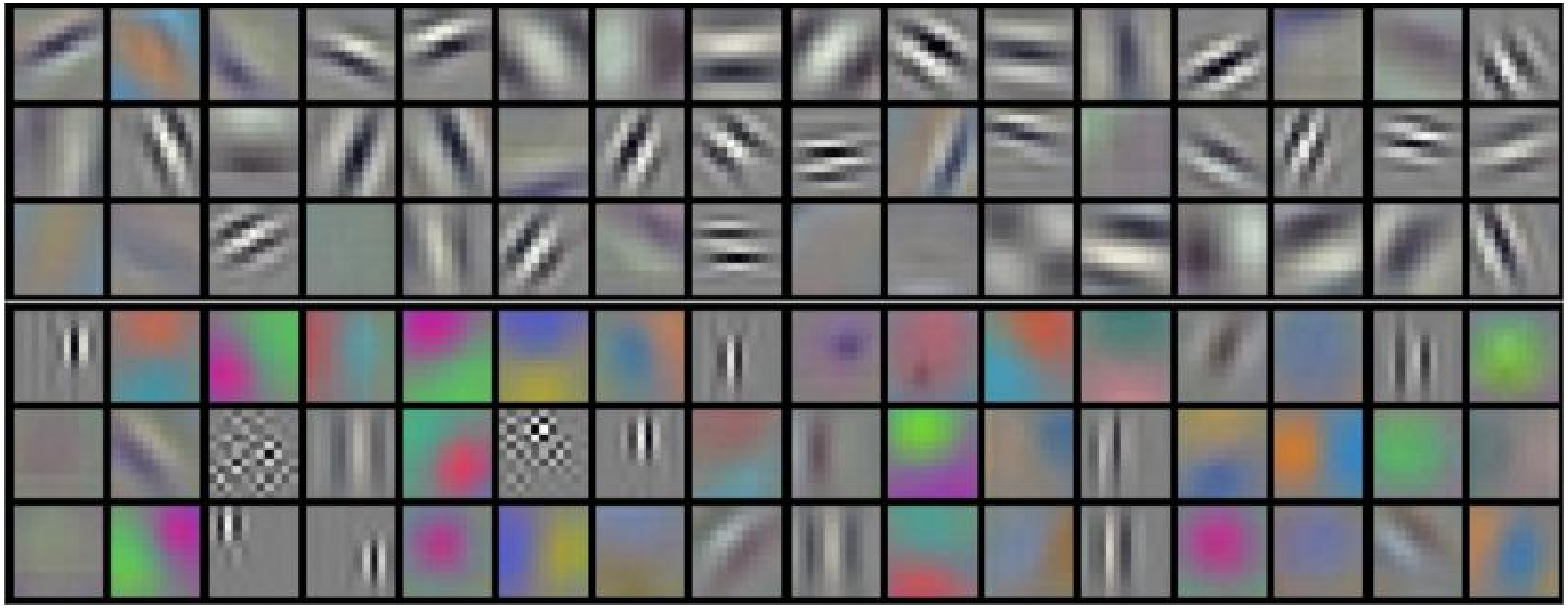


Figure 4: Top regions for six pool$_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

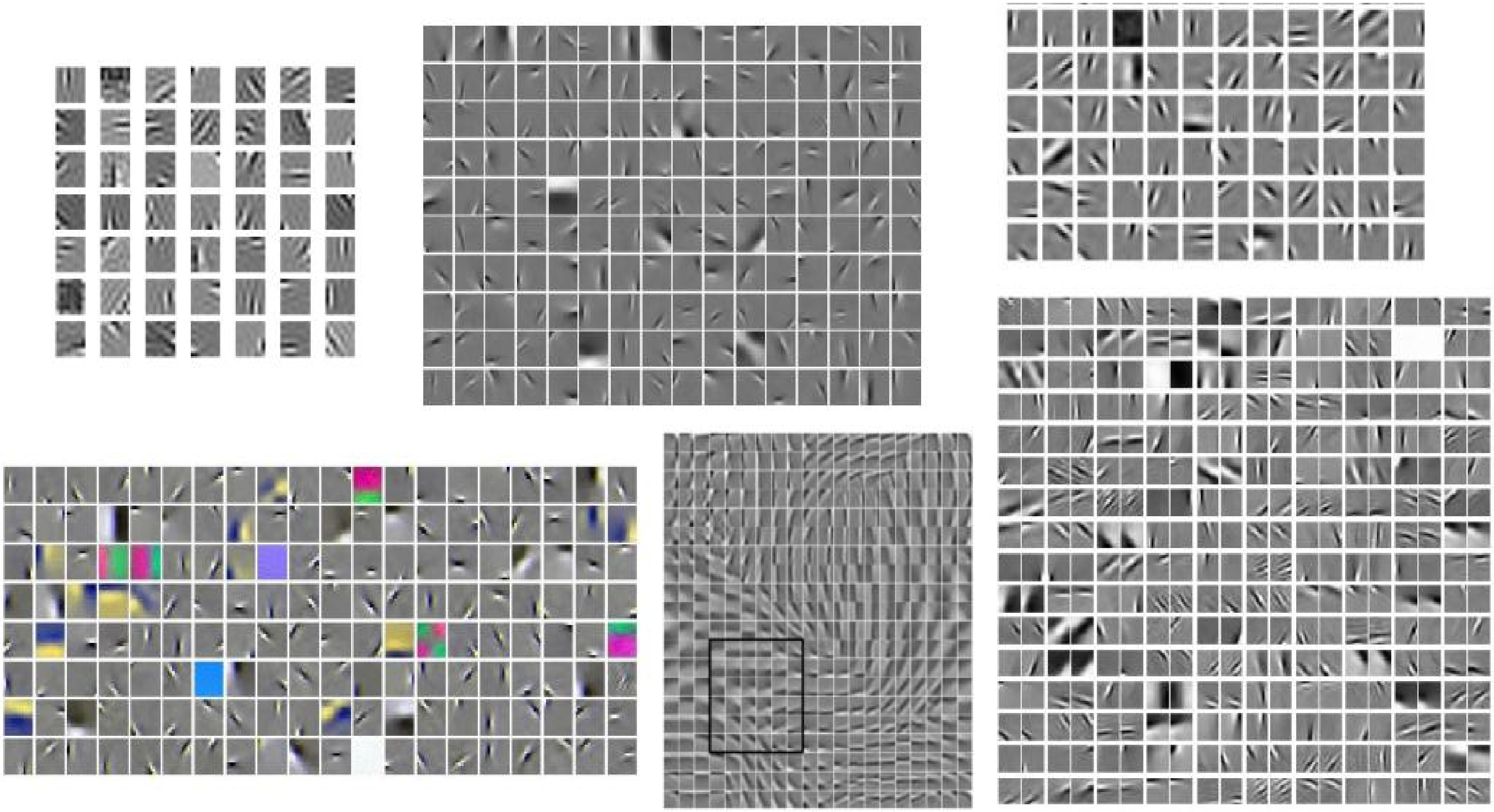
Rich feature hierarchies for accurate object detection and semantic segmentation [Girshick, Donahue, Darrell, Malik]

Visualize the raw weights on the first layer



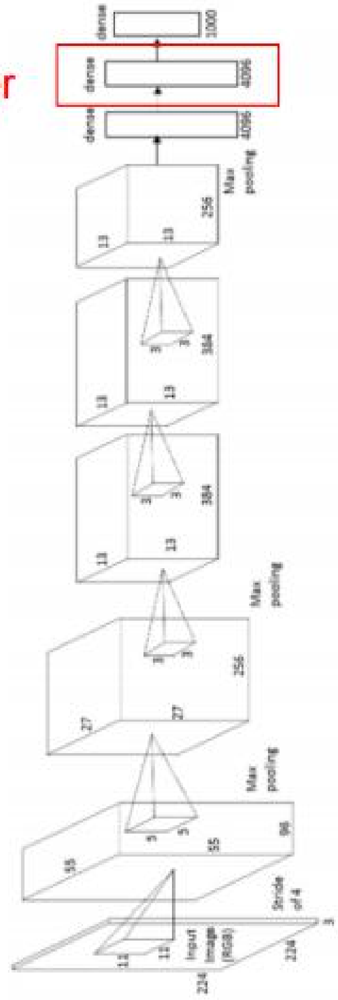
96 convolutional kernels learned by the first layer. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2.

The edge-like filters



Visualizing the representation

fc7 layer



4096-dimensional "code" for an image
(layer immediately before the classifier)

can collect the code for many images

Visualizing the representation

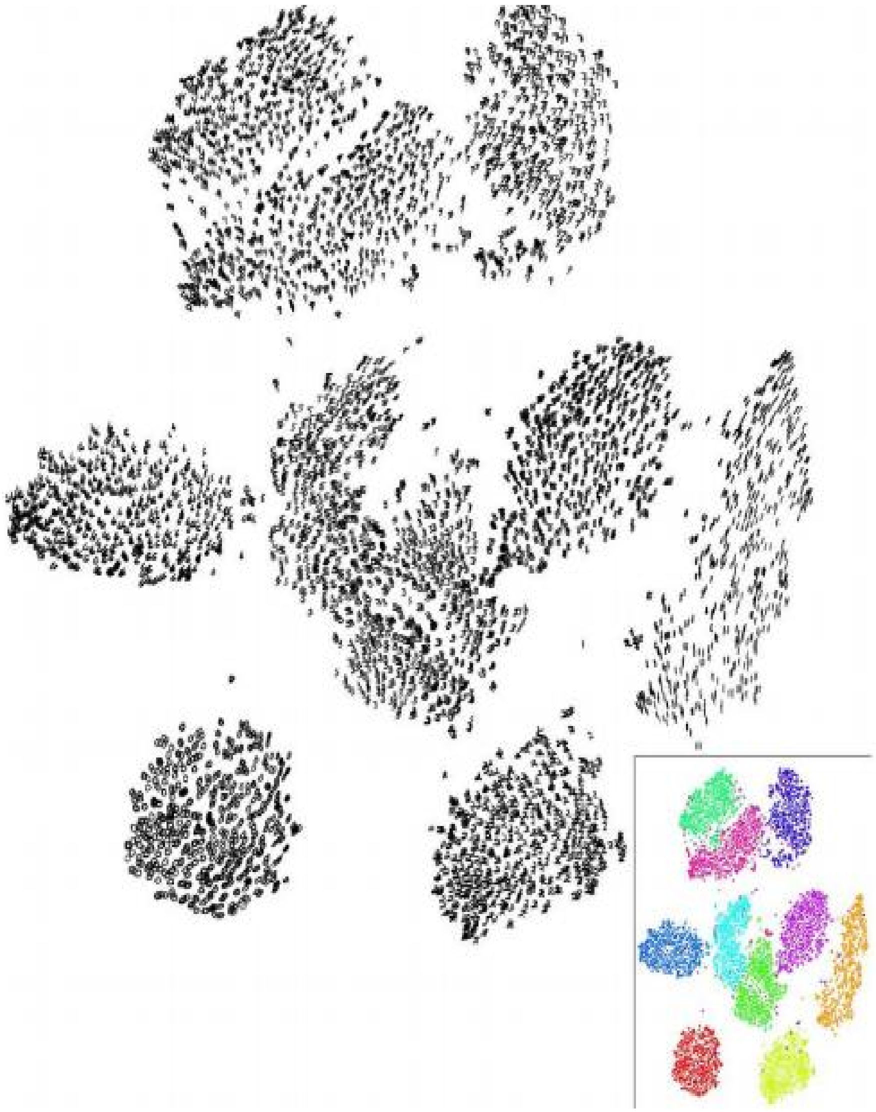
t-SNE visualization

[van der Maaten & Hinton]

