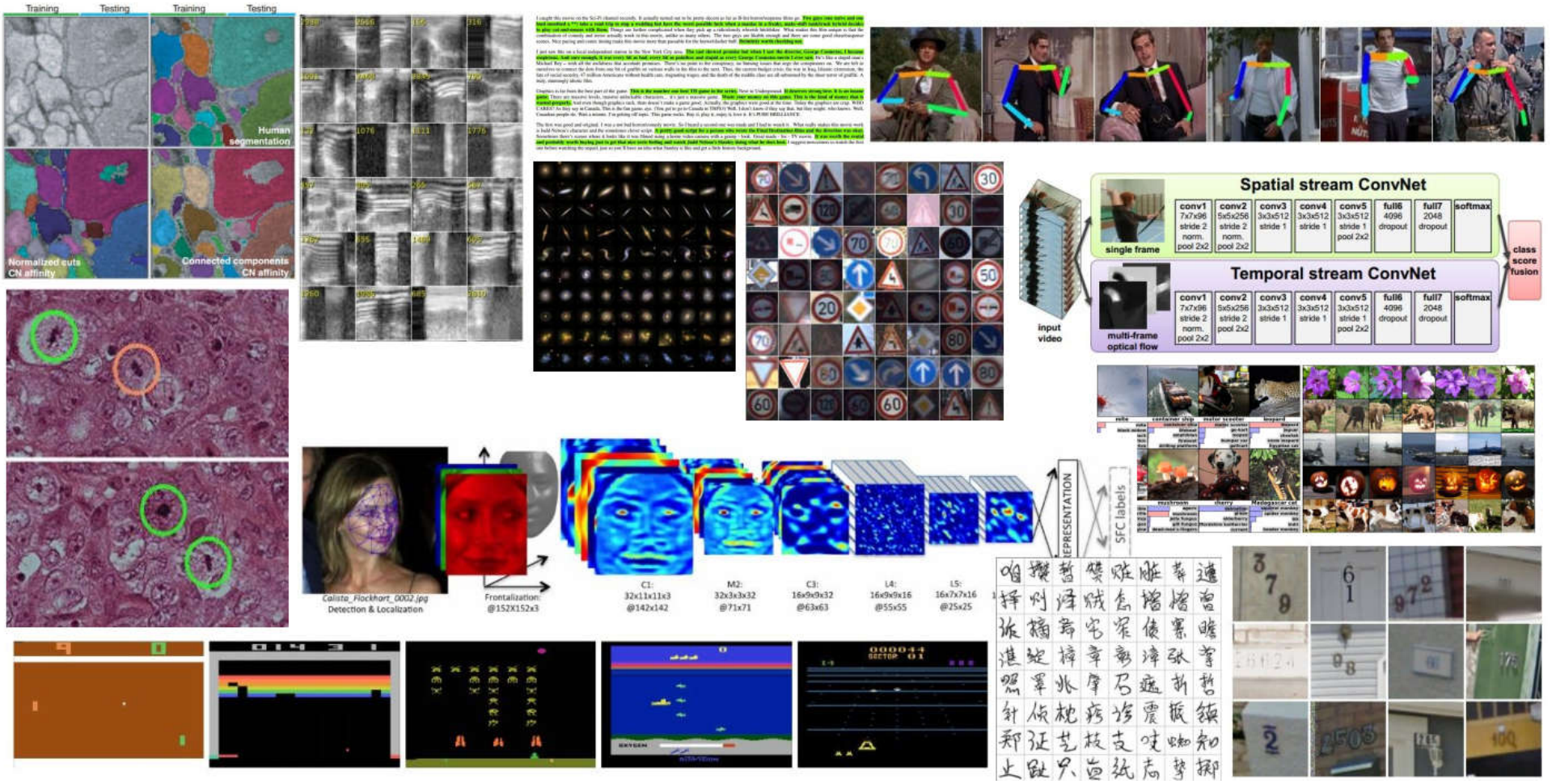
Lecture 09 – Understanding and Visualizing Convolutional Neural Networks

박사과정 김성빈 chengbinjin@inha.edu,
지도교수 김학일 교수 hikim@inha.ac.kr
인하대학교 컴퓨터비전 연구실

CNNs in a Wide Variety of Applications



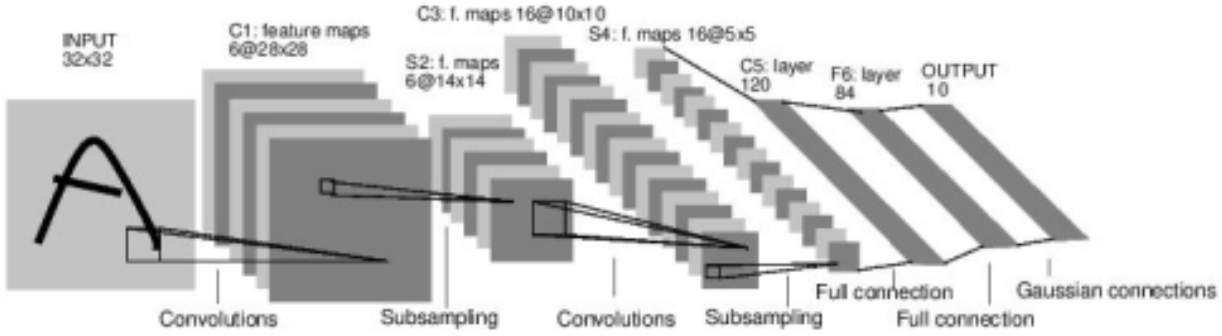
Slide from CS231n

ConvNets Case Study

Case Studies

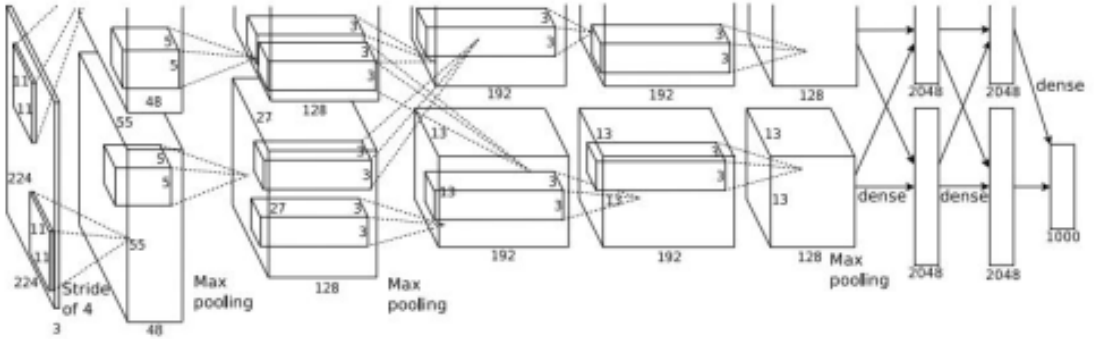
LeNet

(1998)



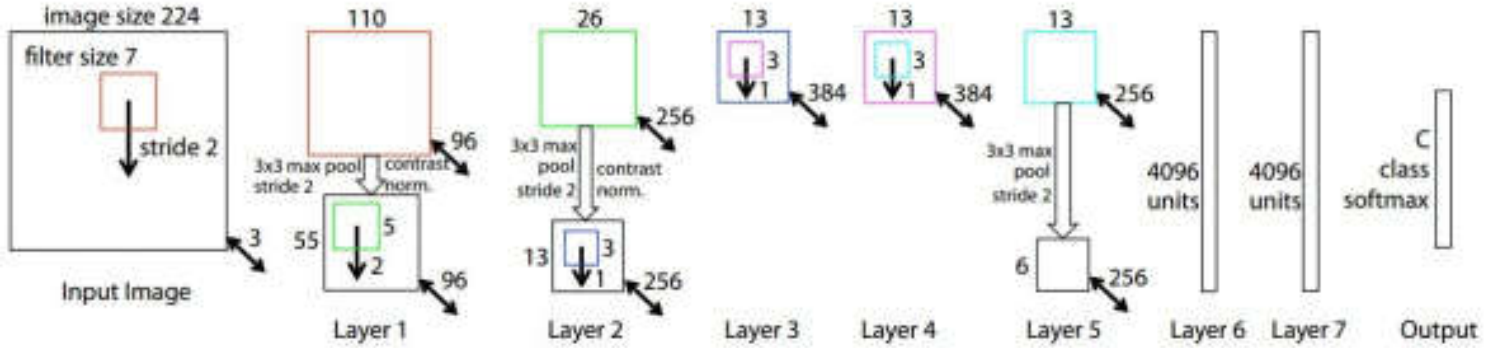
AlexNet

(2012)



ZFNet

(2013)



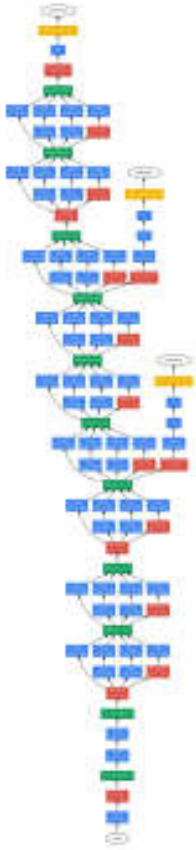
Slide from CS231n

ConvNets Case Study

Case Studies

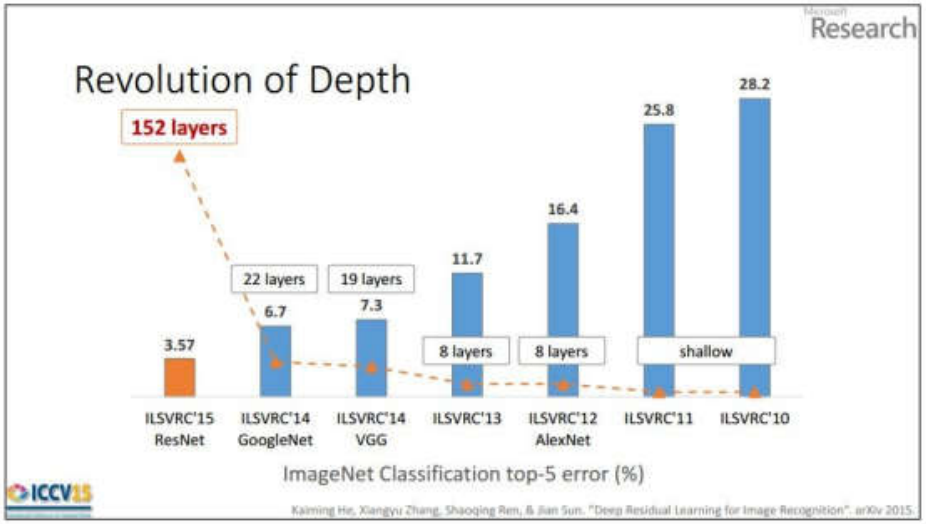
D	E
16 weight layers	19 weight layers
ge)	
conv3-64 conv3-64	conv3-64 conv3-64
conv3-128 conv3-128	conv3-128 conv3-128
conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool	
FC-4096	
FC-4096	
FC-1000	
soft-max	

VGG
(2014)



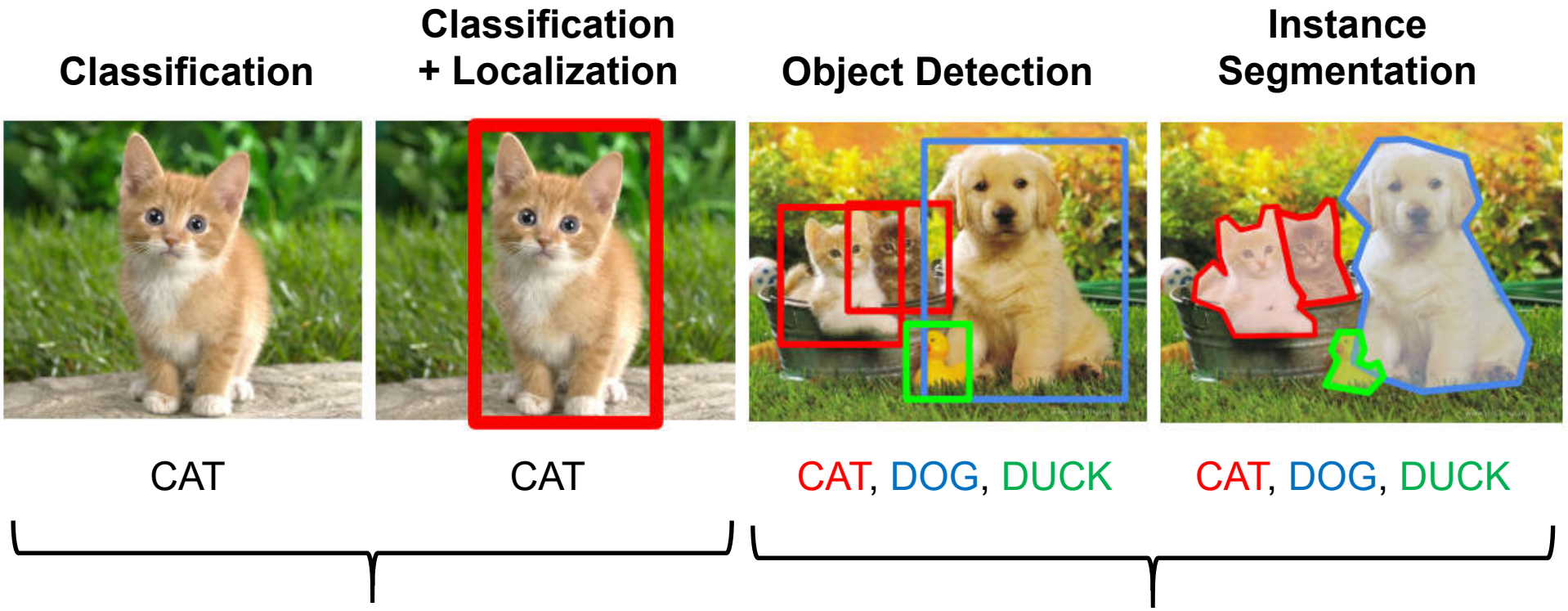
GoogLeNet
(2014)

ResNet
(2014)



Slide from CS231n

Computer Vision Tasks



Slide from CS231n

Understanding ConvNets

I. Visualize patches that maximally activate neurons

II. Visualize the weights

III. Visualize the representation space (e.g. with t-SNE)

IV. Occlusion experiments

V. Deconv approaches (single backward pass)

VI. Optimization over image approaches (optimization)

I. Visualize Patches that Maximally Activate Neurons

Visualize Patches that Maximally Activate Neurons

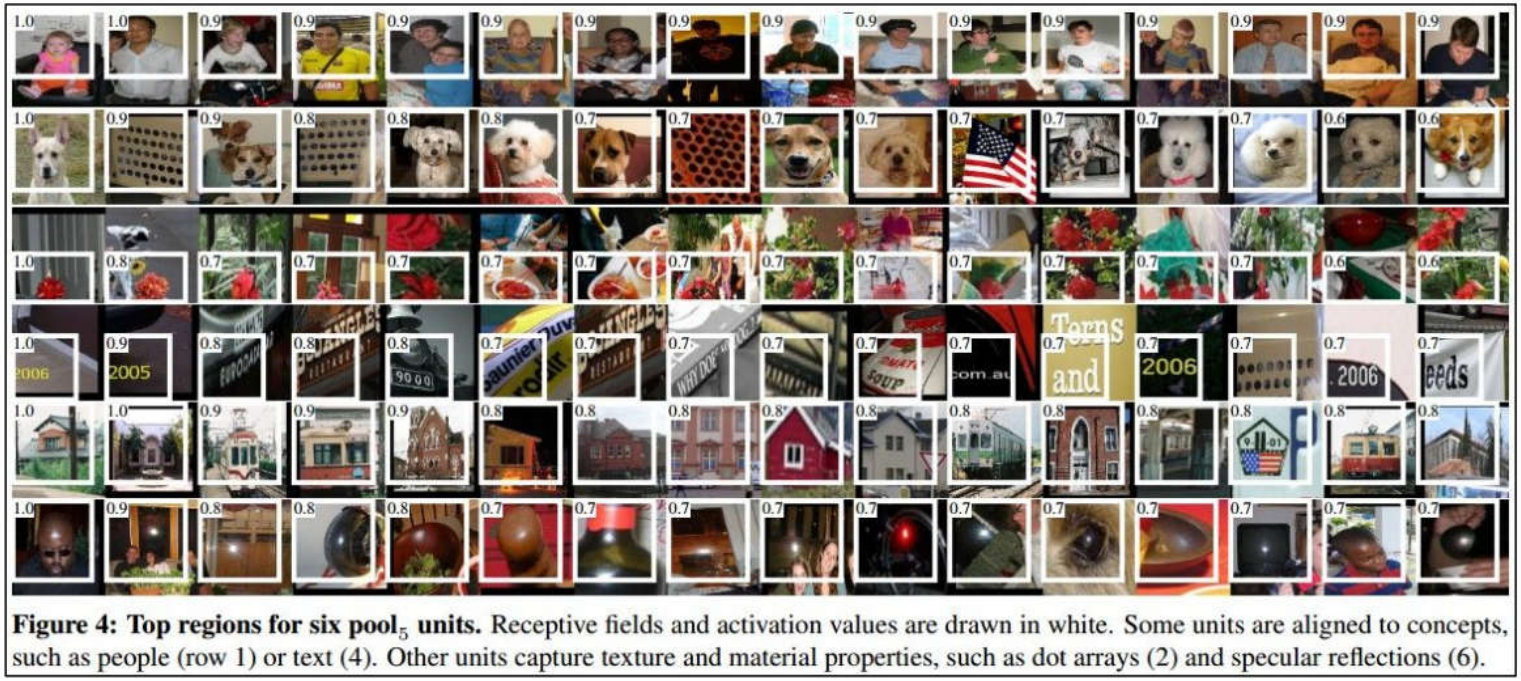
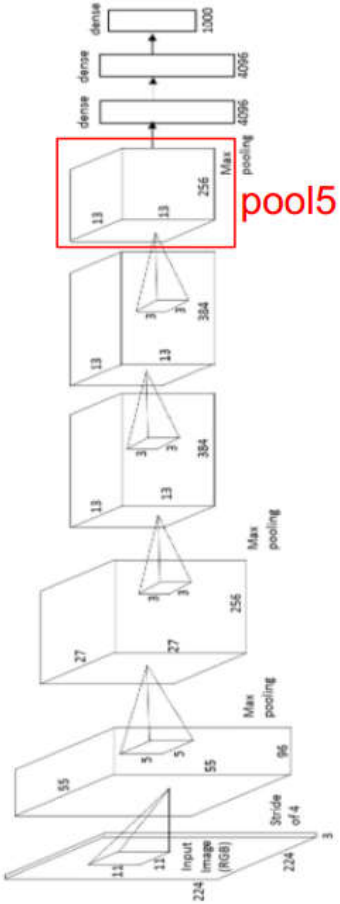


Figure 4: Top regions for six pool$_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

one-stream AlexNet

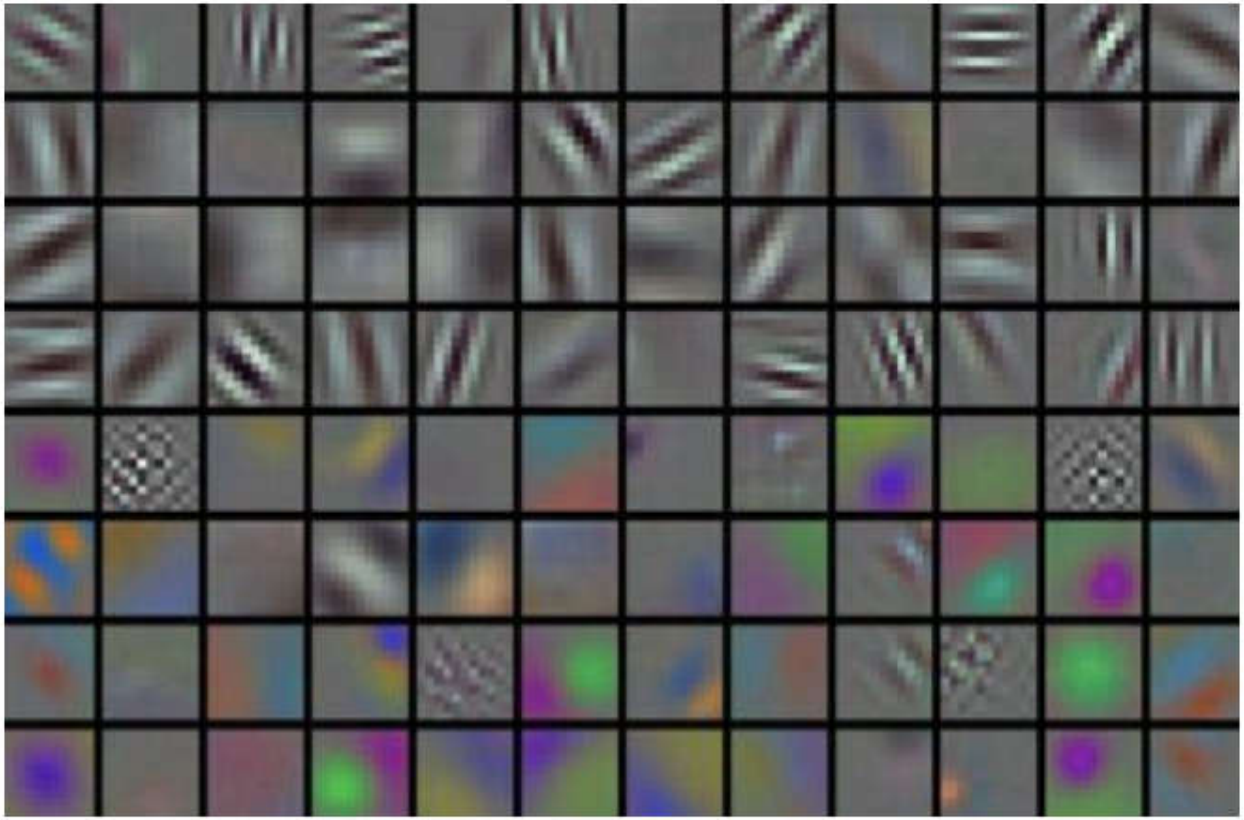


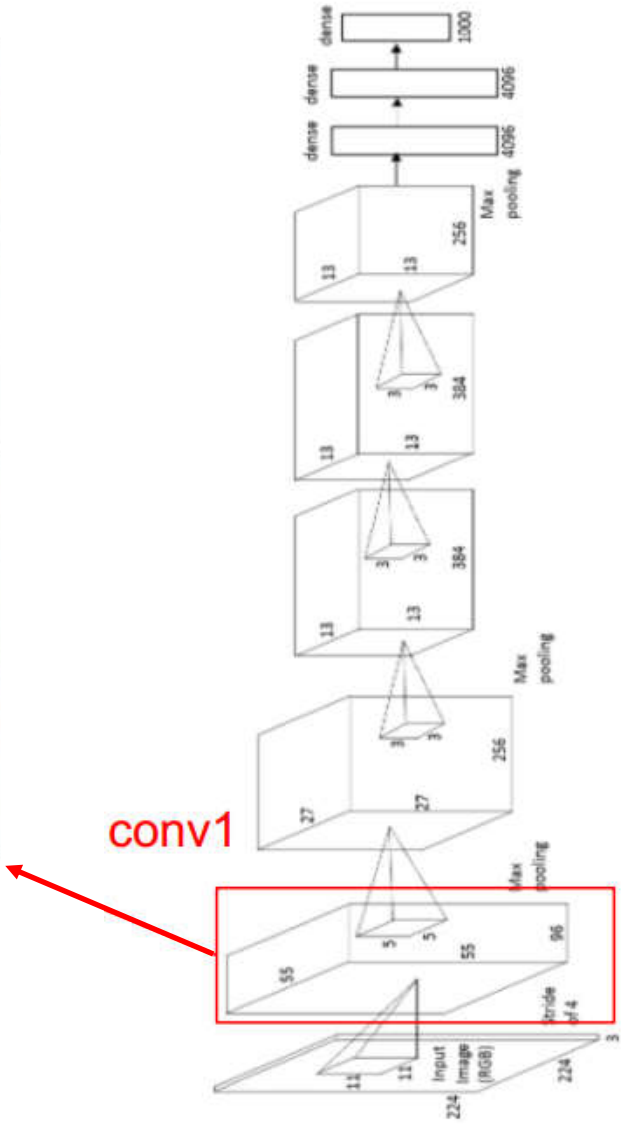
Girshick, et al., Rich feature hierarchies for accurate object detection and semantic segmentation, **CVPR2014**

