CS 4803 / 7643: Deep Learning

Topics:

– Visualizing CNNs

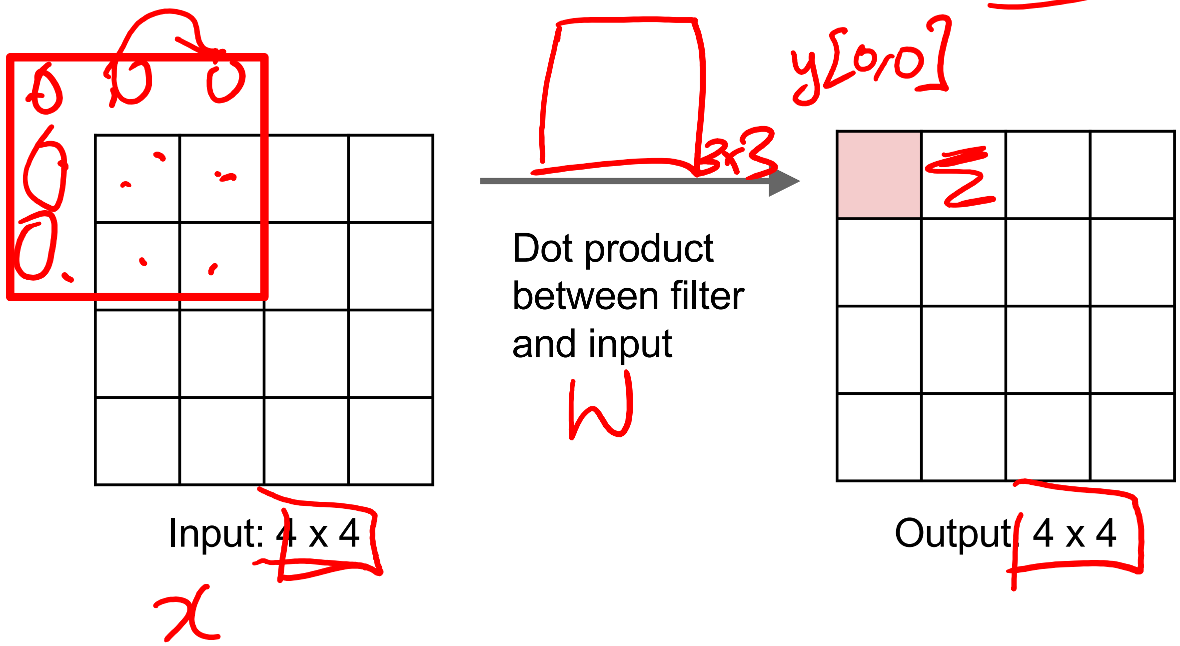
Dhruv Batra

Georgia Tech

Recap from last time

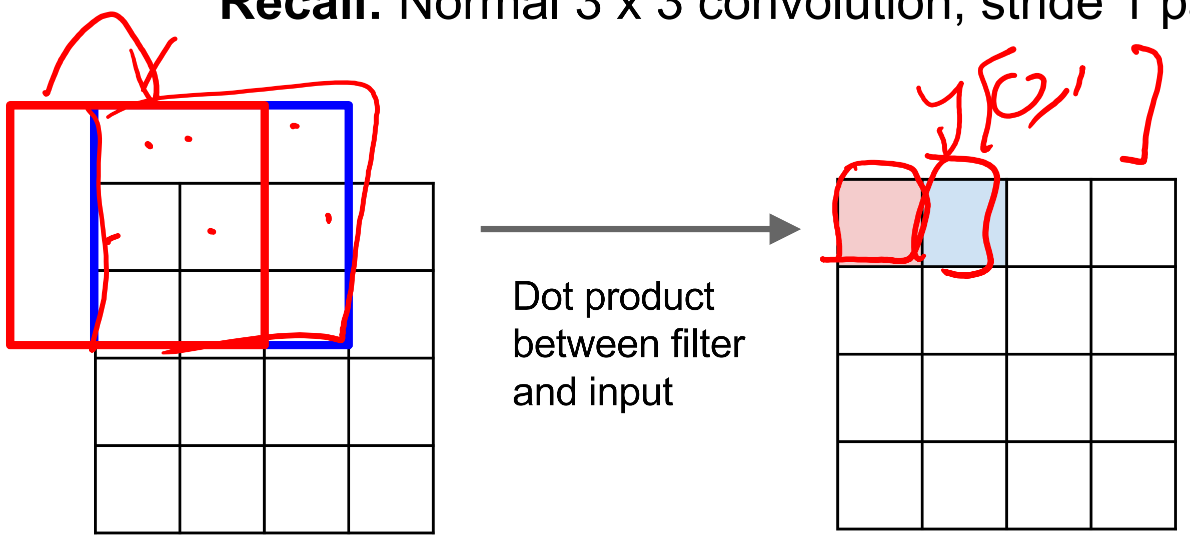
Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1



Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1

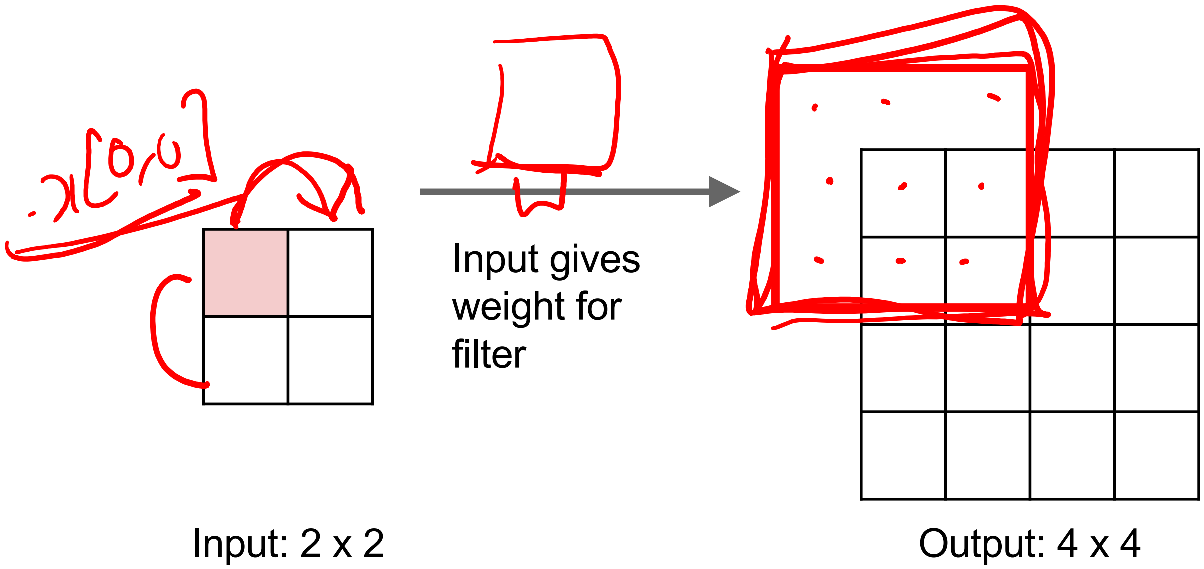


Input: 4 x 4

Output: 4 x 4

Learnable Upsampling: Transpose Convolution

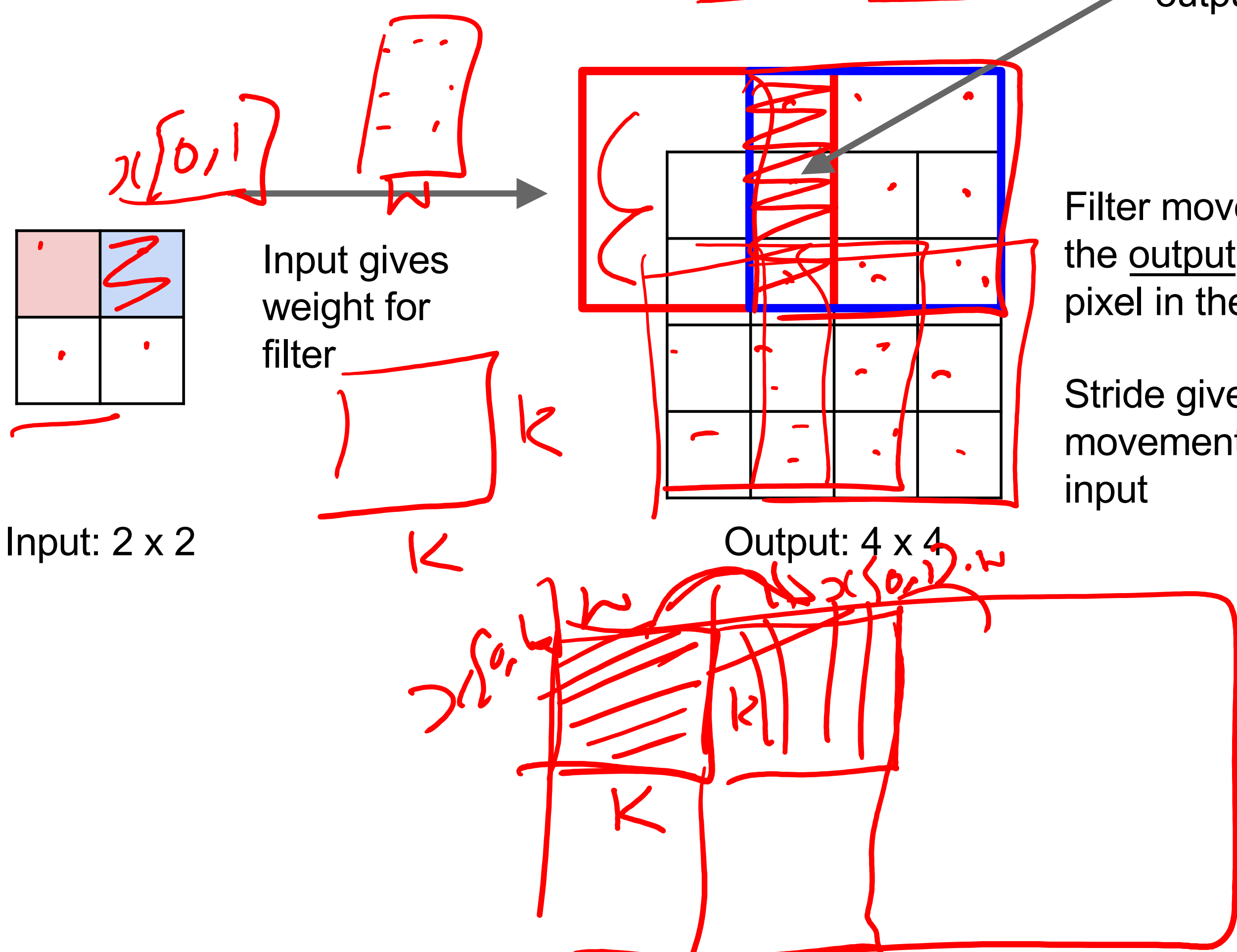
3 x 3 **transpose** convolution, stride 2 pad 1



Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1

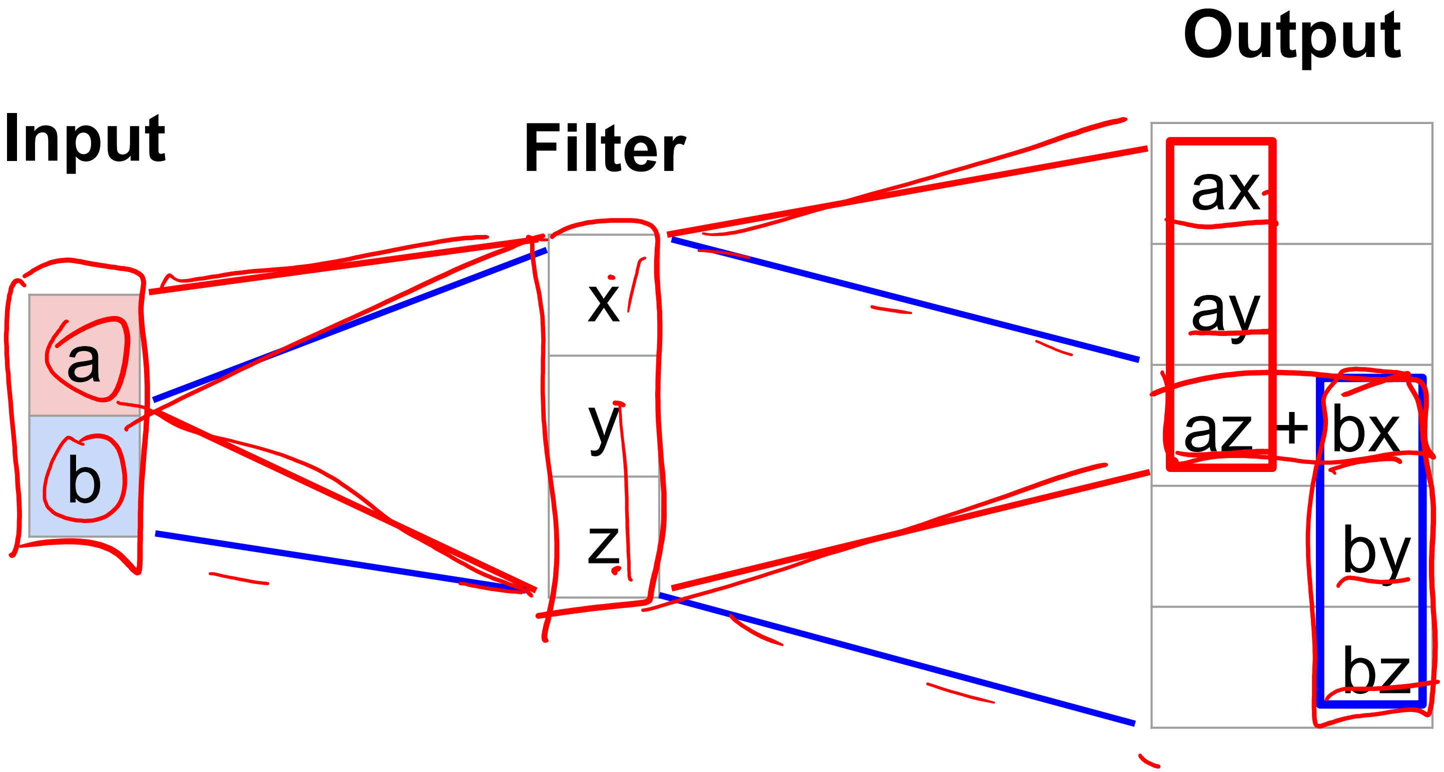
Sum where output overlaps



Filter moves 2 pixels in the output for every one pixel in the input

Stride gives ratio between movement in output and input

Transpose Convolution: 1D Example



Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Need to crop one pixel from output to make output exactly 2x input