Embed high-dimensional points so that locally, pairwise distances are conserved

i.e. similar things end up in similar places. dissimilar things end up wherever

Right: Example embedding of MNIST digits (0-9) in 2D

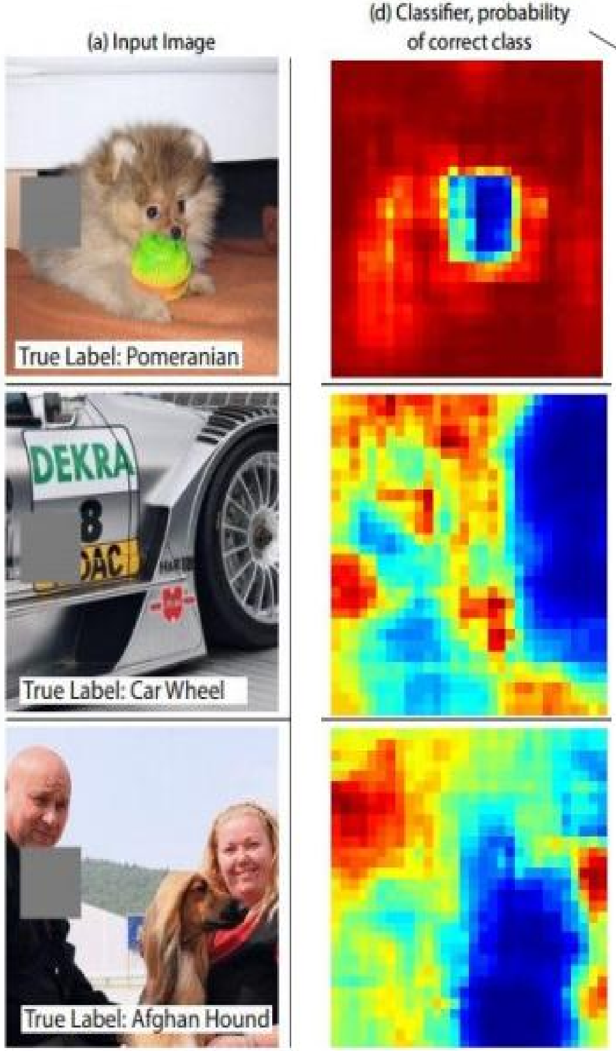


t-SNE visualization:

two images are placed nearby if their CNN codes are close.

See more: http://cs.stanford.edu/people/karpathy/cnnembed/

Occlusion experiments
[Zeiler & Fergus 2013]



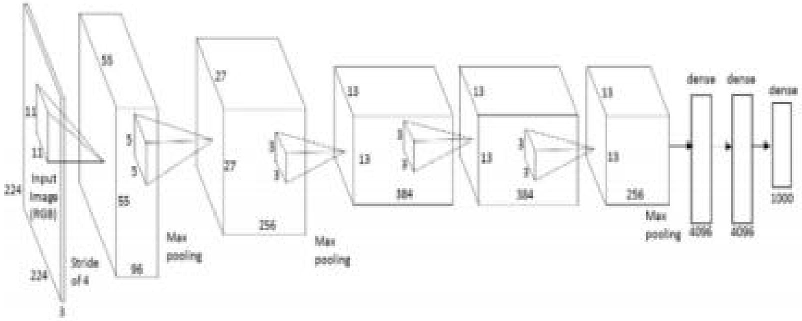
(as a function of the position of the square of zeros in the original image)

Visualizing Activations

https://www.youtube.com/watch?v=AgkfIQ4IGaM

Deconv approaches

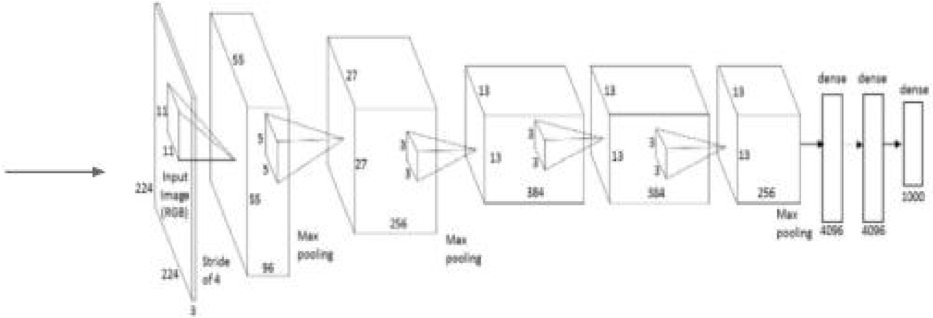
1. Feed image into net



Q: how can we compute the gradient of any arbitrary neuron in the network w.r.t. the image?

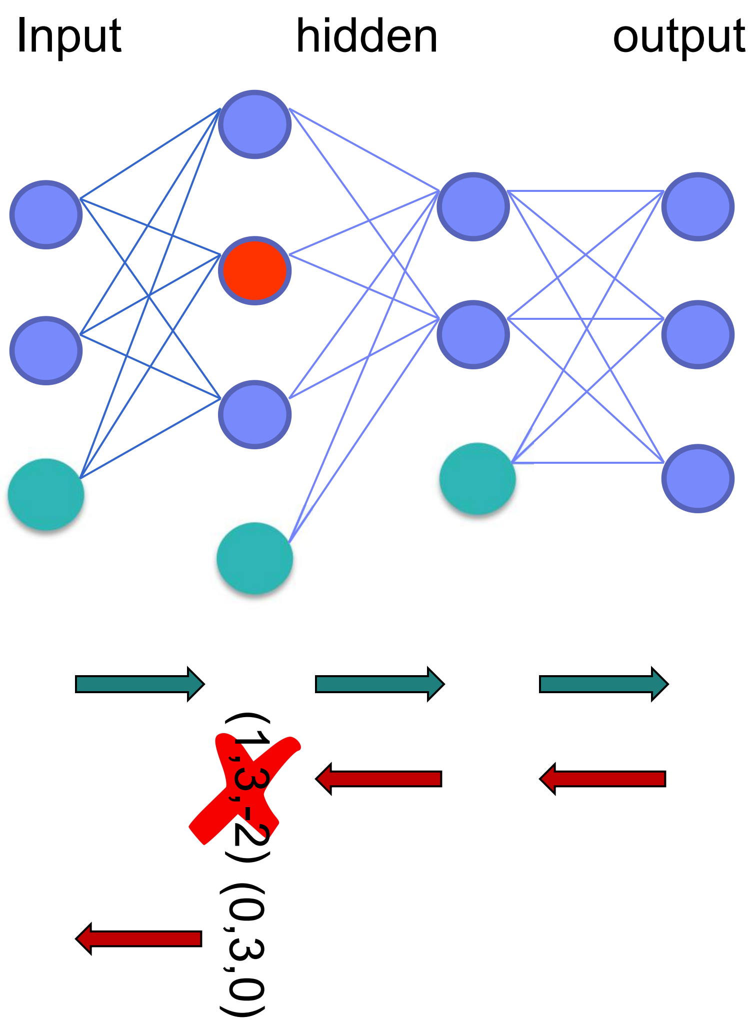
Deconv approaches

1. Feed image into net



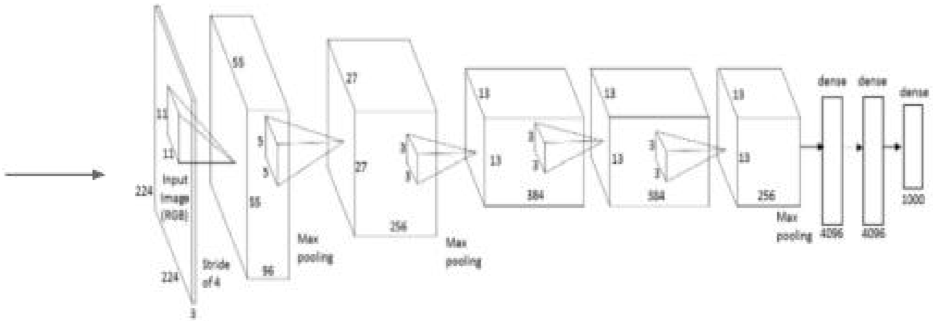
2. Pick a layer, set the gradient there to be all zero except for one 1 for some neuron of interest
3. Backprop to image:

