Administrative

- A2 has a number of corrections on Pizza. They are fixed in most recent .zip file.
- Btw CNNs in Matlab: http://www.vlfeat.org/matconvnet/

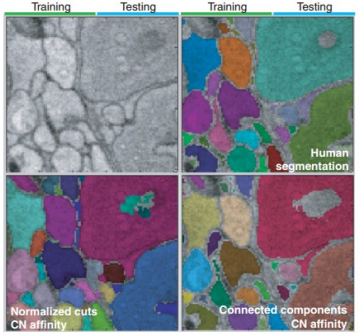
Lecture 8:

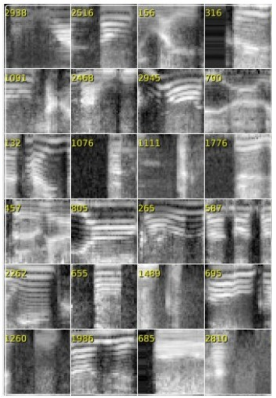
Visualizing and Understanding Convolutional Neural Networks

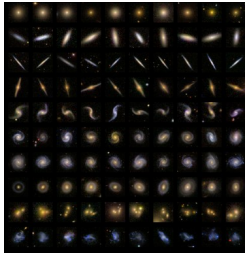
dumbbell cup dalmatian

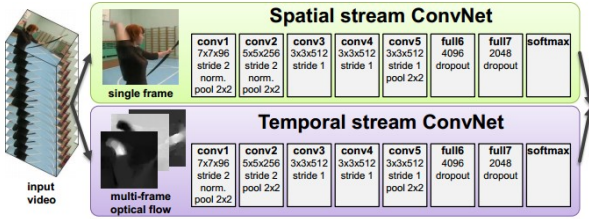
[Simonyan et al. 2014]

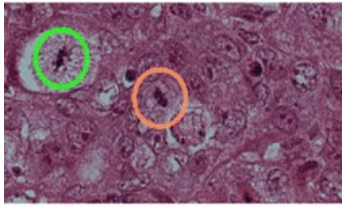
Where we are...

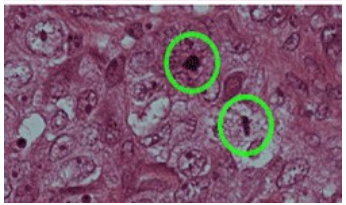

Training
Testing
Training
Testing
Human segmentation
Normalized cuts CN affinity
Connected components CN affinity

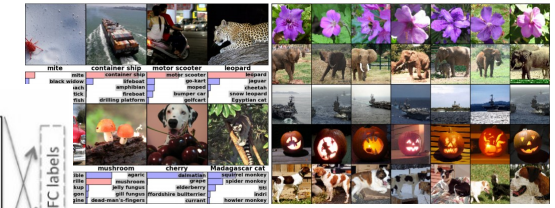


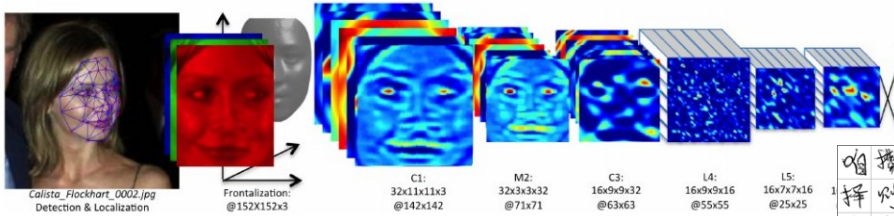


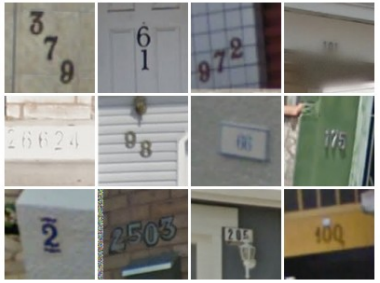

Spatial stream ConvNet
single frame
conv1 7x7x96 stride 2 norm. pool 2x2
conv2 5x5x256 stride 2 norm. pool 2x2
conv3 3x3x512 stride 1
conv4 3x3x512 stride 1
conv5 3x3x512 stride 1 pool 2x2
full6 4096 dropout
full7 2048 dropout
softmax
Temporal stream ConvNet
multi-frame optical flow
input video
class score fusion



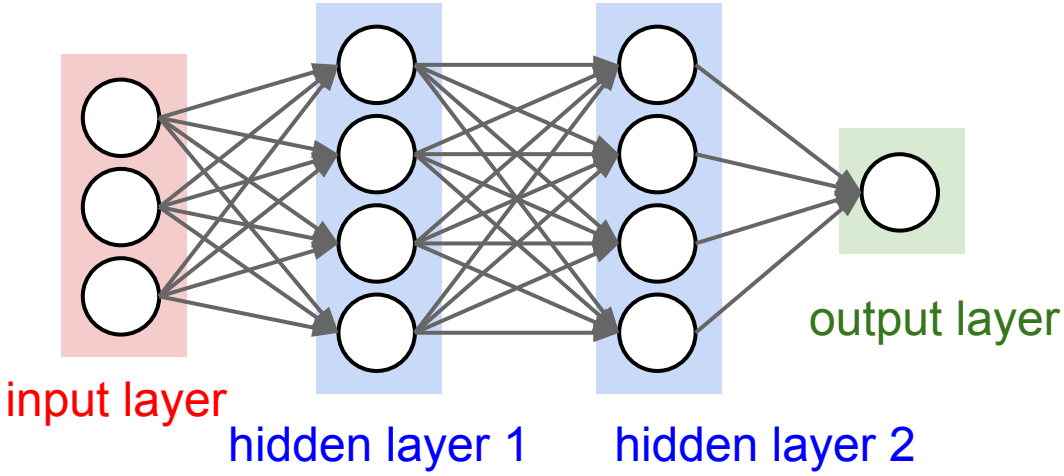



mite
container ship
motor scooter
leopard
mushroom
cherry
Madagascar cat

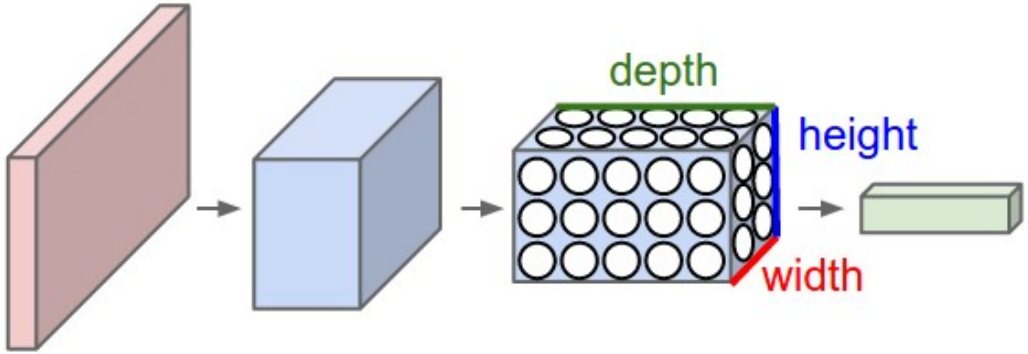

Calista_Flockhart_0002.jpg
Detection & Localization
Frontalization: @152X152x3
C1: 32x11x11x3 @142x142
M2: 32x3x3x32 @71x71
C3: 16x9x9x32 @63x63
L4: 16x9x9x16 @55x55
L5: 16x7x7x16 @25x25
REPRESENTATION
SFC labels



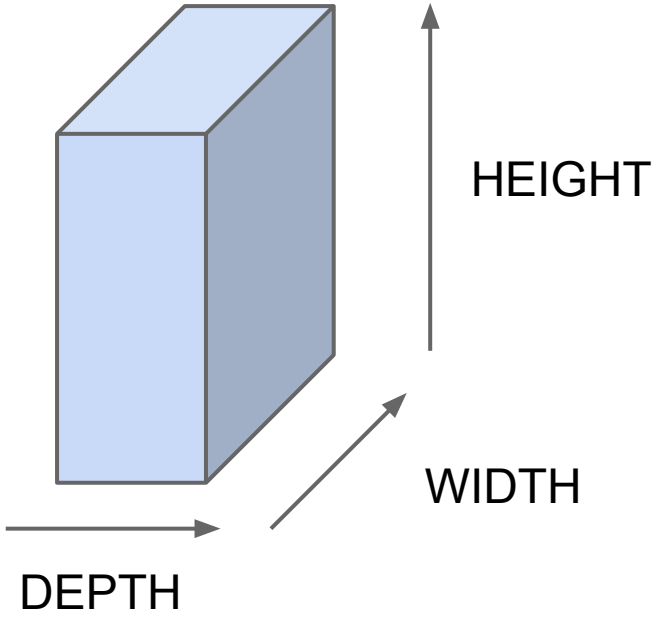
before:

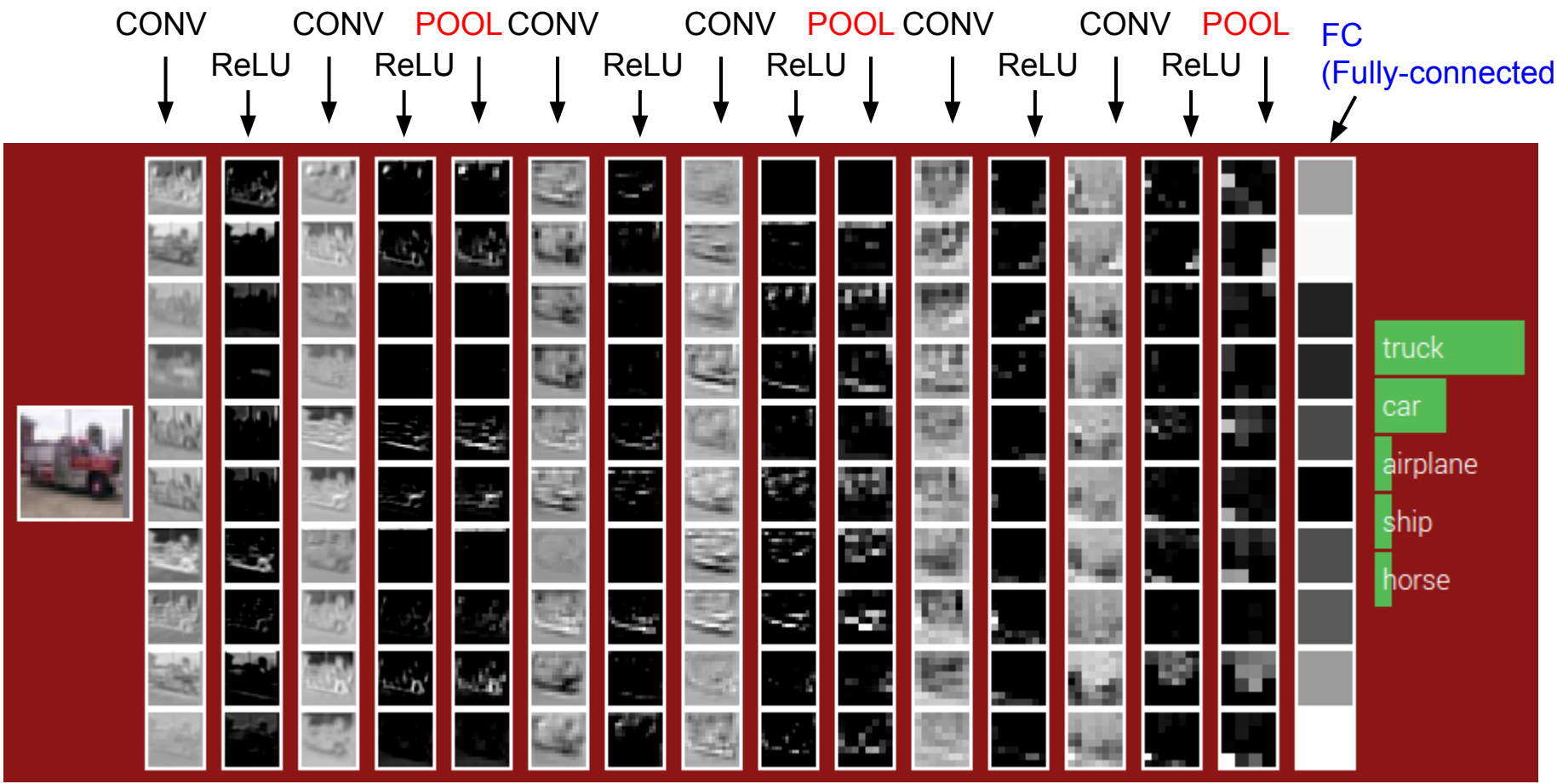

input layer
hidden layer 1
hidden layer 2
output layer

now:


depth
height
width

Every stage in a ConvNet has activations of three dimensions:




CONV
ReLU
CONV
ReLU
POOL
CONV
ReLU
CONV
ReLU
POOL
CONV
ReLU
CONV
ReLU
POOL
FC
(Fully-connected)
truck
car
airplane
ship
horse

Typical ConvNets look like:

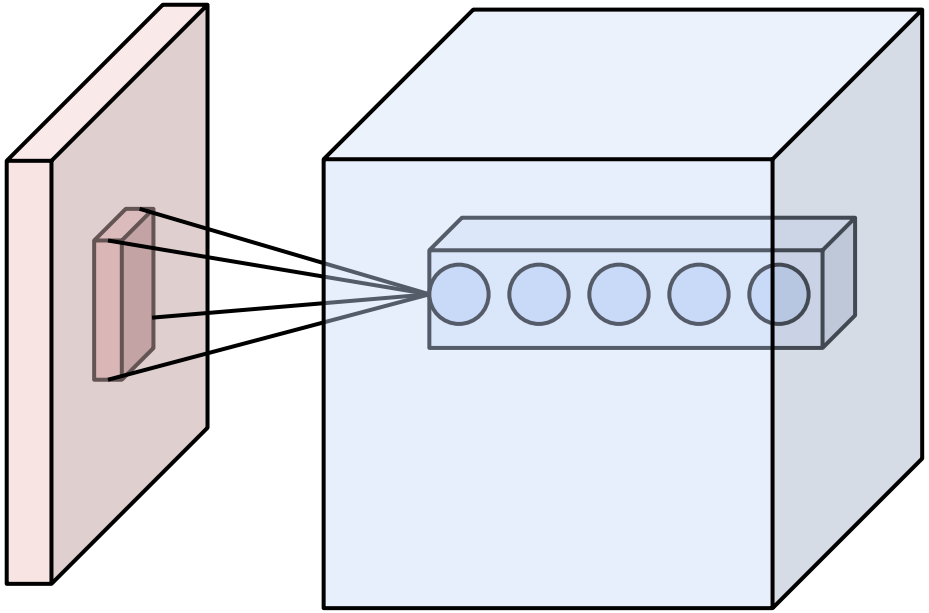
[CONV-RELU-POOL]xN,[FC-RELU]xM,FC,SOFTMAX or

[CONV-RELU-CONV-RELU-POOL]xN,[FC-RELU]xM,FC,SOFTMAX

N >= 0, M >=0

Note:
(last FC layer should not have RELU - these are the class scores)

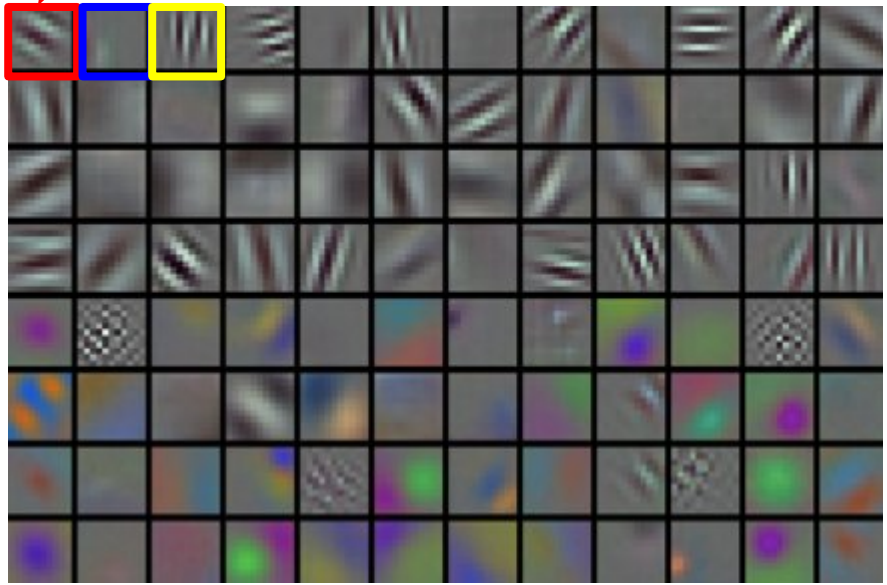
Convolutional Layer

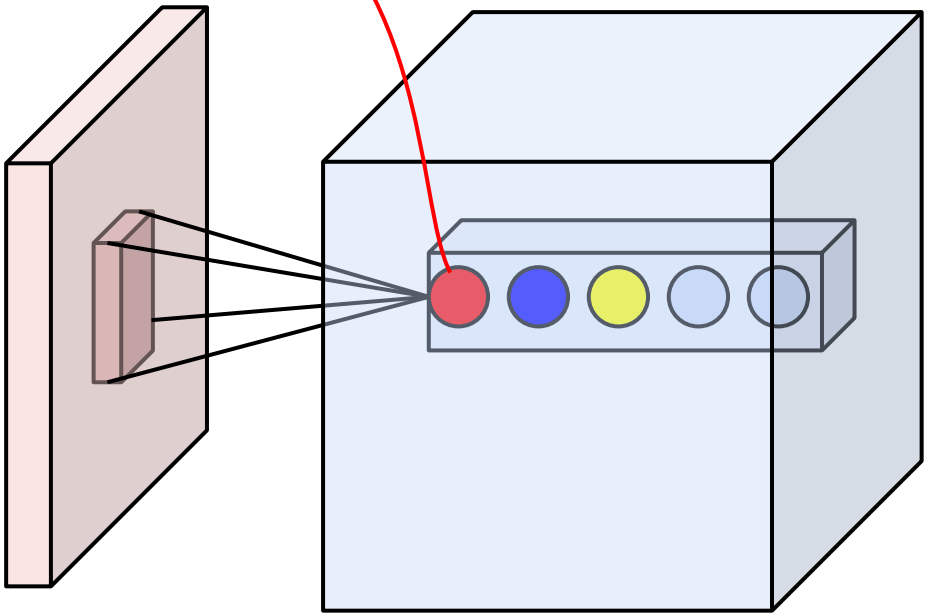


Just like normal Hidden Layer BUT:

- Connect neurons to the input in a **local receptive field**
- All neurons in a single depth slice **share weights**

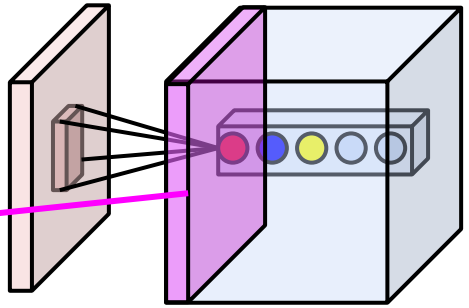
The weights of this neuron visualized



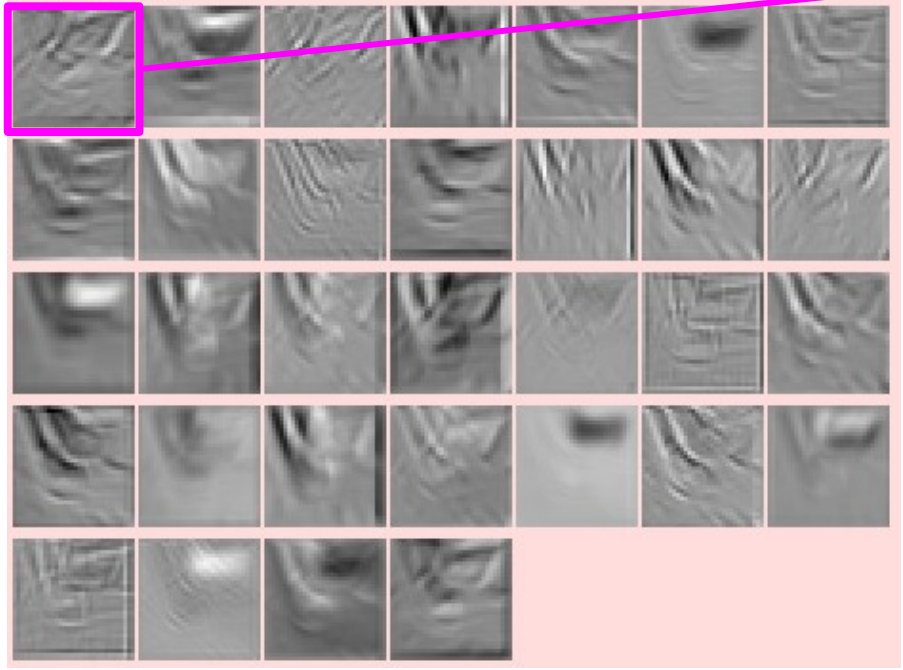


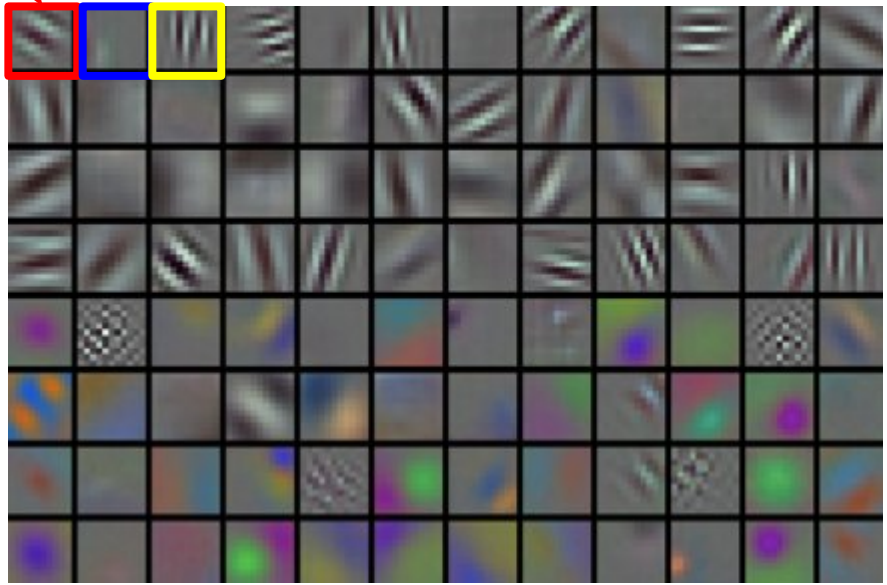
Activations:

convolving the first filter in the input gives the first slice of depth in output volume

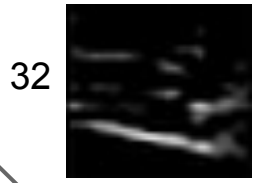


Activations:





Max Pooling Layer



downsampling

Single depth slice

x

1	1	2	4
5	**6**	7	**8**
3	2	1	0
1	2	3	**4**

y

max pool with 2x2 filters
and stride 2

6	8
3	4

Pooling layer downsamples every activation map in the input independently with max.

Modern CNN trend toward:

- Small filter sizes (3x3 and less)
- Small pooling sizes (2x2 and less)
- Small strides (stride = 1, ideally)
- Deep

- Conv Layers should **pad with zeros** to not reduce spatial size
- Pool Layers should reduce size once in a while
- Eventually Fully-Connected Layers take over

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

Note:

Most memory is in early CONV

Most params are in late FC

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

Lecture 8:

Visualizing and Understanding Convolutional Neural Networks

dumbbell cup dalmatian

[Simonyan et al. 2014]

Q: What are the properties of the learned CNN representation?

...
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

"CNN code"

A CNN transforms the image to 4096 numbers that are then linearly classified.