Slide from CS231n

II. Visualize the Weights

Visualize the Filters/Kernels (Raw Weights)





Different rotated Gabor like features

- Only interpretable on the first layer ☹

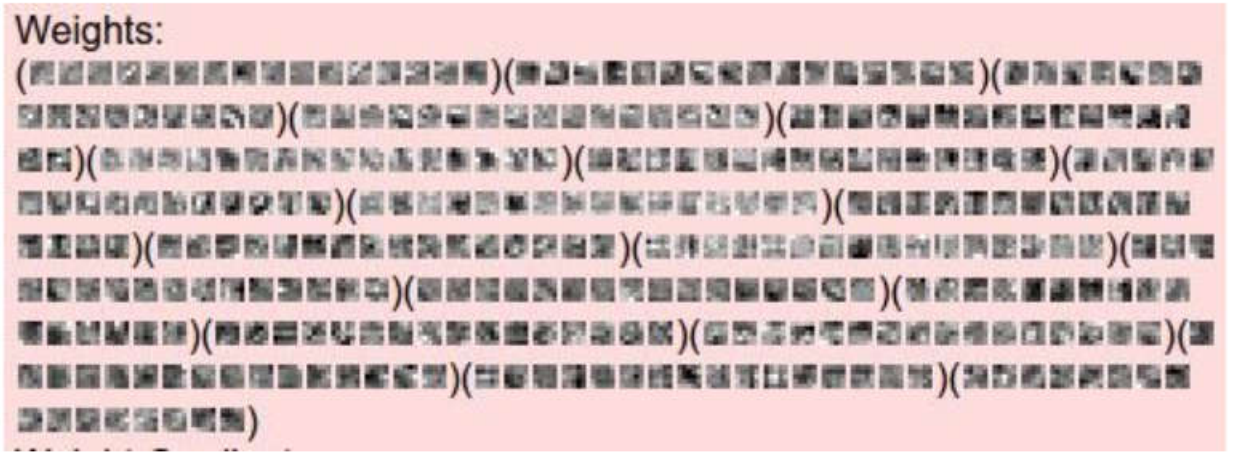
Visualize the Filter/Kernels (Raw Weights)

You can still do it for higher layers, it's just not that interesting

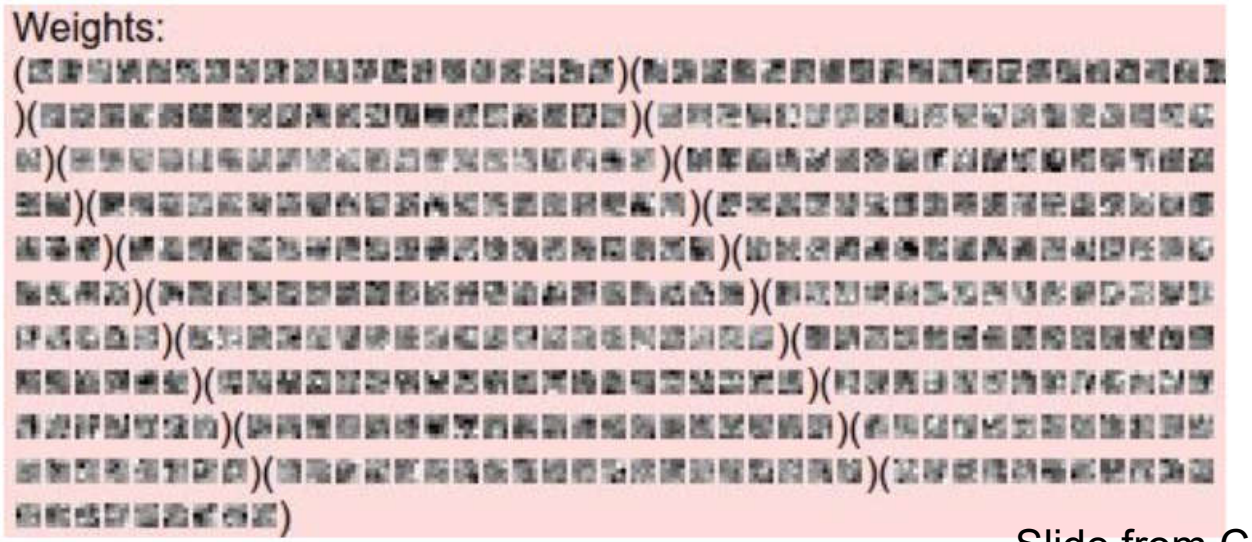
(these are taken from ConvNetJS CIFAR-10 demo)

<u>This doesn't make as much sense and it's not as nice to look at</u>

Layer 1 weights



Layer 2 weights



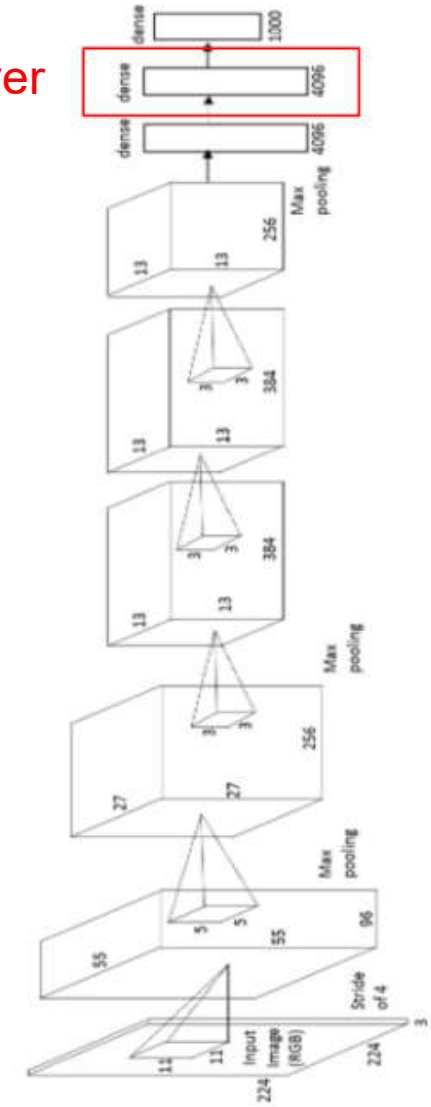
Layer 3 weights

Slide from CS231n

III. Visualize the Representation Space

Visualizing the Representation

fc7 layer



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

Can collect the code for many images

Slide from CS231n

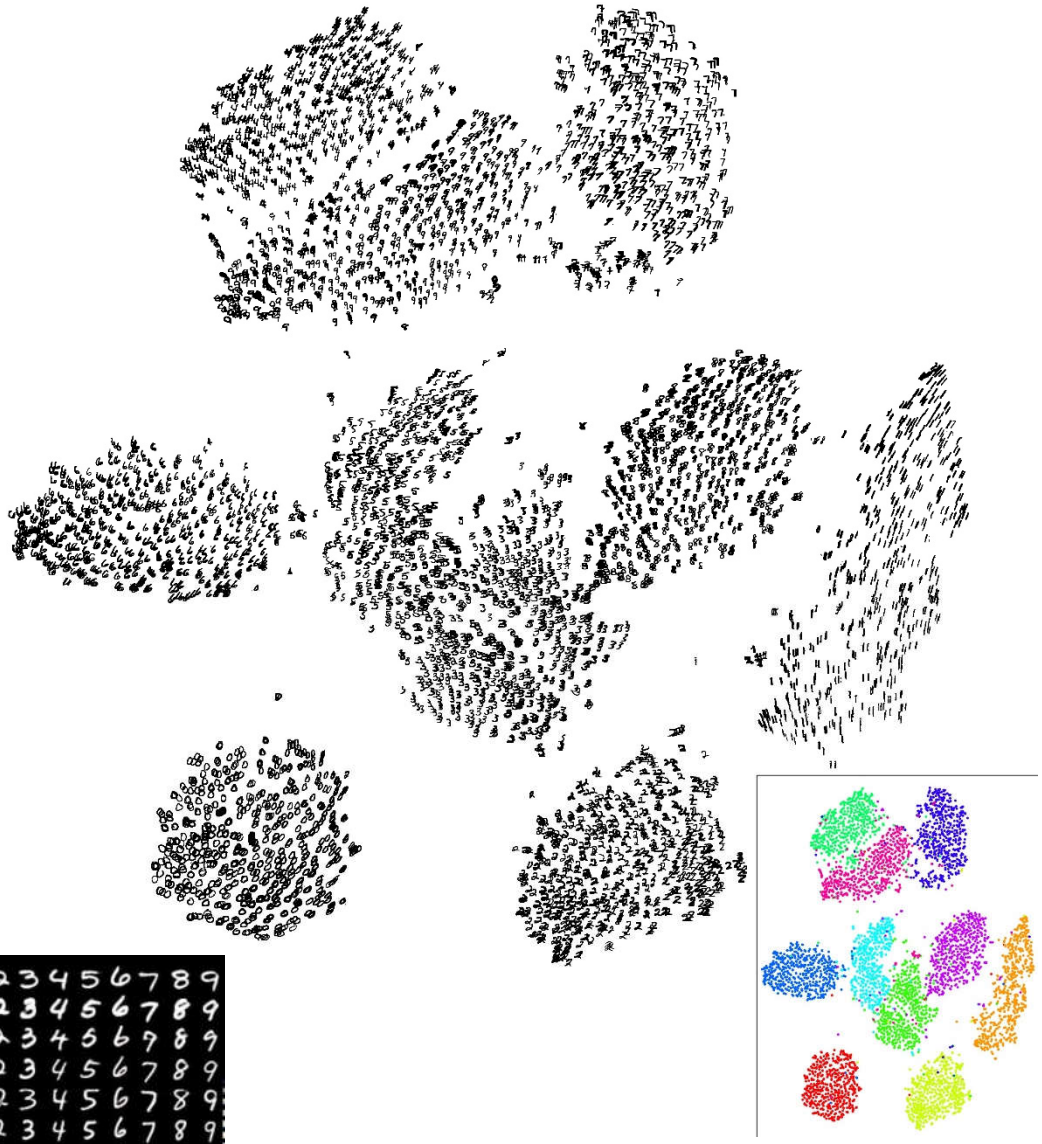
Visualizing the Representation

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

Slide from CS231n

t-SNE Visualization

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

See more:

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

Slide from CS231n

IV. Occlusion Experiments

Occlusion Experiments

[Zeiler & Fergus 2013]

(a) Input image

(d) Classifier, probability of correct class

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

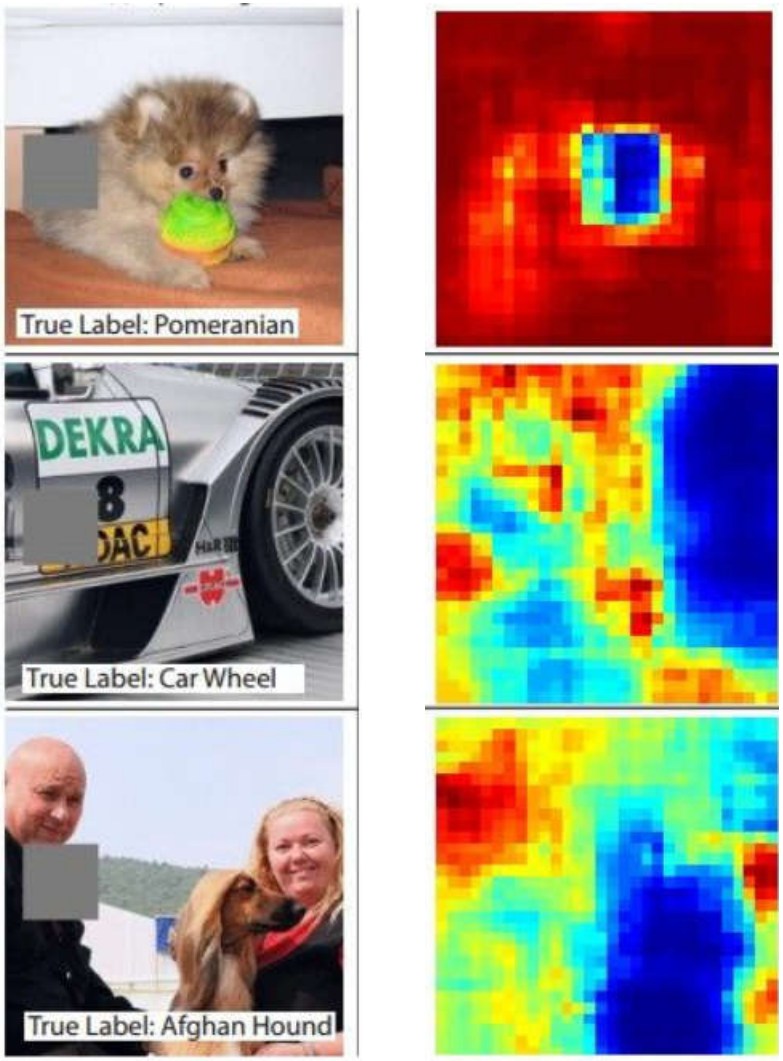
Slide from CS231n

Occlusion Experiments

[Zeiler & Fergus 2013]

(a) Input image

(d) Classifier, probability of correct class



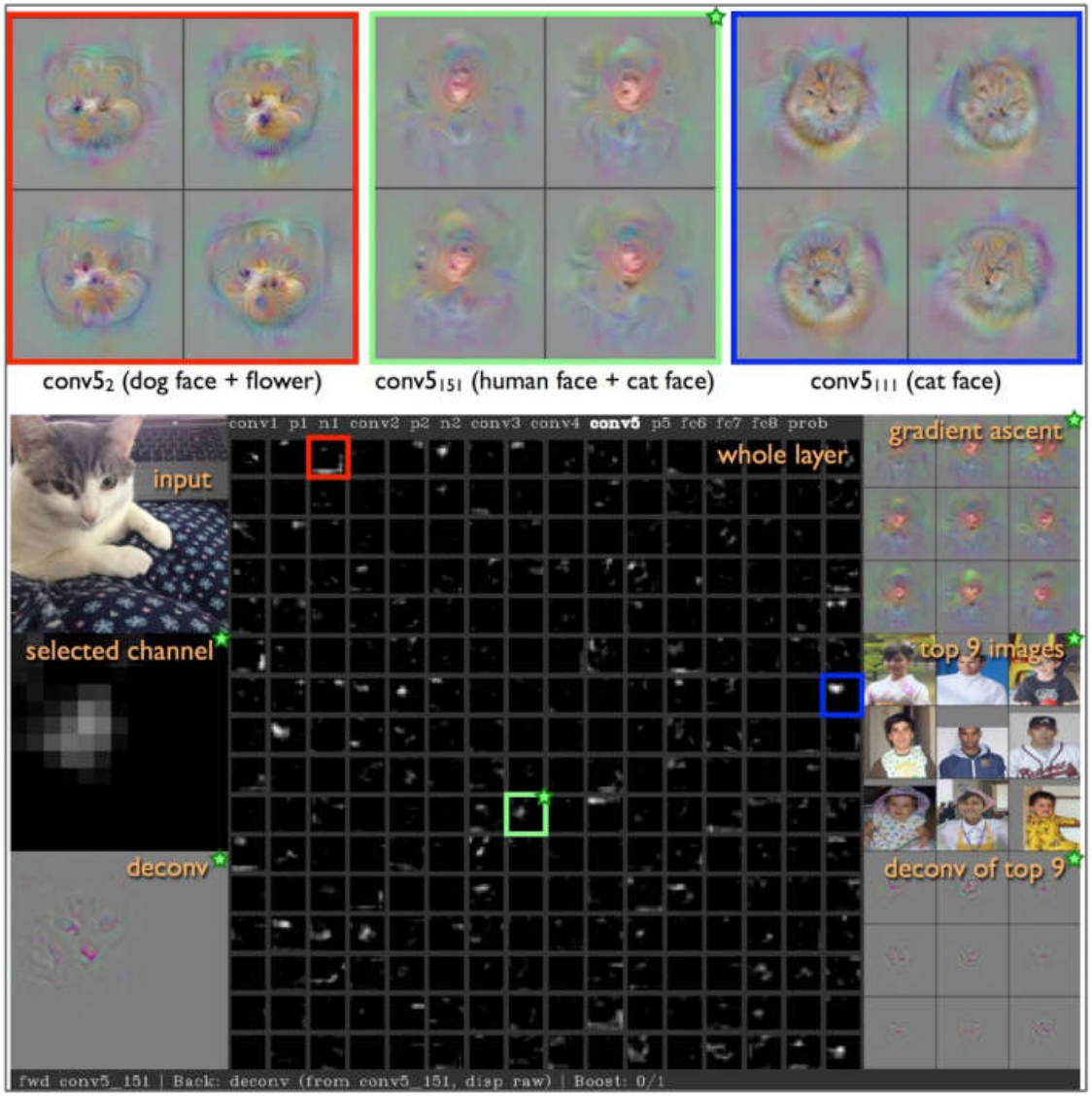
(as a function of the position of the **square of zeros** in the original image)

Slide from CS231n

Visualizing Activations

http://yosinski.com/deepvis

- Deconv-based approach
- Optimization based approach



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

Slide from CS231n

Visualizing Activations

Deep Visualization Toolbox

yosinski.com/deepvis

#deepvis


Jason Yosinski


Jeff Clune


Anh Nguyen


Thomas Fuchs


Hod Lipson

Cornell University

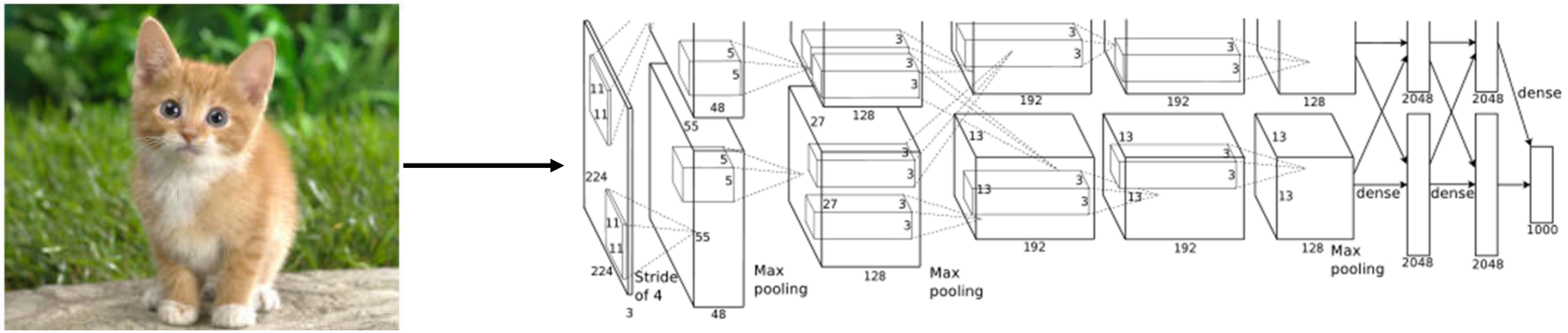
University of Wyoming

Jet Propulsion Laboratory
California Institute of Technology

V. Deconv-Based Visualization

Deconv Approach

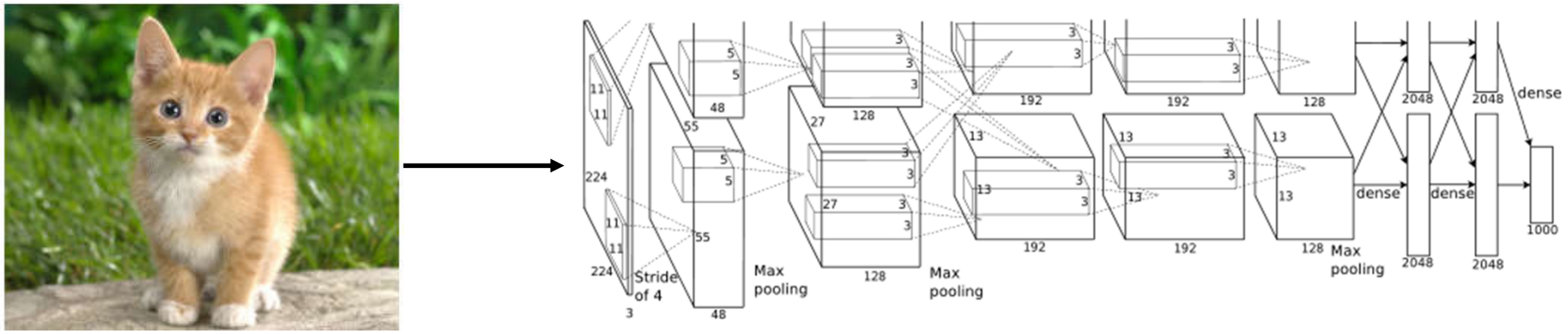
1. Feed image into net

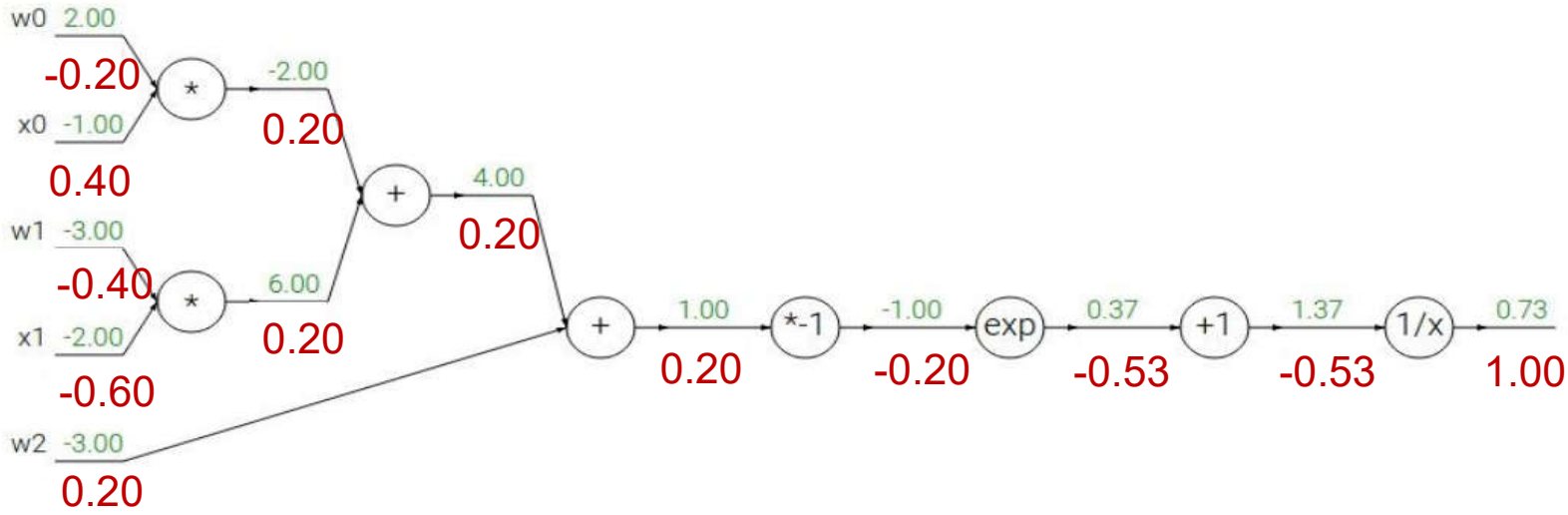


Question: how can we compute the gradient of any arbitrary neuron in the network w.r.t the image?