Deconv approaches



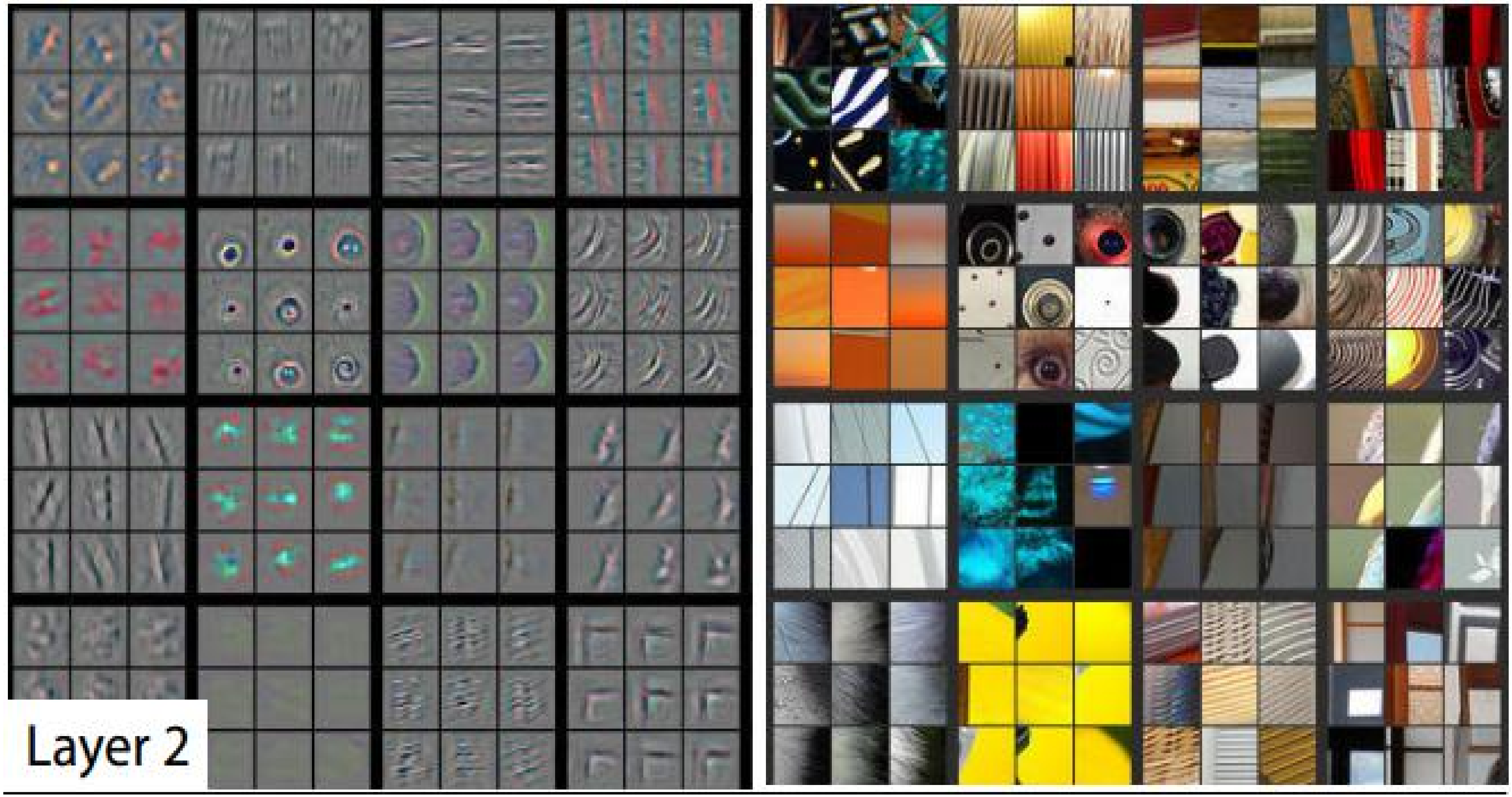
Deconv approaches

1. Feed image into net

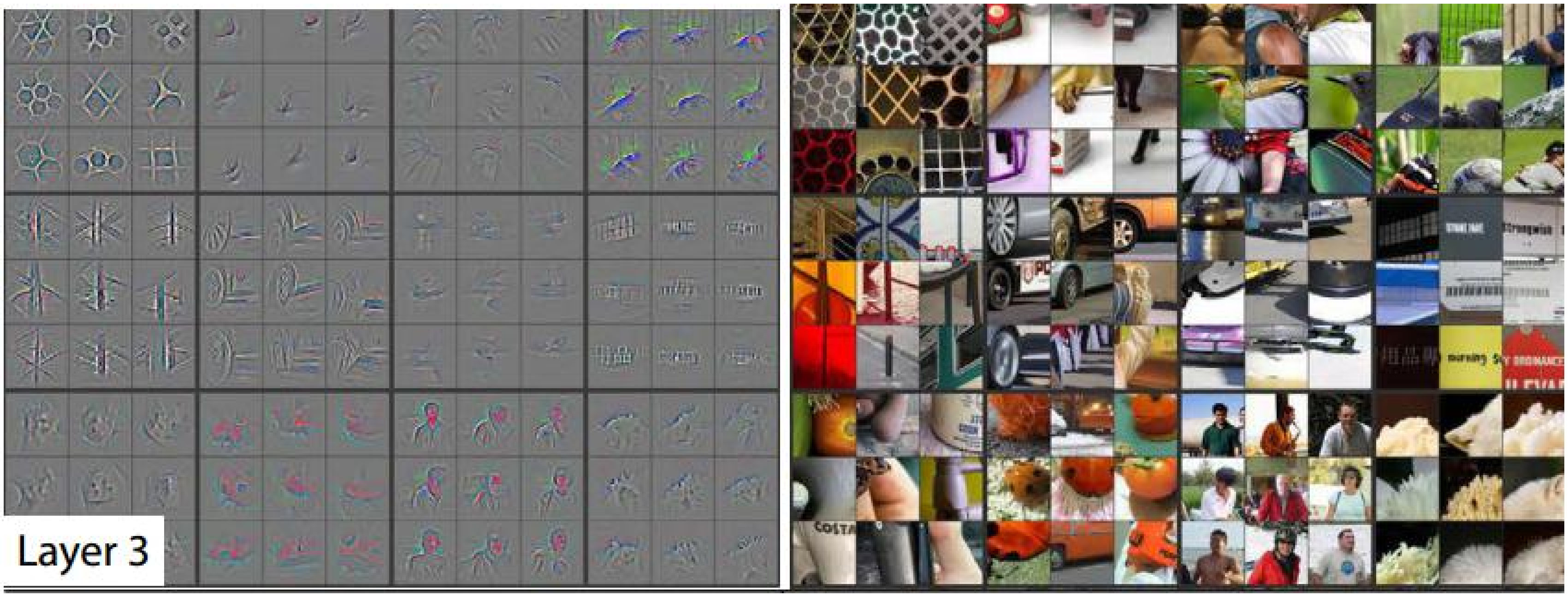


2. Pick a layer, set the gradient there to be all zero except for one 1 for some neuron of interest
3. Backprop to image:

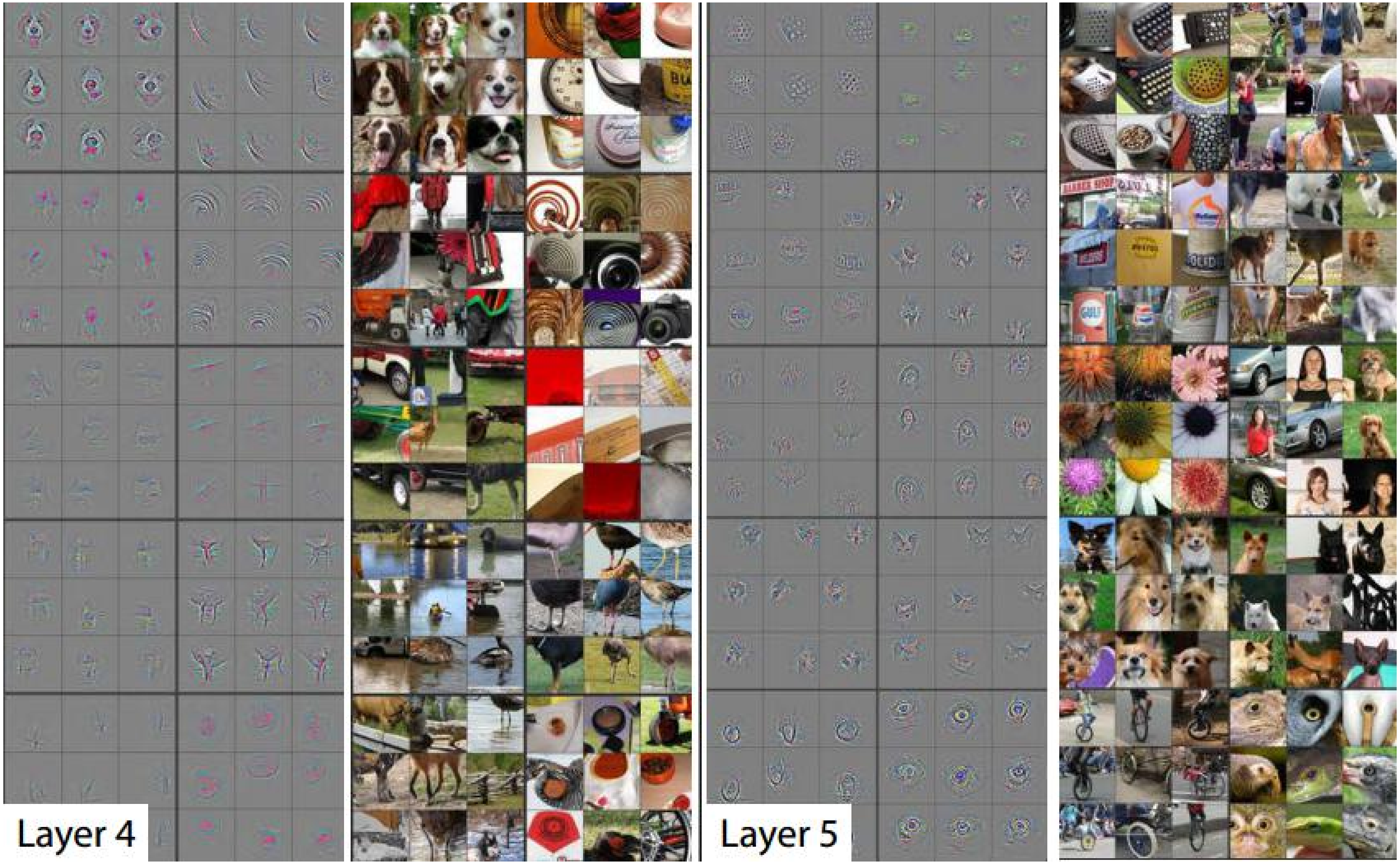
Visualizing and Understanding Deep Neural Networks



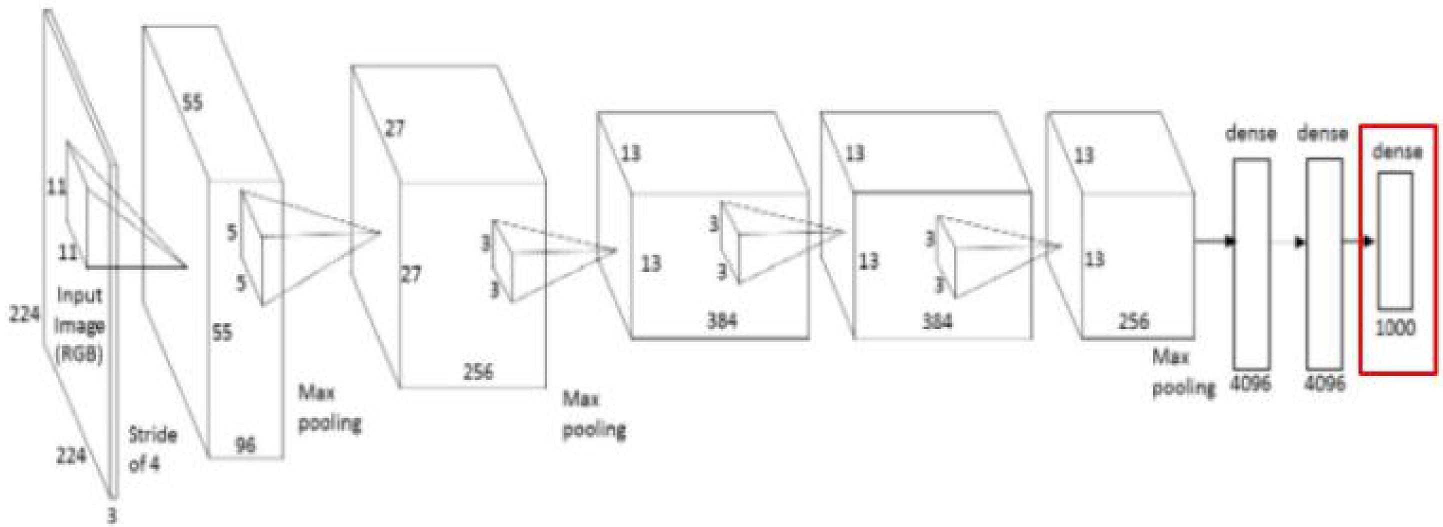
Visualizing and Understanding Deep Neural Networks



Visualizing and Understanding Deep Neural Networks



Optimization to Image

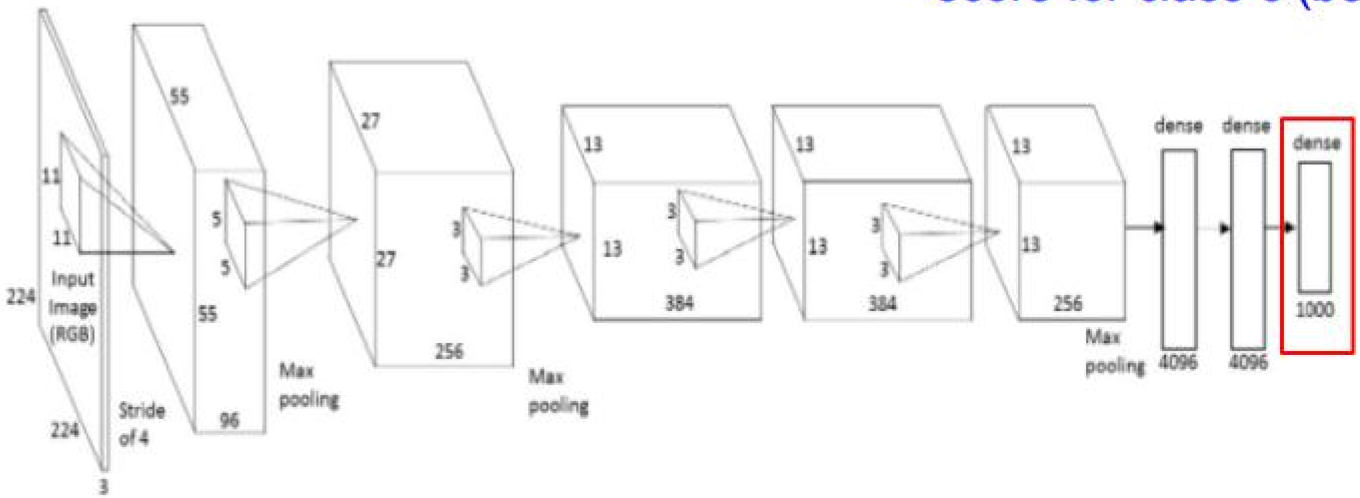


Q: can we find an image that maximizes some class score?