In-Network upsampling: “Unpooling”

stride=2

Nearest Neighbor

$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

1	2
3	4

Input: 2 x 2

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

Output: 4 x 4

“Bed of Nails”

$\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$

1	2
3	4

Input: 2 x 2

1	0	2	0
0	0	0	0
3	0	4	0
0	0	0	0

Output: 4 x 4

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

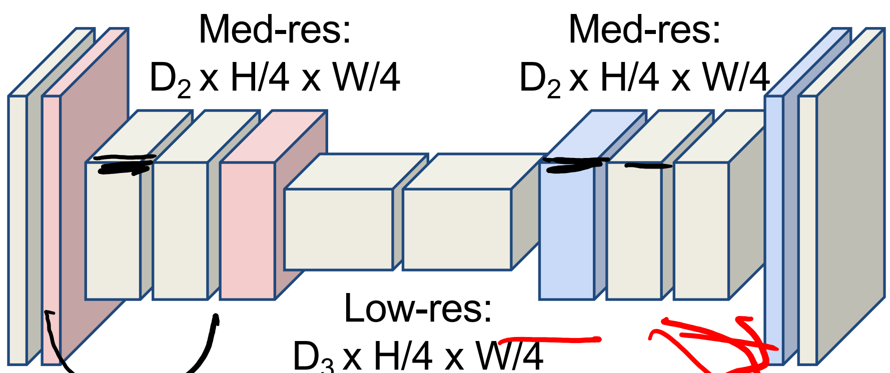
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
Unpooling or strided transpose convolution

Input:
$3 \times H \times W$



Predictions:
$H \times W$

Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015
Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Why this operation?

What is deconvolution?

- (Non-blind) Deconvolution

$y = x * w$

$\vec{y} = W \vec{x}$

What is deconvolution?

$\vec{y} = W\vec{x}$

$\vec{w} = [-1 \ 0 \ +1]$

$W^T W = I$

- (Non-blind) Deconvolution

$$y = w * x$$

$$\begin{bmatrix} w_k & 0 & \dots & 0 & 0 \\ w_{k-1} & w_k & \dots & 0 & 0 \\ w_{k-2} & w_{k-1} & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_1 & w_{k-2} & \dots & w_k & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & w_1 & \dots & w_{k-1} & w_k \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \vdots & w_1 & w_2 \\ 0 & 0 & \vdots & 0 & w_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

$W^{-1} = W$

$\vec{y} = W\vec{x}$

$\tilde{x} = W^T y$

What does “deconvolution” have to do with “transposed convolution”?

“transposed convolution” is a convolution!

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

"transposed convolution" is a convolution!

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

[x, y, z]

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

[z, y, x]

“transposed convolution” is a convolution!

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T\vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

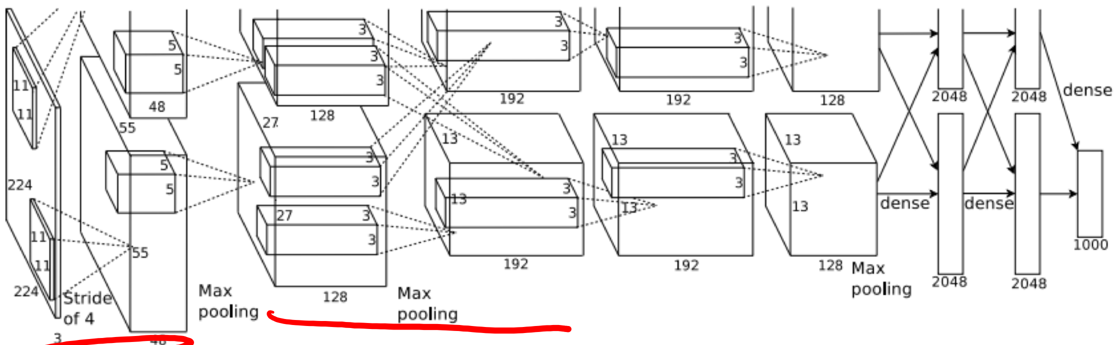
Plan for Today

- Visualizing CNNs
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations
  - How to evaluate visualizations?
  - Creating “prototypical” images for a class
  - Creating adversarial images
  - Deep dream
  - Feature inversion

What's going on inside ConvNets?

This image is CC0 public domain

Input Image:
3 x 224 x 224



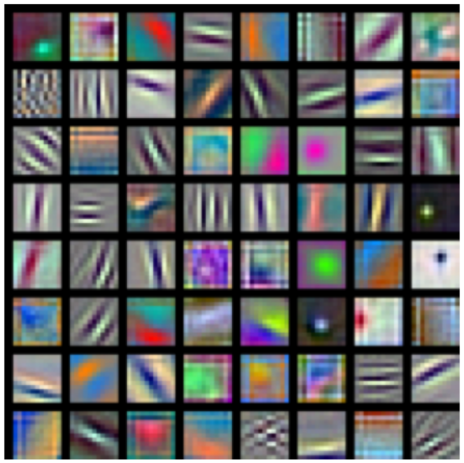
Class Scores:
1000 numbers

What are the intermediate features looking for?

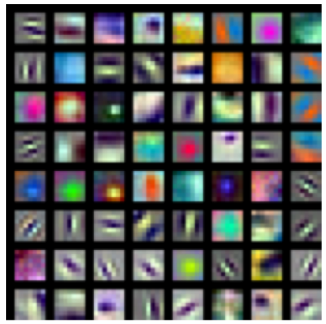
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figure reproduced with permission.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

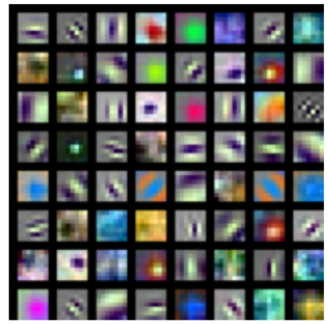
First Layer: Visualize Filters



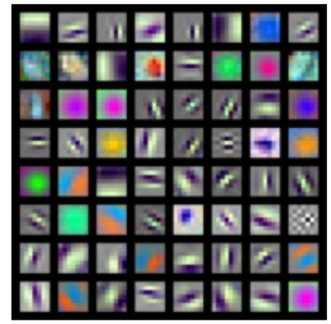
AlexNet:
64 x 3 x 11 x 11



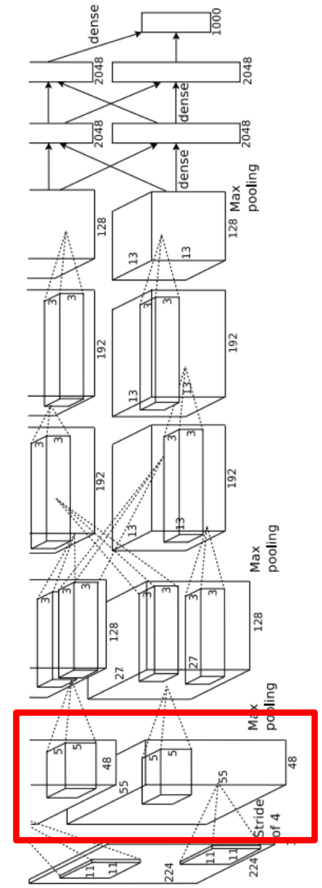
ResNet-18:
64 x 3 x 7 x 7



ResNet-101:
64 x 3 x 7 x 7



DenseNet-121:
64 x 3 x 7 x 7



Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

Visualize the filters/kernels (raw weights)

We can visualize filters at higher layers, but not that interesting

(these are taken from ConvNetJS CIFAR-10 demo)

layer 1 weights

$\boxed{16} \times 3 \times 7 \times 7$

layer 2 weights

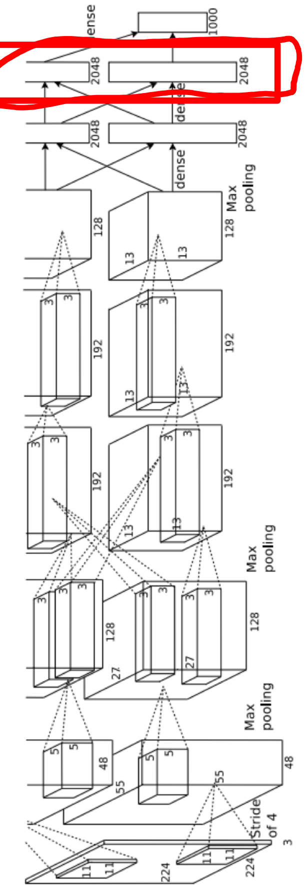
$20 \times 16 \times 7 \times 7$

layer 3 weights

$20 \times 20 \times 7 \times 7$

Last Layer

FC7 layer



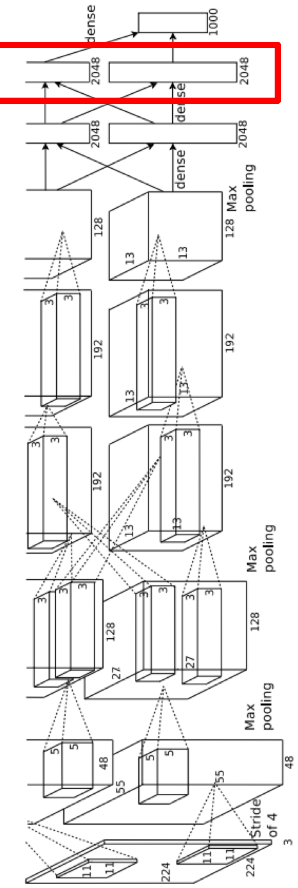
4096-dimensional feature vector for an image
(layer immediately before the classifier)