Deconv Approach

1. Feed image into net

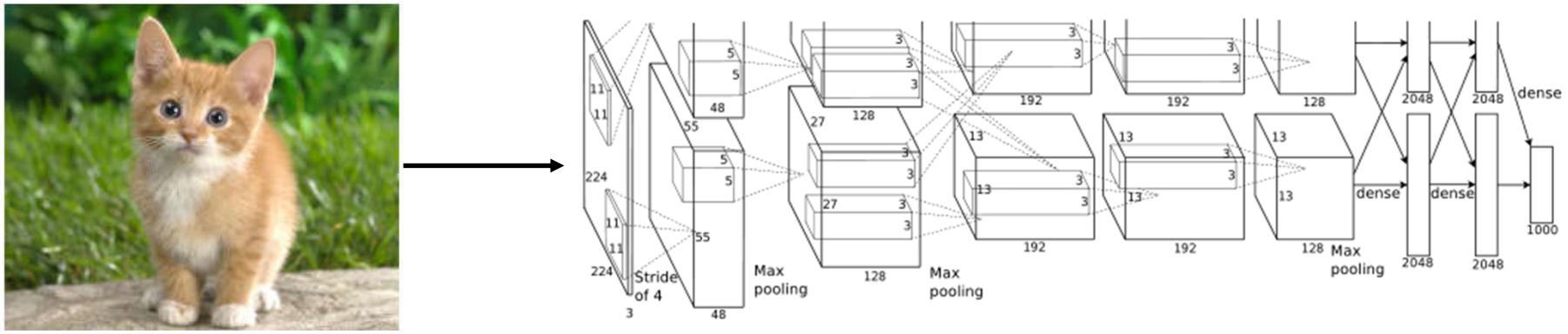




Slide from CS231n

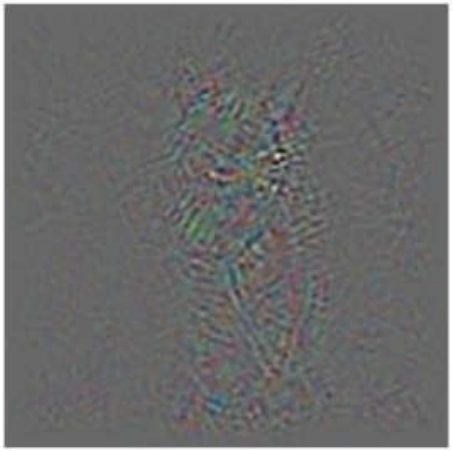
Deconv Approach

1. Feed image into net



2. Pick a layer, set the gradient there to be all zeros except for one 1 for some neuron of interest

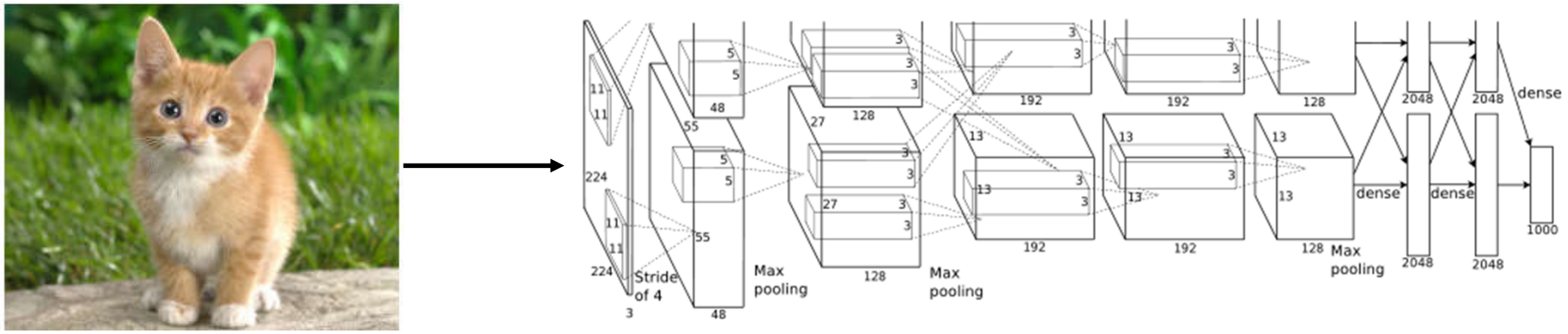
3. Backprop to image:



Slide from CS231n

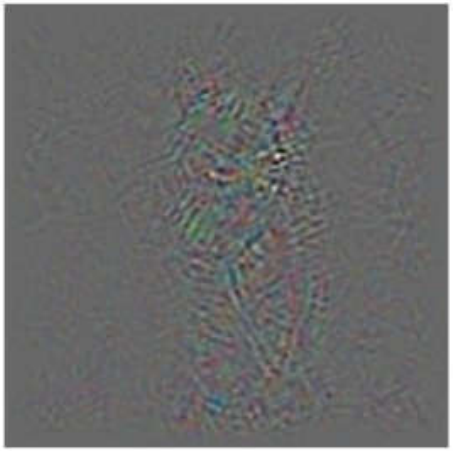
Deconv Approach

1. Feed image into net



2. Pick a layer, set the gradient there to be all zeros except for one 1 for some neuron of interest

3. Backprop to image:

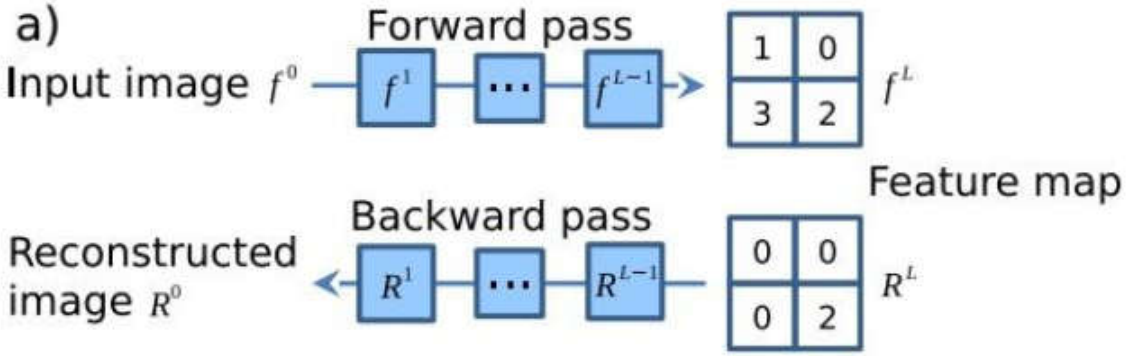


"Guided backpropagation:" instead

Slide from CS231n

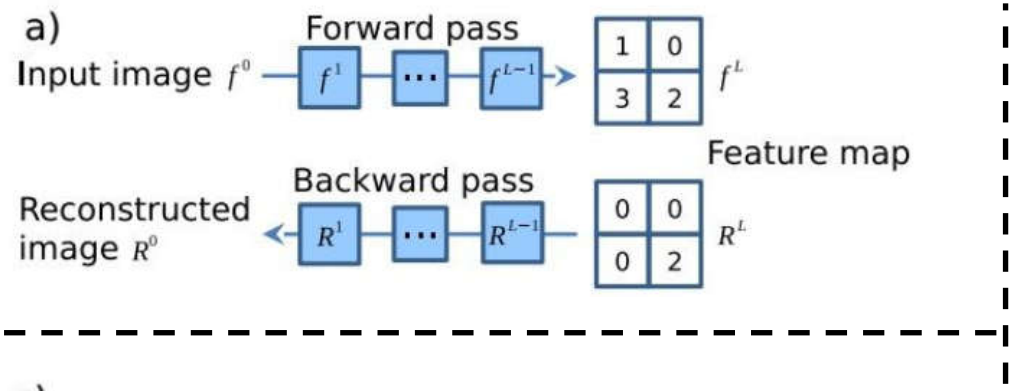
Deconv Approach

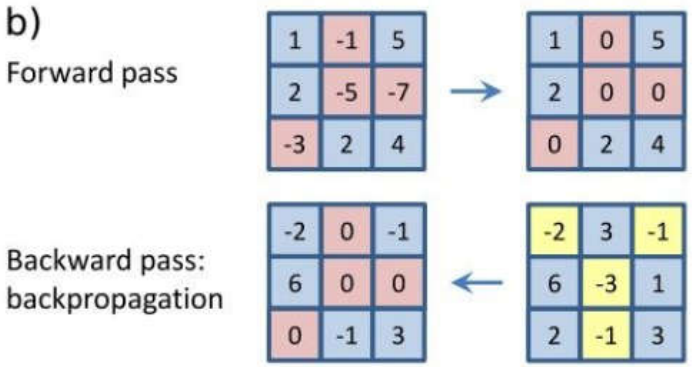
[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]
[Deep Inside Convolutional Networks: Visualizing Image Classification Models and Saliency Maps, Simonyan et al., 2014]
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]



Deconv Approach

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





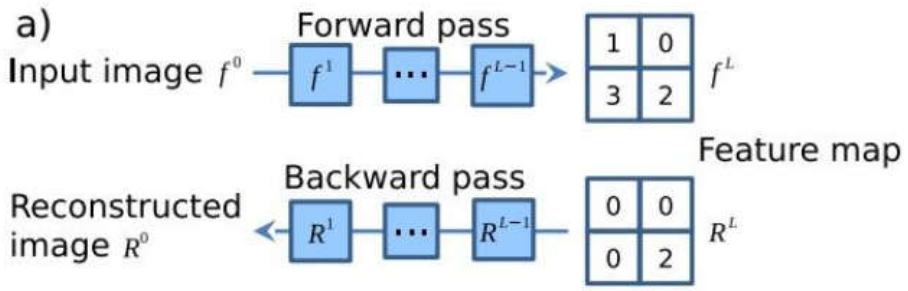
c) activation: $f_i^{l+1} = relu(f_i^l) = \max(f_i^l, 0)$

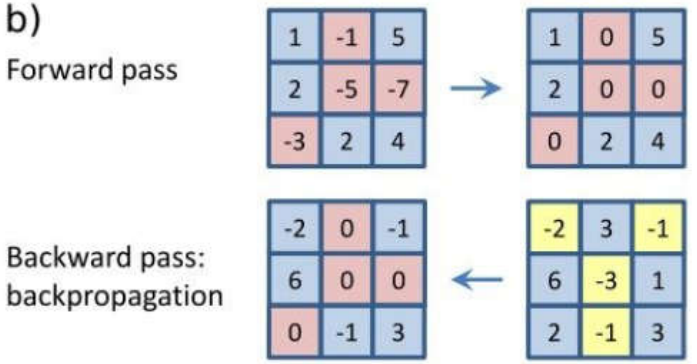
backpropagation: $R_i^l = (f_i^l > 0) \cdot R_i^{l+1}$, where $R_i^{l+1} = \frac{\partial f^{out}}{\partial f_i^{l+1}}$

Backward pass for a ReLu (will be changed in **Guided Backprop**)

Deconv Approach

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



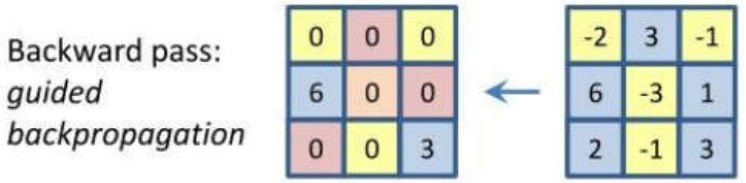


c)

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



Slide from CS231n

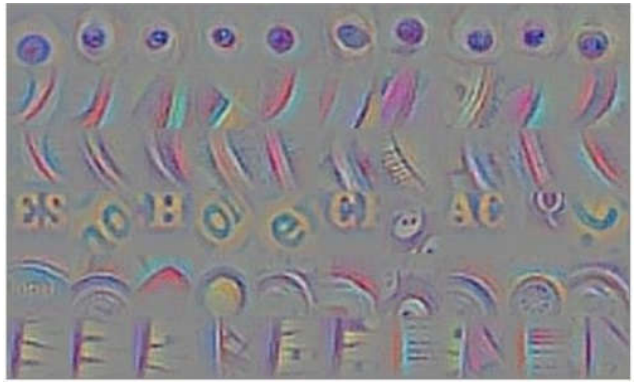
Deconv Approach

Visualization of patterns learned by the layer **conv6** (top) and layer **conv9** (bottom) of the network trained on ImageNet.

<u>Each row corresponds to one filter.</u>

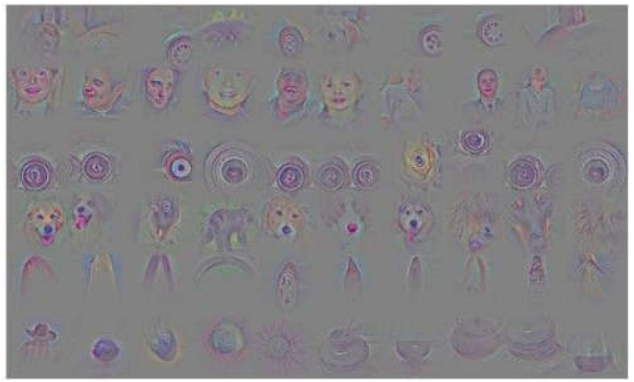
The visualization using "guided backpropagation" is based on the top 10 image patches activating this filter taken from the ImageNet dataset.

Guided backpropagation

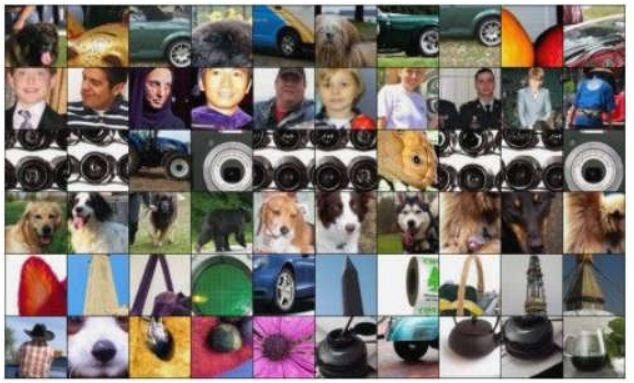


Corresponding image crops

Guided backpropagation



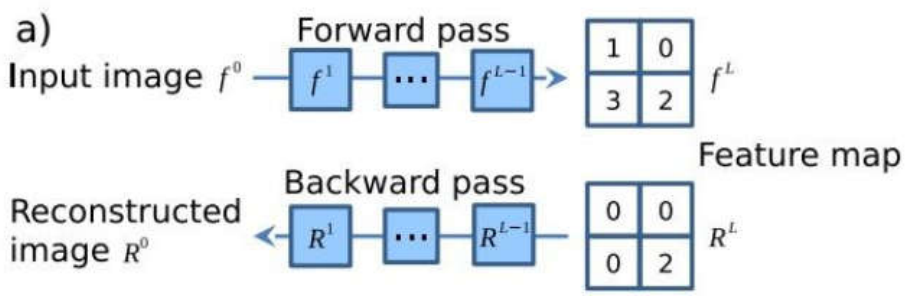
Corresponding image crops

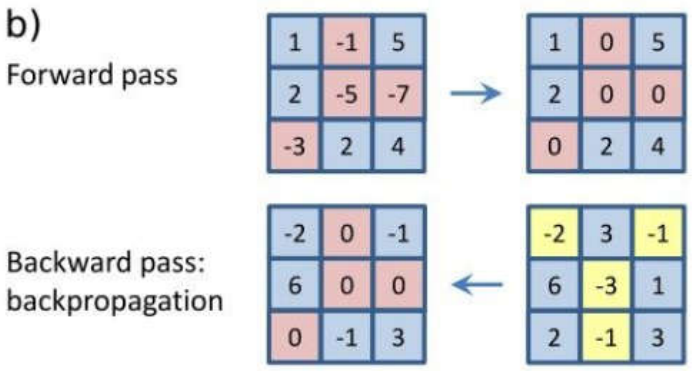


[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]

Deconv Approach

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





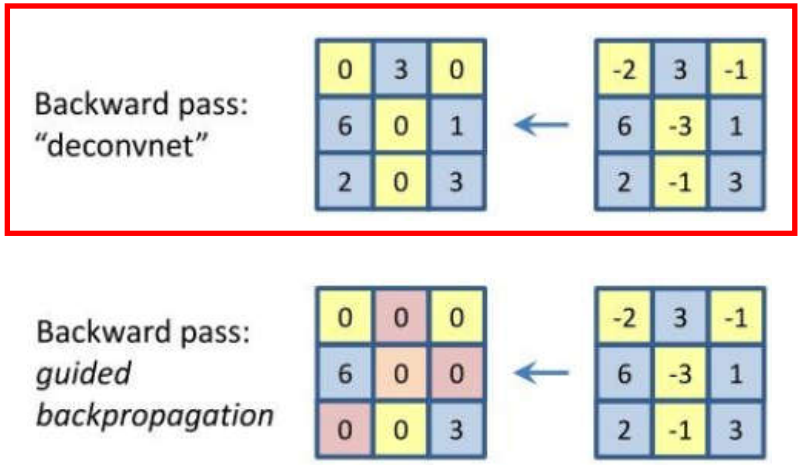
c)

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 $R_i^{l+1} = \frac{\partial f^{out}}{\partial f_i^{l+1}}$

backward 'deconvnet': $R_i^l = (R_i^{l+1} > 0) \cdot R_i^{l+1}$

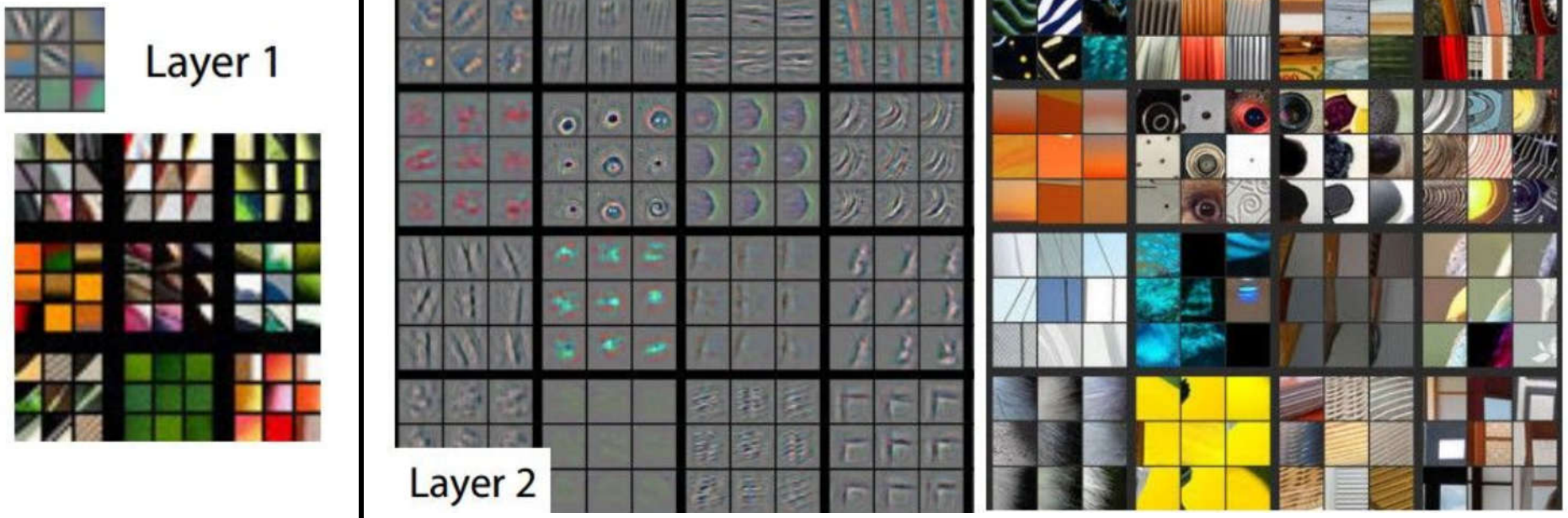
guided backpropagation: $R_i^l = (f_i^l > 0) \cdot (R_i^{l+1} > 0) \cdot R_i^{l+1}$