Optimization to Image

$$\arg\max_{I} \boxed{S_c(I)} - \lambda \|I\|_2^2$$

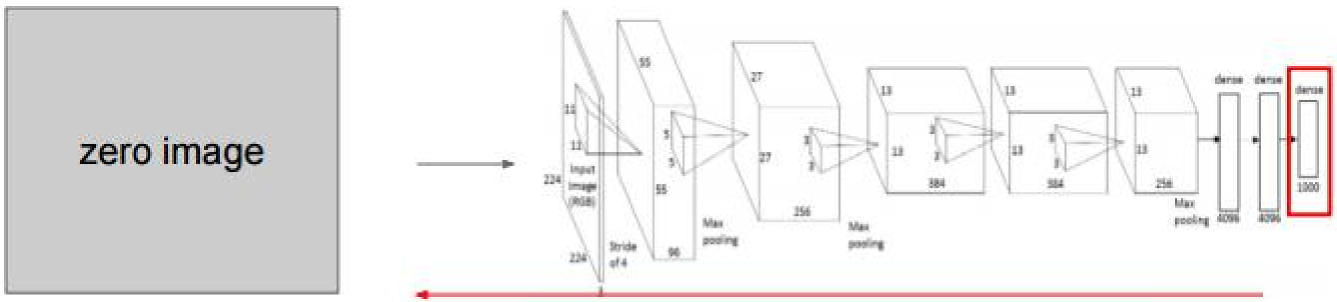
score for class c (before Softmax)



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

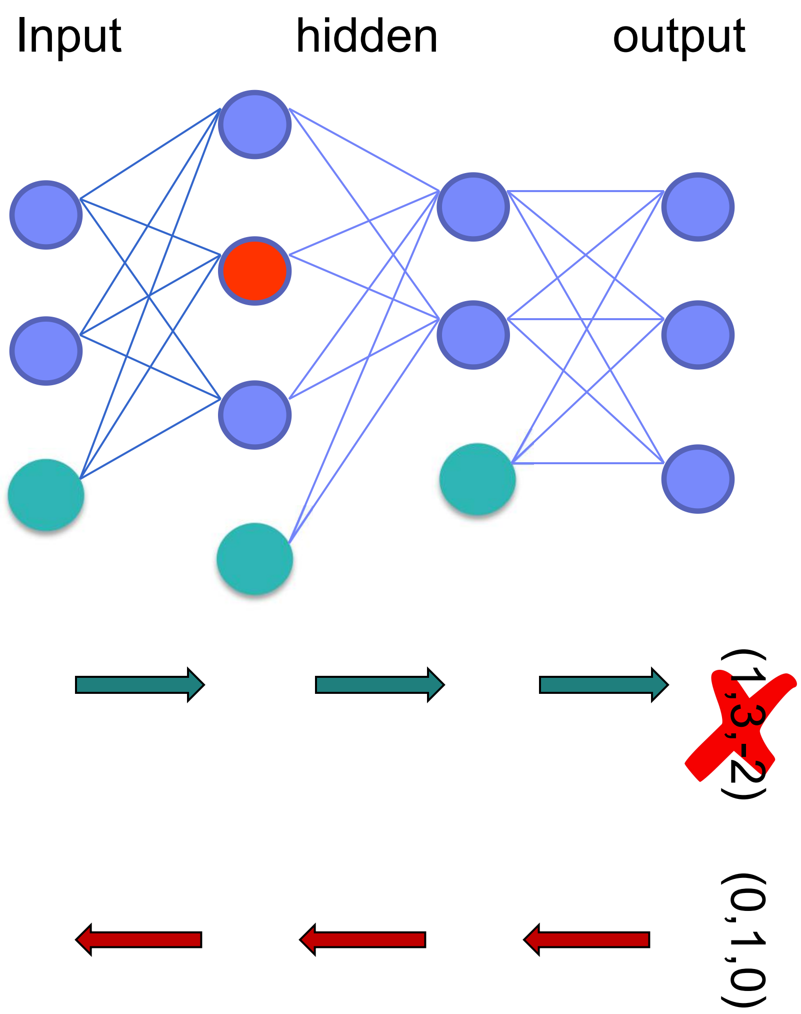
Optimization to Image

1. feed in zeros.



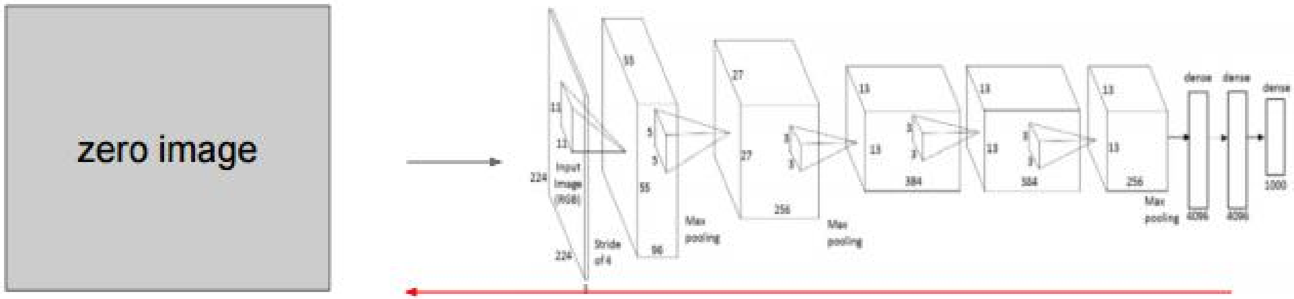
2. set the gradient of the scores vector to be [0,0,....1,....,0], then backprop to image

Optimization to image



Optimization to Image

1. feed in zeros.

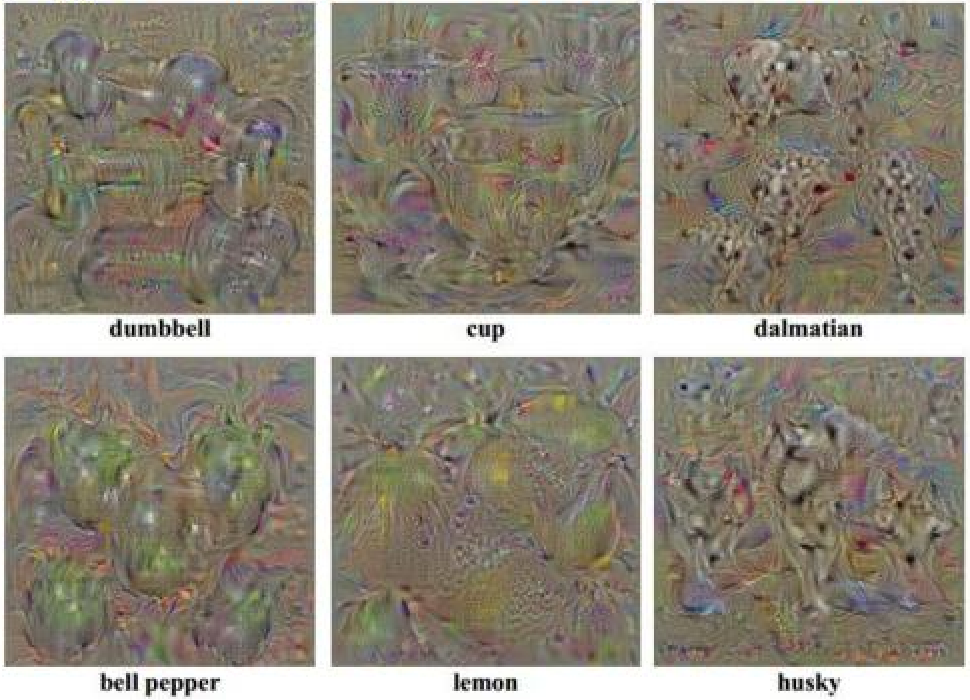


2. set the gradient of the scores vector to be [0,0,....1,....,0], then backprop to image
3. do a small “image update”
4. forward the image through the network.
5. go back to 2.

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

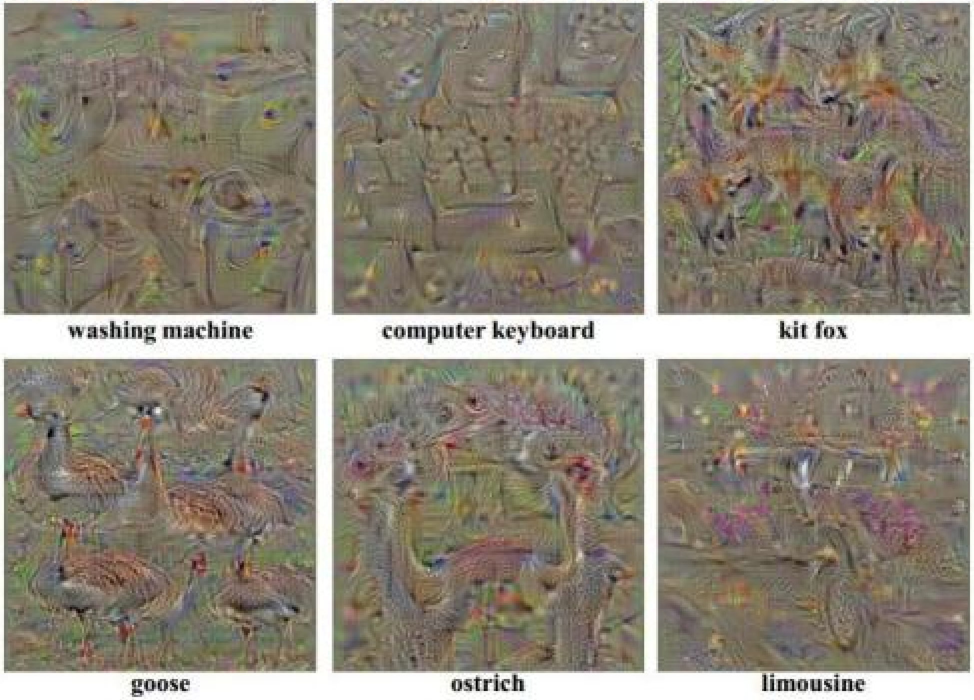
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014