Method 3: Visualizing the CNN code representation

("CNN code" = 4096-D vector before classifier)

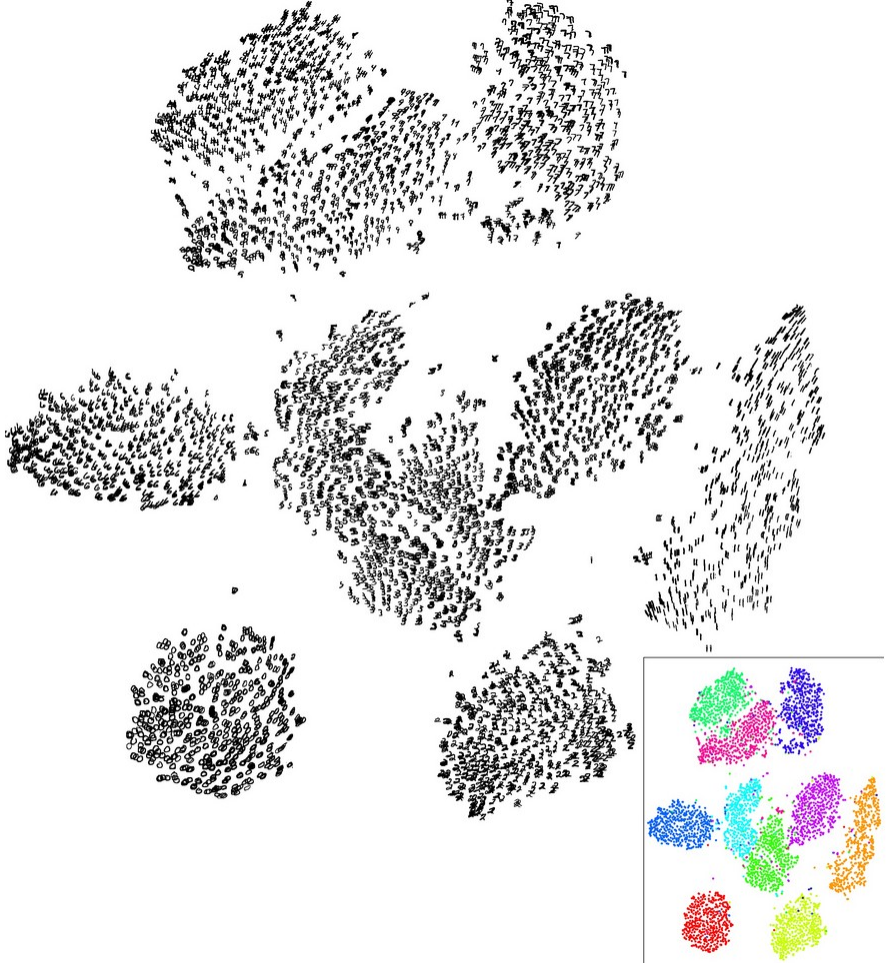
query image

nearest neighbors in the "code" space

(But we'd like a more global way to visualize the distances)

t-SNE visualization

[van der Maaten & Hinton]



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

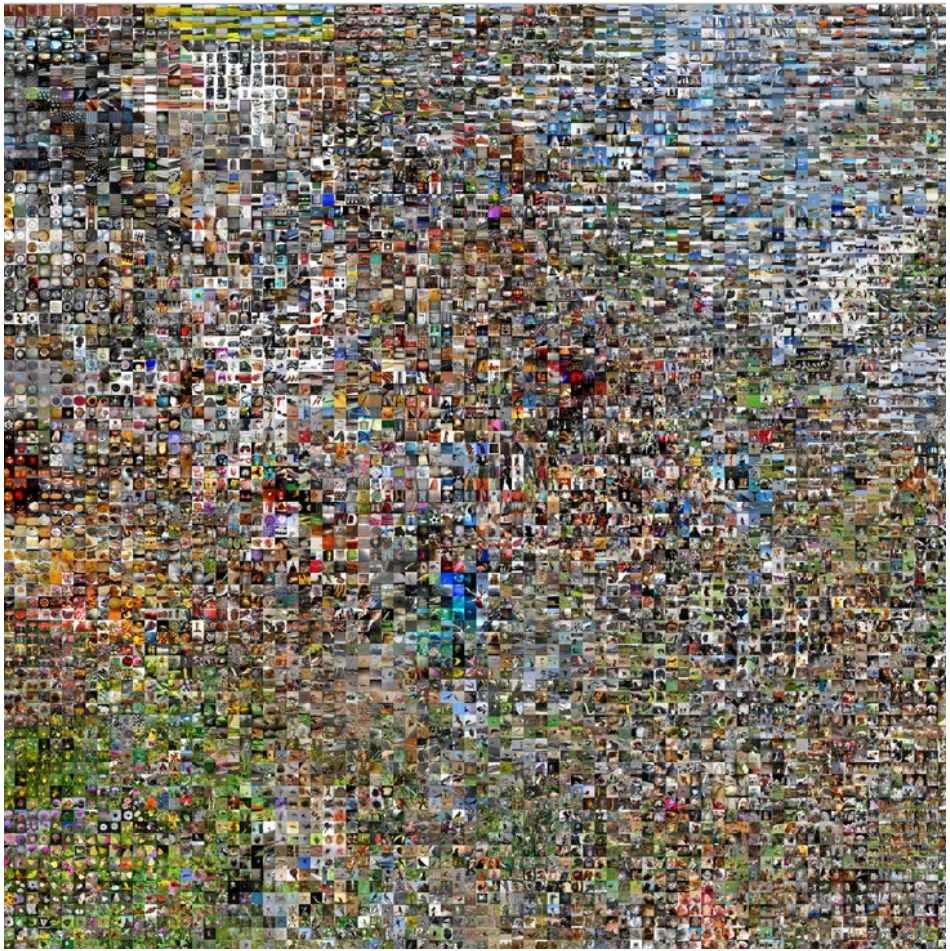
i.e. similar things end up in similar places. dissimilar things end up wherever

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/



t-SNE visualization

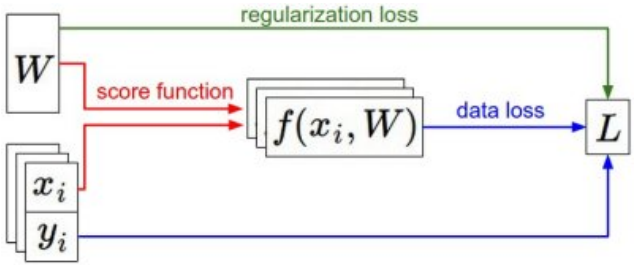
Q: What images maximize the score of some class in a ConvNet?

1. Find images that maximize some class score:

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

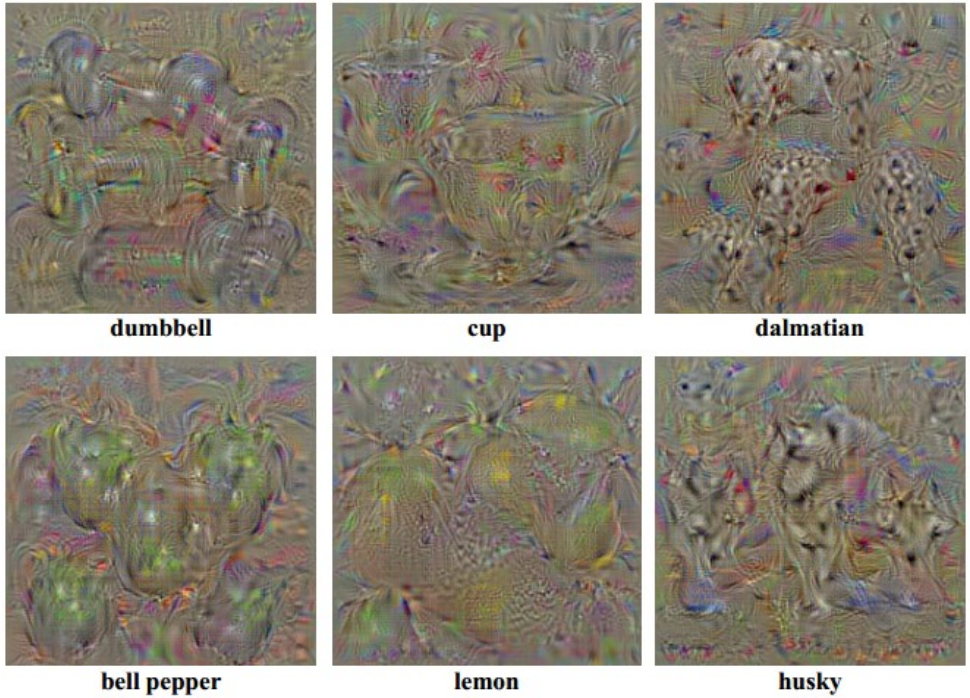
Score for class c
(before Softmax)

Remember:



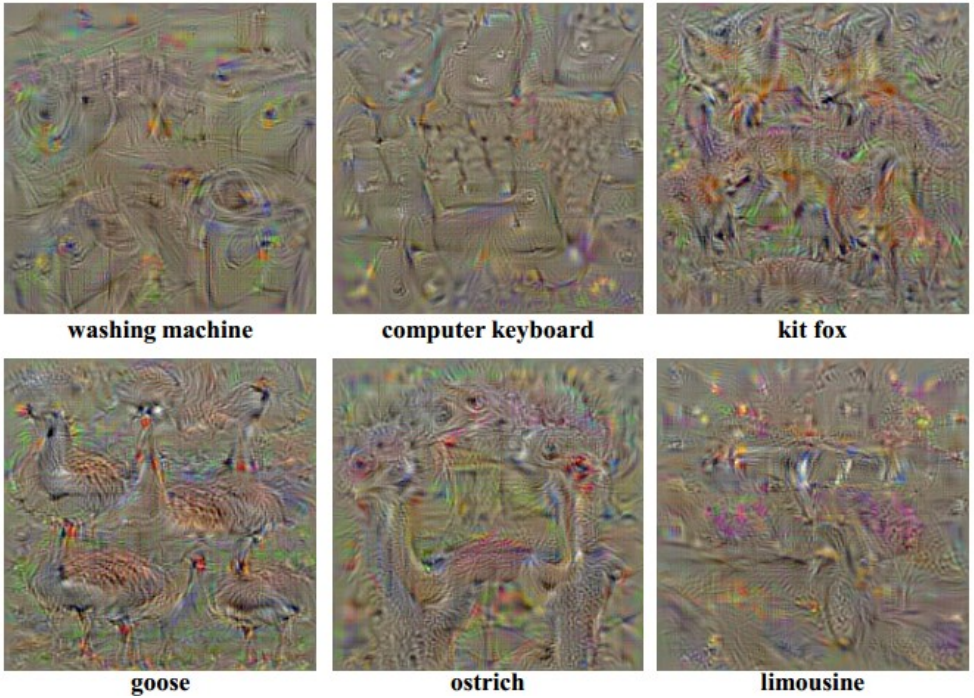
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014

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:



2. Visualize the Data gradient:



M = ?