Slide from CS231n

Visualizing Arbitrary Neurons Along the Way to the Top…

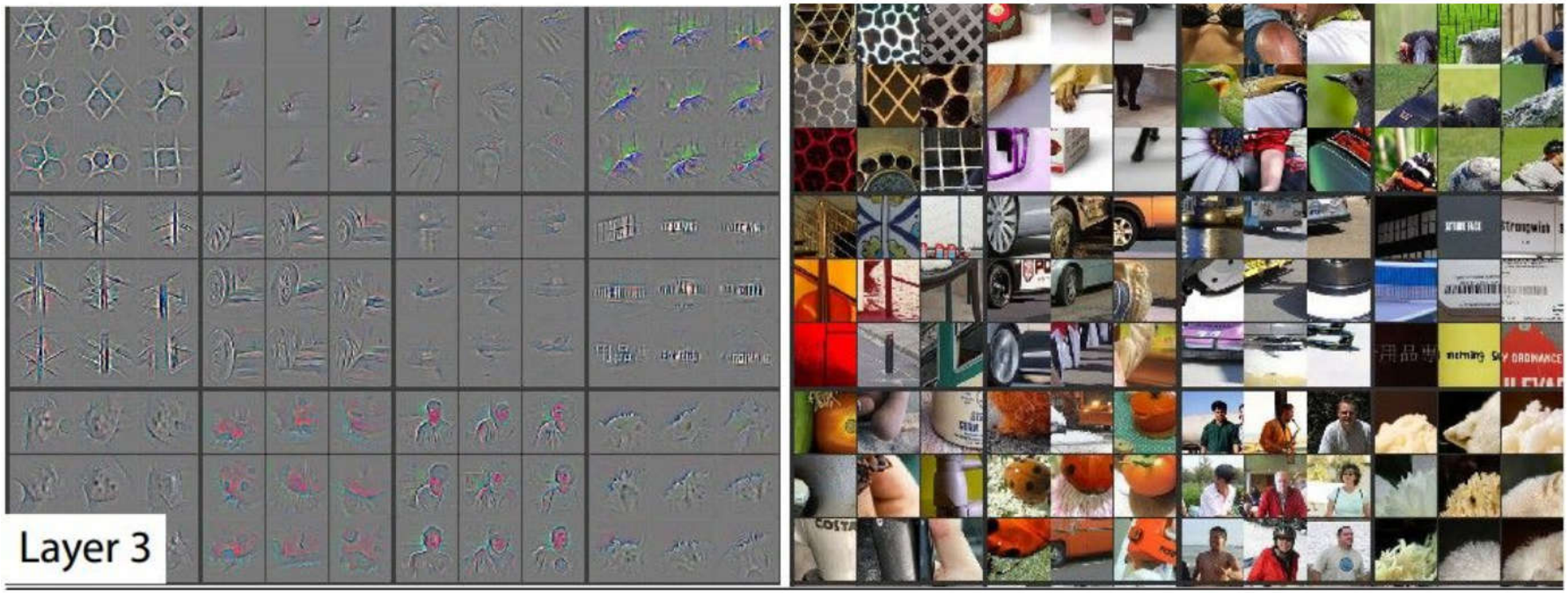
Visualizing and Understanding Convolutional Networks, Zeiler & Fergus, 2013



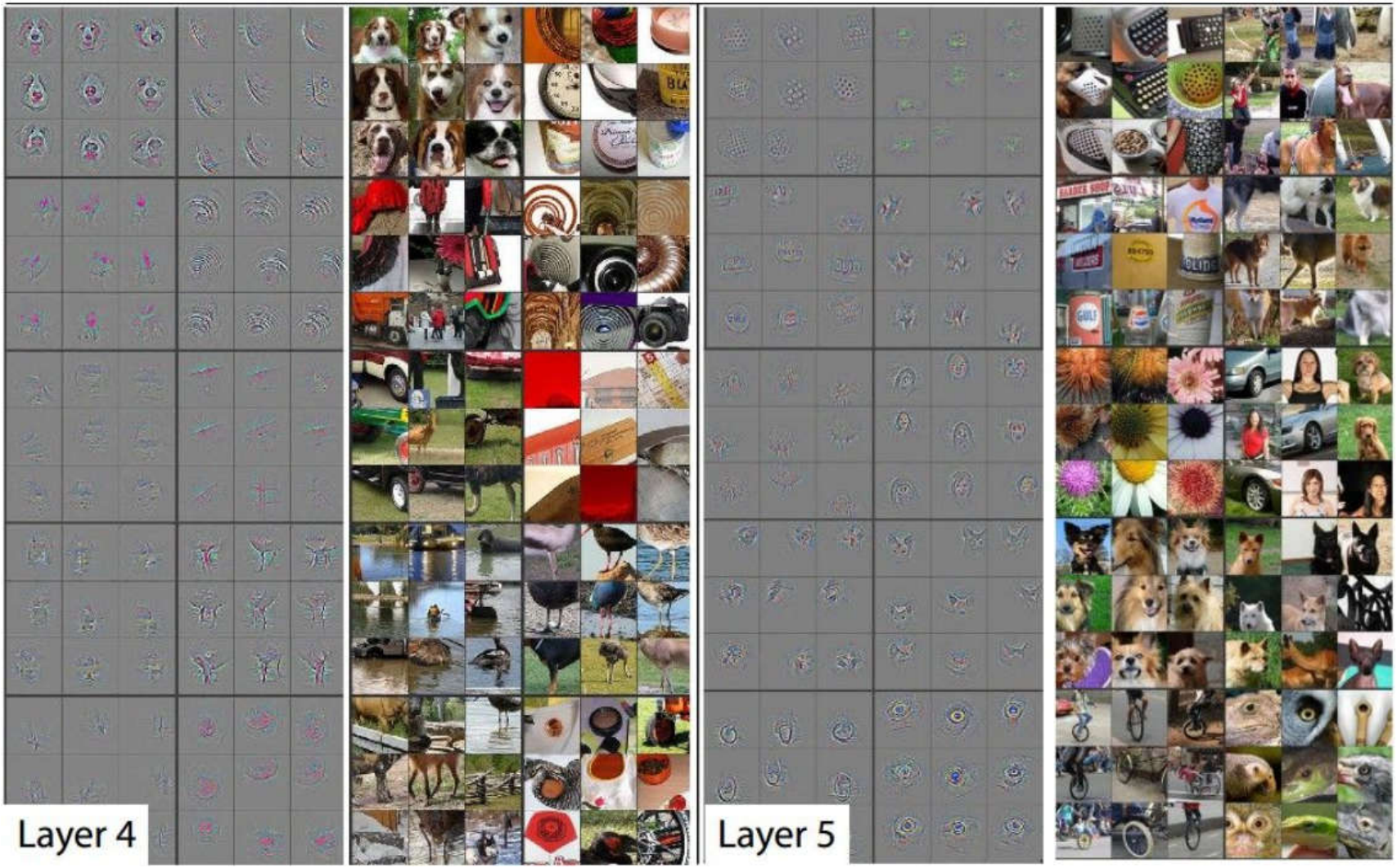
- It produces much **cleaner pictures**, because we only through the positive influences through the backward chain

Slide from CS231n

Visualizing Arbitrary Neurons Along the Way to the Top…



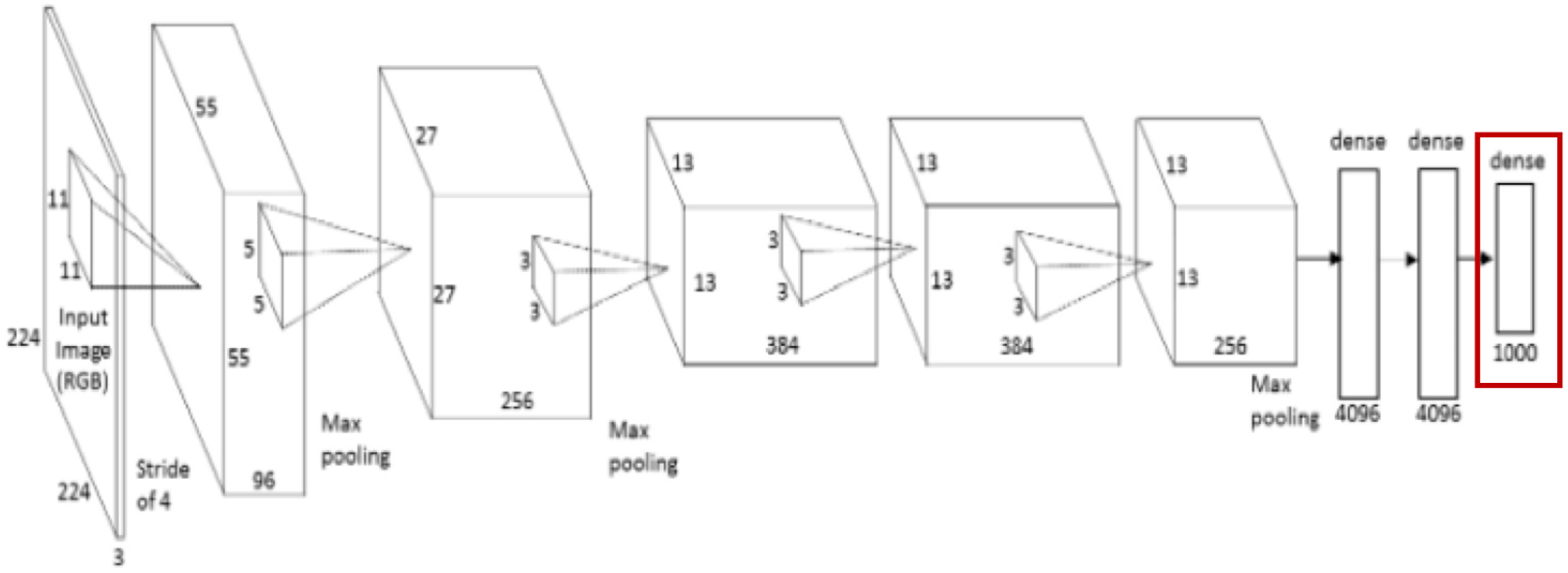
Visualizing Arbitrary Neurons Along the Way to the Top…



- Optimization based visualization which makes much more sense and are much more intuitive in terms of what's going on

VI. Optimization-Based Visualization

Optimization to Image



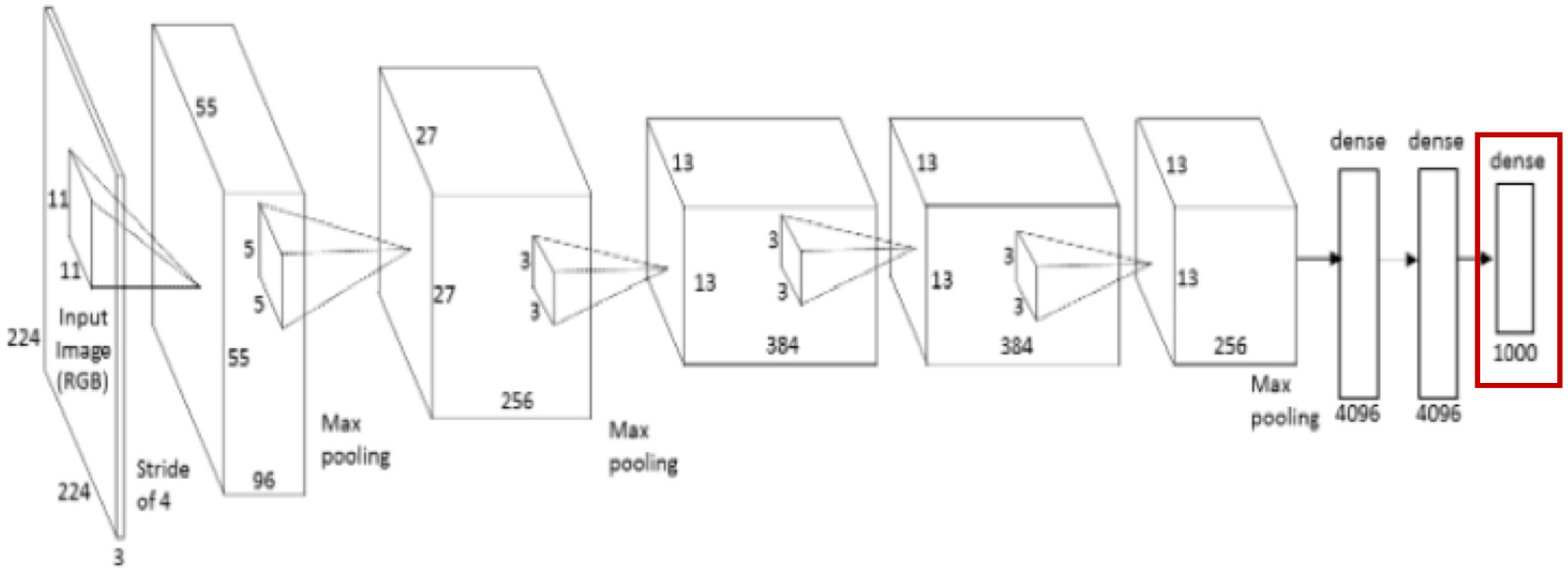
Question: can we find an image that maximizes some class score?

Slide from CS231n

Optimization to Image

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

Score for class c (before Softmax)

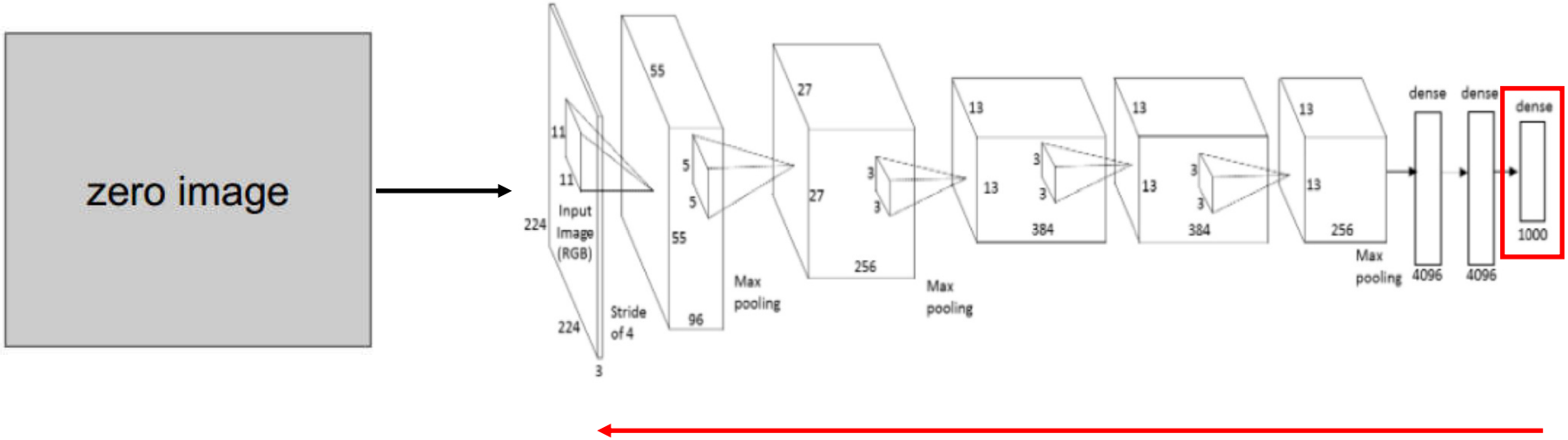


Question: can we find an image that maximizes some class score?

Slide from CS231n

Optimization to Image

1. Feed in zeros



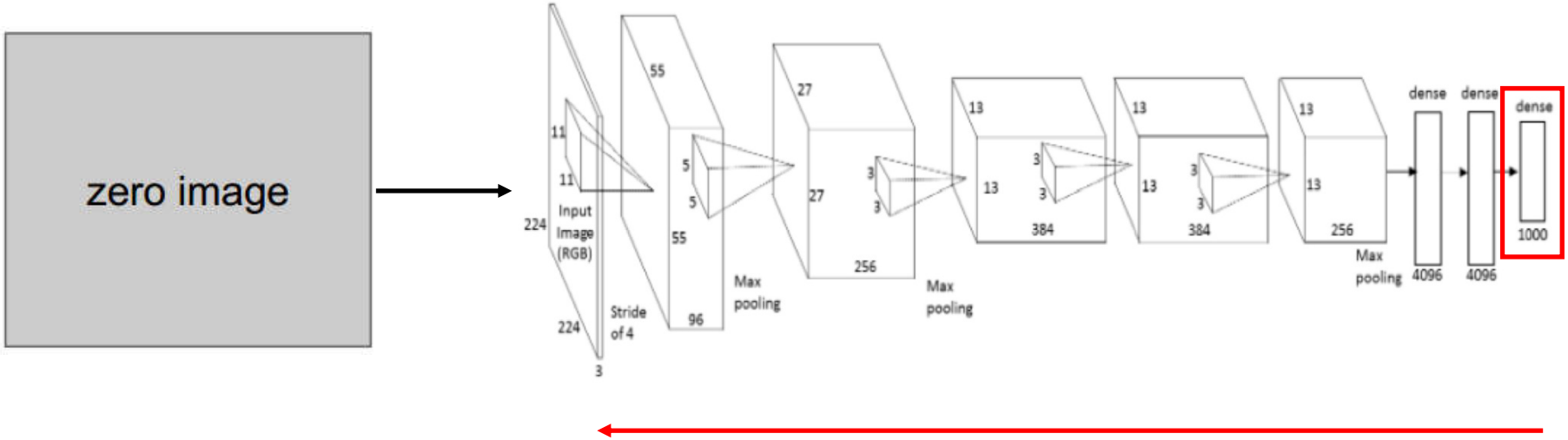
2. Set the gradient of the scores vector to be [0,0,…1,…,0], then backprop to image

Question: Set zero image as initialization… is it can be a problem?

Slide from CS231n

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.

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

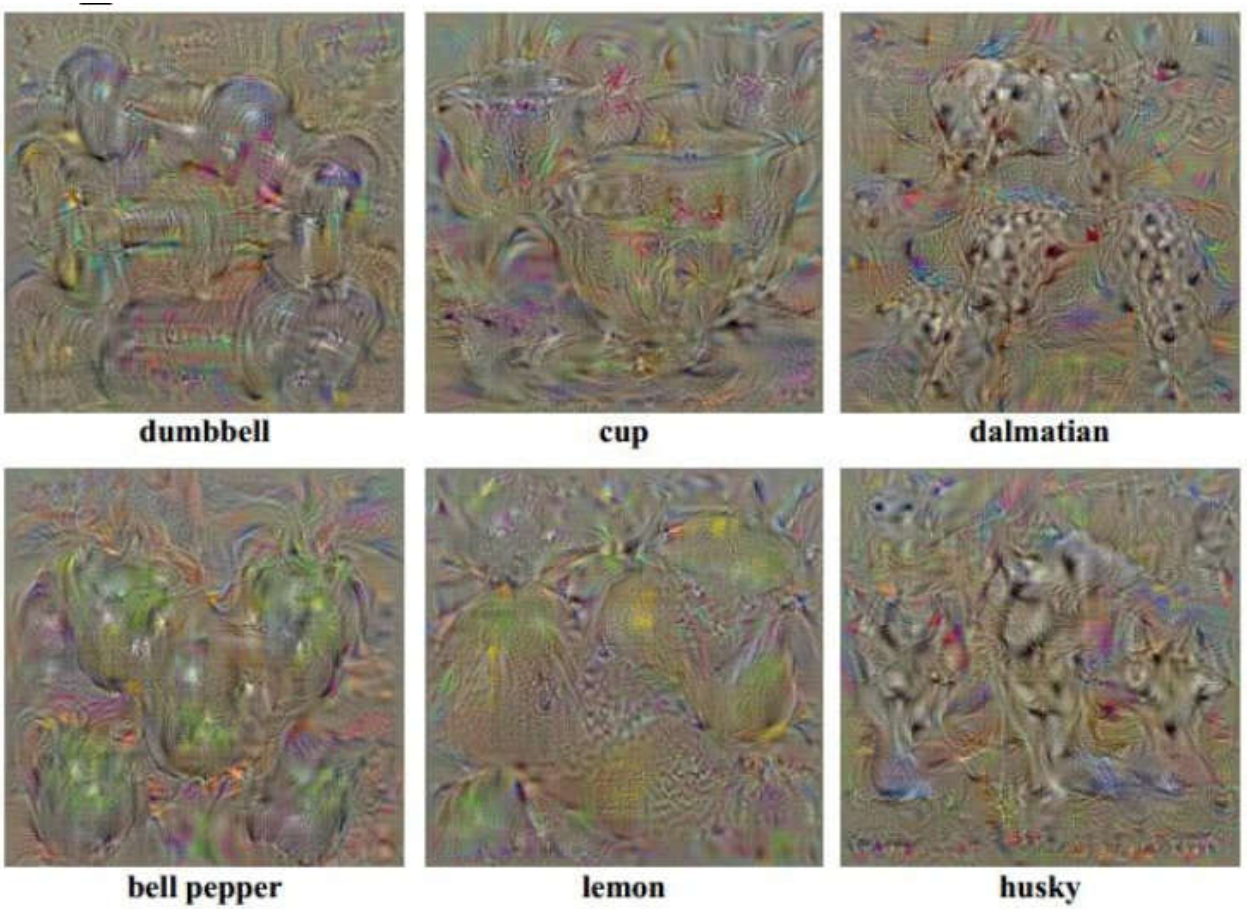
Score for class c (before Softmax)

Slide from CS231n

Optimization to Image

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

1. Find images that maximize some class score:

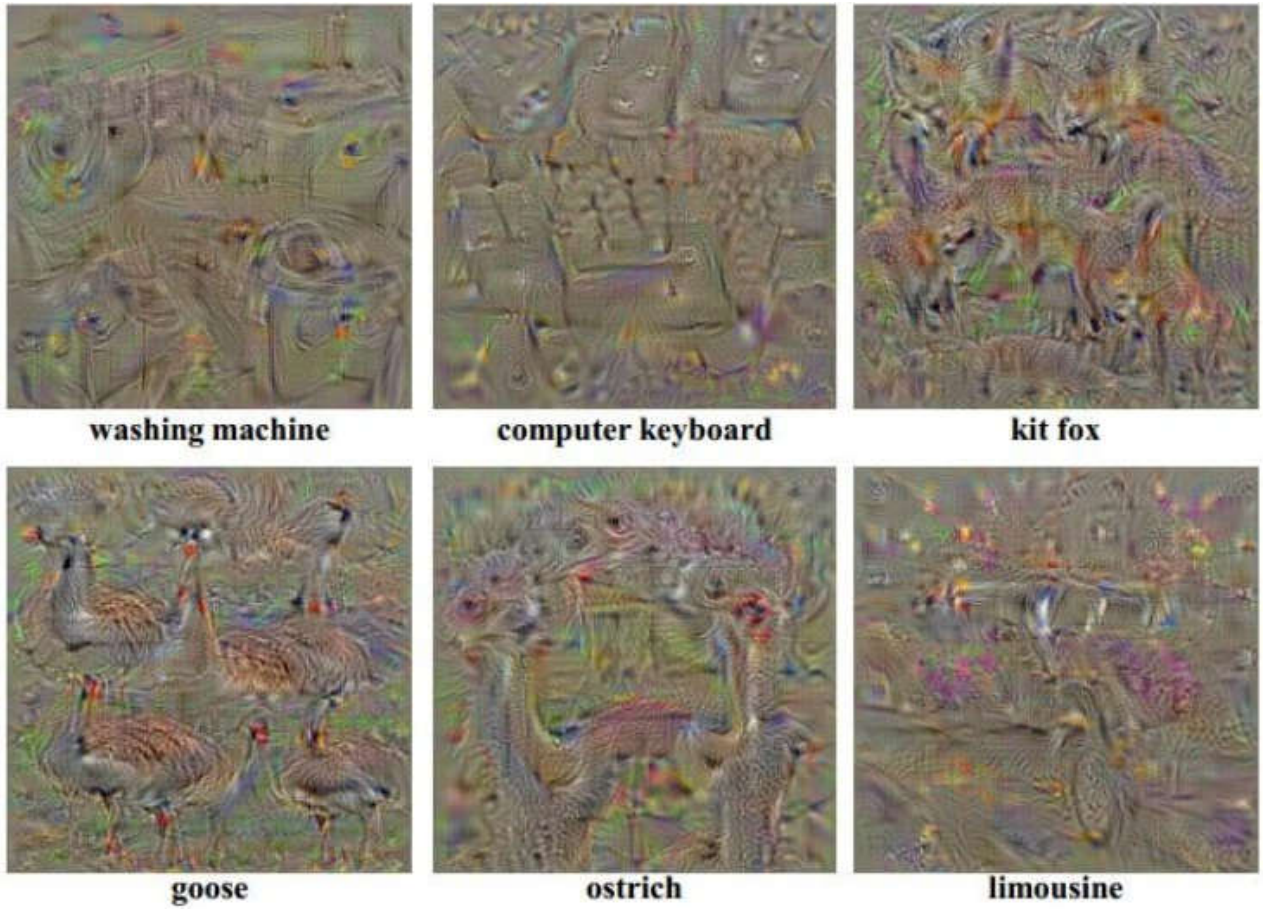


Slide from CS231n

Optimization to Image

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

1. Find images that maximize some class score:



Slide from CS231n

Optimization to Image

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

2. Visualize the Data gradient:



M = ?