Run the network on many images, collect the feature vectors

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

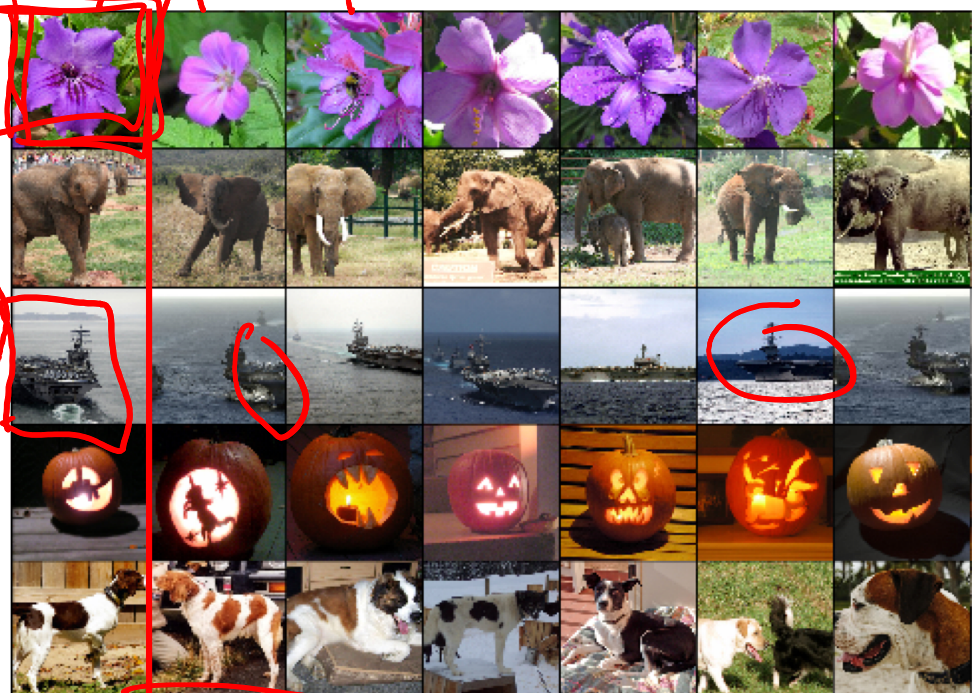
Last Layer: Nearest Neighbors

4096-dim vector

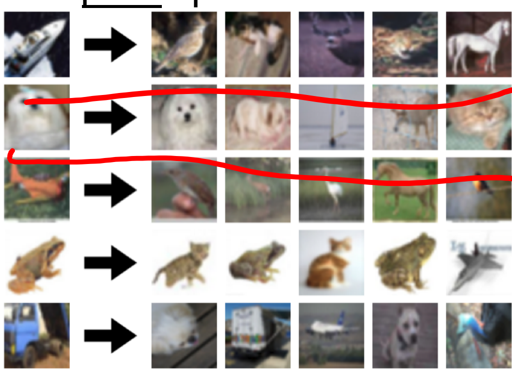


Test image

L2 Nearest neighbors in feature space



Recall: Nearest neighbors in pixel space



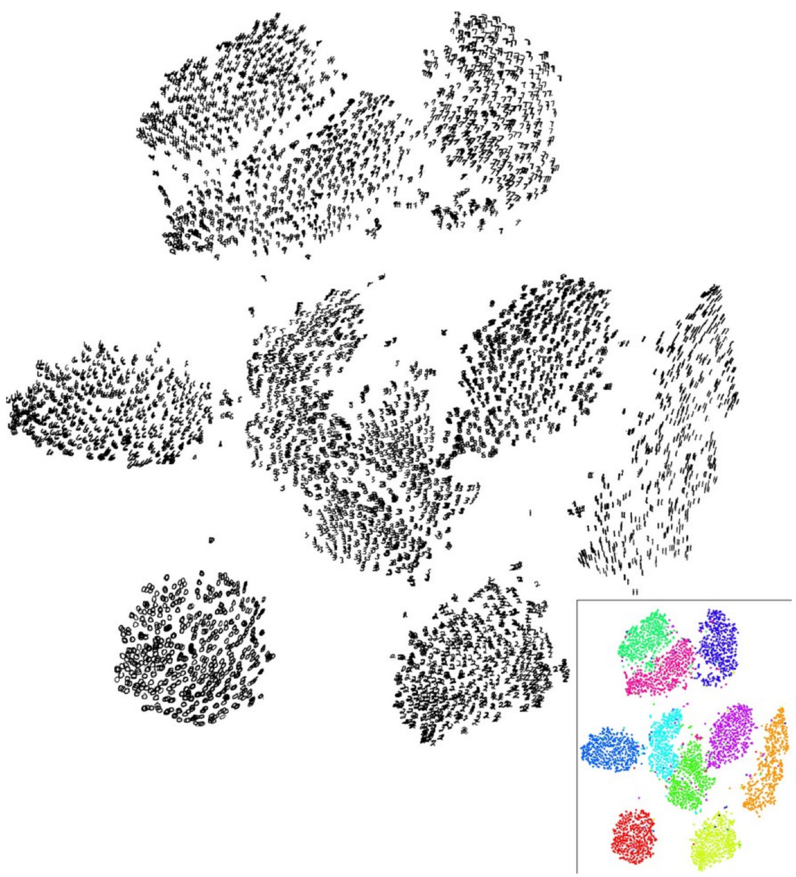
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figures reproduced with permission.

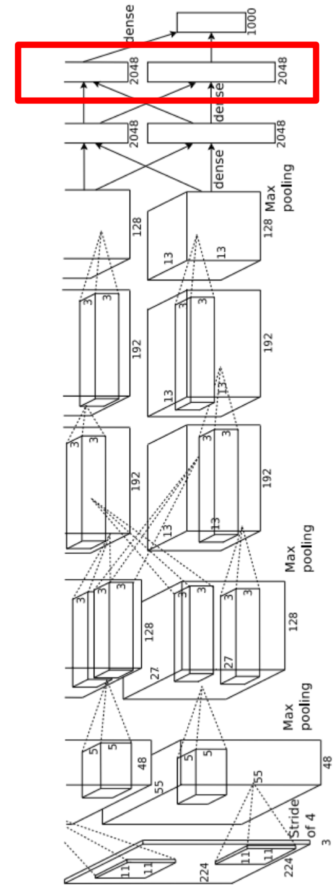
Last Layer: Dimensionality Reduction

Visualize the "space" of FC7 feature vectors by reducing dimensionality of vectors from 4096 to 2 dimensions

Simple algorithm: Principal Component Analysis (PCA)

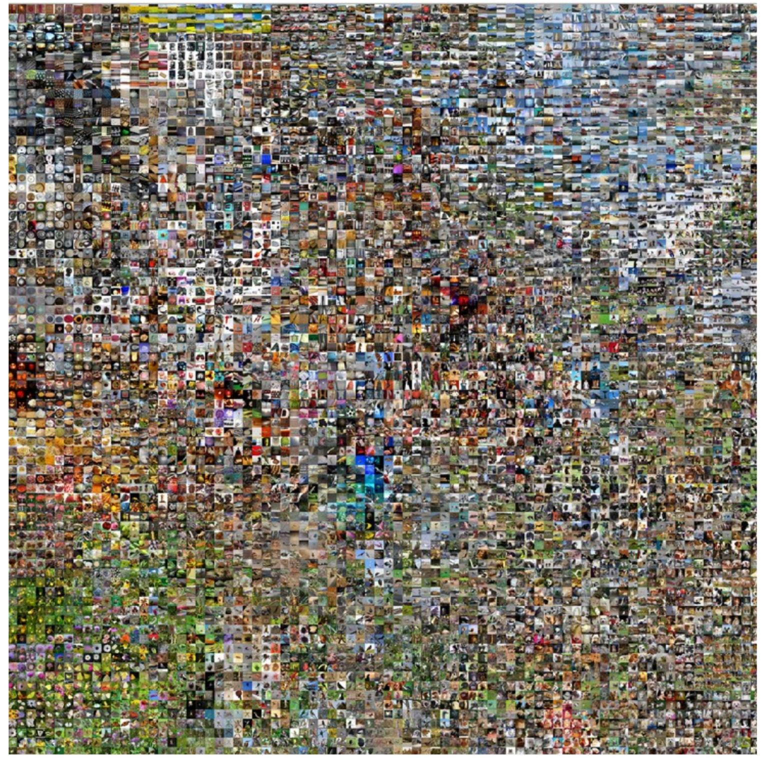
More complex: **t-SNE**

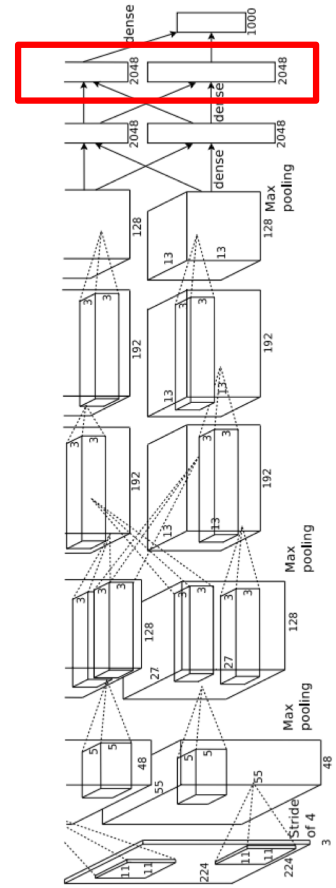




Van der Maaten and Hinton, "Visualizing Data using t-SNE", JMLR 2008

Last Layer: Dimensionality Reduction

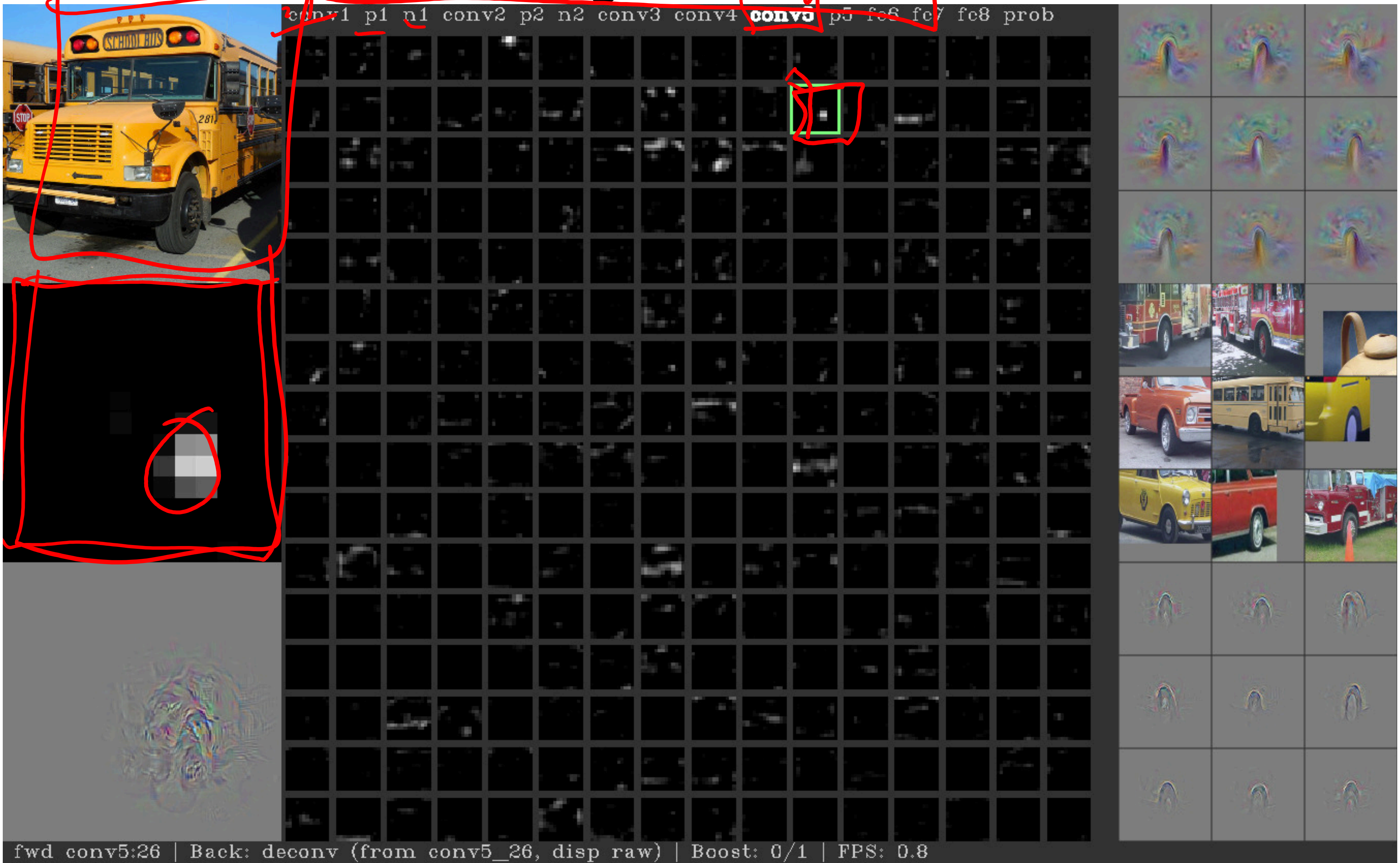




Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008
Krizhevsky et al, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012.
Figure reproduced with permission.

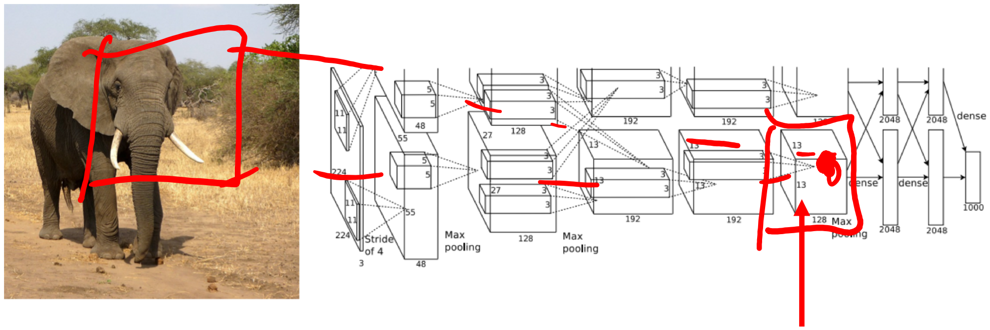
See high-resolution versions at
http://cs.stanford.edu/people/karpathy/cnnembed/

Visualizing Activations



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Maximally Activating Patches



Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick channel 17/128

Run many images through the network, record values of chosen channel

Visualize image patches that correspond to maximal activations

Springenberg et al, “Striving for Simplicity: The All Convolutional Net”, ICLR Workshop 2015

Plan for Today

- Visualizing CNNs
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations
  - How to evaluate visualizations?
  - Creating “prototypical” images for a class
  - Creating adversarial images
  - Deep dream
  - Feature inversion

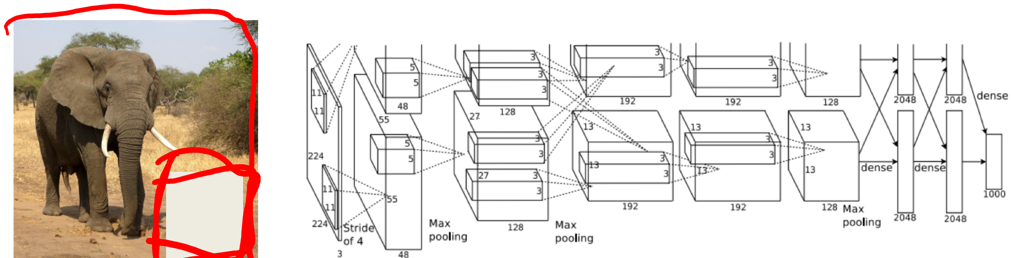
Visual Explanations

Where does an intelligent system "look" to make its predictions?