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

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

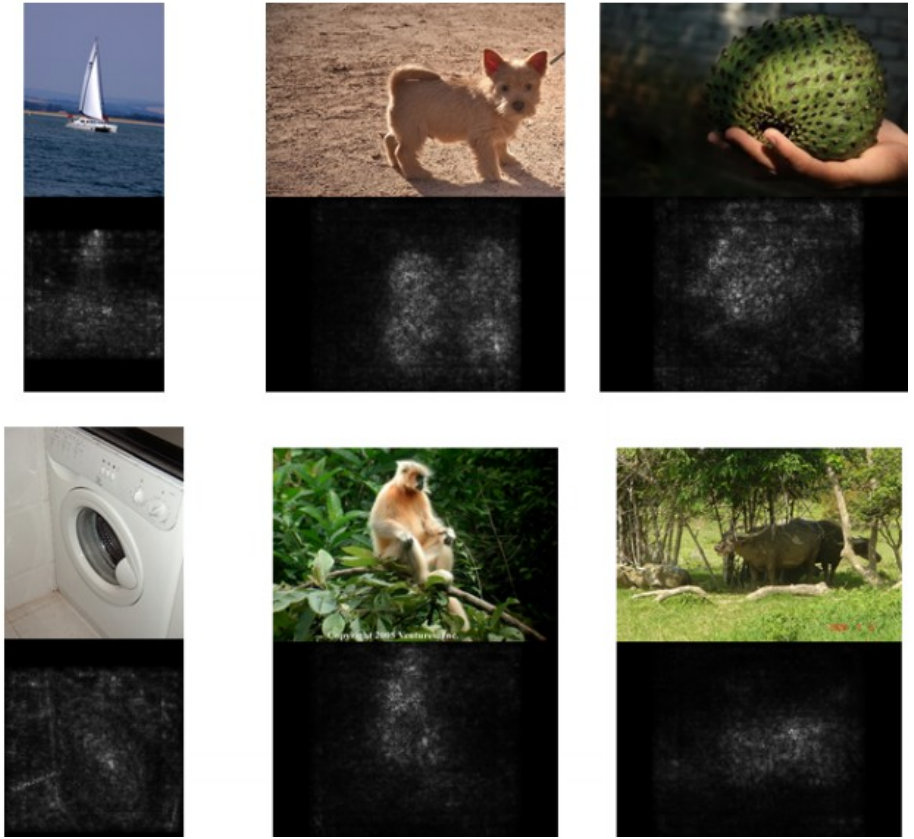
(at each pixel take abs val, and max over channels)

2. Visualize the Data gradient:

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

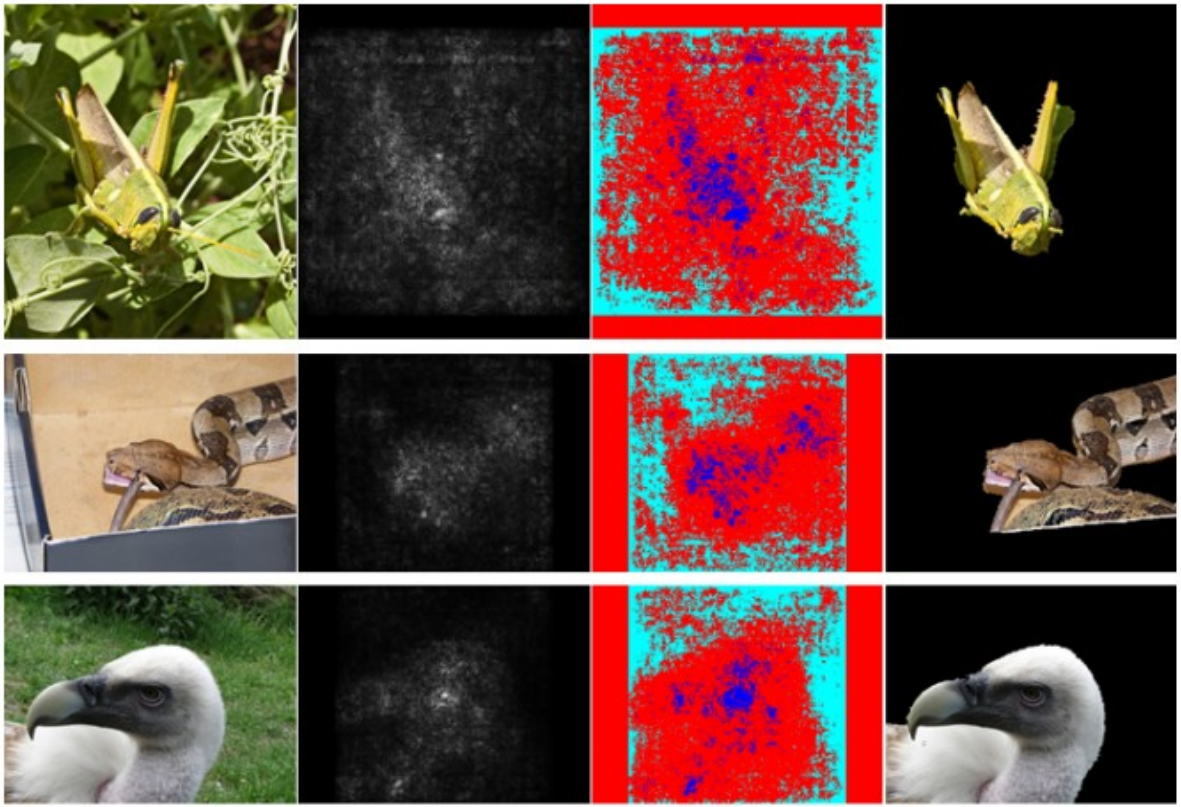
$$M_{ij} = \max_c |w_{h(i,j,c)}|$$

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

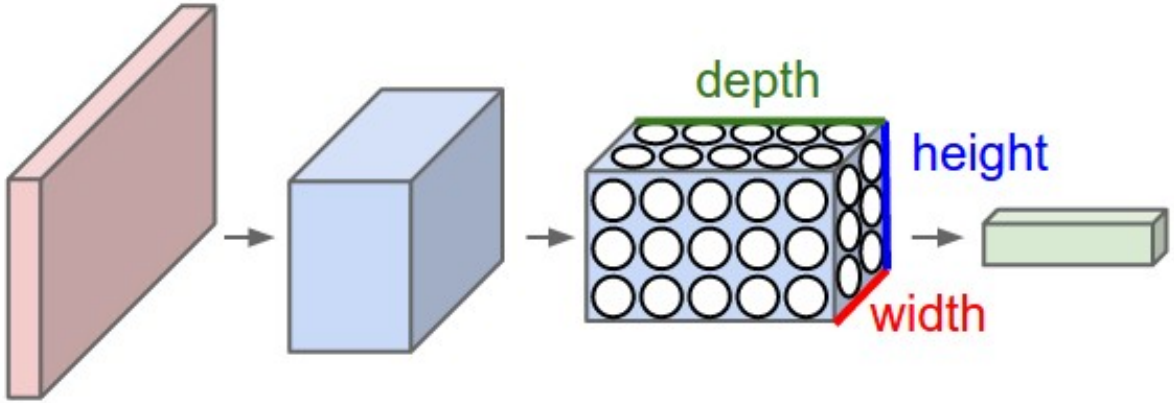


Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014

- Use **grabcut** for segmentation



Q: What do the individual neurons look for in an image?



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

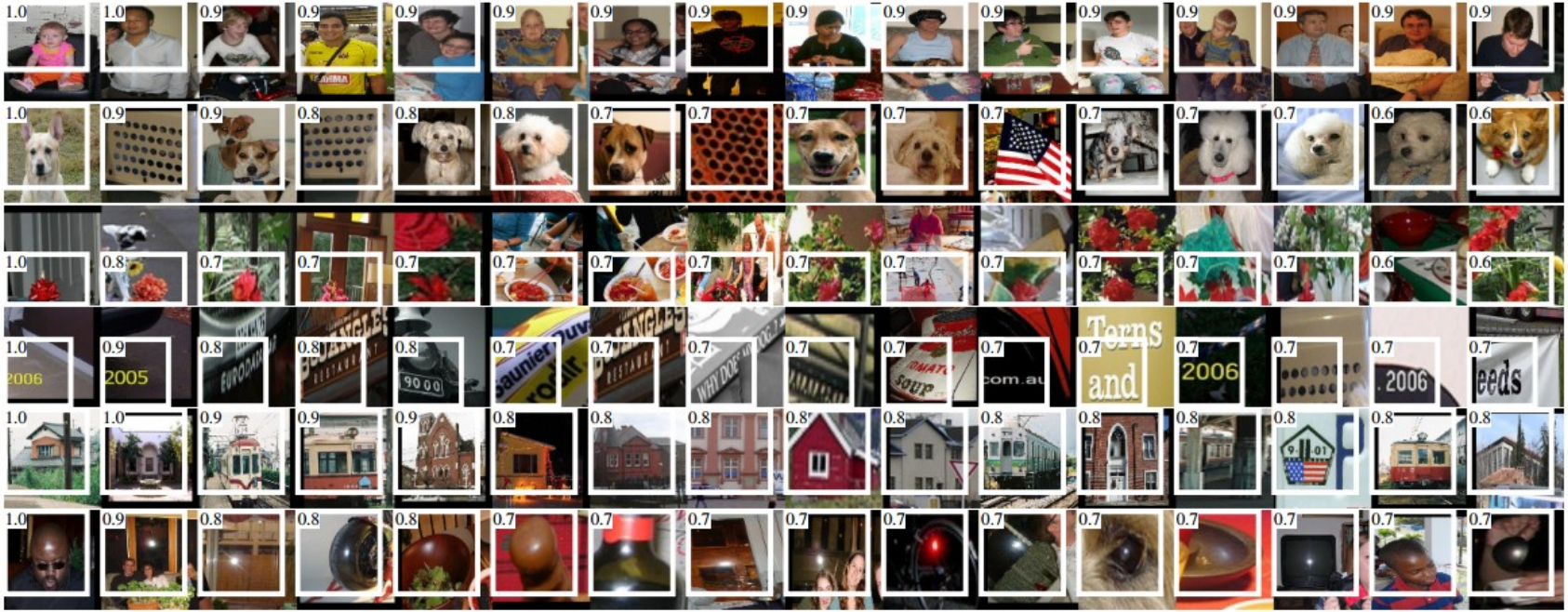
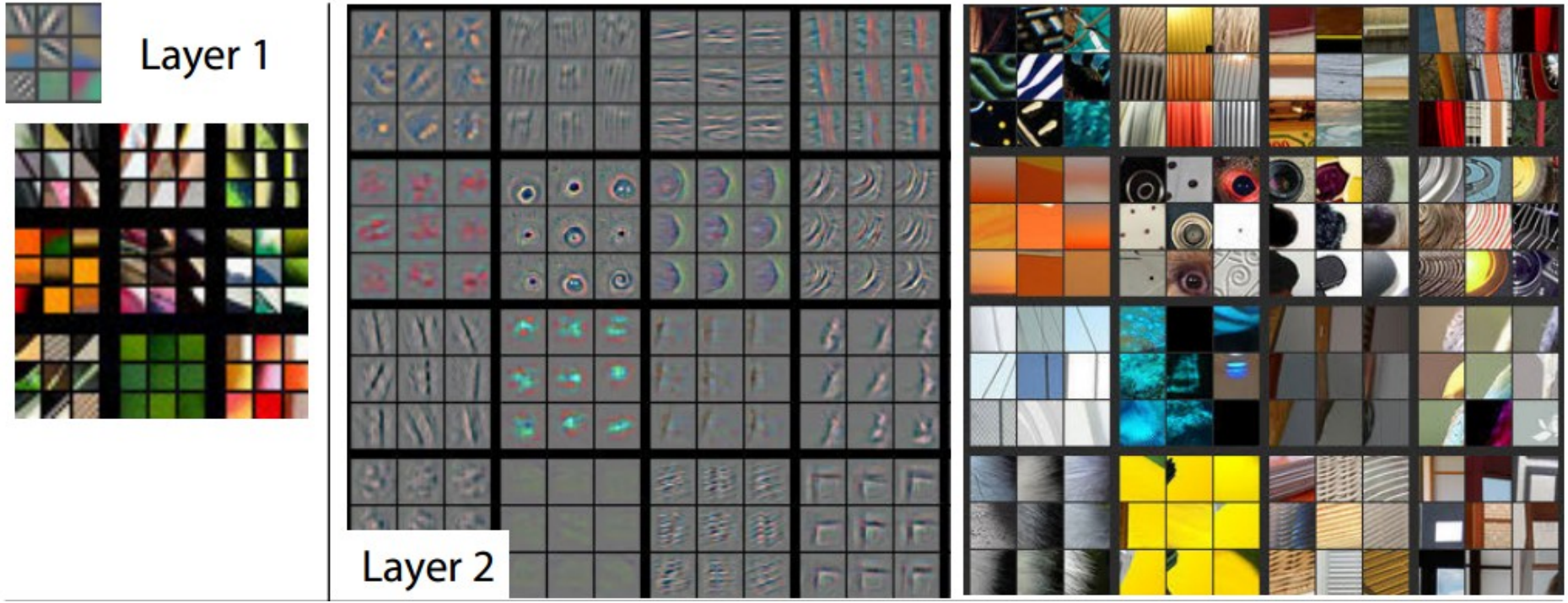