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

Hit map

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

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

Optimization to Image

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

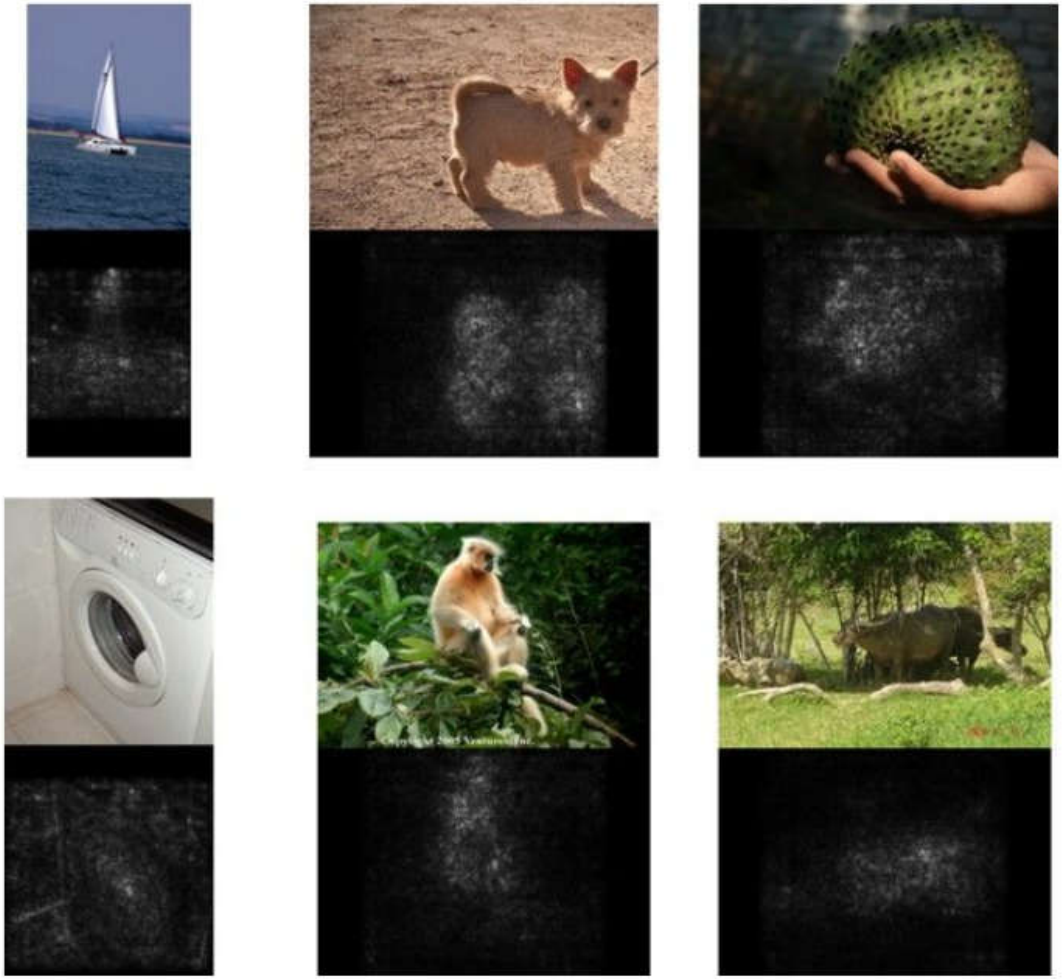
2. Visualize the Data gradient:

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

Hit map

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

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



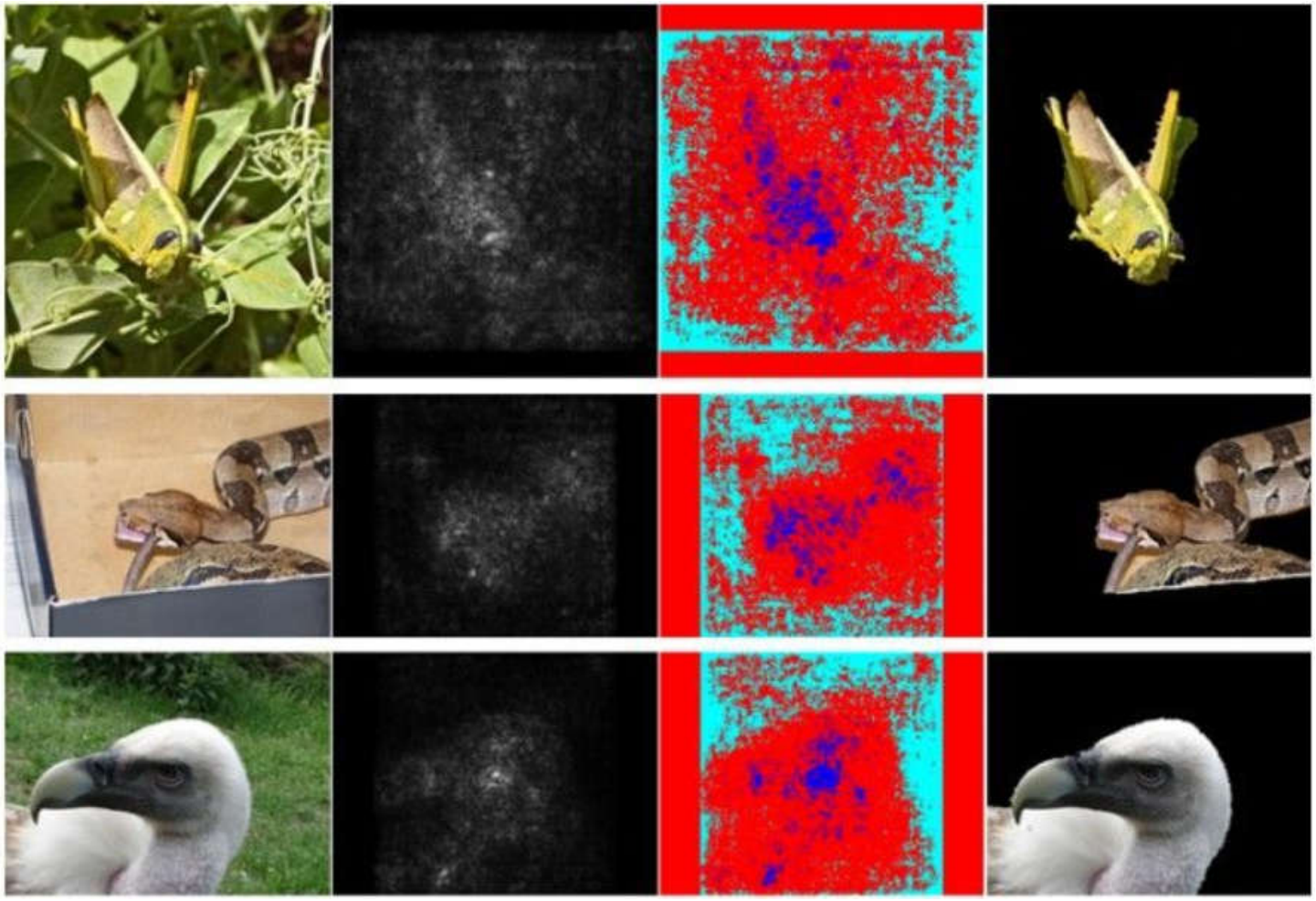
Slide from CS231n

Optimization to Image

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

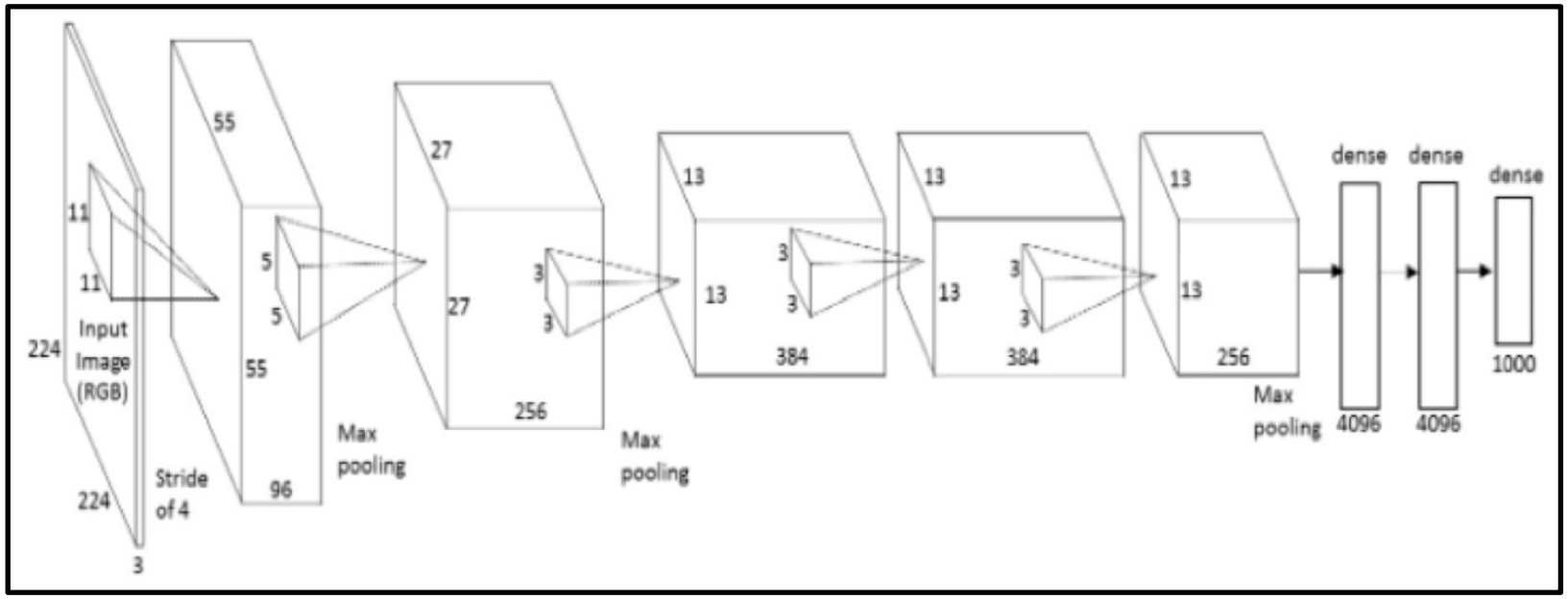
- Use **grabcut**

for segmentation



Slide from CS231n

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”

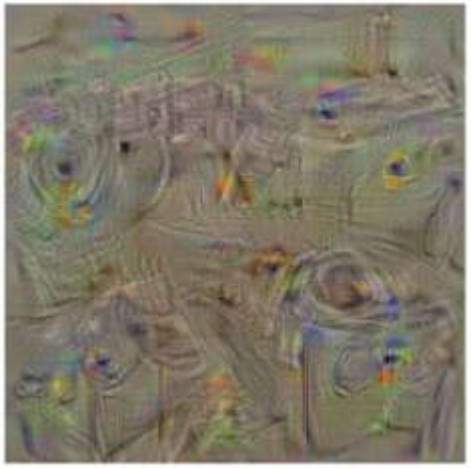
Slide from CS231n

We Can In Fact Do This for Arbitrary Neurons Along the ConvNet

[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:

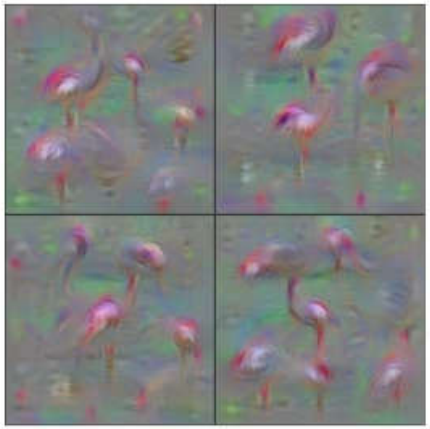
Repeat:

- 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)

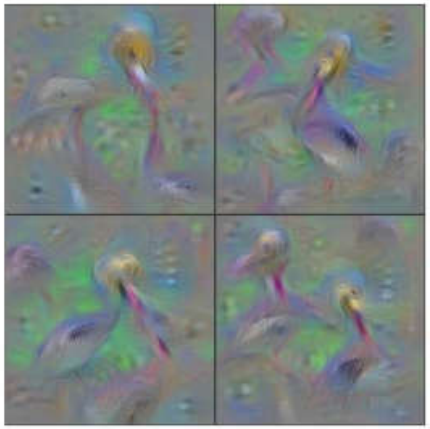
We Can In Fact Do This for Arbitrary Neurons Along the ConvNet

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

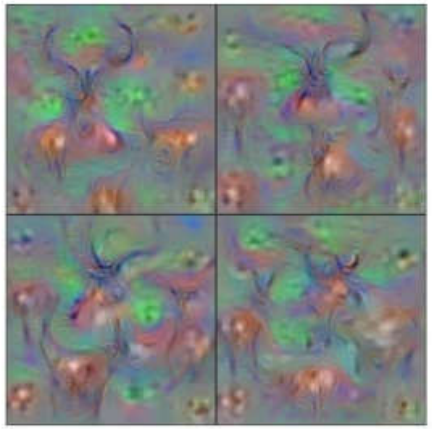
http://yosinski.com/deepvis



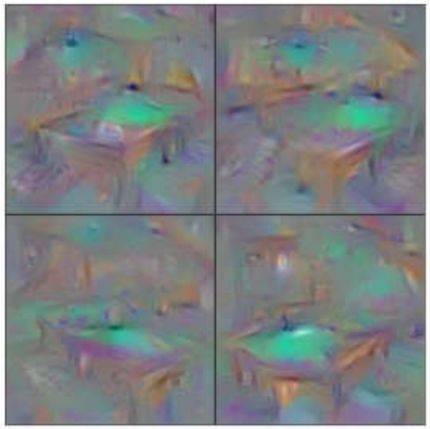
Flamingo



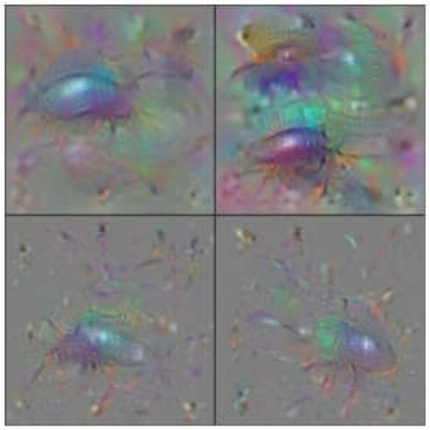
Pelican



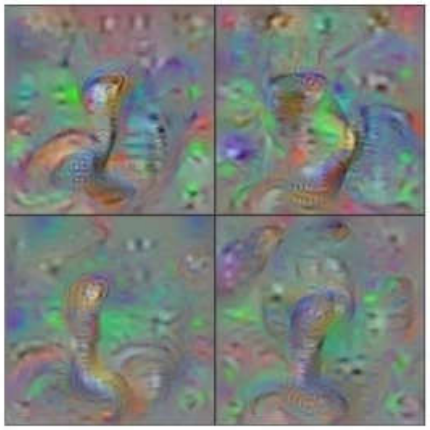
Hartebeest



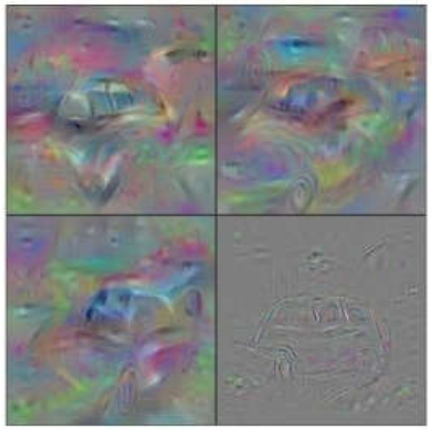
Billiard Table



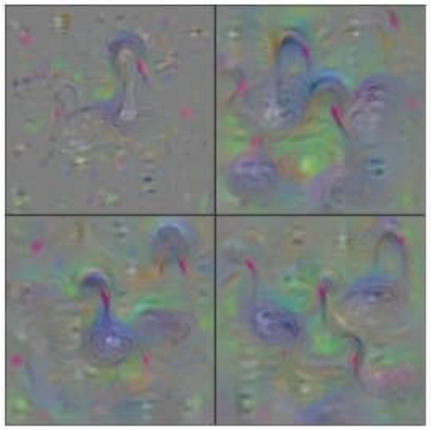
Ground Beetle



Indian Cobra



Station Wagon

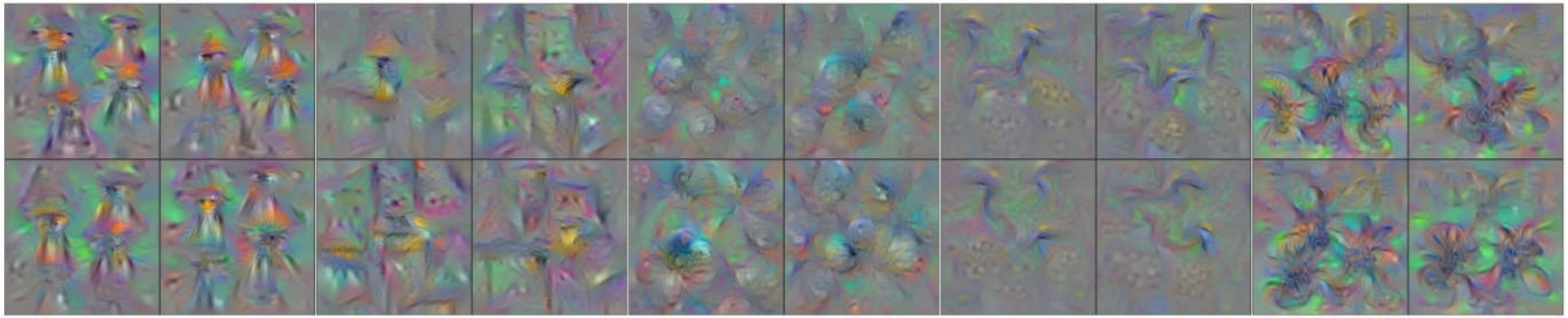


Black Swan

Slide from CS231n

We Can In Fact Do This for Arbitrary Neurons Along the ConvNet

Layer 7

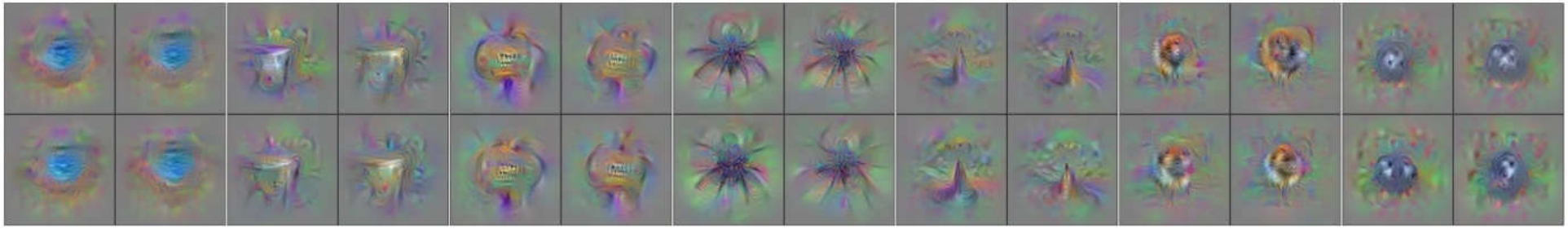


Slide from CS231n

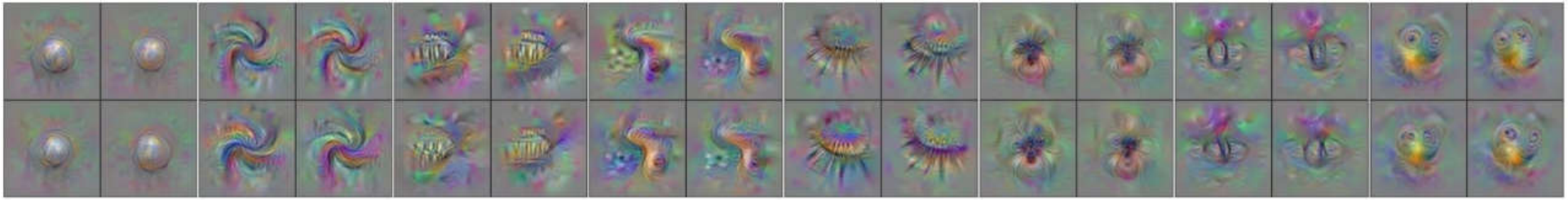
We Can In Fact Do This for Arbitrary Neurons Along the ConvNet