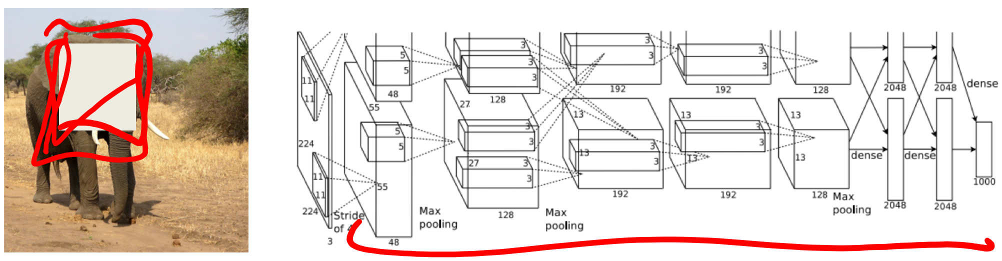
Which pixels matter: Occlusion Maps

0.99

Mask part of the image before feeding to CNN, check how much predicted probabilities change



P(elephant) = 0.95

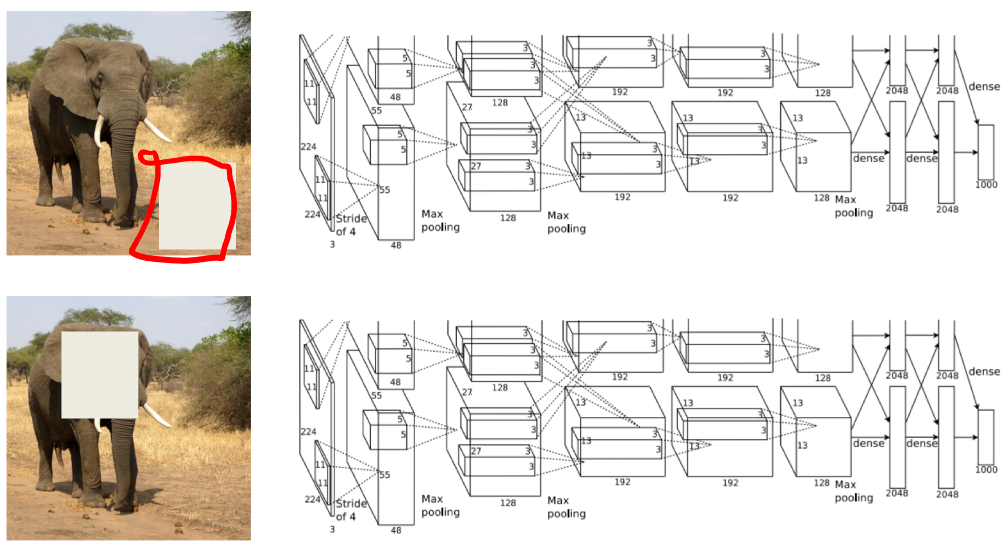


P(elephant) = 0.75

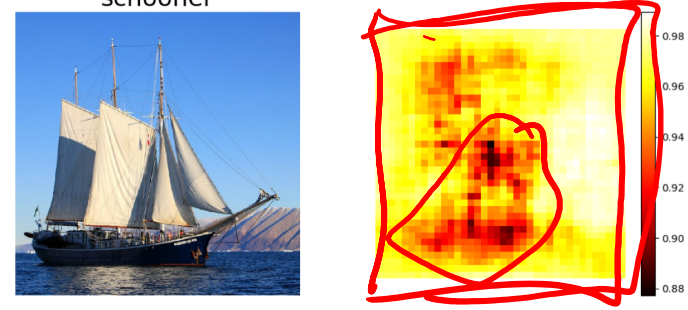
Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

Which pixels matter: Occlusion Maps

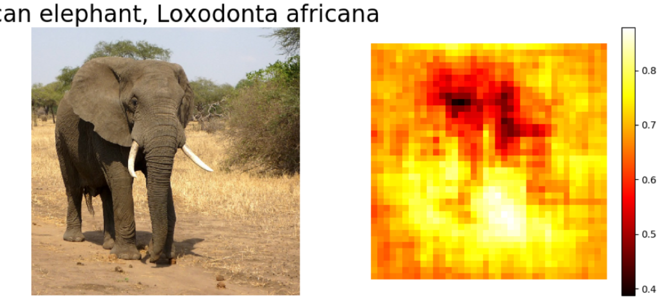
Mask part of the image before feeding to CNN, check how much predicted probabilities change



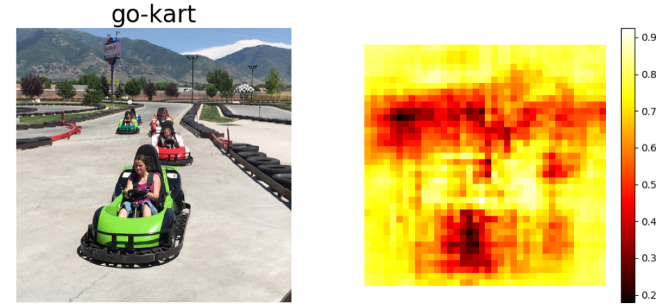
schooner



African elephant, Loxodonta africana



go-kart



Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Boat image is CC0 public domain
Elephant image is CC0 public domain
Go-Karts image is CC0 public domain

What if our model was linear?

$$\langle \mathbf{w}_c \, , \, \mathbf{x} \rangle + b = S_c(\mathbf{x})$$

What if our model was linear?

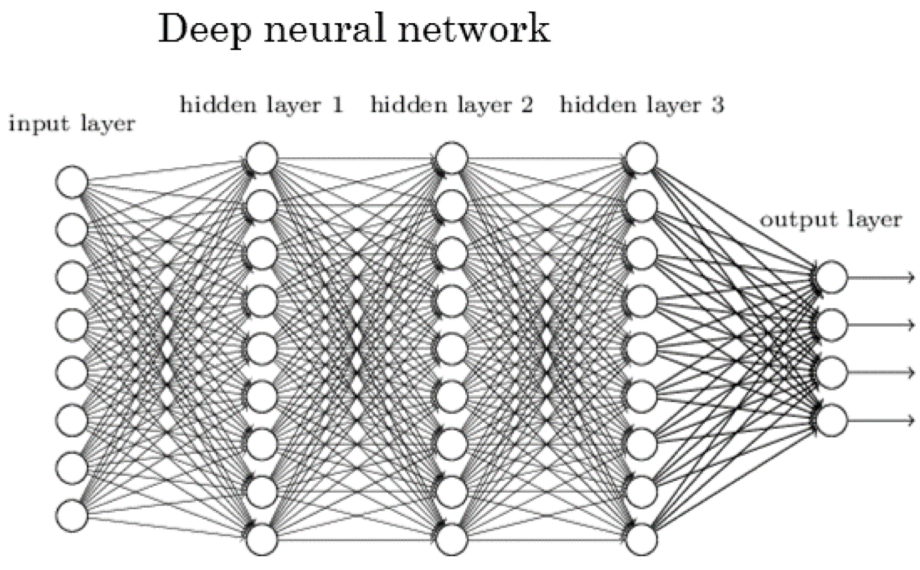
$$\left\langle \begin{bmatrix} \mathbf{100} \\ 0.1 \\ -0.1 \\ \mathbf{510} \\ \mathbf{-200} \end{bmatrix}, \begin{bmatrix} 1 \\ 0.9 \\ -0.2 \\ 0.5 \\ -0.9 \end{bmatrix} \right\rangle + b = S_c(\mathbf{x})$$

But it's not ☹

$$\langle \mathbf{w}_c , \mathbf{x} \rangle + b = S_c(\mathbf{x})$$

Can we make it linear?

$$\underbrace{f(\mathbf{x})} = \underbrace{S_c(\mathbf{x})}$$



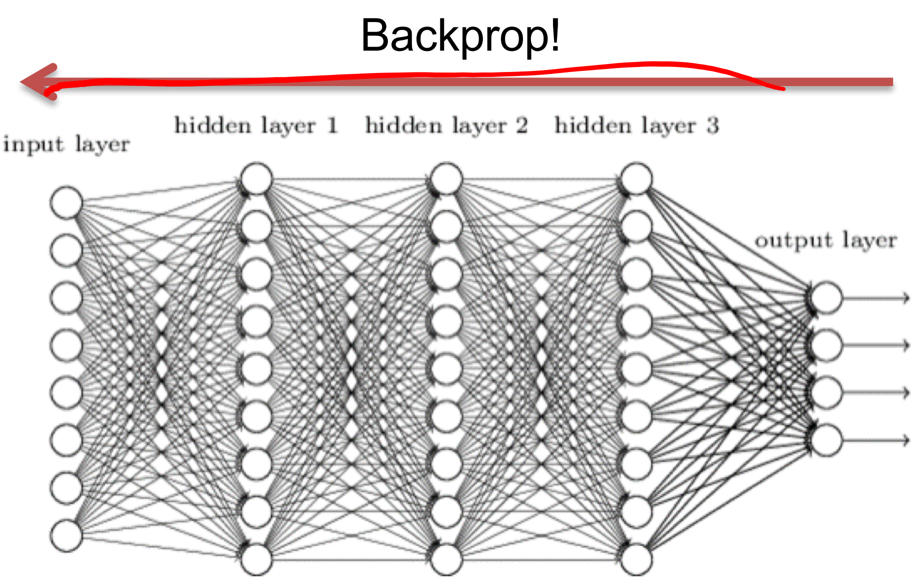
Taylor Series

$$\boxed{f(x)} \approx f(x_0) + f'(x_0)(x - x_0)$$

Feature Importance in Deep Models

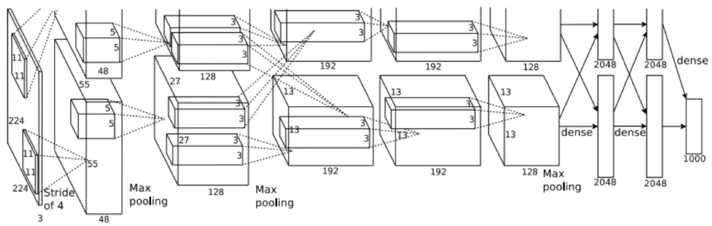
$$\mathbf{w}_c = \left.\boxed{\frac{\partial S_c}{\partial \mathbf{x}}}\right|_{\mathbf{x}_0}$$

$$\langle \boxed{\mathbf{w}_c}, \mathbf{x} \rangle + b \approx S_c(\mathbf{x})$$



Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities

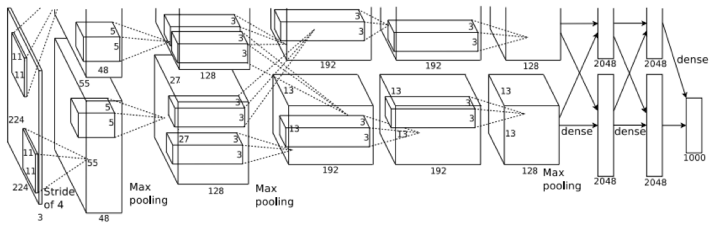


Dog

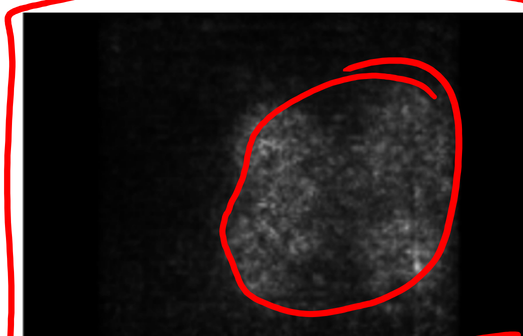
Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities



Dog



Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels

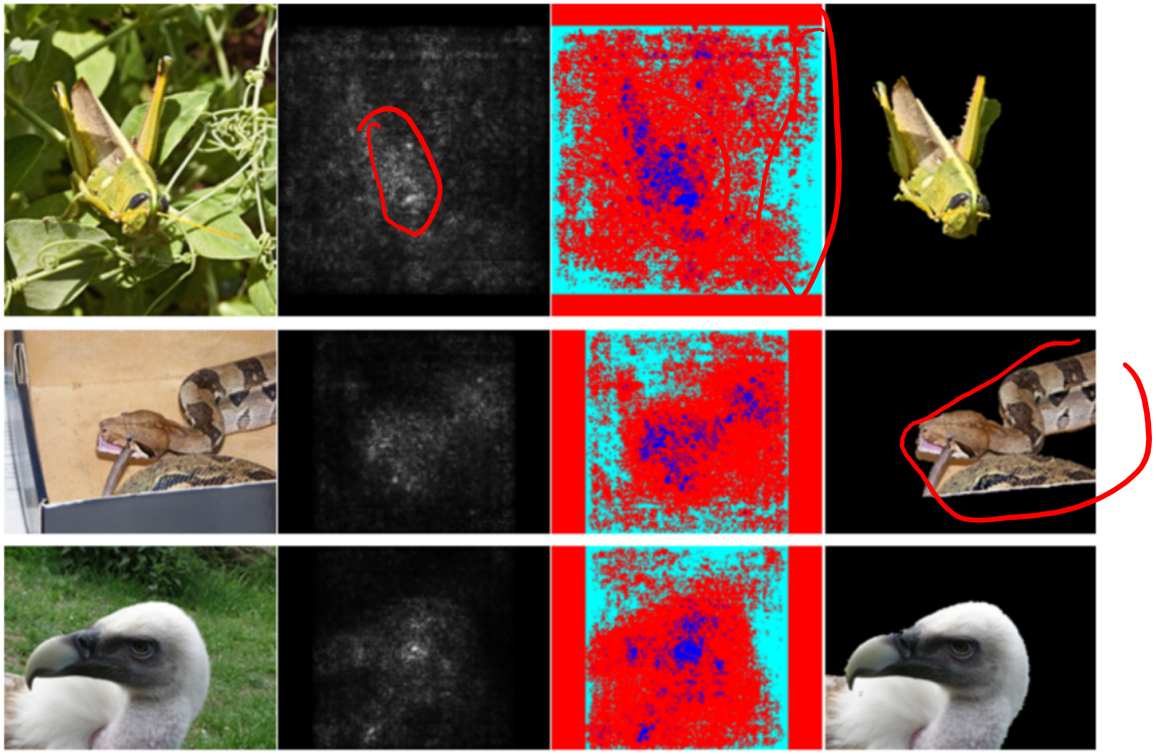
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Saliency Maps

Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

Saliency Maps: Segmentation without supervision

Use GrabCut on saliency map



Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Rother et al, "Grabcut: Interactive foreground extraction using iterated graph cuts", ACM TOG 2004

Gradient-based Visualizations

- Raw Gradients
  - [Simoyan et al. ICLRw '14]
- 'Deconvolution'
  - [Zeiler & Fergus, ECCV '14]
- Guided Backprop
  - [Springenber et al. ICLR '15]

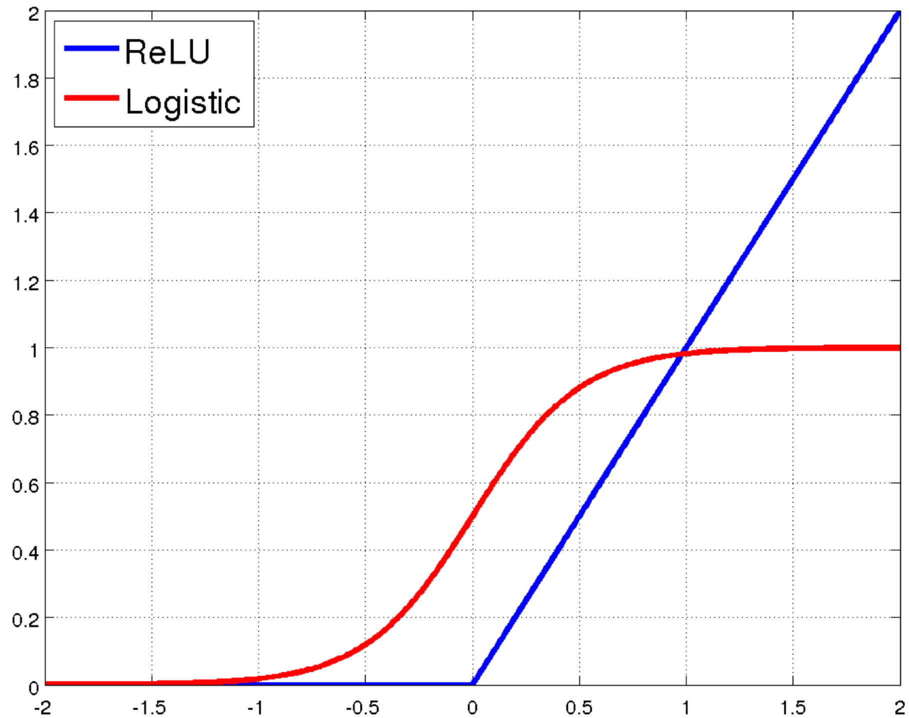
Identical for all layers except ReLU

$$\frac{\partial S_c}{\partial x}$$

Remember ReLUs?

$$h^{l+1} = \mathrm{ReLU}(h^l) = \max\{0, h^l\}$$

$$\frac{\partial h^{l+1}}{\partial h^l} = \begin{cases} \boxed{0} & \text{if } h^l < 0 \\ \boxed{1} & \text{if } h^l > 0 \end{cases} = \boxed{[[h^l > 0]]}$$



$$h^{l+1} = \max\{0, h^l\}$$

Forward pass

$$h^l \quad \begin{bmatrix} 1 & -1 & 5 \\ 2 & -5 & -7 \\ -3 & 2 & 4 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 5 \\ 2 & 0 & 0 \\ 0 & 2 & 4 \end{bmatrix} \quad h^{l+1}$$

$$\frac{\partial L}{\partial h^l} = [\![h^l > 0]\!] \frac{\partial L}{\partial h^{l+1}}$$

Backward pass: backpropagation

$$\begin{bmatrix} -2 & 0 & -1 \\ 6 & 0 & 0 \\ 0 & -1 & 3 \end{bmatrix} \leftarrow \begin{bmatrix} -2 & 3 & -1 \\ 6 & -3 & 1 \\ 2 & -1 & 3 \end{bmatrix} \quad \frac{\partial L}{\partial h^{l+1}}$$

$\frac{\partial L}{\partial h^{l+1}} > 0$

$$\frac{\partial L}{\partial h^l} = [\![h^{l+1} > 0]\!] \frac{\partial L}{\partial h^{l+1}}$$

Backward pass: "deconvnet"

$$\begin{bmatrix} 0 & 3 & 0 \\ 6 & 0 & 1 \\ 2 & 0 & 3 \end{bmatrix} \leftarrow \begin{bmatrix} -2 & 3 & -1 \\ 6 & -3 & 1 \\ 2 & -1 & 3 \end{bmatrix} \quad \frac{\partial L}{\partial h^{l+1}}$$

$\frac{\partial L}{\partial h^{l+1}} > 0$

$$\frac{\partial L}{\partial h^l} = [\![(h^l > 0) \,\&\&\, (h^{l+1} > 0)]\!] \frac{\partial L}{\partial h^{l+1}}$$

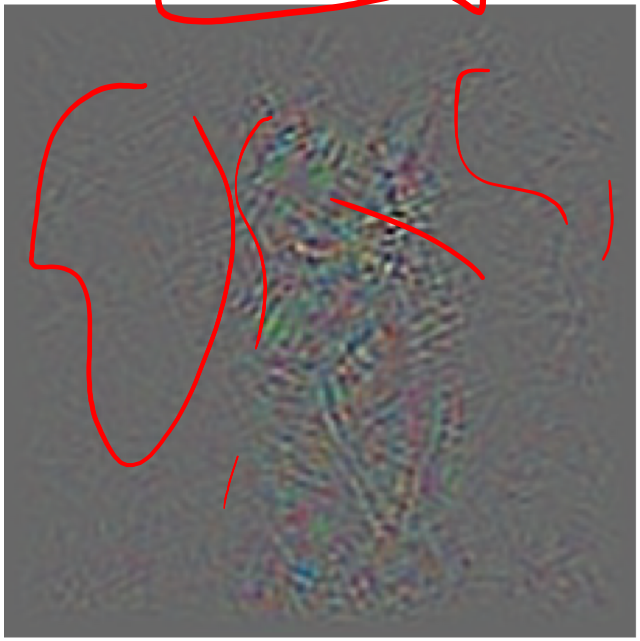
Backward pass: *guided backpropagation*

$$\begin{bmatrix} 0 & 0 & 0 \\ 6 & 0 & 0 \\ 0 & 0 & 3 \end{bmatrix} \leftarrow \begin{bmatrix} -2 & 3 & -1 \\ 6 & -3 & 1 \\ 2 & -1 & 3 \end{bmatrix} \quad \frac{\partial L}{\partial h^{l+1}}$$

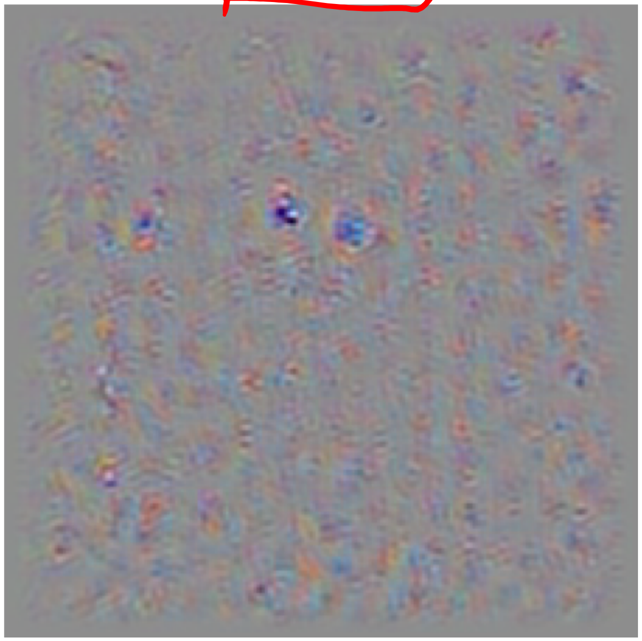
Backprop vs Deconv vs Guided BP

- Guided Backprop tends to be "cleanest"

Backprop



Deconv



Guided Backprop

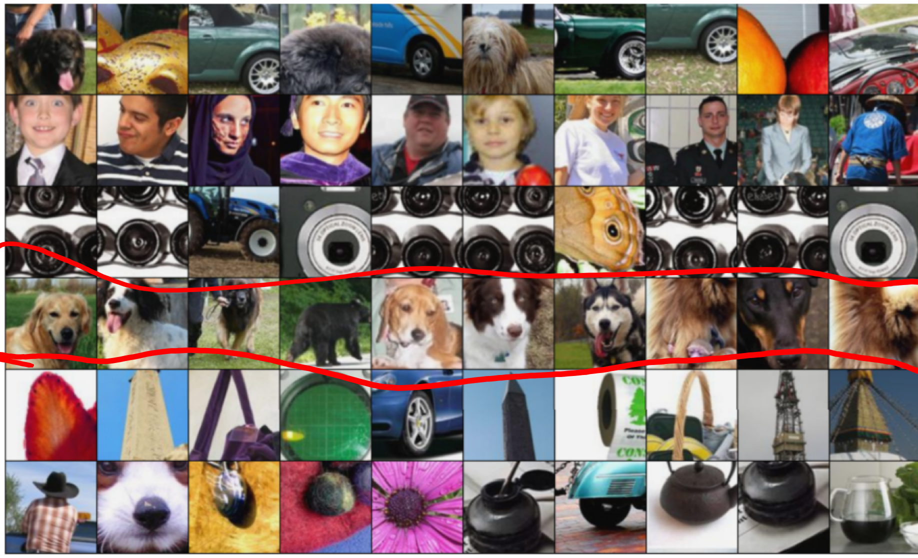
Intermediate features via (guided) backprop

Maximally activating patches
(Each row is a different neuron)

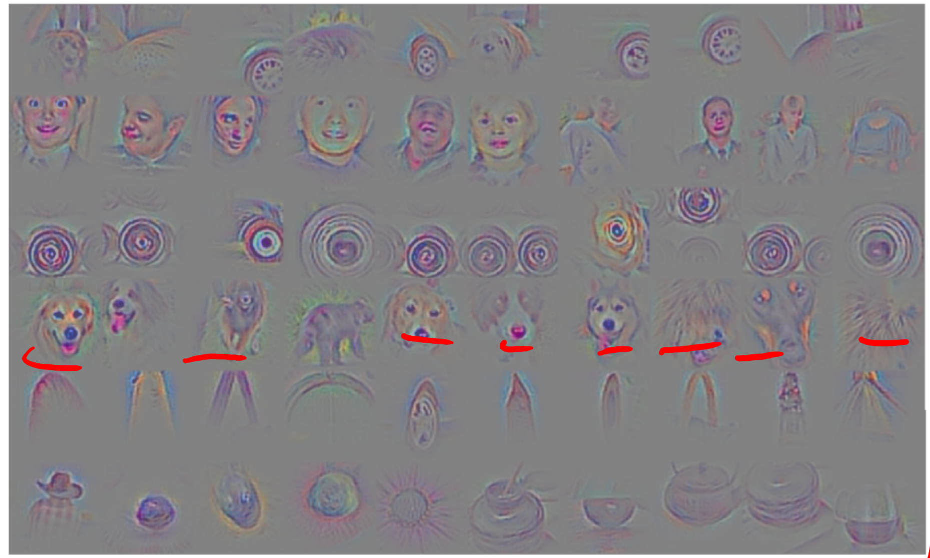
Guided Backprop

Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014
Springenberg et al, “Striving for Simplicity: The All Convolutional Net”, ICLR Workshop 2015