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

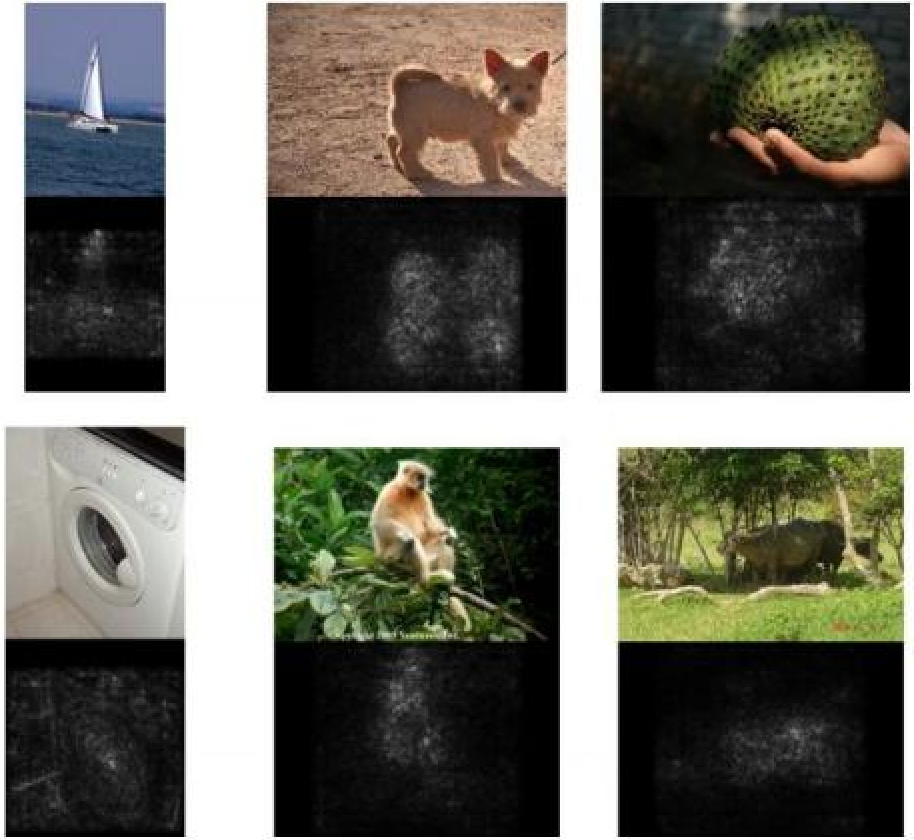
1. Find images that maximize some class score:



washing machine, computer keyboard, kit fox, goose, ostrich, limousine

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

2. Visualize the Data gradient:

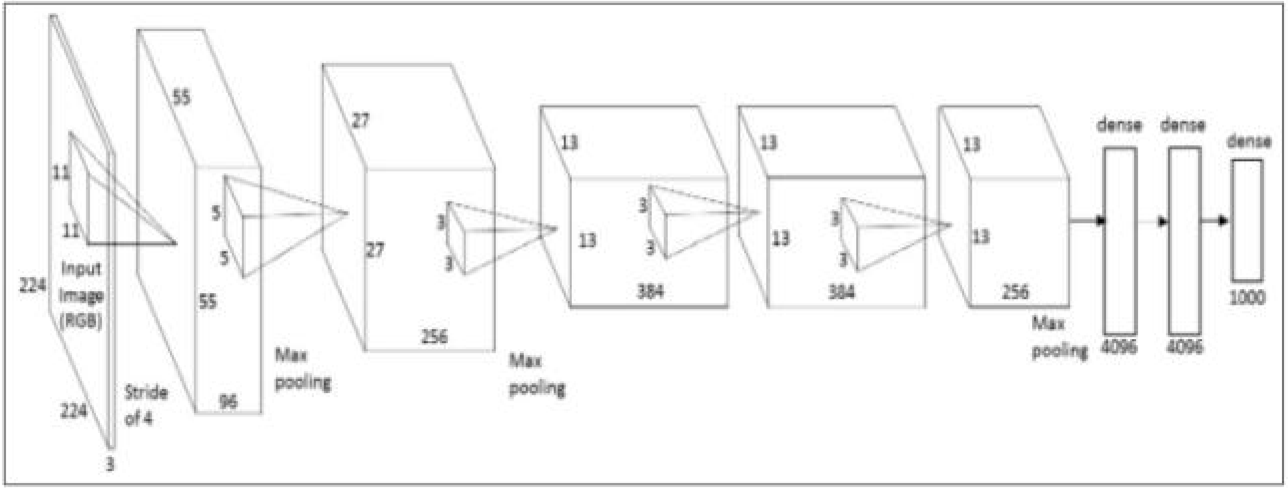


(note that the gradient on data has three channels. Here they visualize M, s.t.:

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

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

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



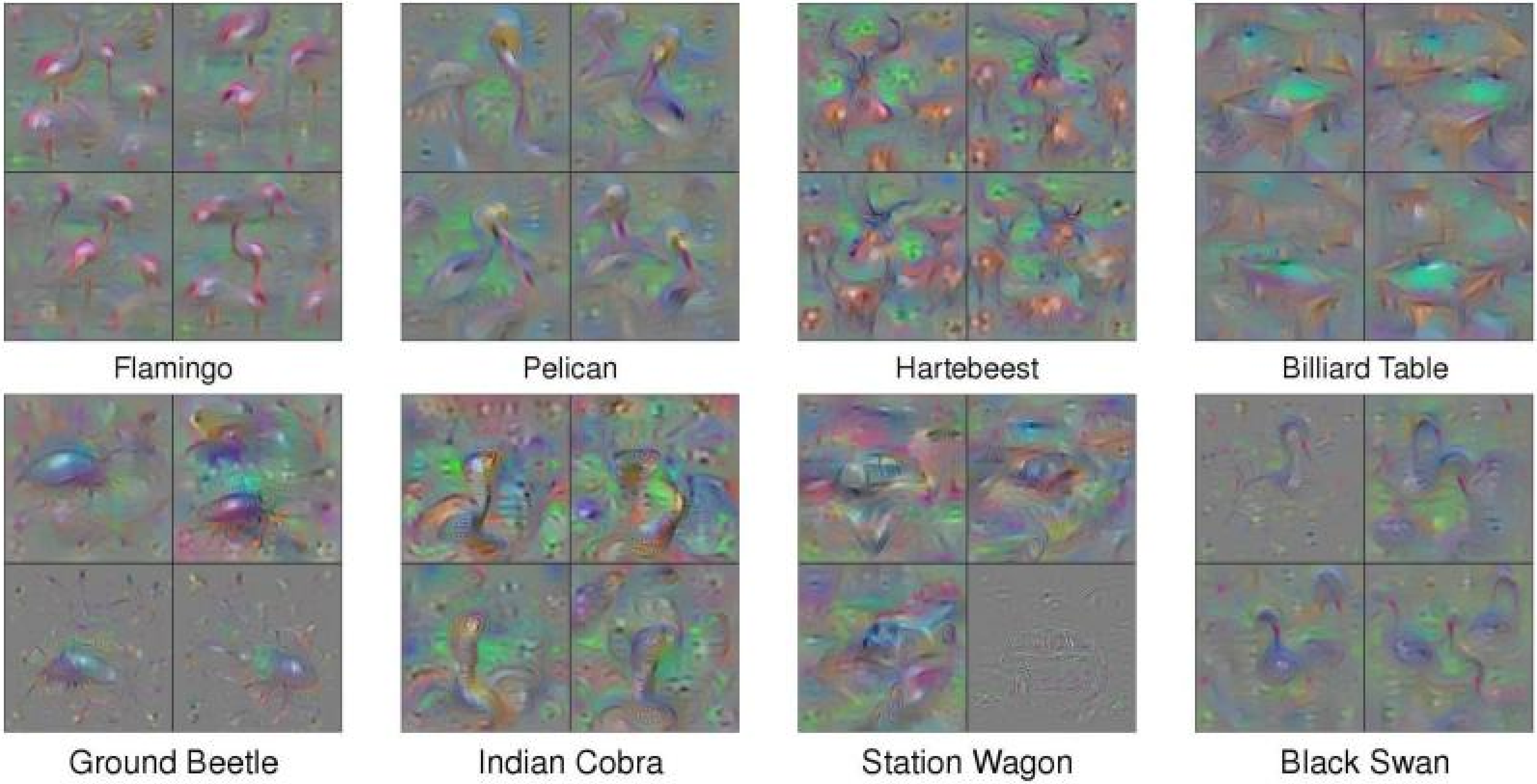
Repeat:

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"

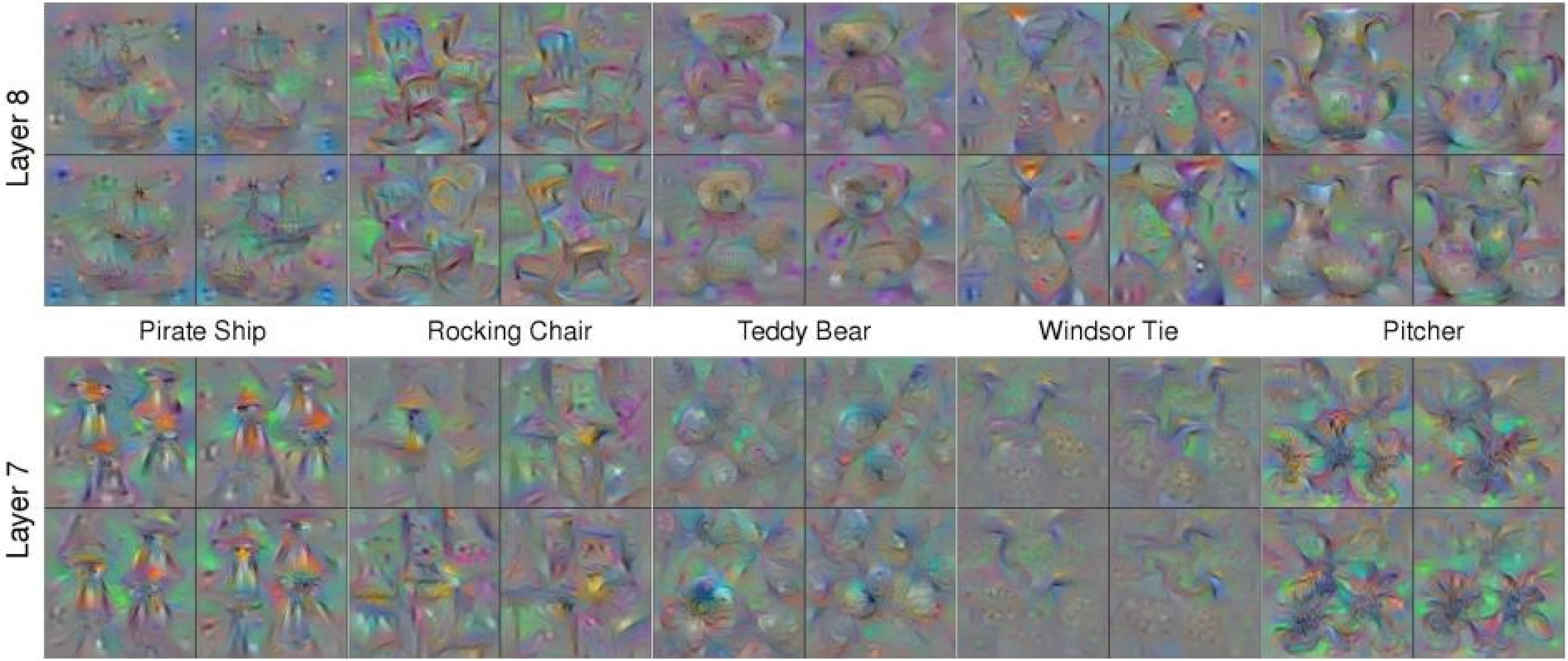
Images that maximize the activtion of neurons