Layer 6

Layer 5



Layer 4

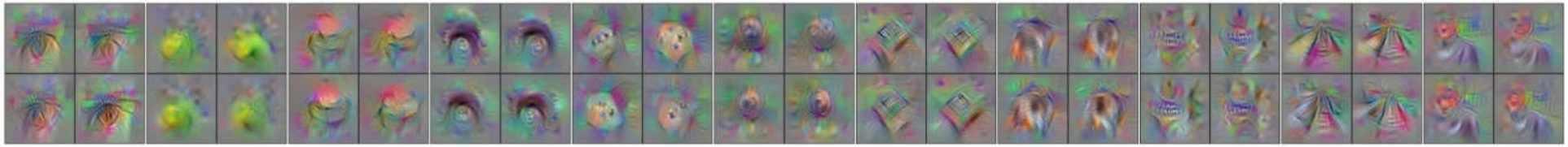


Slide from CS231n

We Can In Fact Do This for Arbitrary Neurons Along the ConvNet

Layer 4

Layer 3



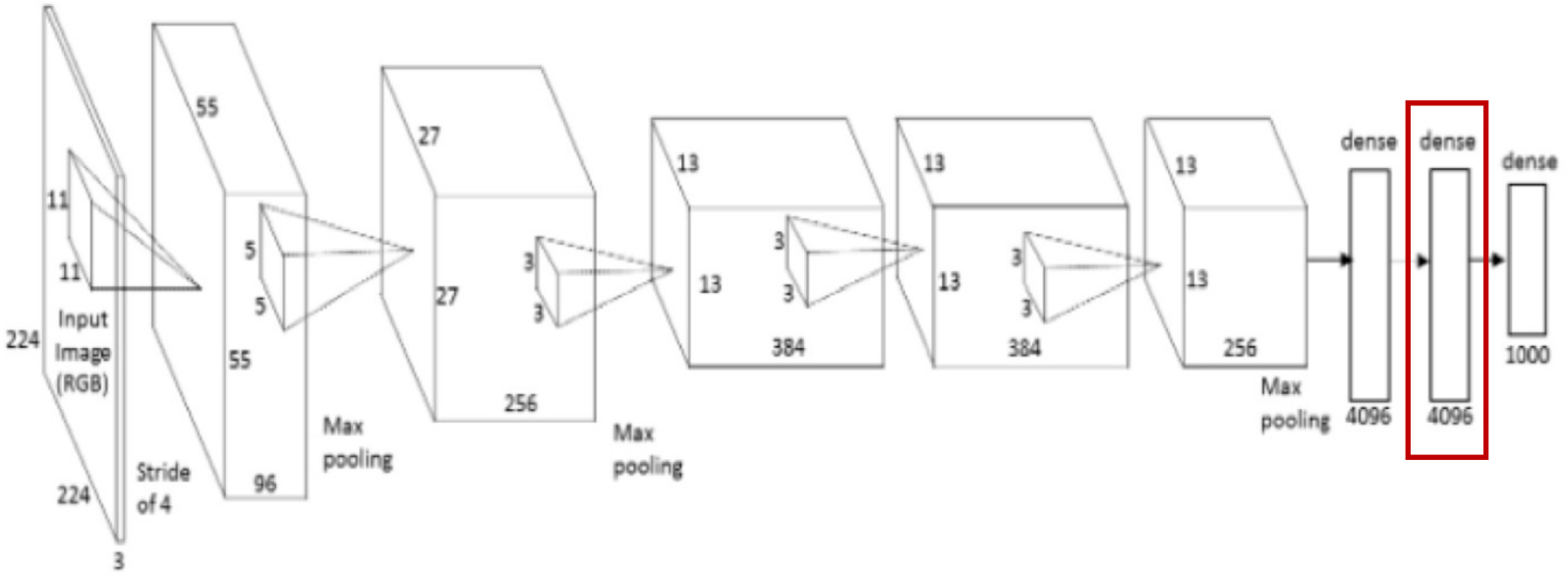
Layer 2

Layer 1

Reconstruction from CNN Code

Reconstruction from CNN Code

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



- Some companies might try to only store the features not the raw images

Slide from CS231n

Reconstruction from CNN Code

Find an image such that:

- Its code is similar to a given code
- It "looks natural" (image prior regularization)

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbf{R}^{H \times W \times C}}{\arg\min} \ \ell\left(\Phi(\mathbf{x}), \Phi_0\right) + \lambda R(\mathbf{x})$$

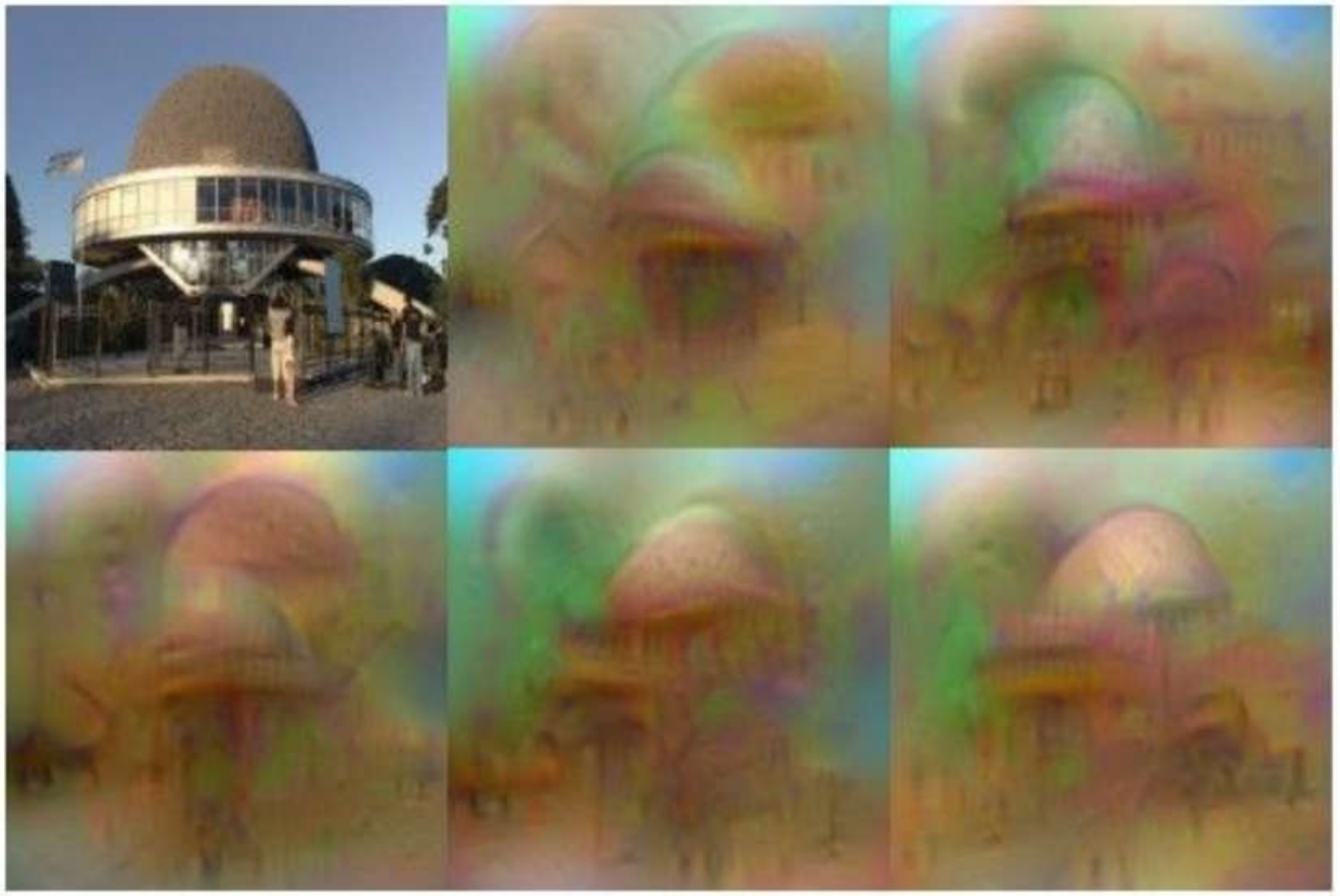
$$\ell\left(\Phi(\mathbf{x}), \Phi_0\right) = \left\|\Phi(\mathbf{x}) - \Phi_0\right\|^2$$

Slide from CS231n

Reconstruction from CNN Code

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

Original image



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

Slide from CS231n

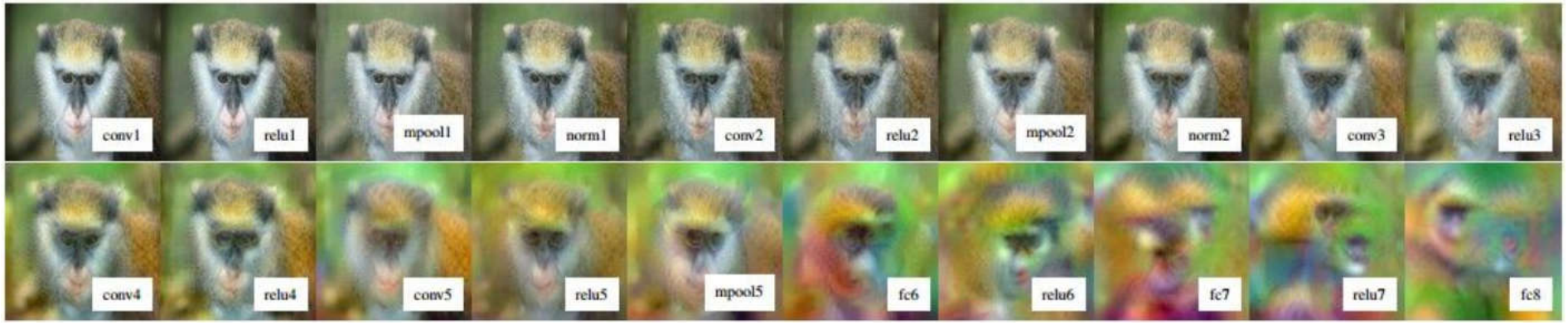
Reconstruction from CNN Code

Reconstructions from the representation after last pooling layer

(immediately before the first Fully Connected layer)

Reconstruction from CNN Code

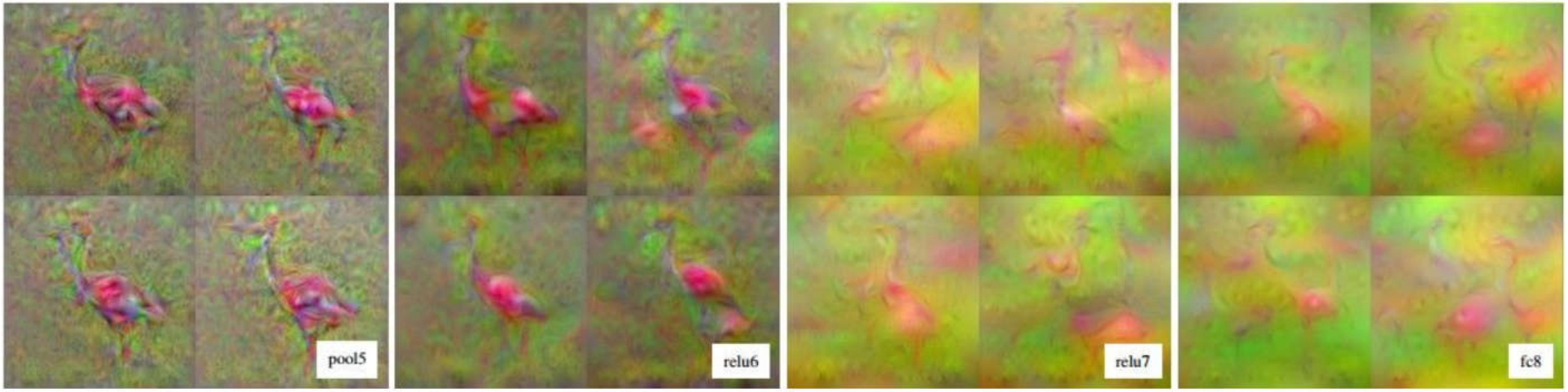
Reconstructions from intermediate layers



Slide from CS231n

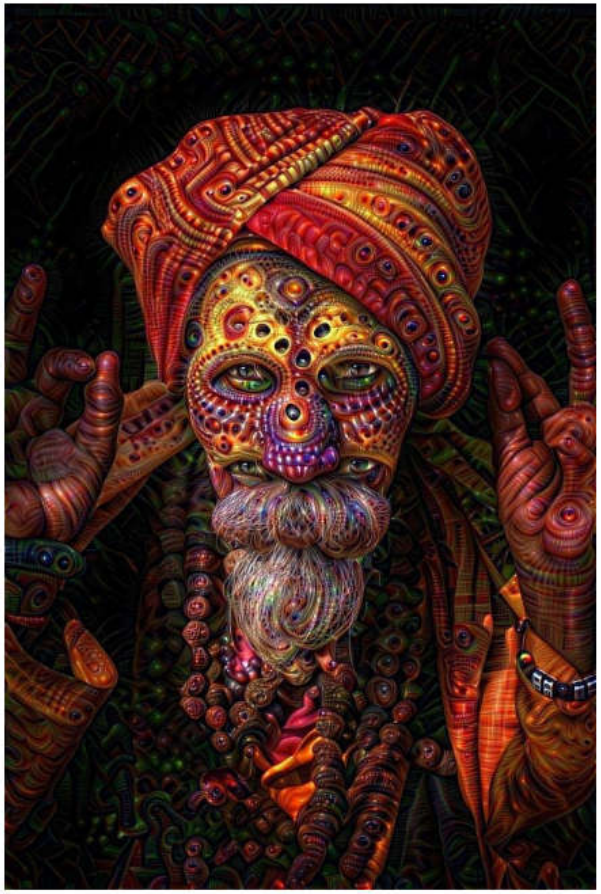
Reconstruction from CNN Code

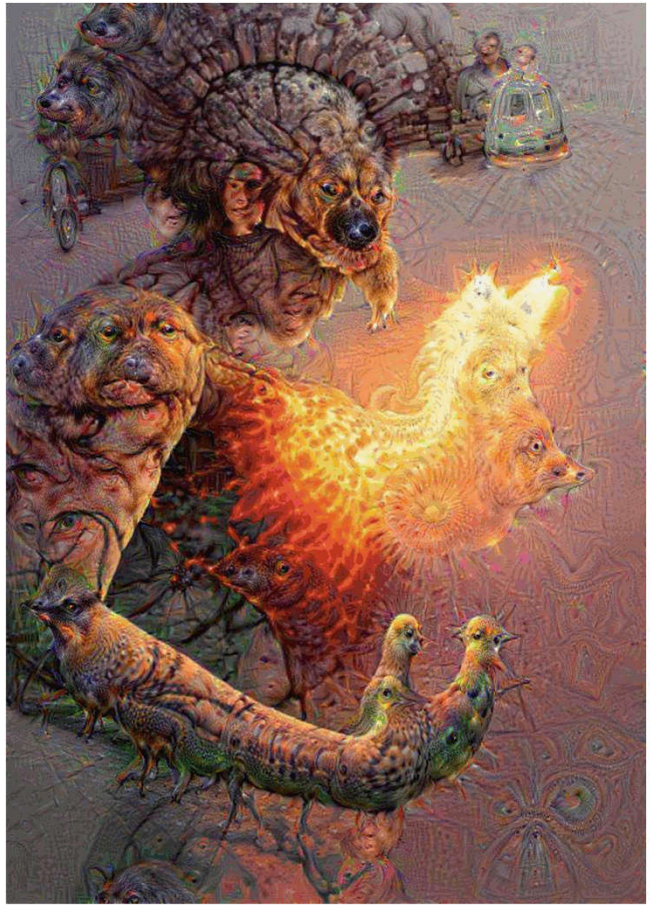
Multiple reconstructions. Images in quadrants all “look” the same to the CNN (same code)



DeepDream

DeepDream





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

Slide from CS231n

DeepDream

```
def objective_L2(dst):
    dst.diff[:] = dst.data

def make_step(net, step_size=1.5, end='inception_4c/output',
              jitter=32, clip=True, objective=objective_L2):
    '''Basic gradient ascent step.'''

    src = net.blobs['data'] # input image is stored in Net's 'data' blob
    dst = net.blobs[end]

    ox, oy = np.random.randint(-jitter, jitter+1, 2)
    src.data[0] = np.roll(np.roll(src.data[0], ox, -1), oy, -2) # apply jitter shift

    net.forward(end=end)
    objective(dst)  # specify the optimization objective
    net.backward(start=end)
    g = src.diff[0]
    # apply normalized ascent step to the input image
    src.data[:] += step_size/np.abs(g).mean() * g

    src.data[0] = np.roll(np.roll(src.data[0], -ox, -1), -oy, -2) # unshift image

    if clip:
        bias = net.transformer.mean['data']
        src.data[:] = np.clip(src.data, -bias, 255-bias)
```

DeepDream: set dx = x :)

Jitter regularizer

"image update"

Slide from CS231n

DeepDream

Inception_4c/output

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

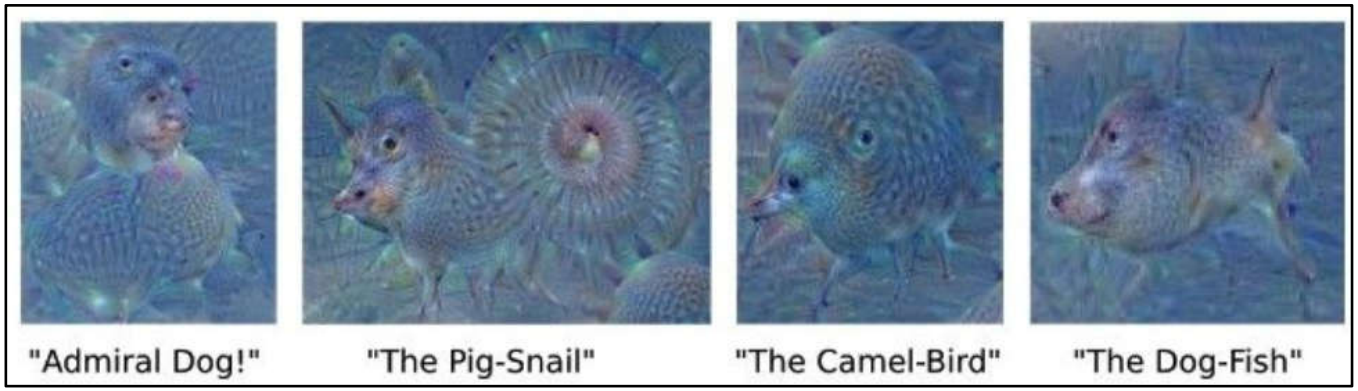
This creates a feedback loop: e.g. any slightly detected dog face will be made more and more dog like over time

Slide from CS231n

DeepDream

Inception_4c/output

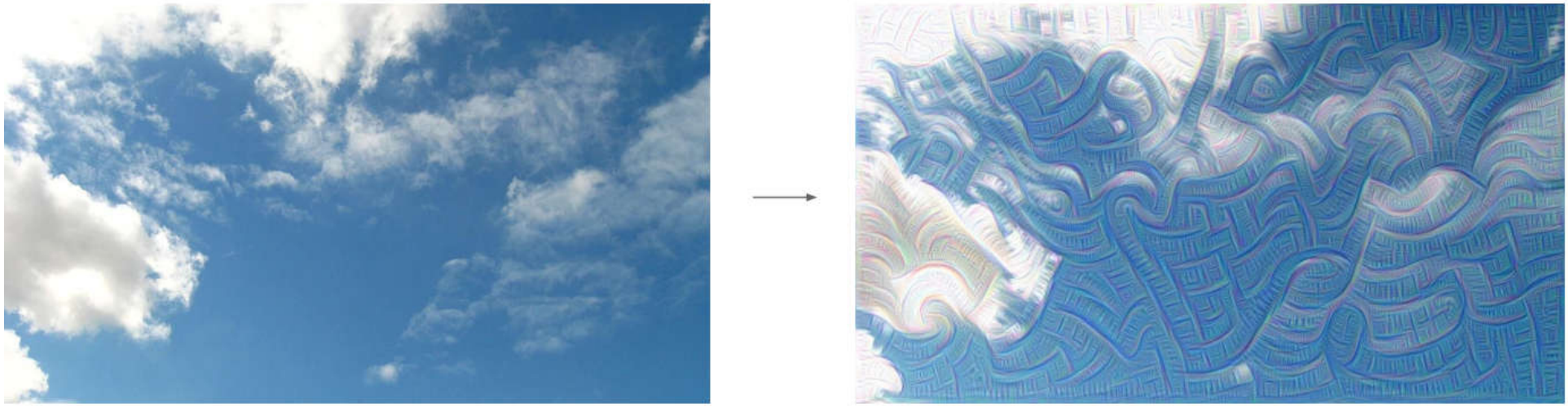
DeepDream modifies the image in a way that “boosts” all activations, at any layer