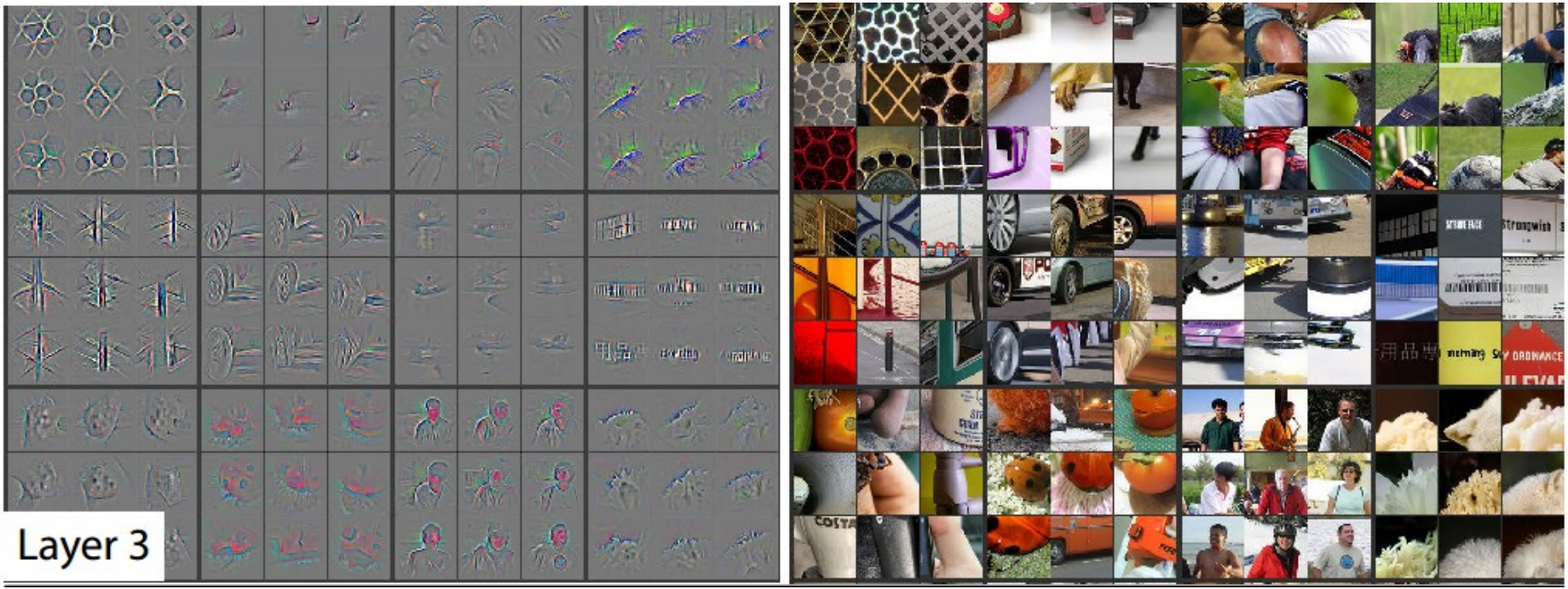
Figure 4: Top regions for six pool$_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

Visualizing and Understanding Convolutional Networks
Zeiler & Fergus, 2013

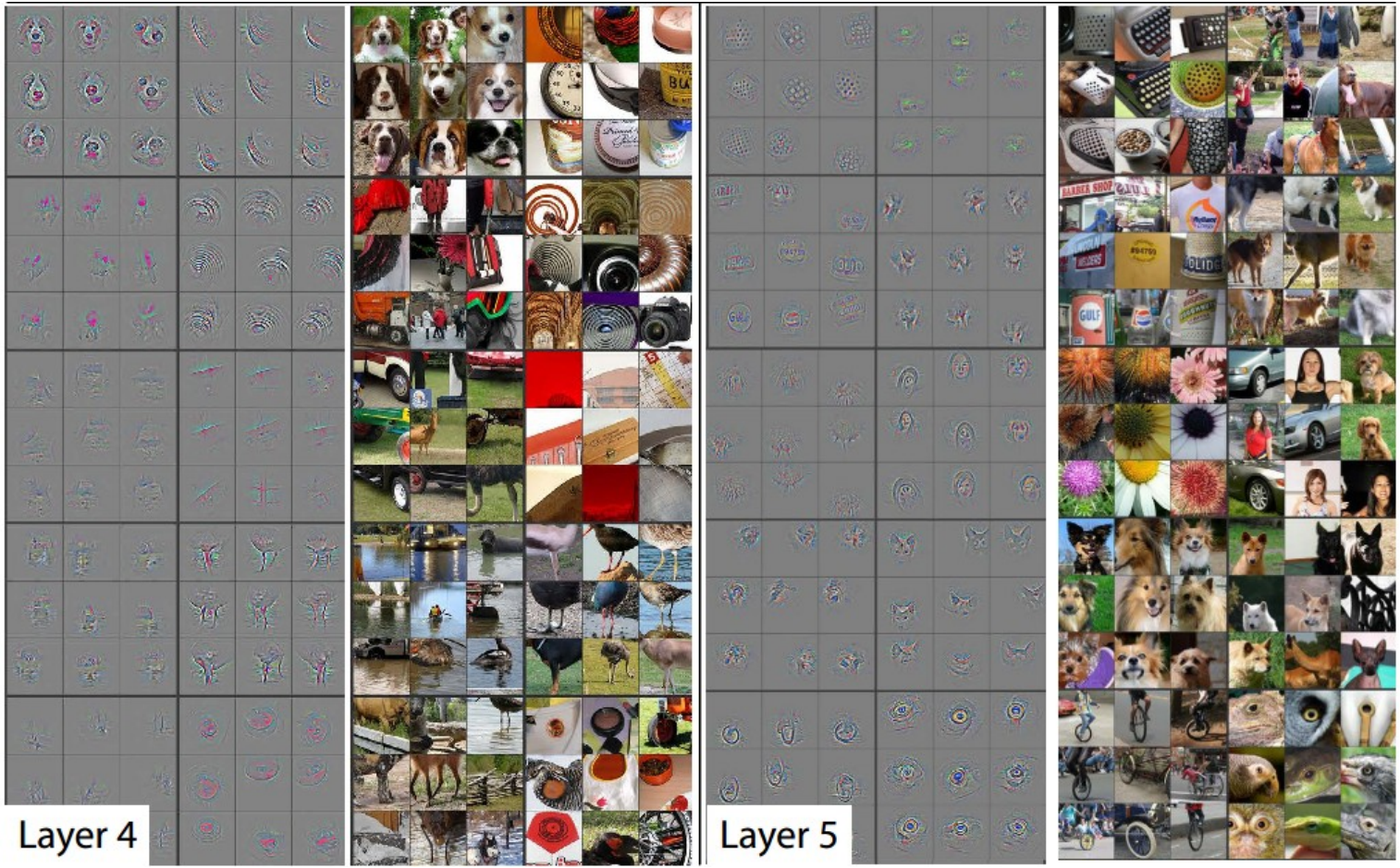
Visualizing arbitrary neurons along the way to the top...

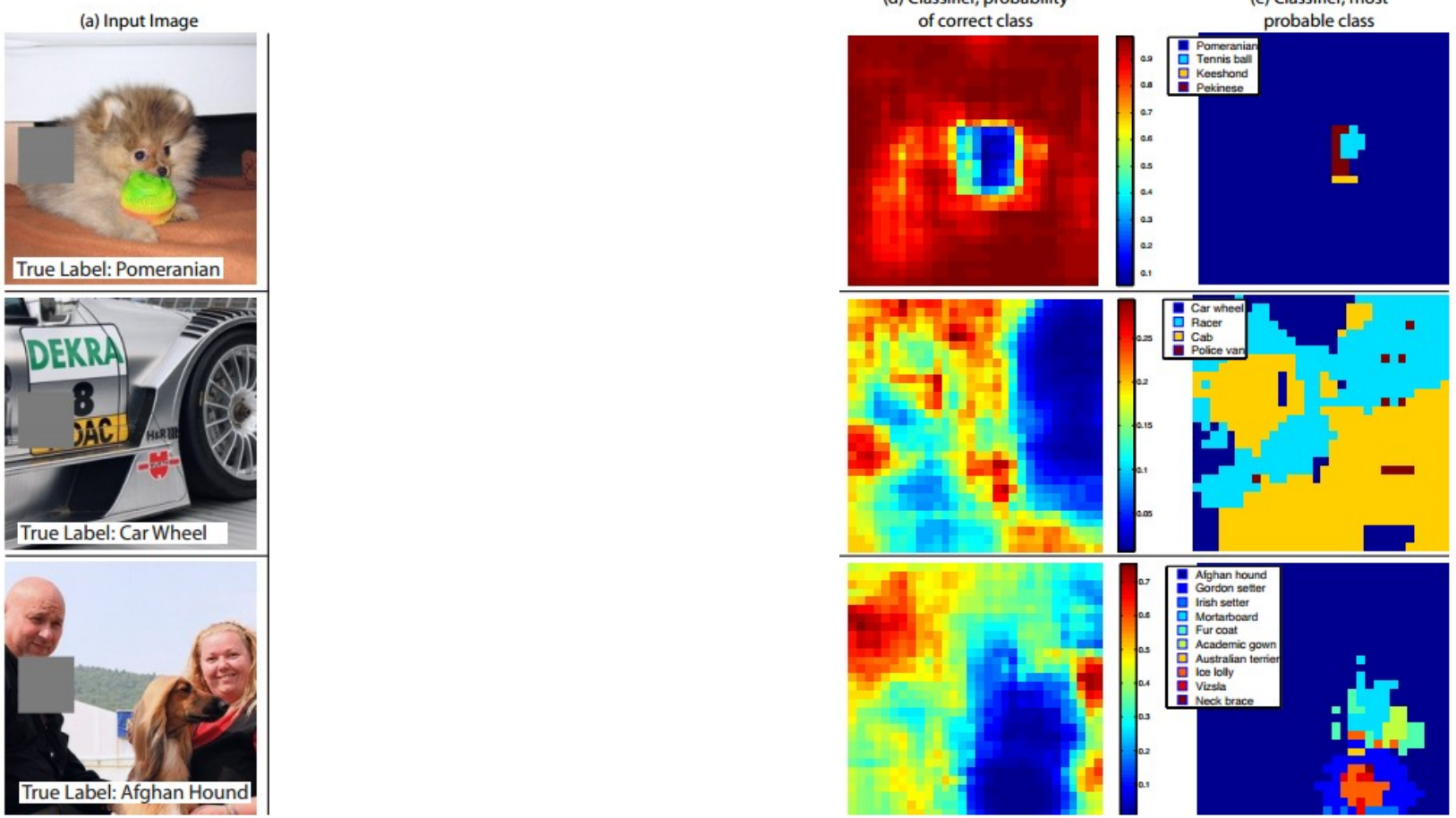


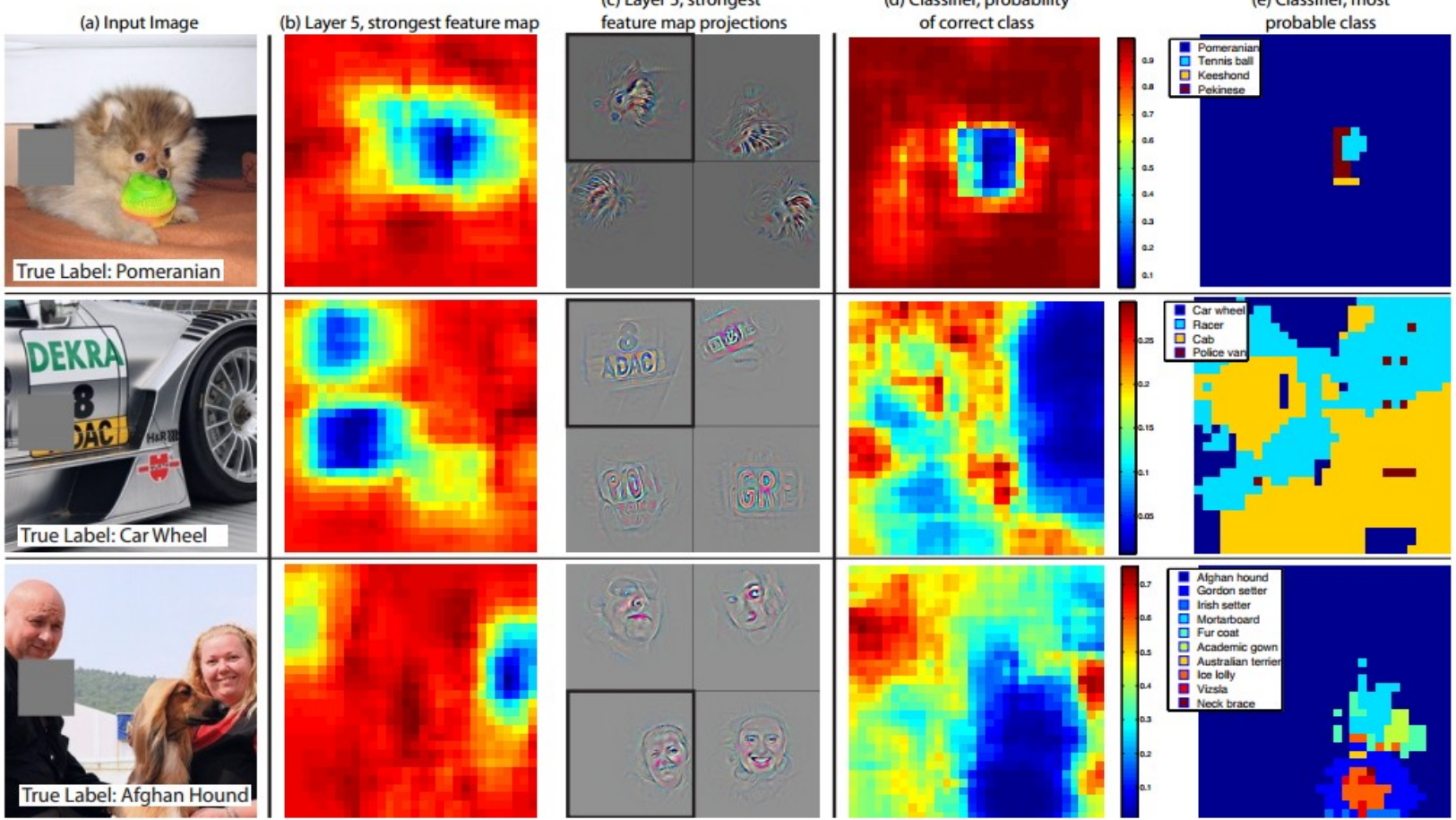
Visualizing arbitrary neurons along the way to the top...