Intermediate features via (guided) backprop



Maximally activating patches
(Each row is a different neuron)



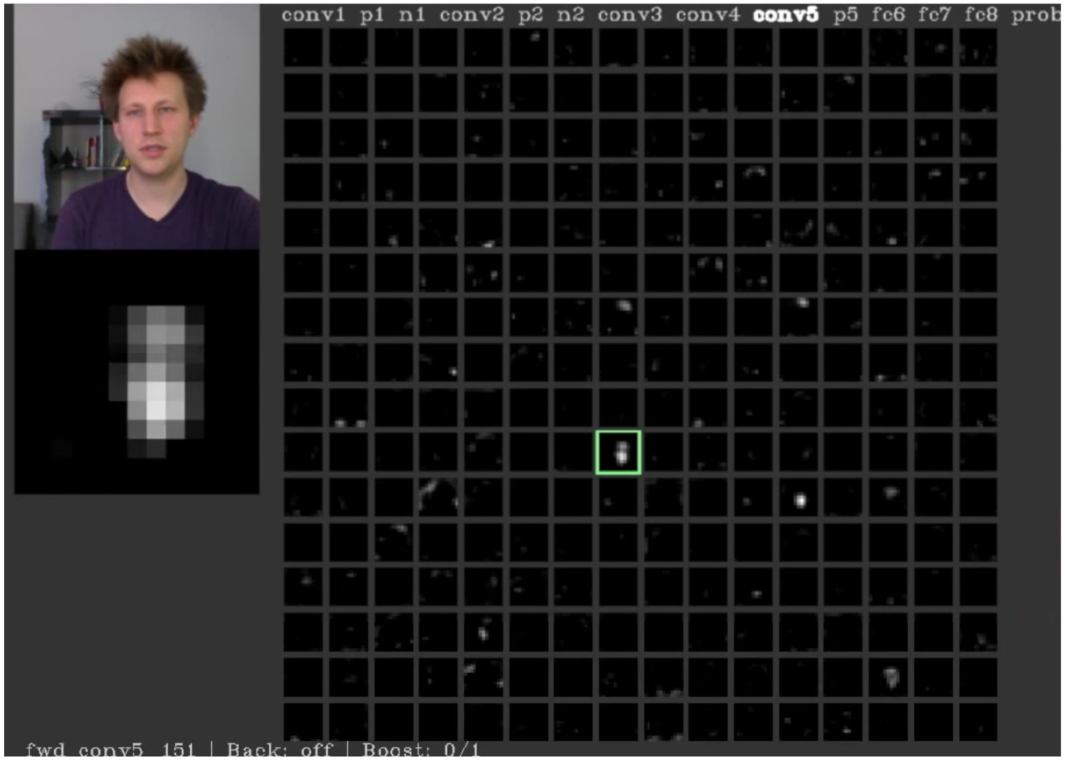
Guided Backprop

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

Visualizing Activations

conv5 feature map is 128x13x13; visualize as 128 13x13 grayscale images

https://youtu.be/AgkfIQ4IGaM?t=92



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Problem with Guided Backup

- Not very “class-discriminative”

GB for “airliner”

GB for “bus”

Slide Credit: Ram Selvaraju

Grad-CAM

Visual Explanations from Deep Networks via Gradient-based Localization

[ICCV '17]

Ramprasaath Selvaraju

Michael Cogswell

Abhishek Das

Ramakrishna Vedantam

Devi Parikh

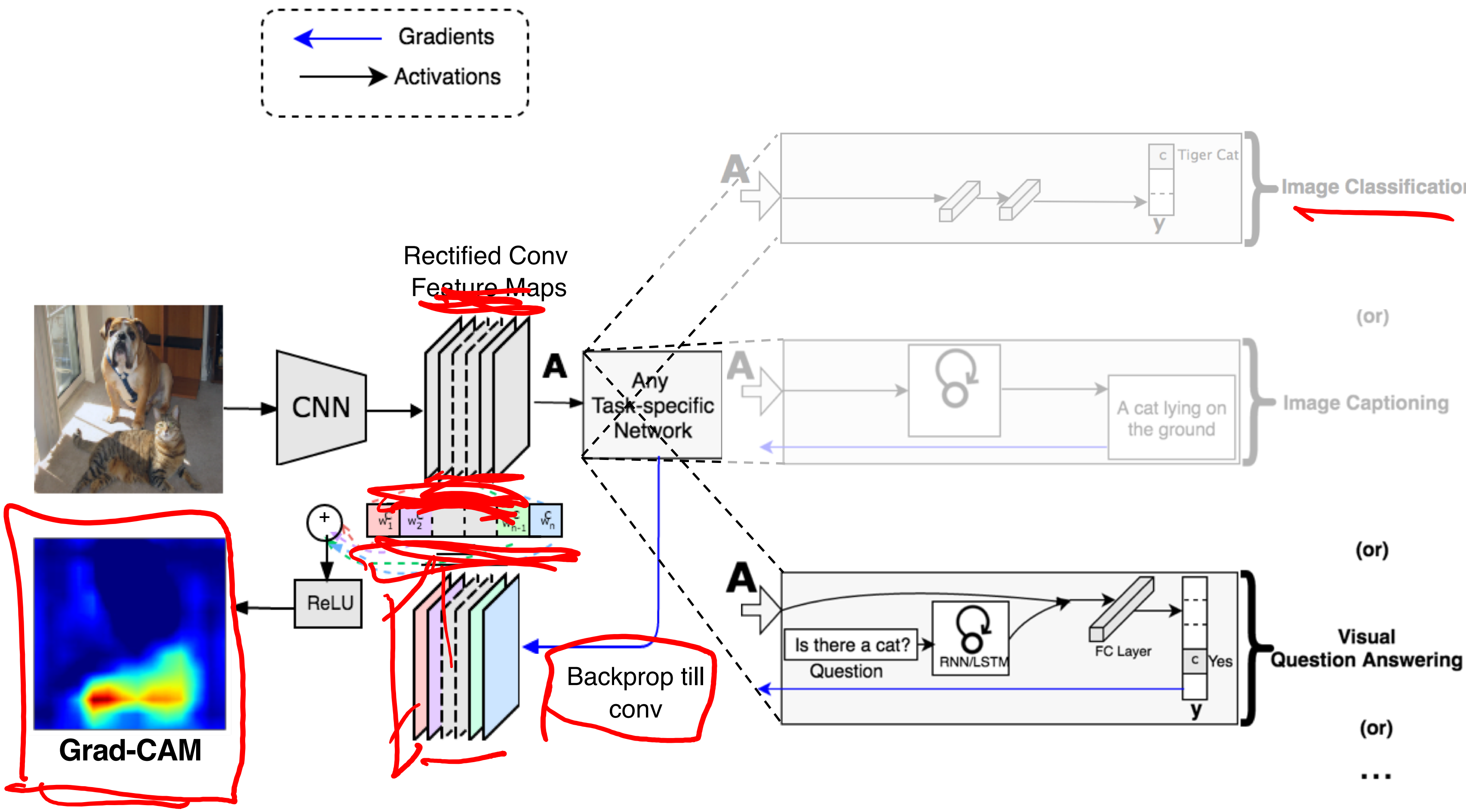
Dhruv Batra

VirginiaTech

facebook research

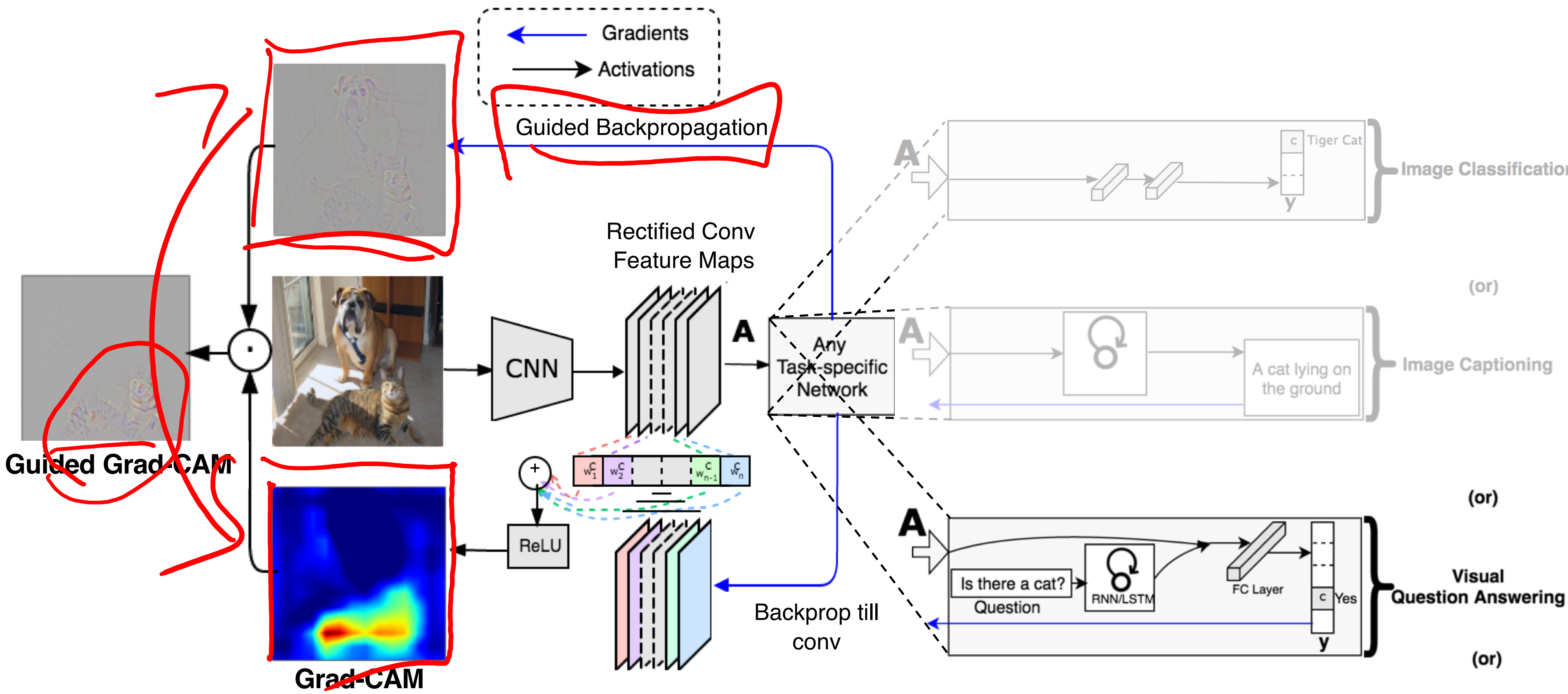
Georgia Institute of Technology

Grad-CAM



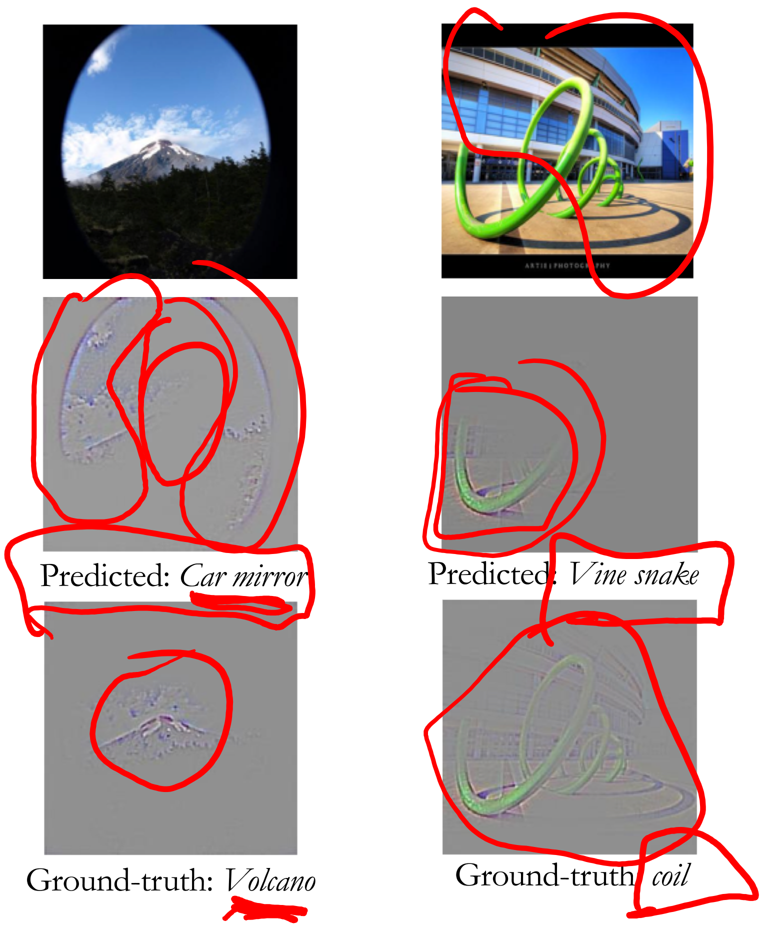
Gradients
Activations
Rectified Conv Feature Maps
CNN
A
Any Task-specific Network
Tiger Cat
y
Image Classification
(or)
A cat lying on the ground
Image Captioning
(or)
Is there a cat?
Question
RNN/LSTM
FC Layer
Yes
y
Visual Question Answering
(or)
...
ReLU
Backprop till conv
Grad-CAM