Slide from CS231n

DeepDream

Inception_3b/5x5_output



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

Slide from CS231n

Bonus Videos

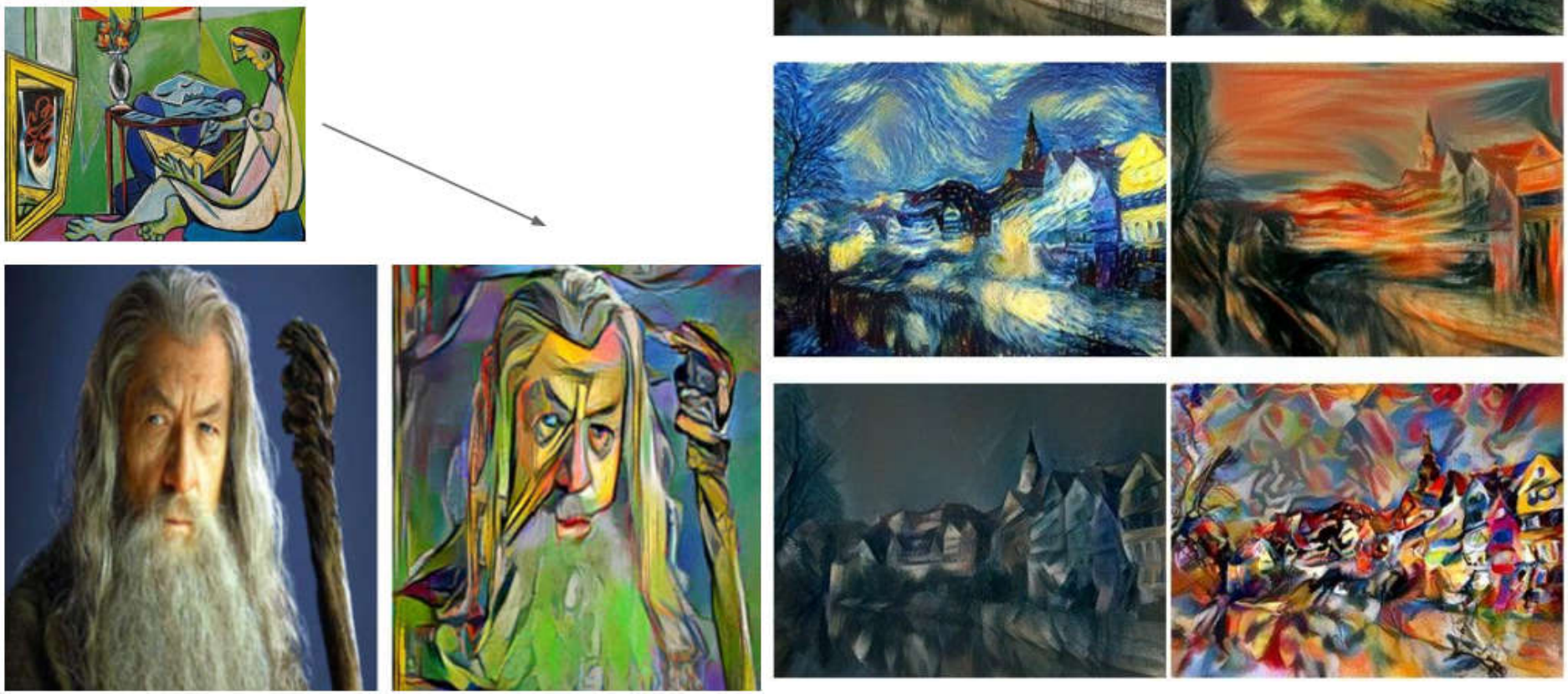
Deep Dreaming Fear & Loathing in Las Vegas: the Great San Francisco Acid Wave:

https://www.youtube.com/watch?v=oyxSerkkP4o

Neural Artistic Style

Neural Artistic Style

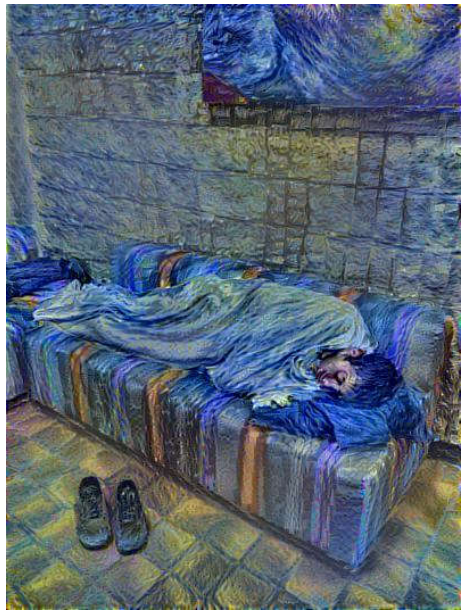
[L. A. Gatys, et al., Image Style Transfer Using Convolutional Neural Networks, **CVPR2016**]

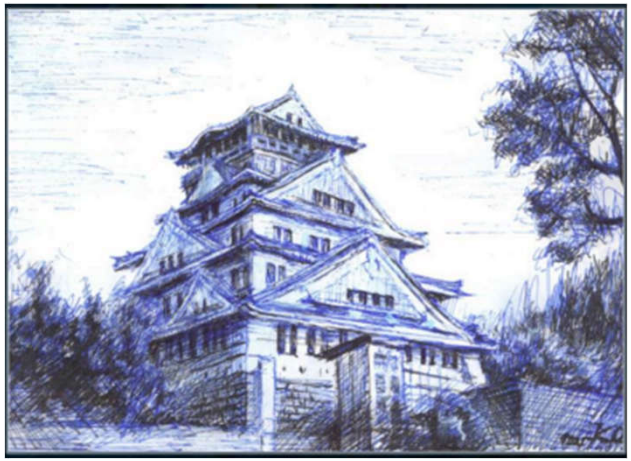
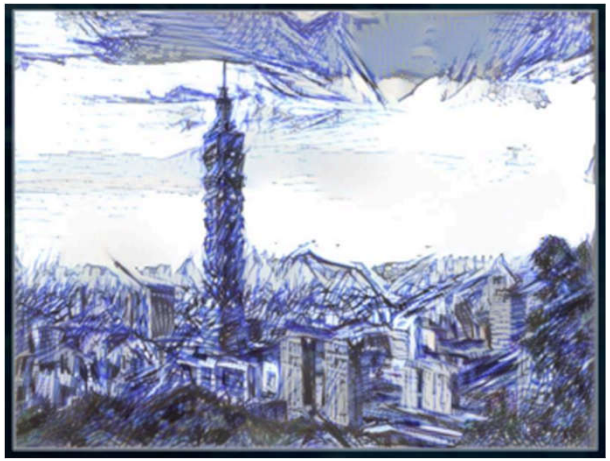


Slide from CS231n

Neural Artistic Style



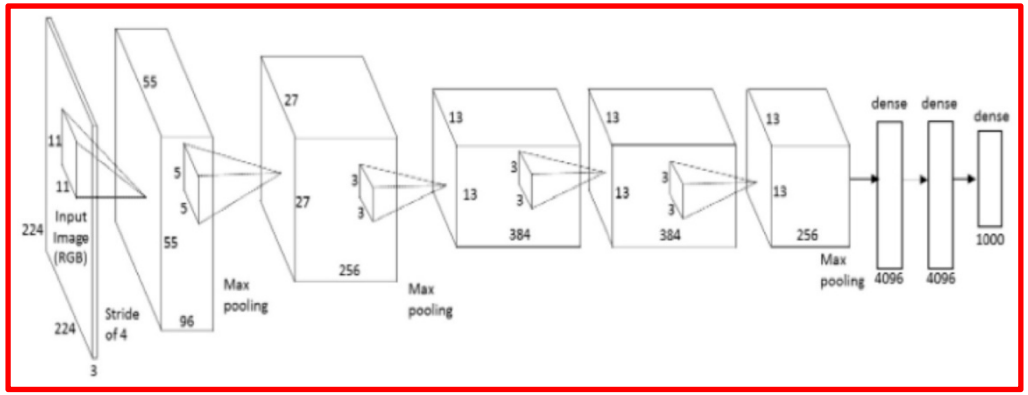


Slide from CS231n

Neural Artistic Style

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



Content activations

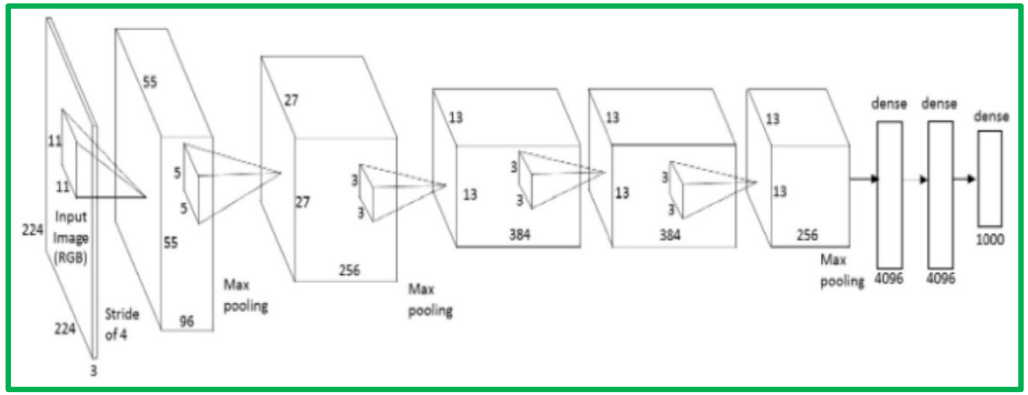
e.g.

At **CONV5_1** layer we would have a **[14x14x512]** array of target activations

Slide from CS231n

Neural Artistic Style

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



Style gram matrices

$$G = V^T V$$

e.g.

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

Slide from CS231n

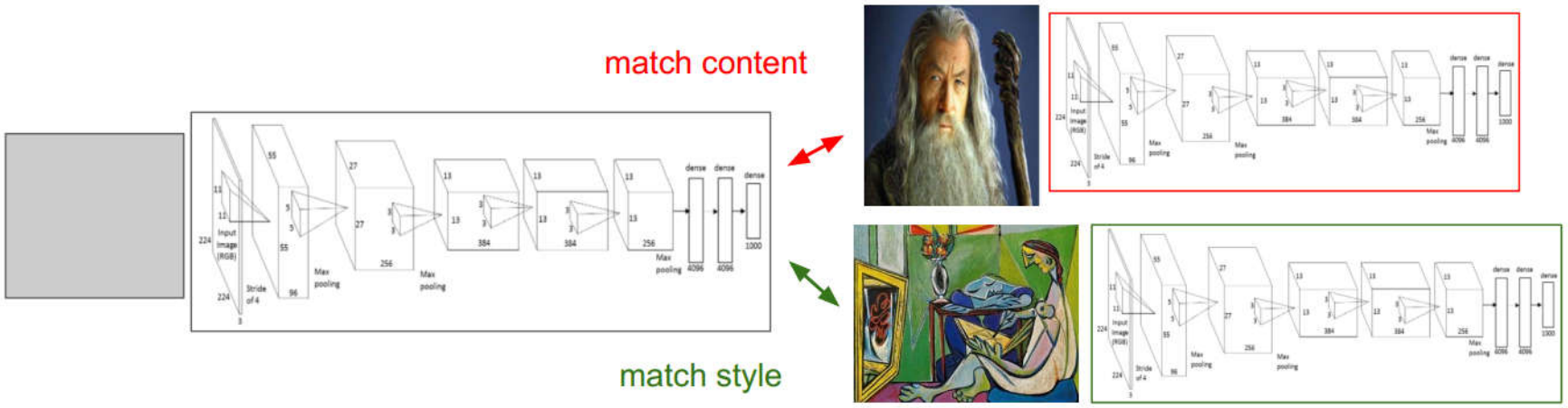
Neural Artistic Style

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)

$$L_{total}\left(\vec{p},\vec{a},\vec{x}\right)=\alpha L_{content}\left(\vec{p},\vec{x}\right)+\beta L_{style}\left(\vec{a},\vec{x}\right)$$

(+ Total Variation regularization)



Slide from CS231n

Fooling Neural Networks

Fooling Neural Networks

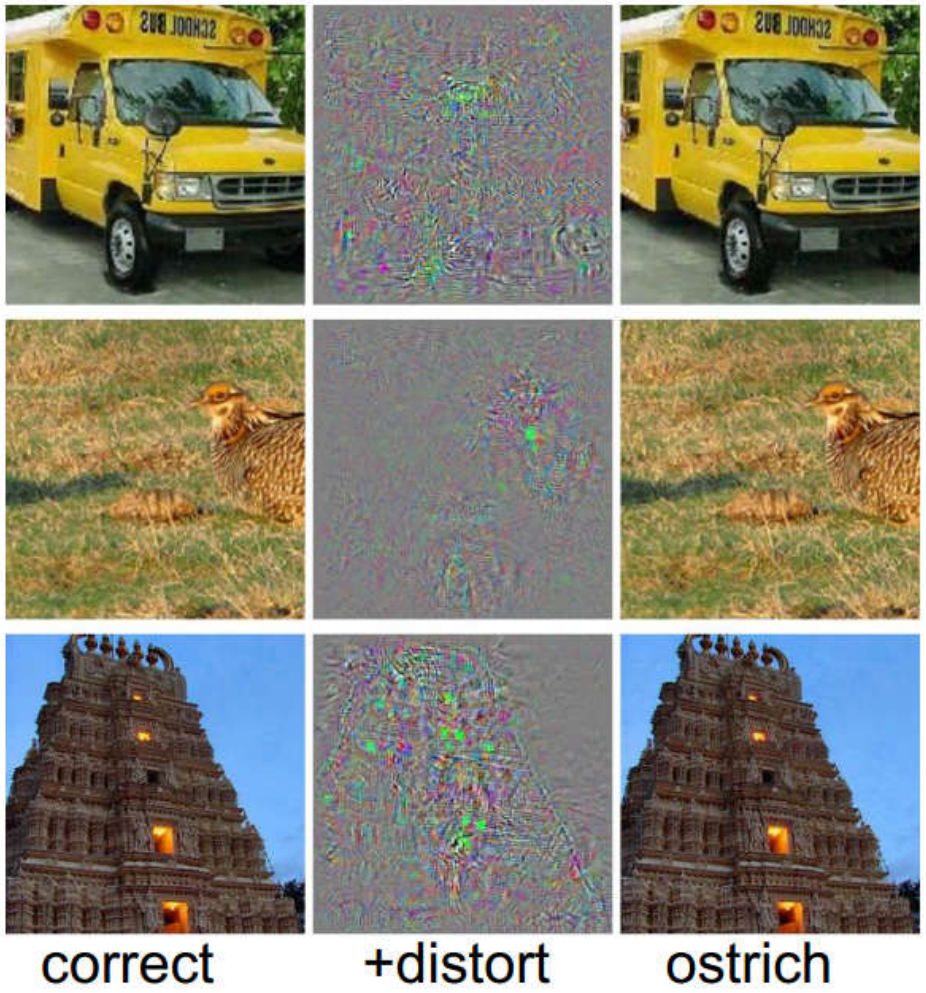
We can pose an optimization over the input image to maximize any class score. That seems useful.

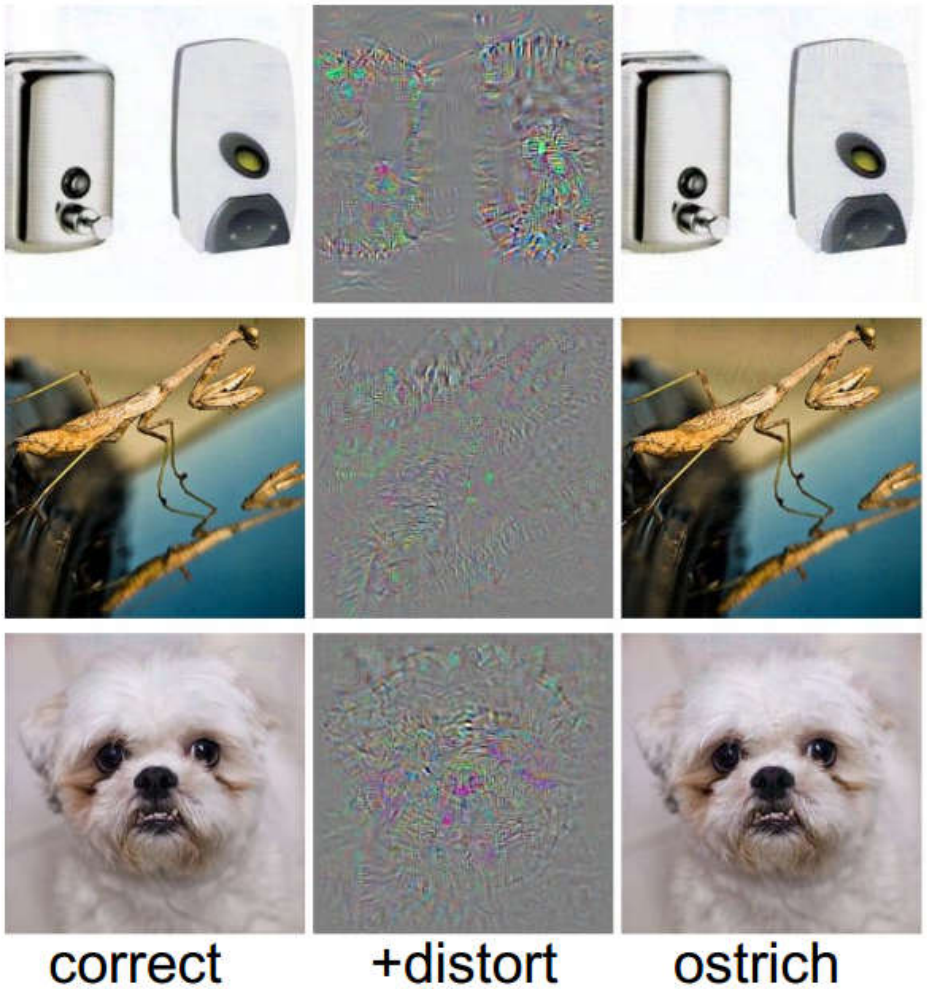
Question: Can we use this to “fool” ConvNets?