Visualizing arbitrary neurons along the way to the top...



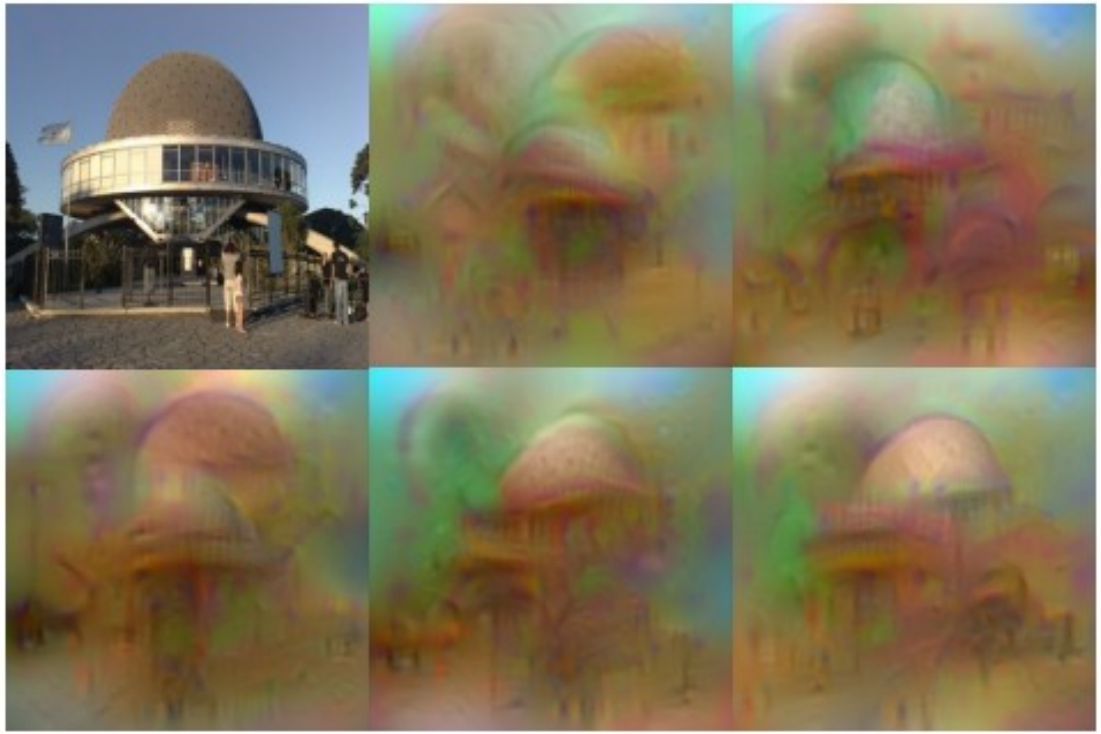

(a) Input Image
(d) Classifier, probability of correct class
(e) Classifier, most probable class
True Label: Pomeranian
Pomeranian
Tennis ball
Keeshond
Pekinese
True Label: Car Wheel
Car wheel
Racer
Cab
Police van
True Label: Afghan Hound
Afghan hound
Gordon setter
Irish setter
Mortarboard
Fur coat
Academic gown
Australian terrier
Ice lolly
Vizsla
Neck brace


(a) Input Image
(b) Layer 5, strongest feature map
(c) Layer 5, strongest feature map projections
(d) Classifier, probability of correct class
(e) Classifier, most probable class
True Label: Pomeranian
Pomeranian
Tennis ball
Keeshond
Pekinese
True Label: Car Wheel
Car wheel
Racer
Cab
Police van
True Label: Afghan Hound
Afghan hound
Gordon setter
Irish setter
Mortarboard
Fur coat
Academic gown
Australian terrier
Ice lolly
Vizsla
Neck brace

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

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

original image



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

Find an image such that:

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

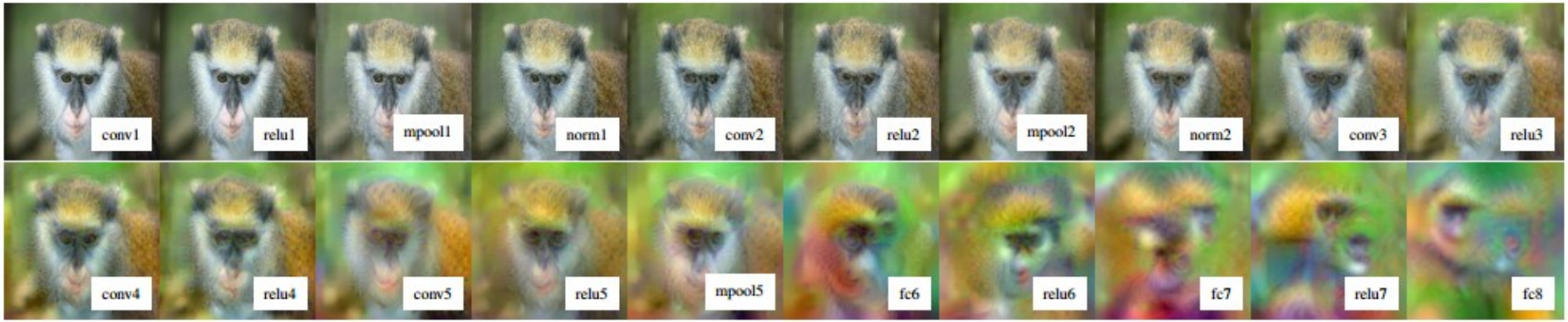
$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\operatorname{argmin}} \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

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

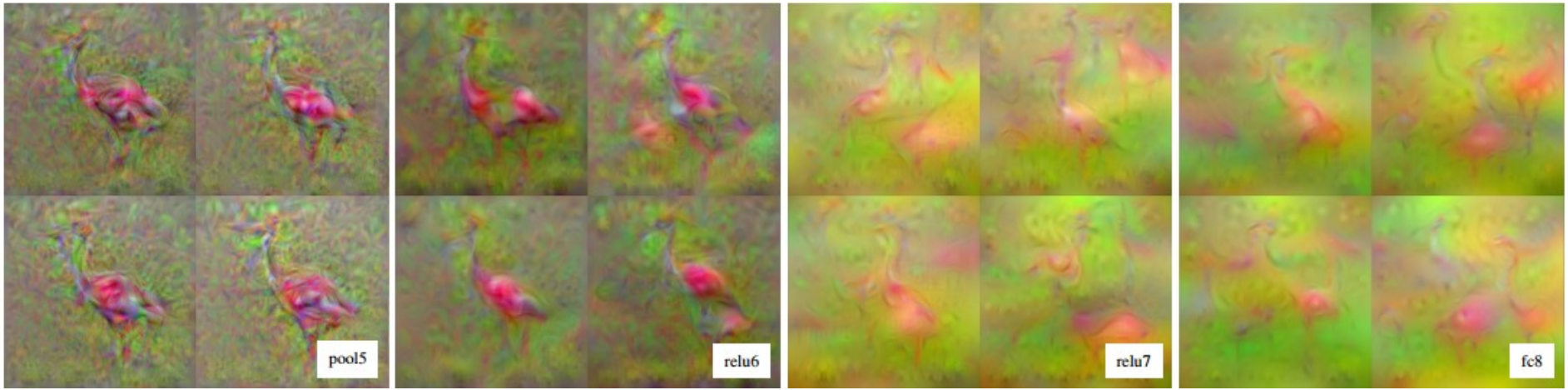
Solve using SGD + Momentum

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)



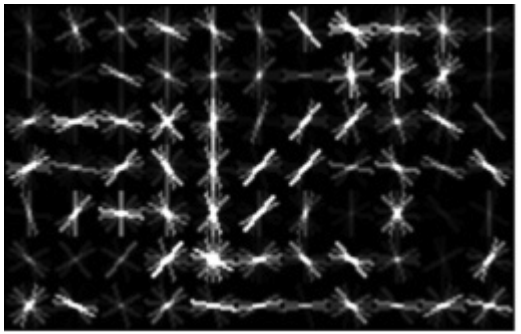
We can pose an optimization over the input image to maximize any class score. That seems useful.

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

Intriguing properties of neural networks
[Szegedy et al.]

These kinds of results were around even before ConvNets…

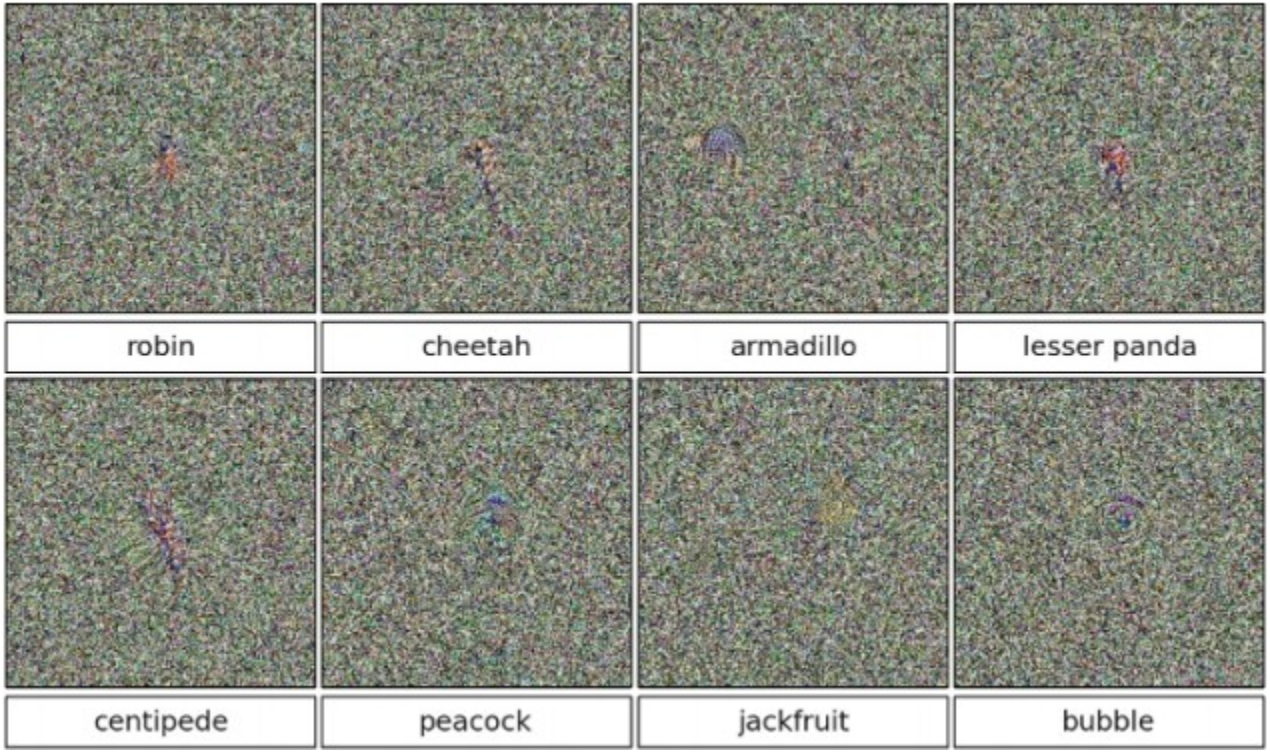
Exploring the Representation Capabilities of the HOG Descriptor
[Tatu et al., 2011]