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

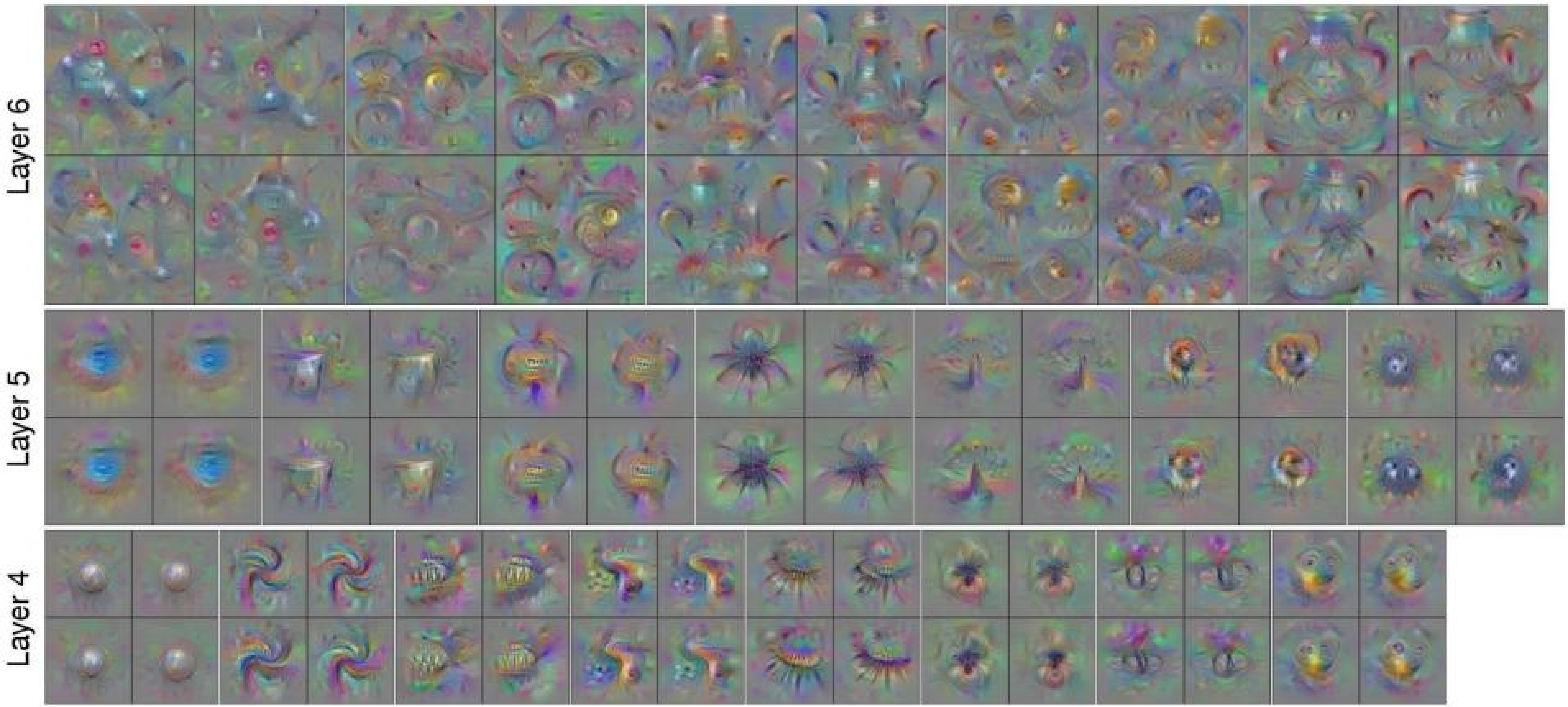
http://yosinski.com/deepvis



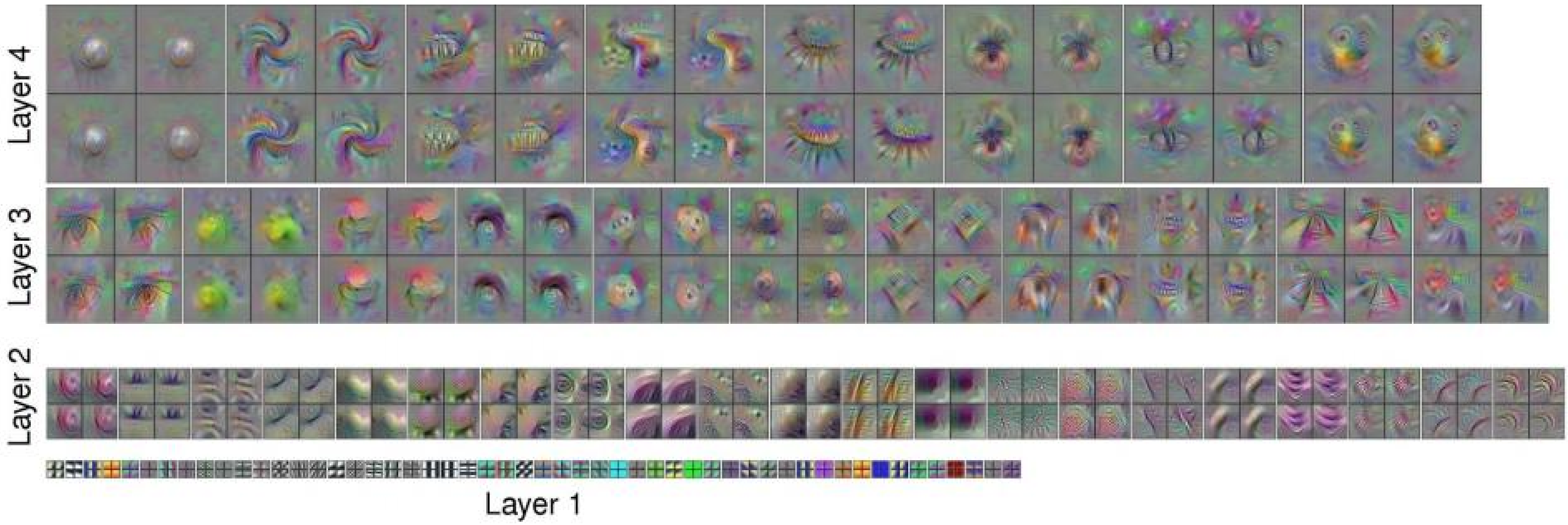
Images that maximize the activtion of neurons



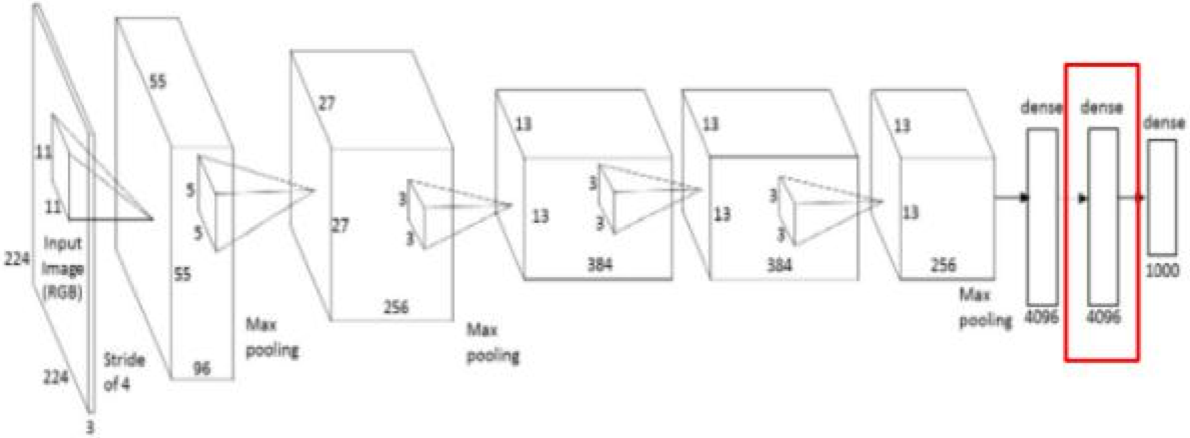
Images that maximize the activtion of neurons



Images that maximize the activtion of neurons



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)

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

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

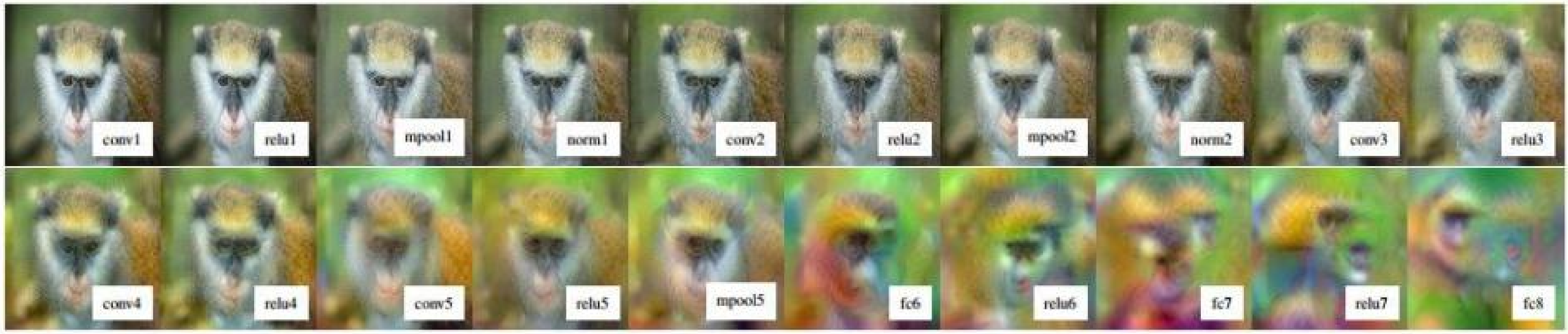
Understanding Deep Image Representations by Inverting Them [Mahendran and Vedaldi, 2014]

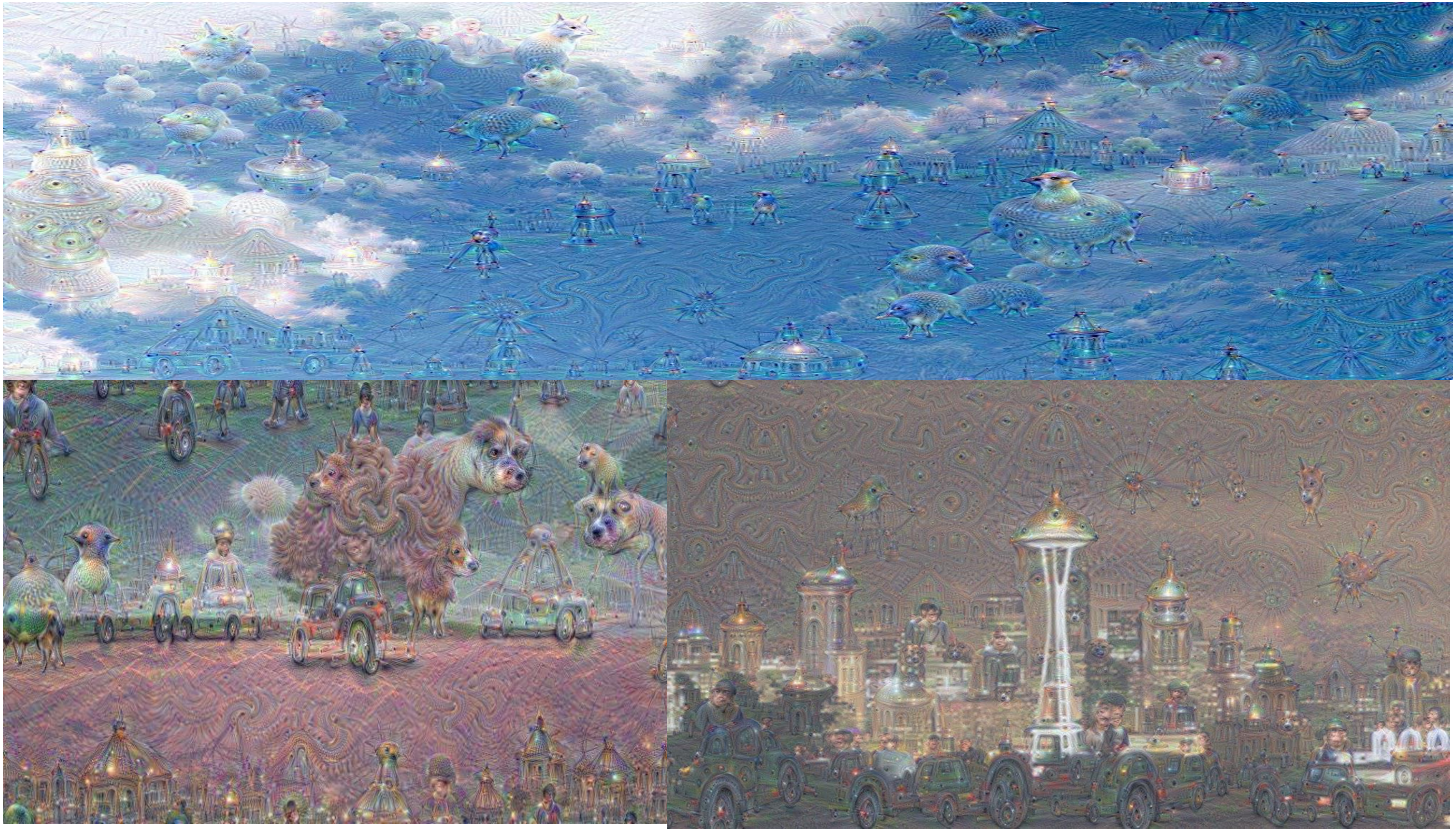
original image

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)