spoiler alert: yeah

Fooling Neural Networks

C. Szegedy, et al., Intriguing properties of neural networks, arXiv2013



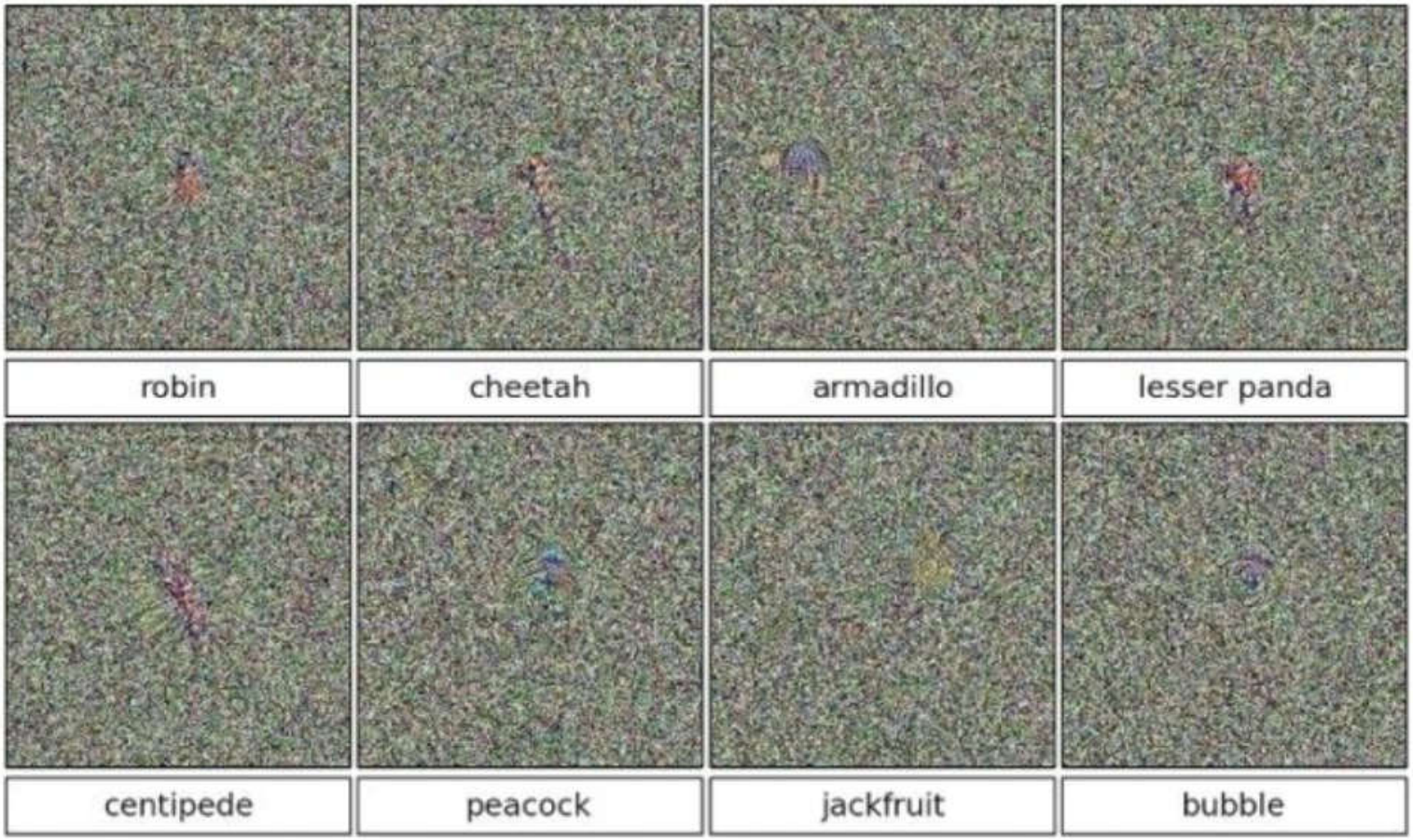


Slide from CS231n

Fooling Neural Networks

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

>99.6% confidences

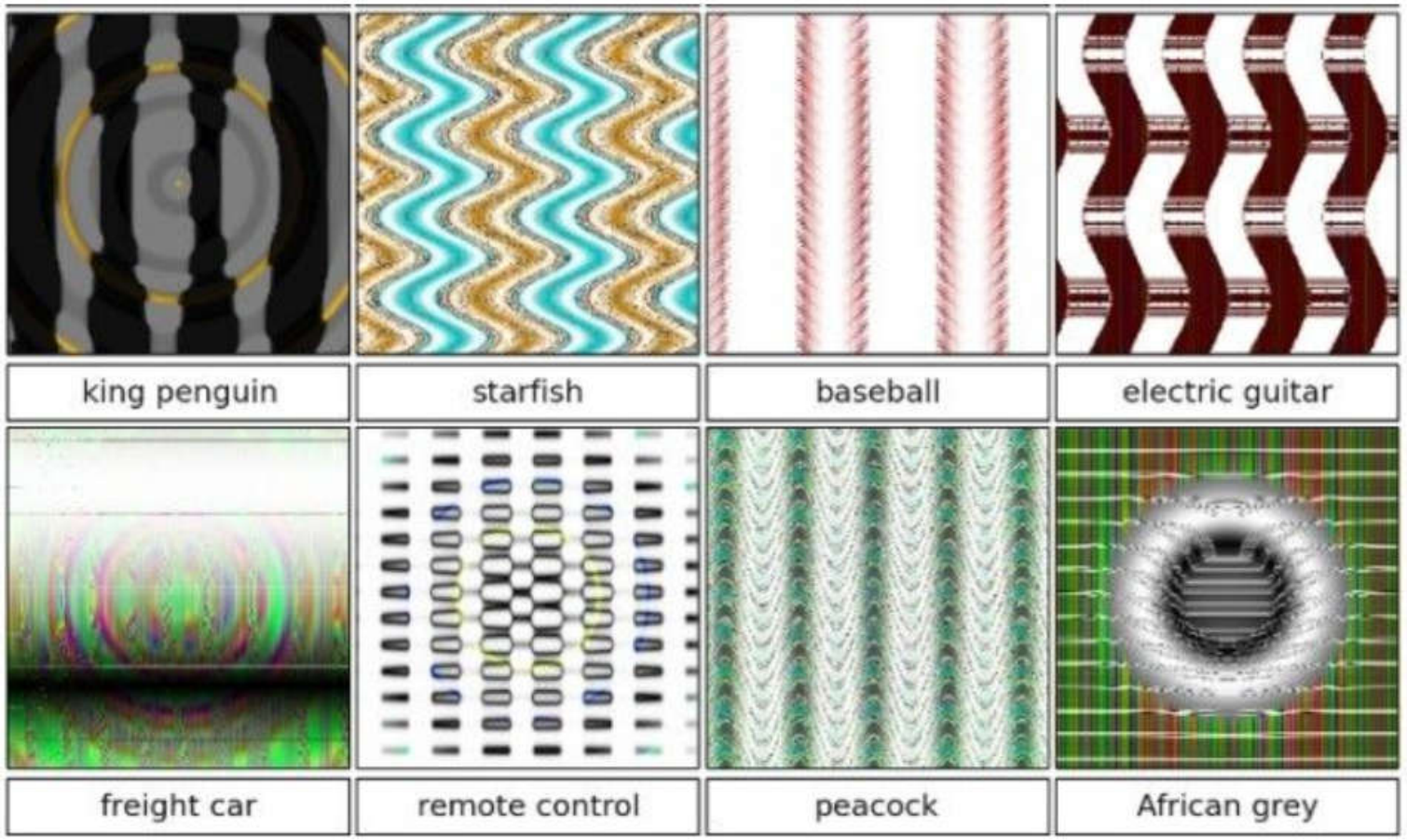


Slide from CS231n

Fooling Neural Networks

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

>99.6% confidences

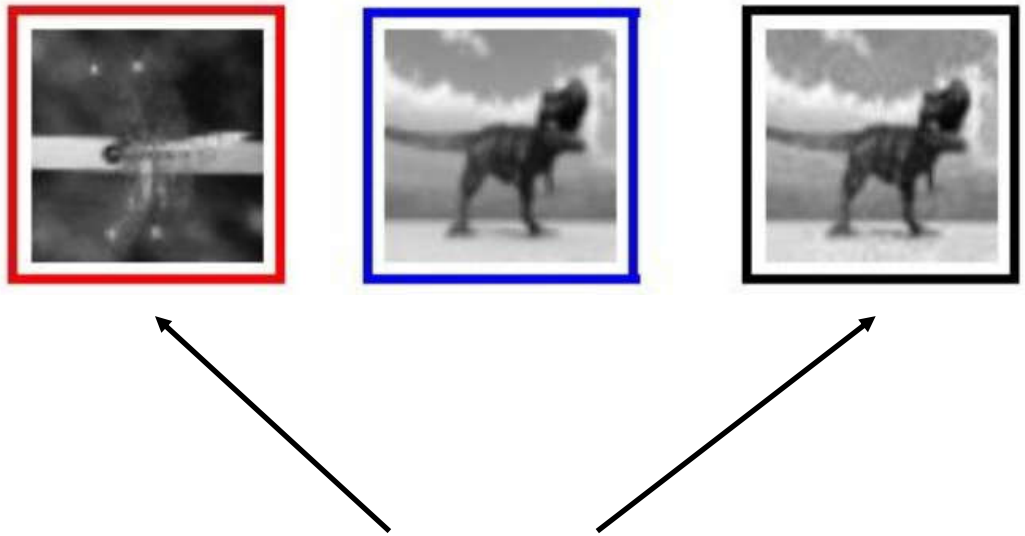


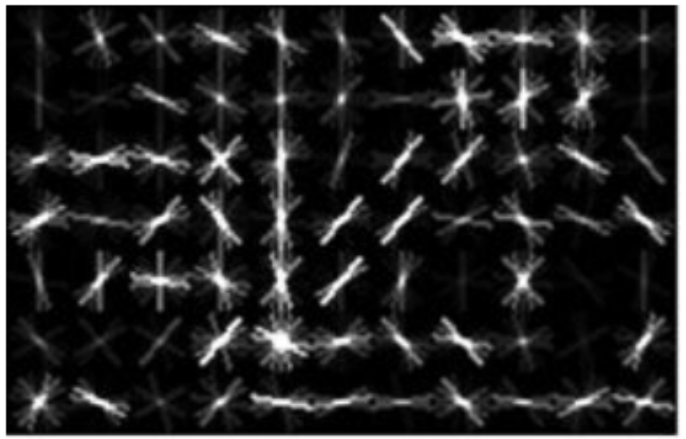
Slide from CS231n

Fooling Neural Networks

These kinds of results were around even before ConvNets…

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



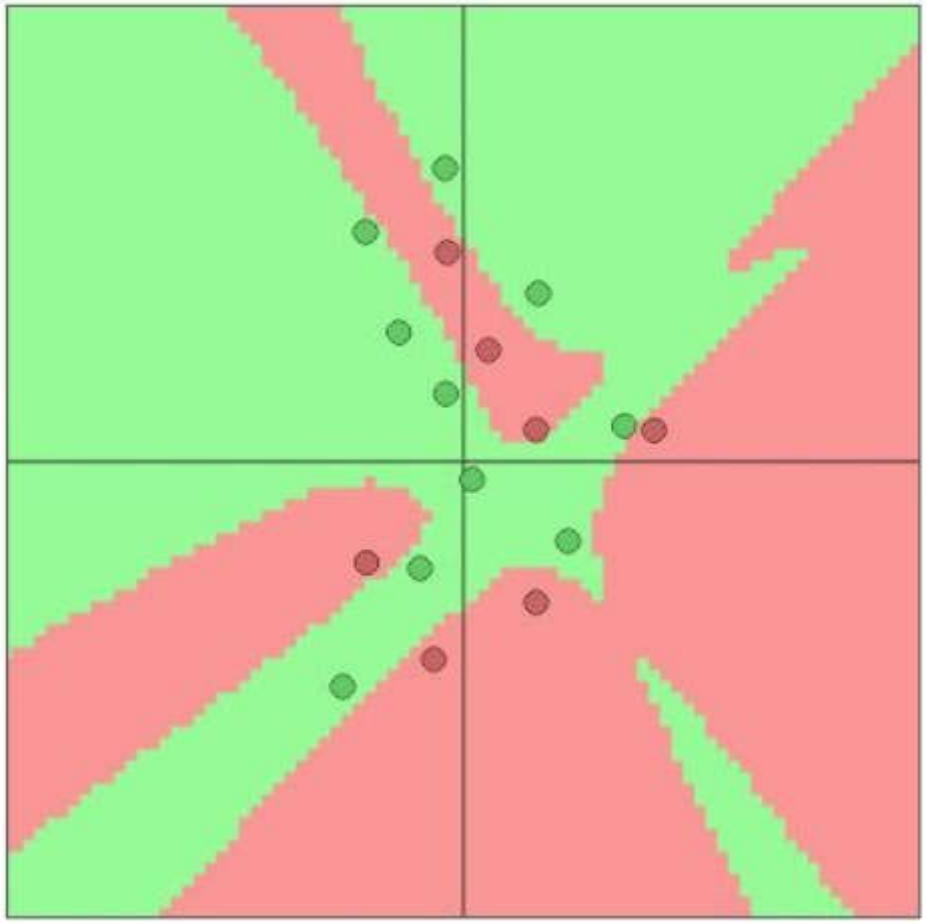


Identical HOG representation

Fooling Neural Networks

Explaining and harnessing adversarial examples, Goodfellow, Shlens & Szegedy, 2014

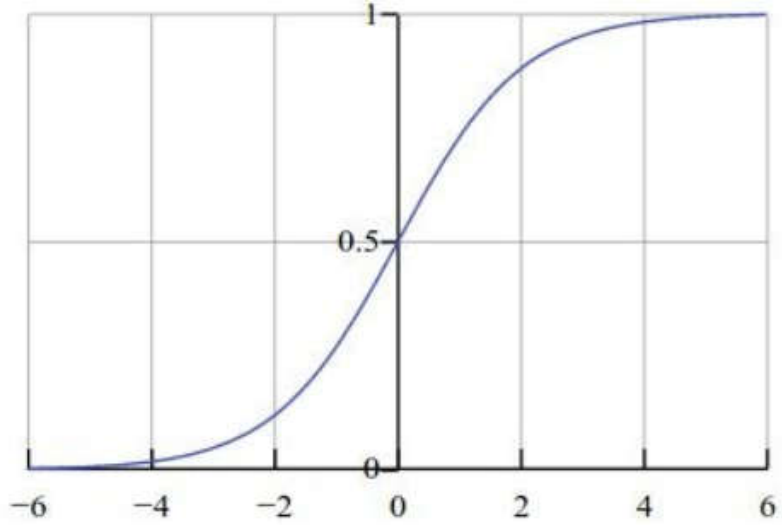
"Primary cause of neural networks' vulnerability to adversarial perturbation is their linear nature"



Slide from CS231n

Lets Fool a Binary Linear Classifier (Logistic Regression)

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



Since the probabilities of class 1 and 0 sum to one, the probability for class 0 is $\mathrm{P}(y = 0 \mid x;\, w,\, b) = 1 - \mathrm{P}(y = 1 \mid x;\, w,\, b)$. Hence, an example is classified as a positive example $(y = 1)$ if $\sigma(w^{\mathrm{T}}x + b) > 0.5$, or equivalently if the score $w^{\mathrm{T}}x + b > 0$.

Slide from CS231n

Lets Fool a Binary Linear Classifier (Logistic Regression)

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1

← input example

← weights

Class 1 score = dot product:

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

$\Rightarrow$ Probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

i.e. the classifier is **95%** certain that this is class 0 example.

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

Slide from CS231n

Lets Fool a Binary Linear Classifier (Logistic Regression)

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1
	?	?	?	?	?	?	?	?	?	?

← input example

← weights

Adversarial x

Class 1 score = dot product:

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

$\Rightarrow$ Probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

i.e. the classifier is **95%** certain that this is class 0 example.

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

Slide from CS231n

Lets Fool a Binary Linear Classifier (Logistic Regression)

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1
	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5

← input example

← weights

Adversarial x

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

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%

Slide from CS231n

Lets Fool a Binary Linear Classifier (Logistic Regression)

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1
	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5

← input example

← weights

Adversarial x

This was only with 10 input dimensions. A 224x224 input image has 150,528.

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

(It's significantly easier with more numbers, need smaller nudge for each)

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

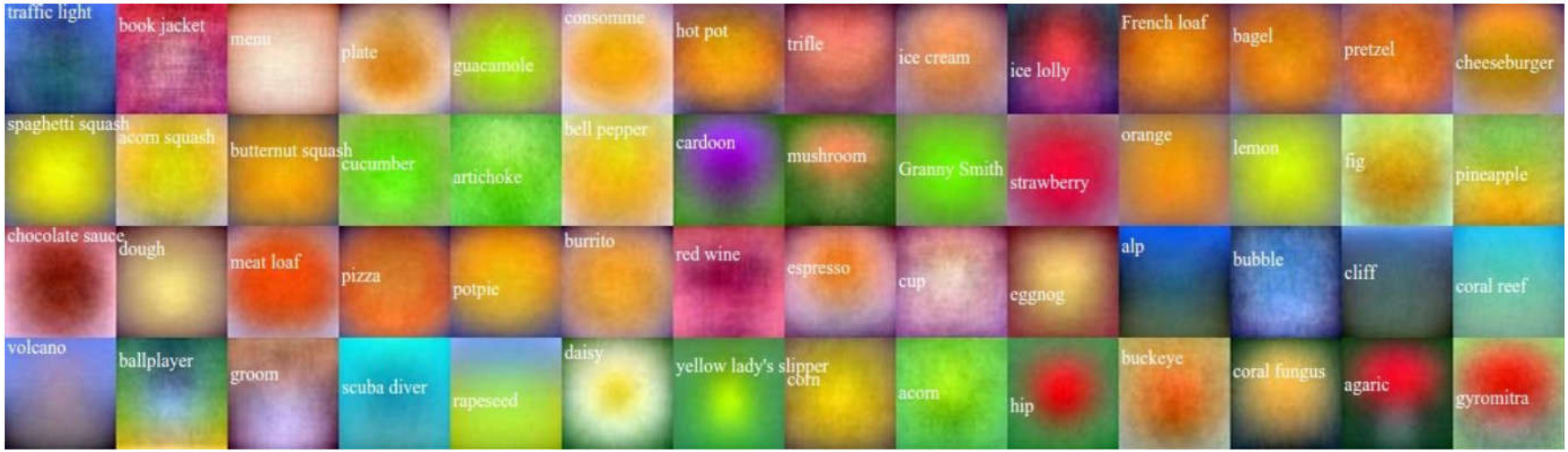
⇒ Probability of class 1 is now 1/(1+e^(-(2))) = 0.88

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

Blog Post: Breaking Linear Classifiers on ImageNet

Recall CIFAR-10 linear classifiers:

ImageNet classifiers:



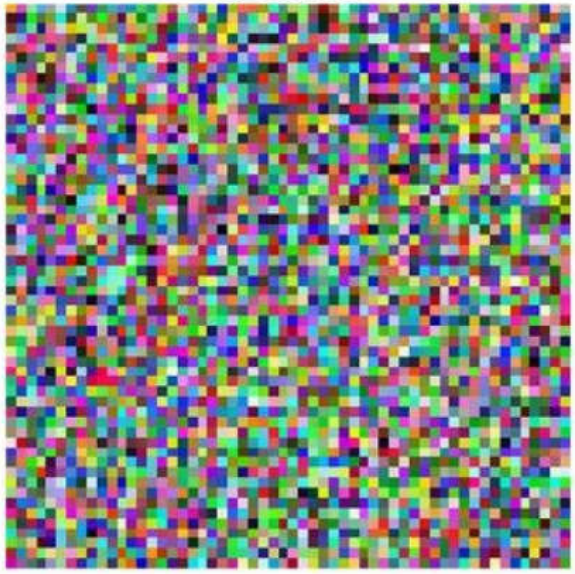
Slide from CS231n

Blog Post: Breaking Linear Classifiers on ImageNet

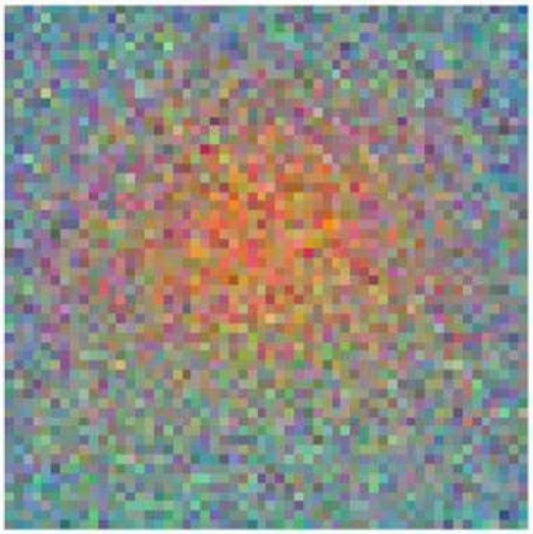
Mix in a tiny bit of

Goldfish classifier weights

0.9% bobsled

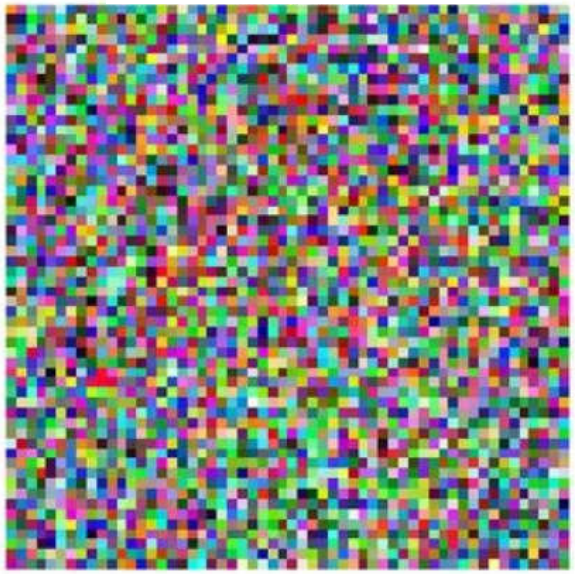


+



=

100.0% goldfish

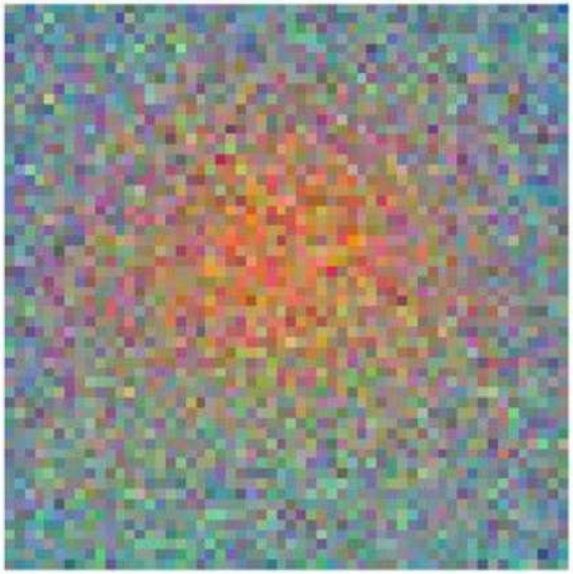


100% Goldfish

Slide from CS231n

Blog Post: Breaking Linear Classifiers on ImageNet

1.0% kit fox

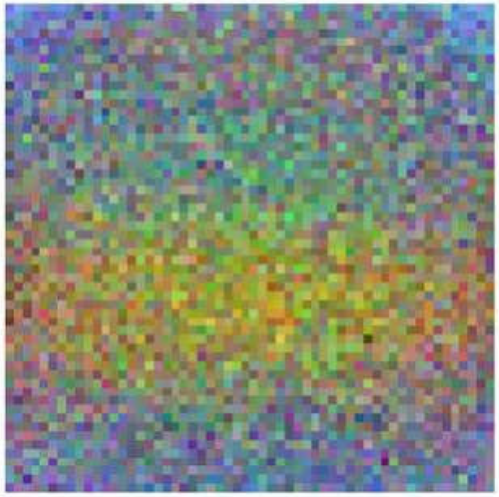


8.0% goldfish

Slide from CS231n

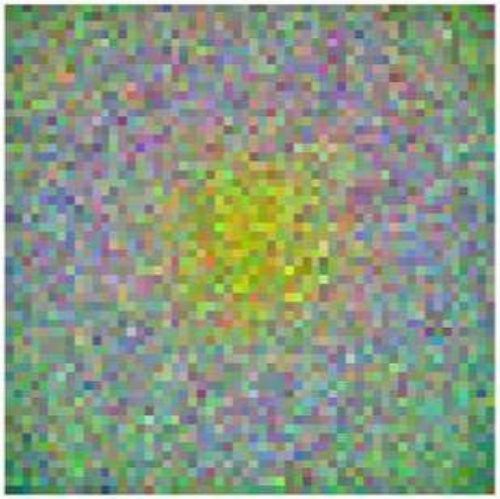
Blog Post: Breaking Linear Classifiers on ImageNet

1.0% kit fox

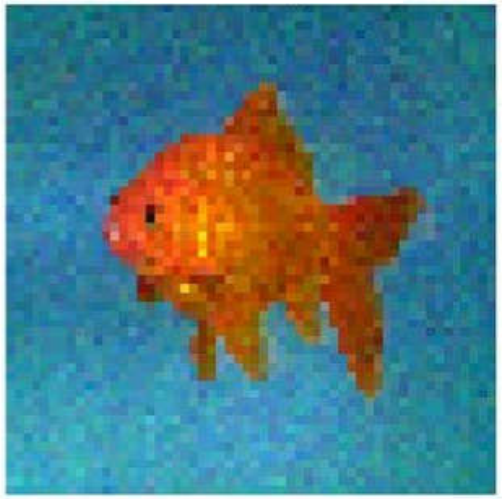


3.9% school bus

8.3% goldfish



12.5% daisy

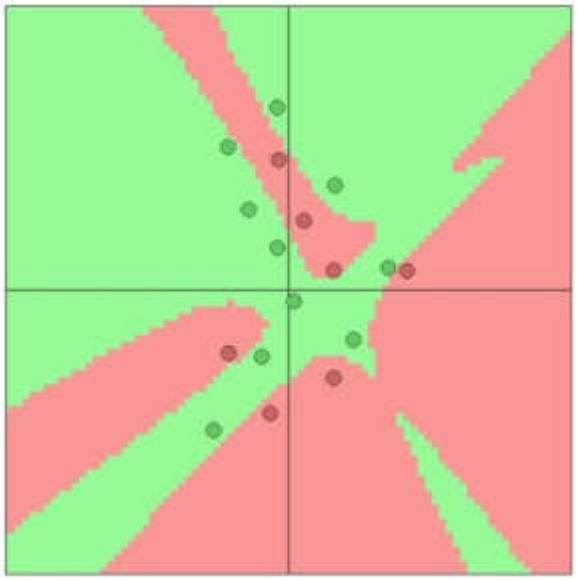


Slide from CS231n

Fooling Deep Neural Networks

Explaining and Harnessing Adversarial Examples [Goodfellow, Shlens & Szegedy, 2014]

"Primary cause of neural networks' vulnerability to adversarial perturbation is their linear nature" (and very high-dimensional, sparsely-populated input spaces)



In particular, 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.

Slide from CS231n

Summary

Backpropping to the image is powerful. It can be used for:

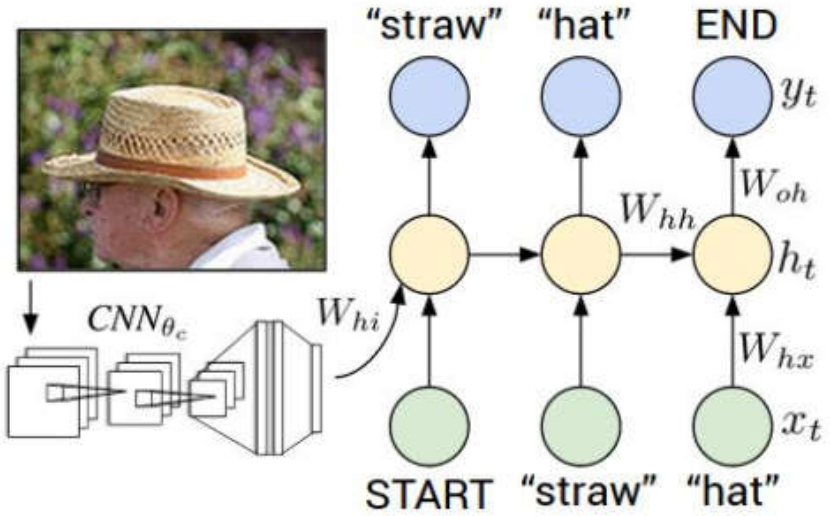
- **Understanding** (e.g. visualize optimal stimuli for arbitrary neurons)
- **Segmenting** objects in the image (kind of)
- **Inverting** codes and introducing privacy concerns
- **Fun** (NeuralStyle/DeepDream)
- **Confusion and chaos** (Adversarial examples)

Next Lecture

Image Captioning

Recurrent Neural Networks

RNN Language Models



"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."

Slide from CS231n

Thank you for your attention!