Guided Grad-CAM



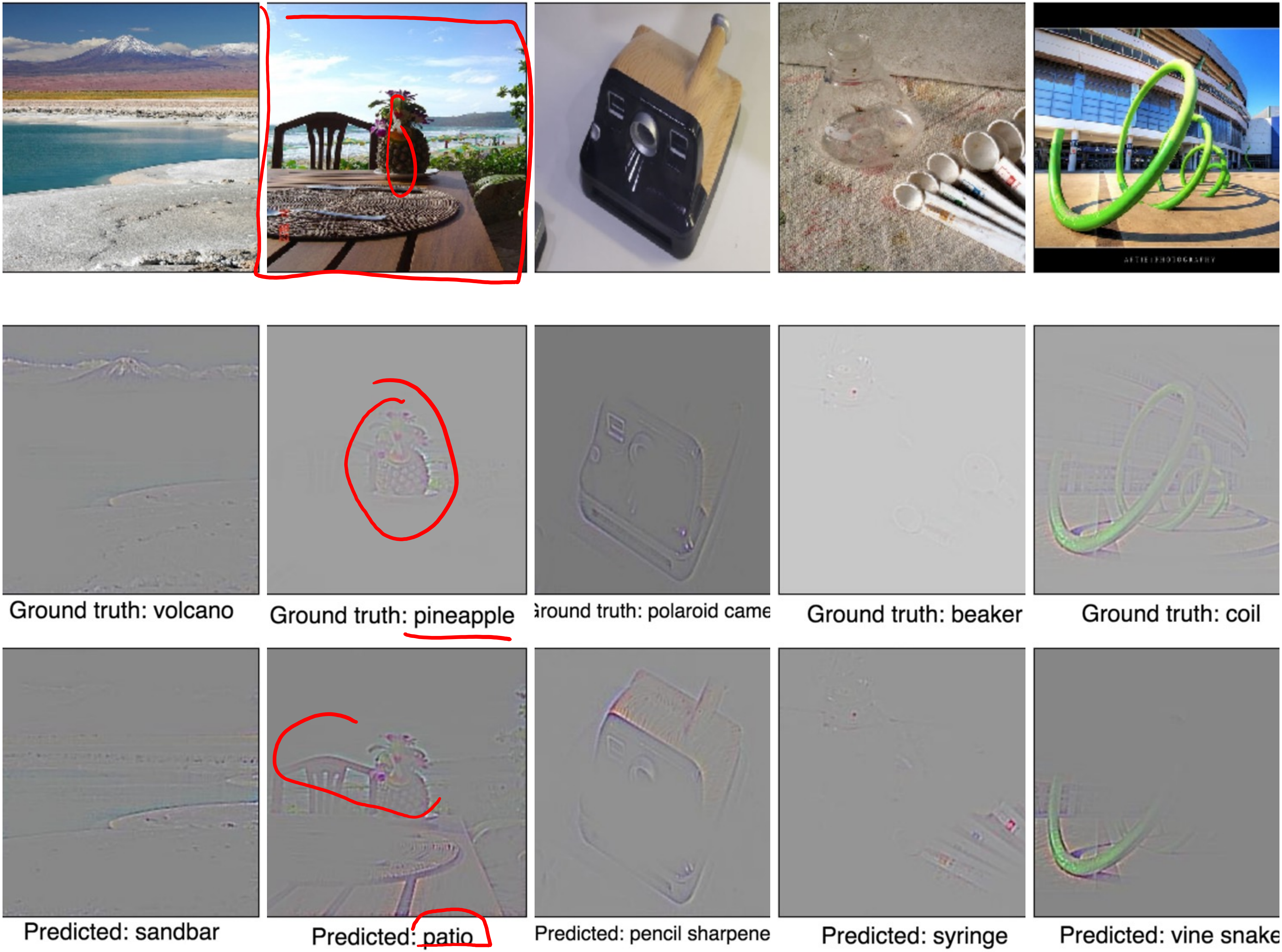
Slide Credit: Ram Selvaraju

Analyzing Failure Modes with Grad-CAM



Reasonable predictions are made in many failure cases.

Slide Credit: Ram Selvaraju


AFTERPHOTOGRAPHY
Ground truth: volcano
Ground truth: pineapple
round truth: polaroid came
Ground truth: beaker
Ground truth: coil
Predicted: sandbar
Predicted: patio
Predicted: pencil sharpene
Predicted: syringe
Predicted: vine snake

Grad-CAM Visual Explanations for Captioning

Guided Backprop | **Grad-CAM** | Guided Grad-CAM

A bathroom with a toilet and a sink

A horse is standing in a field with a fence in the background

1 CloudCV: Large Scale Distrib ×

128.173.88.229:8000/vqa

Apps Bookmarks VT wiki hackathon TTIC sortStory minds and machines Machine Learning Coding attention GT atlanta housing humor » Other Bookmarks

CloudCV[1] Image Stitching[2] Object Detection[3] Decaf-Server[4] Classification[5] VIP[6] Train a new category[7] VQA[8]

Fork me on GitHub

Result of Grad-CAM for Visual Question Answering

Enter the question

Answer(Optional)

Submit

Credits

Code for VQA Model

Built by @deshraj

Plan for Today

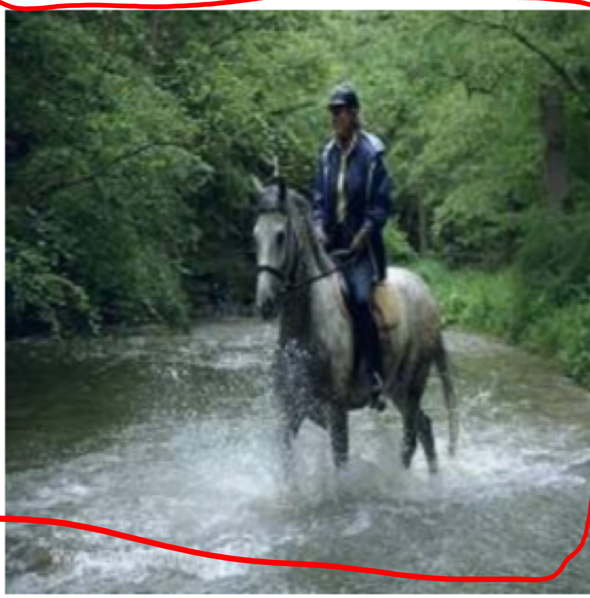
- Visualizing CNNs
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations
  - How to evaluate visualization?
  - Creating “prototypical” images for a class
  - Creating adversarial images
  - Deep dream
  - Feature inversion

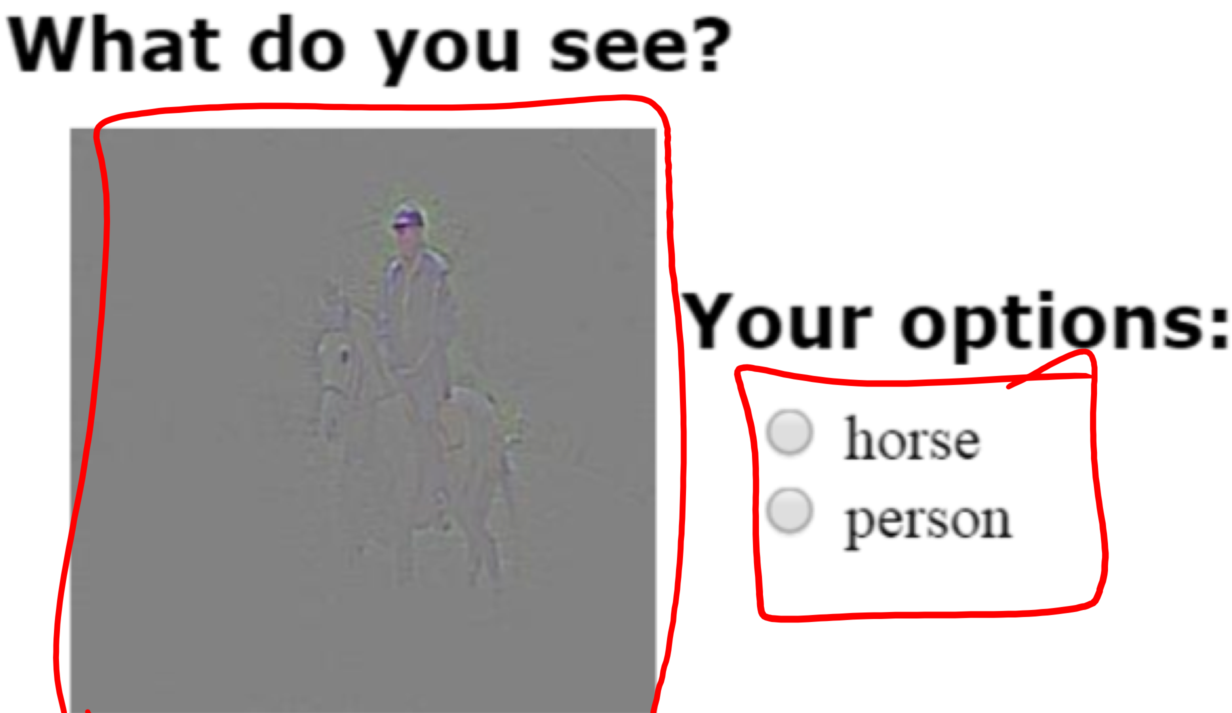
How we evaluate explanations?

- Class-discriminative?
  - Show what they say they found?
- Building Trust with a User?
  - Help users?
- Human-like?
  - Do machines look where humans look?

Is Grad-CAM more class discriminative?

- Can people tell which class is being visualized?
  - Images from Pascal VOC'07 with exactly 2 categories.





- Intuition: A good explanation produces discriminative visualizations for the class of interest.

Is Grad-CAM more class discriminative?

- Human accuracy for 2-class classification

Method	Human Classification Accuracy
Guided Backpropagation	44.44
Guided Grad-CAM	61.23

+17%

Grad-CAM makes existing visualizations class discriminative.

Help establish trust with a user?

- Given explanations from 2 models,
  - VGG16 and AlexNet

  which one is more trustworthy?

- Pick images where both models = correct prediction
- Show these to AMT workers and evaluate

Help establish trust in a user?

Both robots predicted: horse

Robot A based it's decision on

Robot B based it's decision on

Which robot is more reasonable?

1. Robot A seems clearly more reasonable than robot B
2. Robot A seems slightly more reasonable than robot B
3. Both robots seem equally reasonable
4. Robot B seems slightly more reasonable than robot A
5. Robot B seems clearly more reasonable than robot A

Method	Relative Reliability
Guided Backpropagation	+1.00
Guided Grad-CAM	+1.27

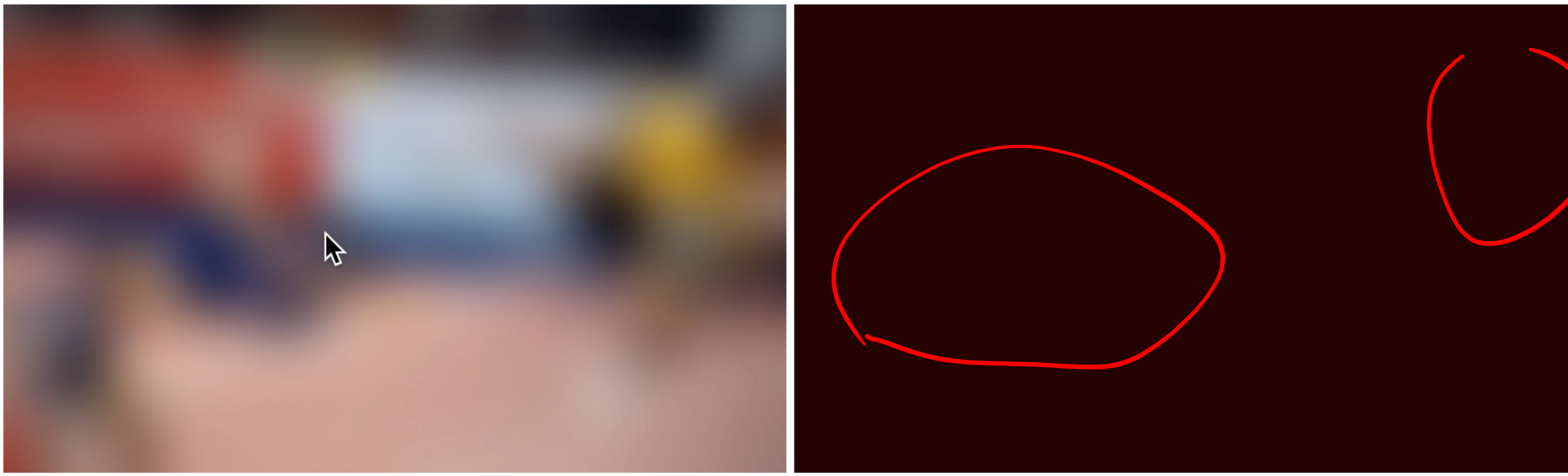
Users place higher trust in a model that generalizes better.