Identical HOG represention

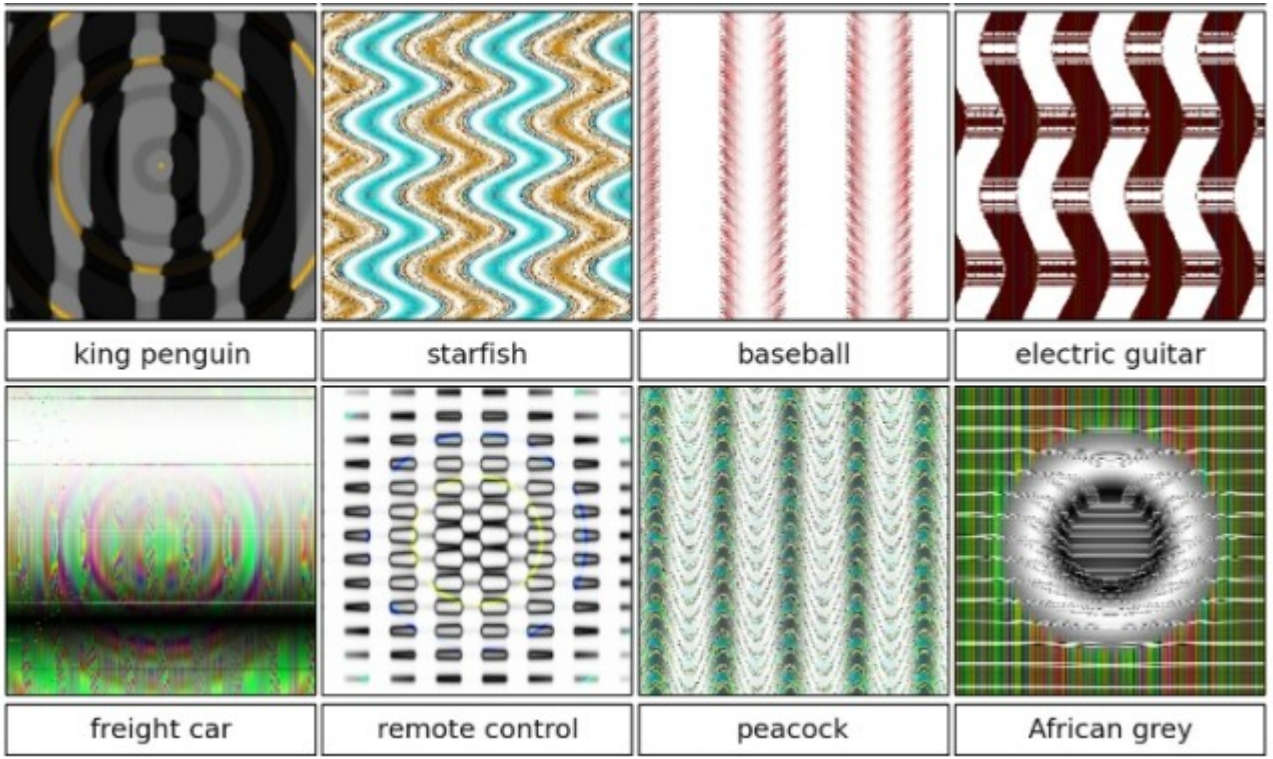
Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images [Nguyen, Yosinski, Clune]

>99.6% confidences



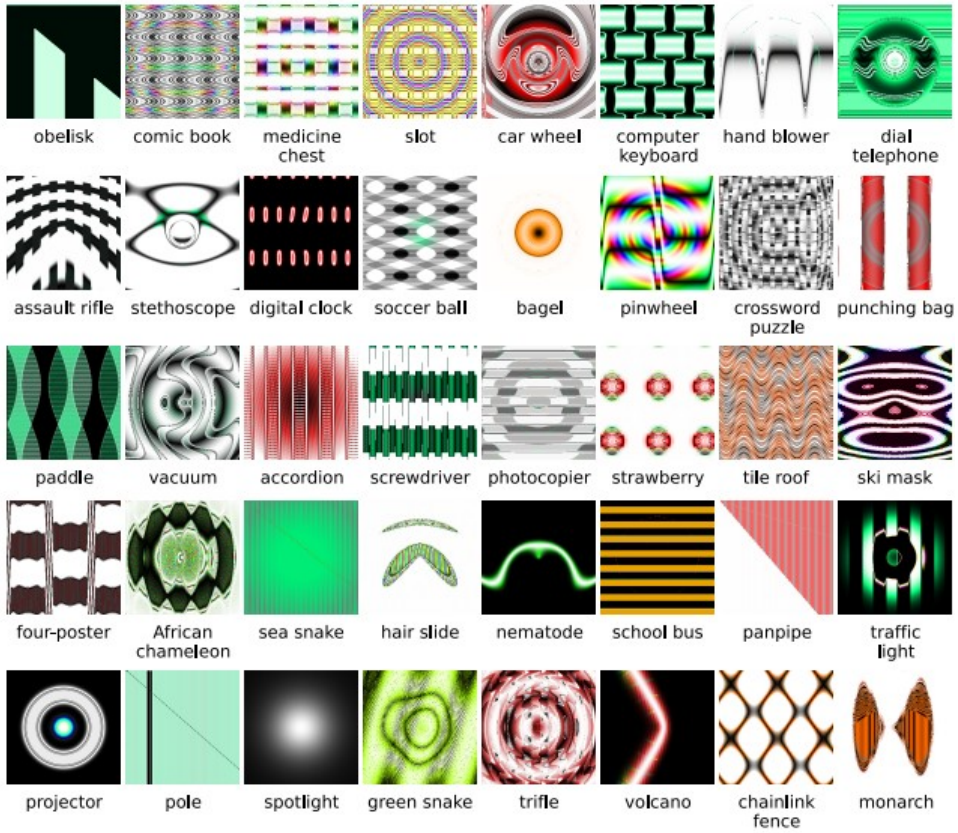
Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images [Nguyen, Yosinski, Clune]

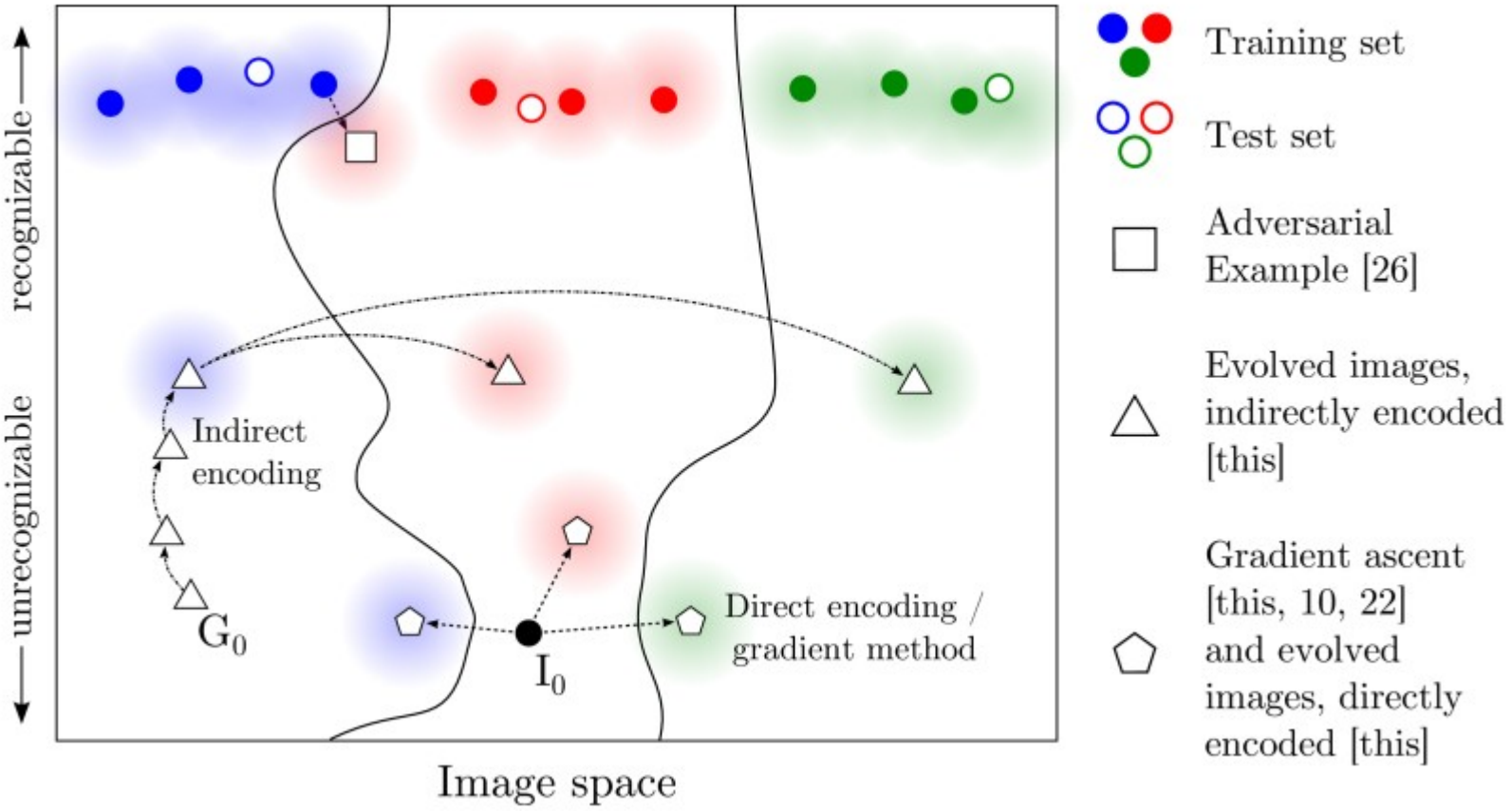
>99.6% confidences



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

>99.12% confidences



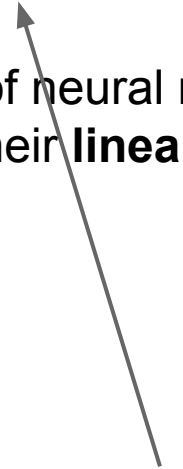

recognizable
unrecognizable
Indirect encoding
G_0
I_0
Direct encoding / gradient method
Image space
Training set
Test set
Adversarial Example [26]
Evolved images, indirectly encoded [this]
Gradient ascent [this, 10, 22] and evolved images, directly encoded [this]

EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES
[Goodfellow, Shlens & Szegedy, 2014]

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

EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES
*[Goodfellow, **Shlens** & Szegedy, 2014]*

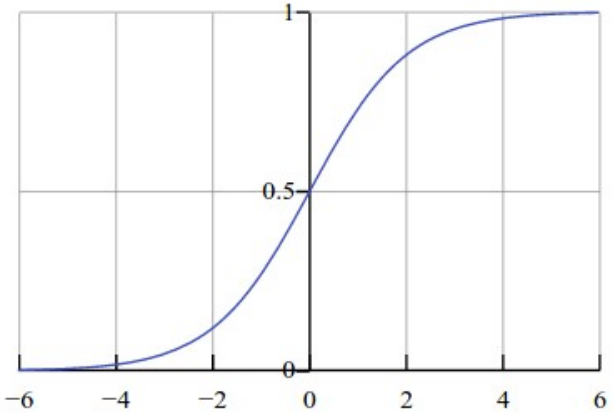
"primary cause of neural networks' vulnerability to adversarial perturbation is their **linear nature**"



(btw Jon Shlens is coming to give a talk in this class on March 2nd)

Lets fool a binary linear classifier: (logistic regression)

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



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

Lets fool a binary linear classifier:

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

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

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1
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
=> 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	?	?	?	?	?	?	?	?	?	?	