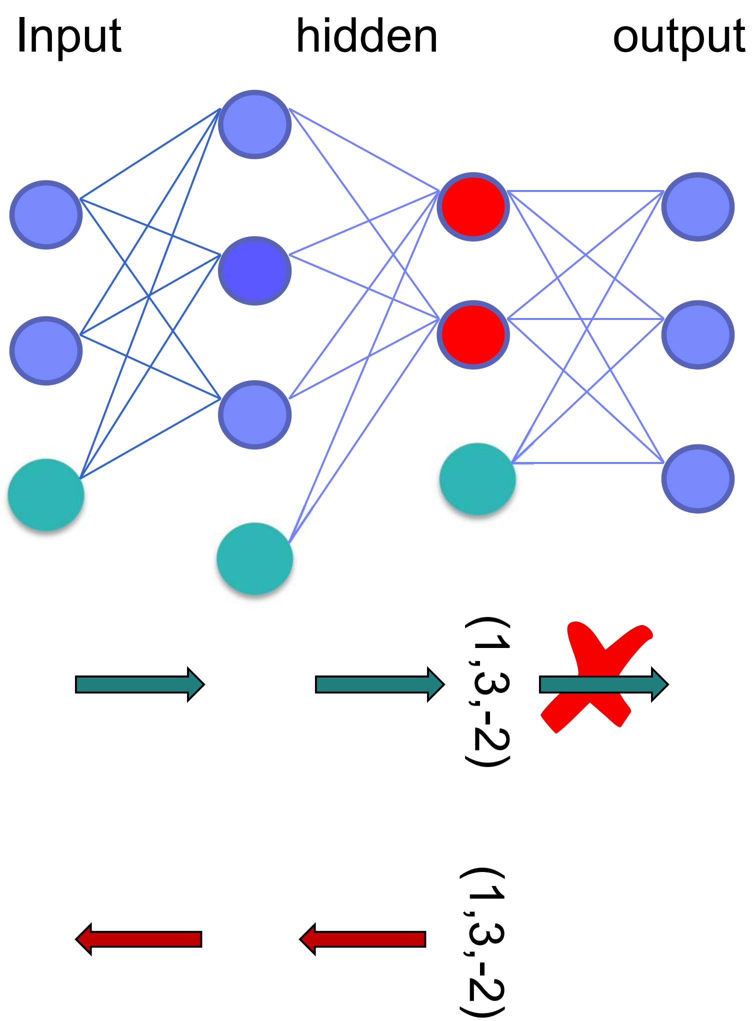
Reconstructions from intermediate layers

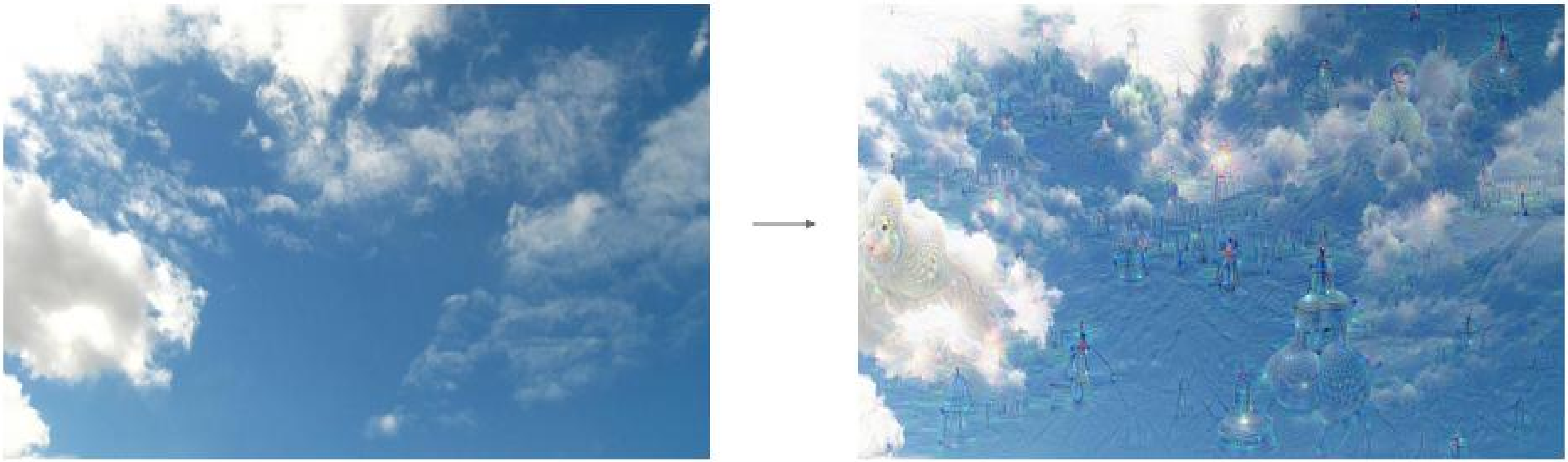




DeepDream https://github.com/google/deepdream

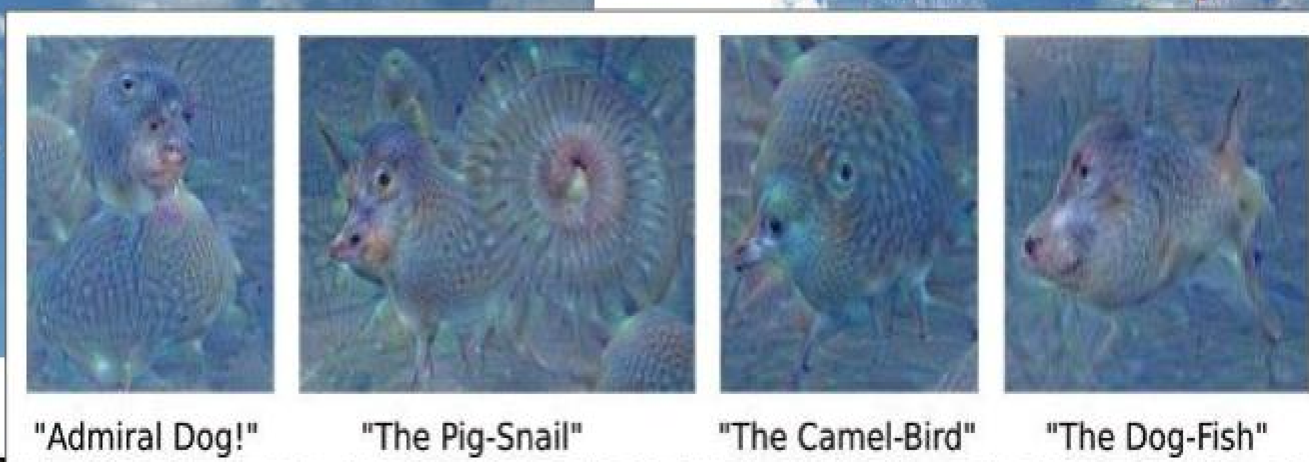
Deep dream





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



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

Bonus videos

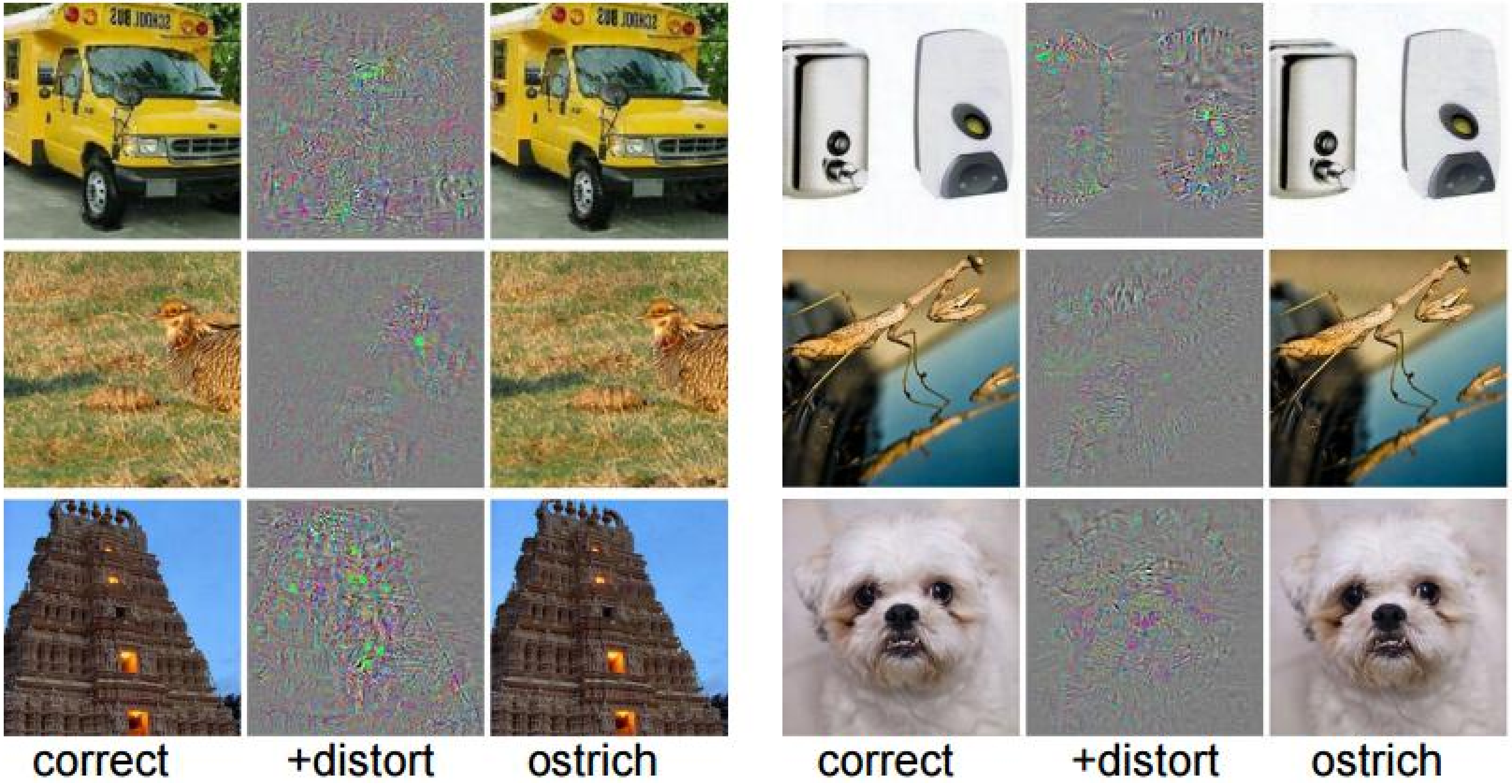
- Deep Dream Grocery Trip
https://www.youtube.com/watch?v=DgPaCWJL7XI

- Deep Dreaming Fear & Loathing in Las Vegas: the Great San Francisco Acid Wave
https://www.youtube.com/watch?v=oyxSerkkP4o

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

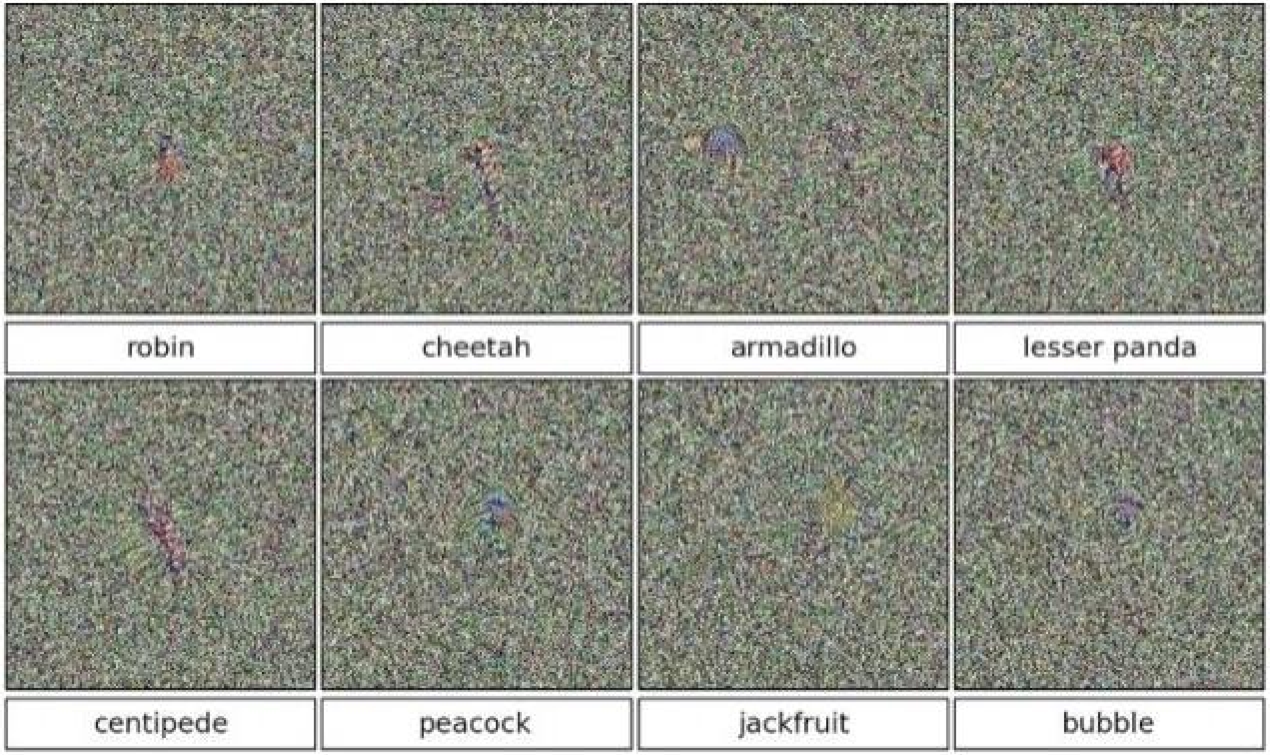
Question: Can we use this to “fool” ConvNets?

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



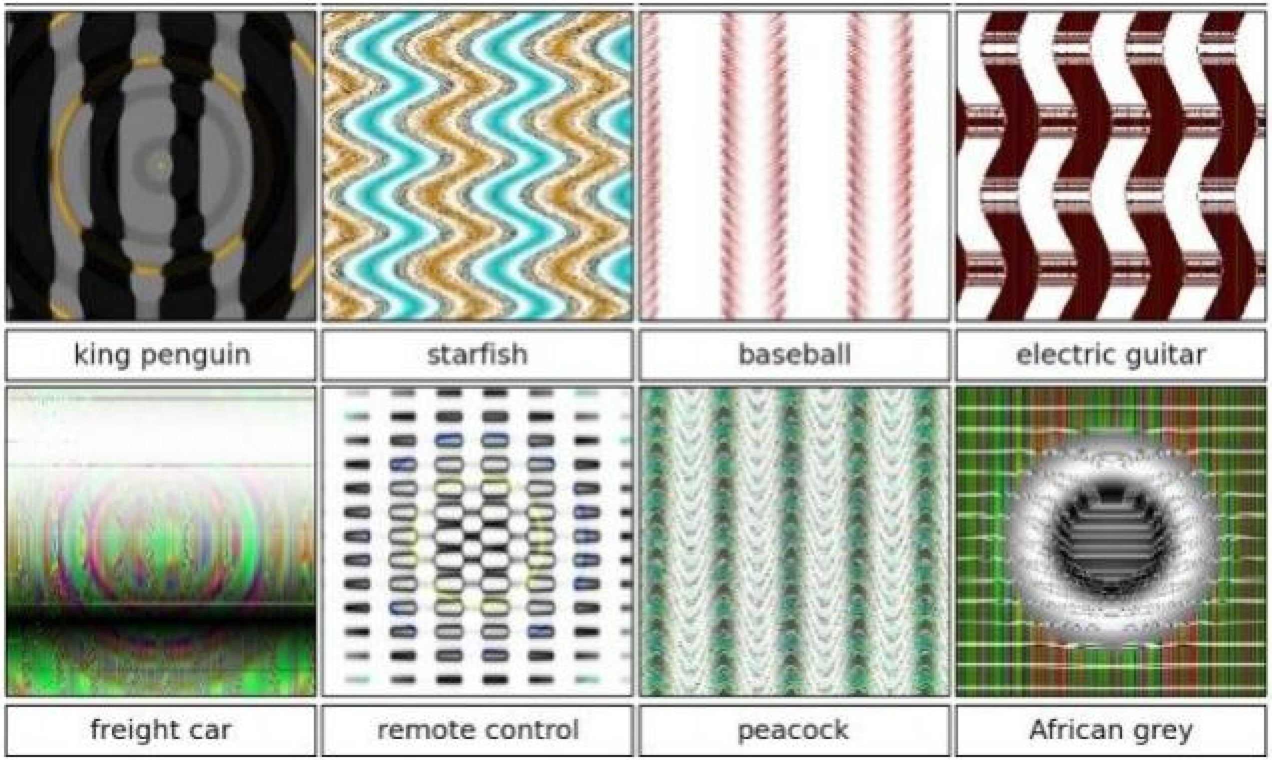
[Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images Nguyen, Yosinski, Clune, 2014]

>99.6% confidences



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

>99.6% confidences



Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1	← input example
w	-1	-1	1	-1	1	-1	1	1	-1	1	← weights

class 1 score = dot product:
= -2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3
=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474
i.e. the classifier is **95%** certain that this is class 0 example.

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x + b)}} = \sigma(w^T x + b)$$

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1	← input example
w	-1	-1	1	-1	1	-1	1	1	-1	1	← weights
adversarial x	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5	

class 1 score before:

-2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3

=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

-1.5+1.5+3.5+2.5+2.5-1.5+1.5-3.5-4.5+1.5 = 2

=> probability of class 1 is now 1/(1+e^(-(2))) = 0.88

i.e. we improved the class 1 probability from 5% to 88%

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x + b)}} = \sigma(w^T x + b)$$

Can we “fool” ConvNets?

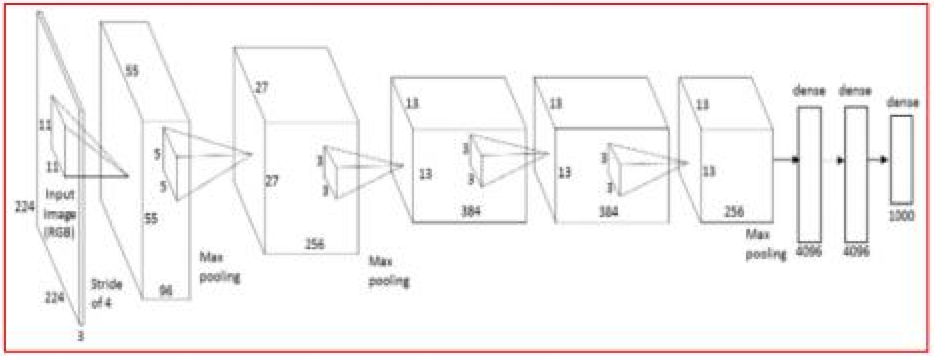
- The primary cause of neural networks vulnerability to adversarial perturbation is their linear nature.

- This is not a problem with Deep Learning, and has little to do with ConvNets specifically. Same issue would come up with Neural Nets in any other modalities.

NeuralStyle

[A Neural Algorithm of Artistic Style by Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge, 2015]
good implementation by Justin in Torch:
https://github.com/jcjohnson/neural-style

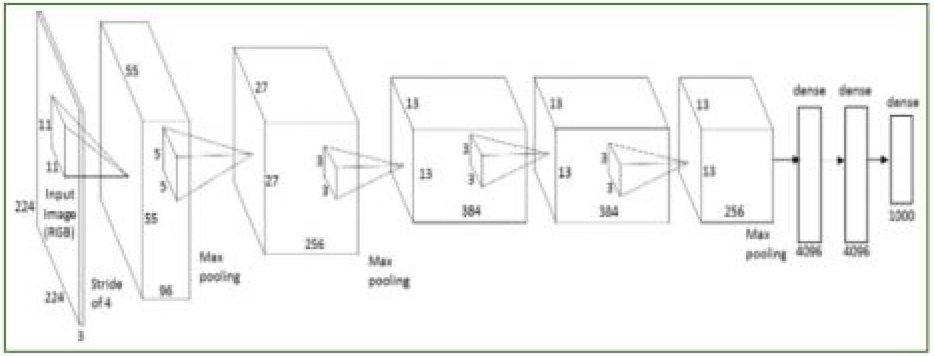
Step 1: Extract **content targets** (ConvNet activations of all layers for the given content image)



content activations

e.g.
at CONV5_1 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

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

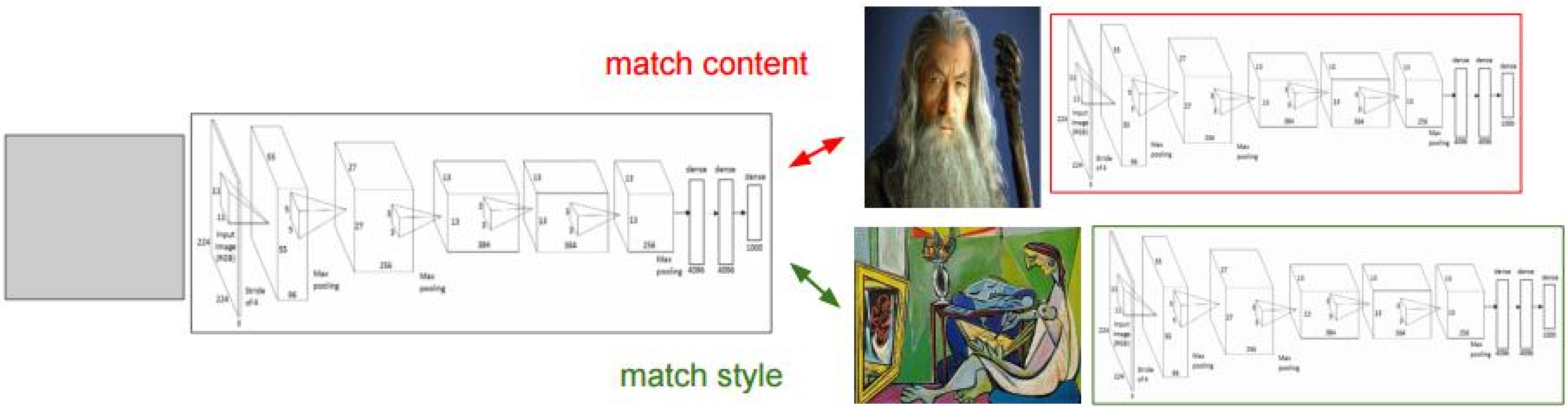
e.g.
at CONV1 layer (with [224x224x64] activations) would give a [64x64] Gram matrix of all pairwise activation covariances (summed across spatial locations)

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)

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

(+Total Variation regularization (maybe))



Questions?