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
w	-1	-1	1	-1	1	-1	1	1	-1	1
adversarial x	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5

← input example

← weights

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%

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

Lets fool a binary linear classifier:

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

← input example

← weights

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%

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

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

EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES
[Goodfellow, Shlens & Szegedy, 2014]

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

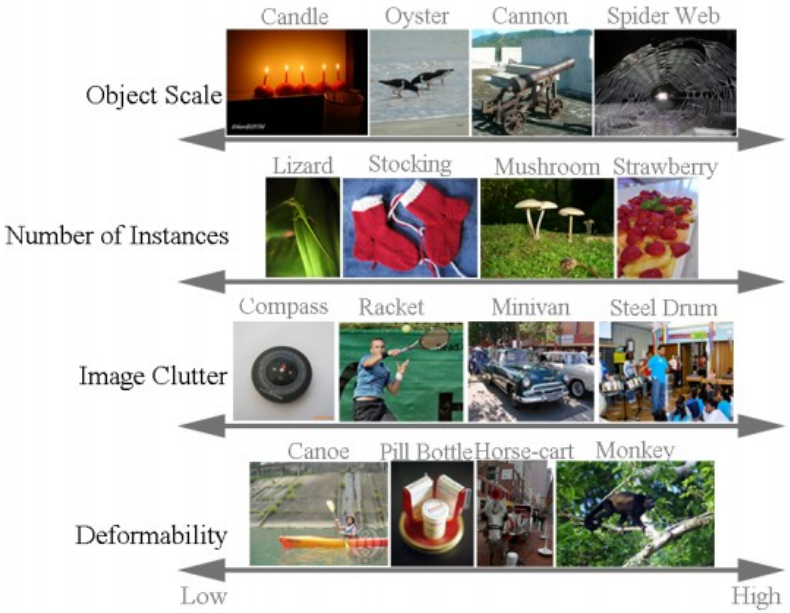
In particular, this is not a problem with Deep Learning, and has little to do with ConvNets specifically. Same issue would come up with Neural Nets in any other modalities.

Question: When does CNN work well and when does it not?

ImageNet (ILSVRC competition) analysis

1. Detecting avocados to zucchinis: what have we done, and where are we going?
2. ImageNet Large Scale Visual Recognition Challenge

[Olga Russakovsky et al.]



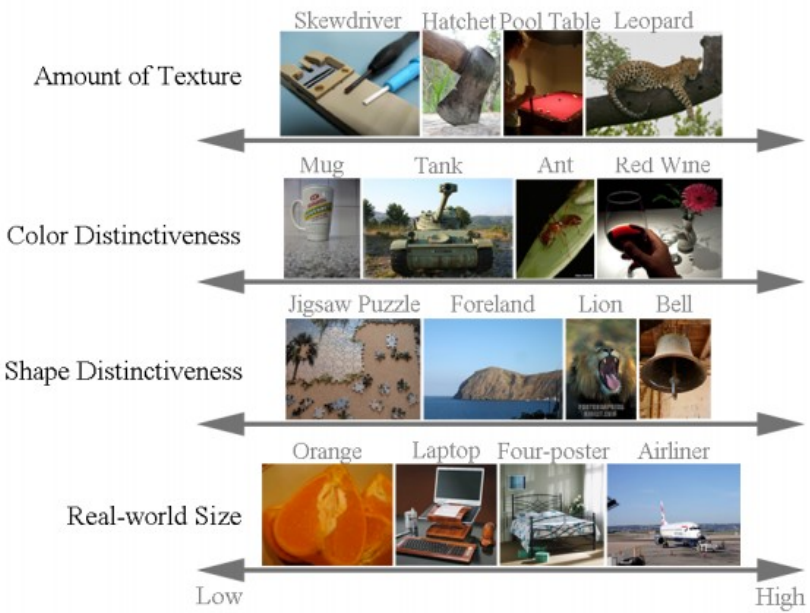


Image classification

Easiest classes

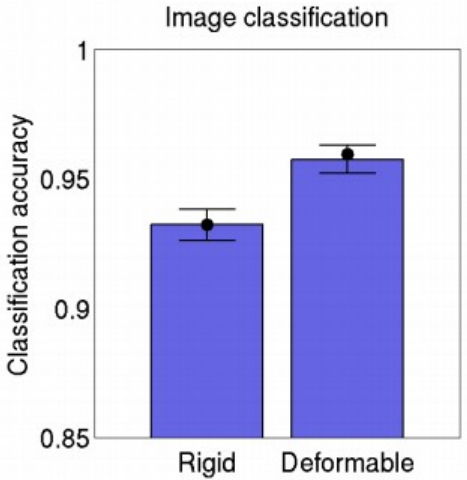
red fox (100) hen-of-the-woods (100) ibex (100) goldfinch (100) flat-coated retriever (100)

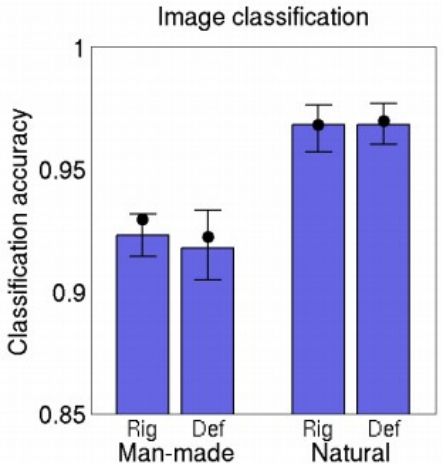
tiger (100) hamster (100) porcupine (100) stingray (100) Blenheim spaniel (100)

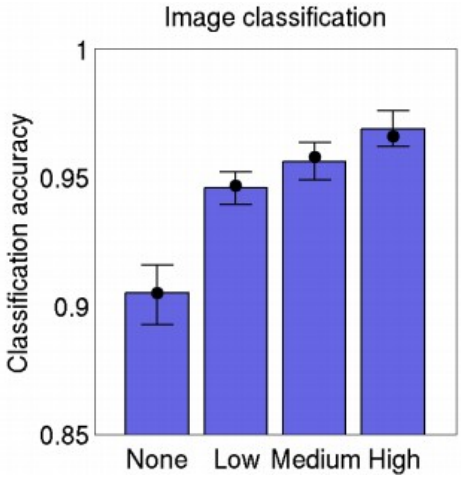
Hardest classes

muzzle (71) hatchet (68) water bottle (68) velvet (68) loupe (66)

hook (66) spotlight (66) ladle (65) restaurant (64) letter opener (59)





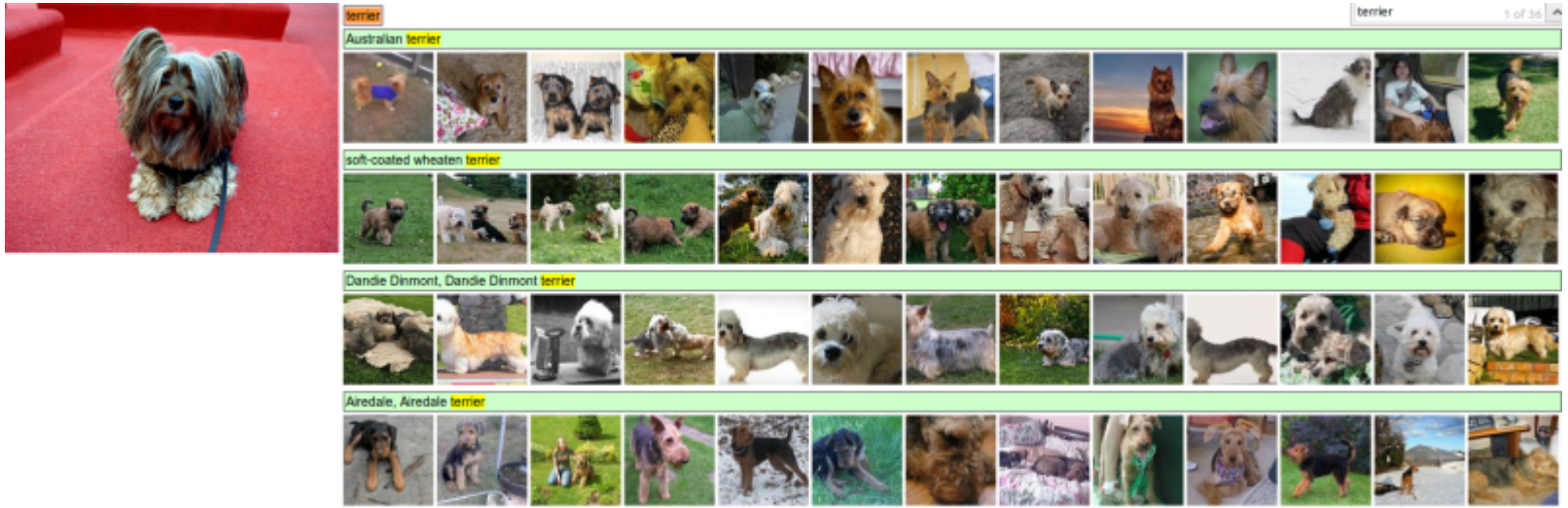


(Amount of texture)

CNN vs. Human

[What I learned from competing against a ConvNet on ImageNet]
Karpathy, 2014: http://bit.ly/humanvsconvnet

Try it out yourself: http://cs.stanford.edu/people/karpathy/ilsvrc/


terrier
terrier
Australian terrier
soft-coated wheaten terrier
Dandie Dinmont, Dandie Dinmont terrier
Airedale, Airedale terrier

:’(

	GoogLeNet correct	GoogLeNet wrong
Human correct	1352/1500 	72/1500 • Objects very small or thin • Abstract representations • Image filters
Human wrong	46/1500 • Fine-grained recognition • Class unawareness • Insufficient training data	30/1500 • Multiple objects • Incorrect annotations

GoogLeNet: 6.8%
Andrej: 5.1% phew...

rule, ruler	king crab, Alaska crab	sidewinder	saltshaker, salt shaker	reel	hatchet	schipperke
pencil box, pencil case	pizza, pizza pie	maze, labyrinth	pill bottle	stethoscope	vase	schipperke
rubber eraser, rubber	strawberry	gar, garfish	water bottle	whistle	pitcher, ewer	groenendael
ballpoint, ballpoint pen	orange	valley, vale	lotion	ice lolly, lolly	coffeepot	doormat, welcome mat
pencil sharpener	fig	hammerhead	hair spray	hair spray	mask	teddy, teddy bear
carpenter's kit, tool kit	ice cream, icecream	sea snake	beer bottle	maypole	cup	jigsaw puzzle

In Summary:

- We looked at several works that try to visualize how ConvNets work and what they learn

- We saw that you can “break them”, but this is not a problem with deep learning (in fact, DL will be the solution), and has little to do with Computer Vision or ConvNets. It’s a problem with the mathematical forms we use in forward pass and training objective.

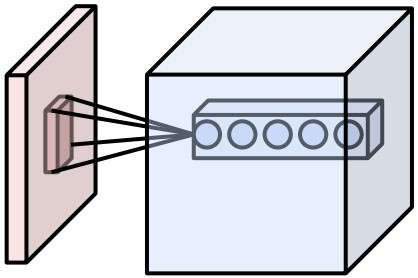
- We looked at where ConvNets work and don’t work

Next Lecture:

Transfer Learning and Finetuning ConvNets

A single neuron is not distinguished in any way. Instead, it's just one of the axes in a representation space.

Intriguing properties of neural networks [Szegedy et al.]



(a) Unit sensitive to white flowers.

(b) Unit sensitive to postures.

(c) Unit senstive to round, spiky flowers.

(d) Unit senstive to round green or yellow objects.

Figure 3: Experiment performed on ImageNet. Images stimulating single unit most (maximum stimulation in natural basis direction). Images within each row share many semantic properties.

(a) Direction sensitive to white, spread flowers.

(b) Direction sensitive to white dogs.

(c) Direction sensitive to spread shapes.

(d) Direction sensitive to dogs with brown heads.

Figure 4: Experiment performed on ImageNet. Images giving rise to maximum activations in a random direction (maximum stimulation in a random basis). Images within each row share many semantic properties.