Slide Credit: R

Where do humans choose to look to answer visual questions?

VQA-HAT (Human ATtention)

Question: How many players are visible in the image?

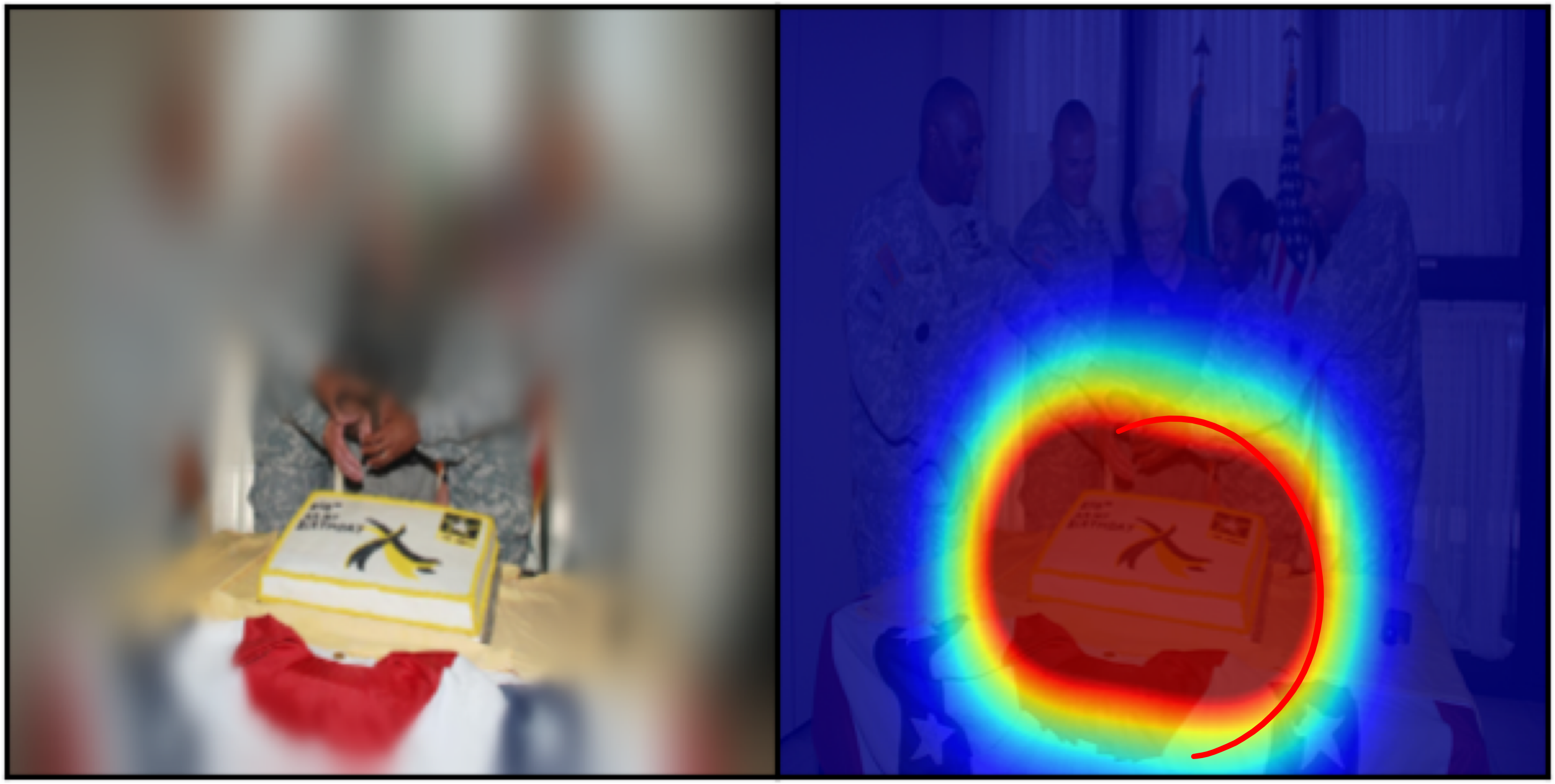


BLUR IMAGE

Answer:

3

VQA-HAT (Human ATtention)

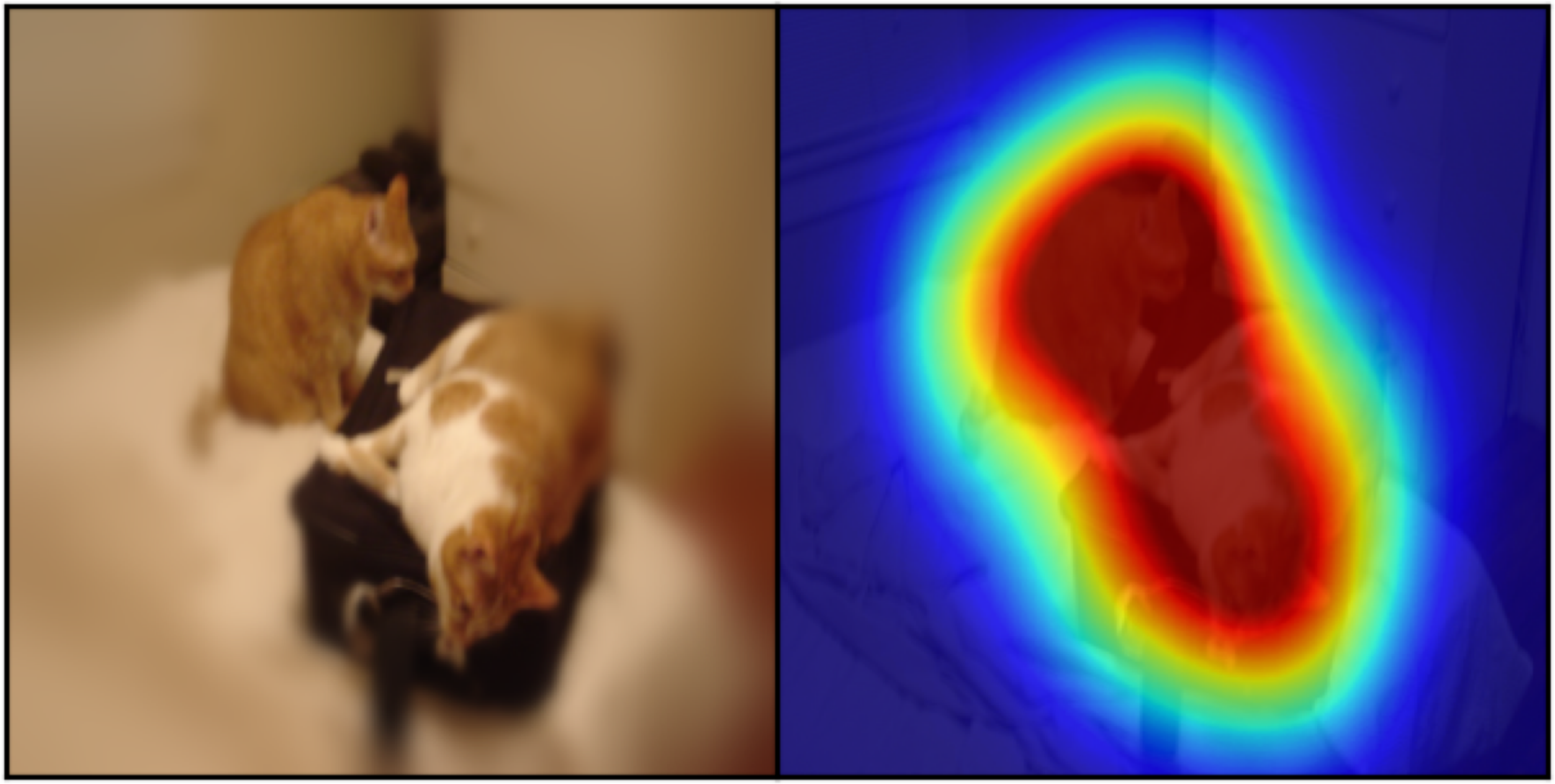


What food is on the table? Cake

VQA-HAT (Human ATtention)

What animal is she riding? Horse

VQA-HAT (Human ATtention)



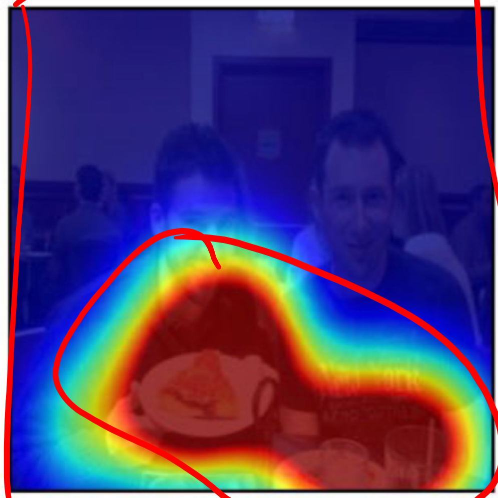
What number of cats are laying on the bed? 2

Are Grad-CAM explanations human-like?

- Correlation with human attention maps
  [Das & Agarwal et al. EMNLP'16]

What are they doing?

Grad-CAM for '*eating*'



Human ATtention map (HAT) for '*eating*'

Method	Rank Correlation w/ HAT
Guided Backpropagation	0.122
Guided Grad-CAM	**0.136**

Current models look at regions more similar to humans than baselines

Plan for Today

- Visualizing CNNs
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations
  - How to evaluate visualizations?
  - Creating “prototypical” images for a class
  - Creating adversarial images
  - Deep dream
  - Feature inversion

Visualizing CNN features: Gradient Ascent on Pixels

(Guided) backprop: Find the part of an image that a neuron responds to

Gradient ascent on pixels: Generate a synthetic image that maximally activates a neuron

$$I^* = \arg\max_I \boxed{f(I)} + \boxed{R(I)}$$

Neuron value

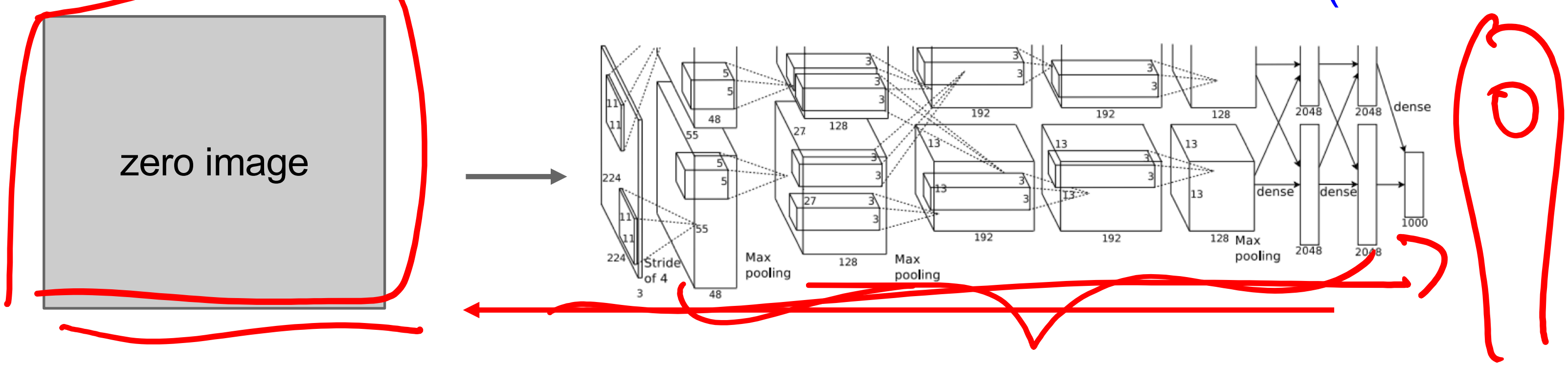
Natural image regularizer

Visualizing CNN features: Gradient Ascent on Pixels

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

score for class c (before Softmax)

1. Initialize image to zeros



Repeat:
2. Forward image to compute current scores
3. Backprop to get gradient of neuron value with respect to image pixels
4. Make a small update to the image

Visualizing CNN features: Gradient Ascent on Pixels

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

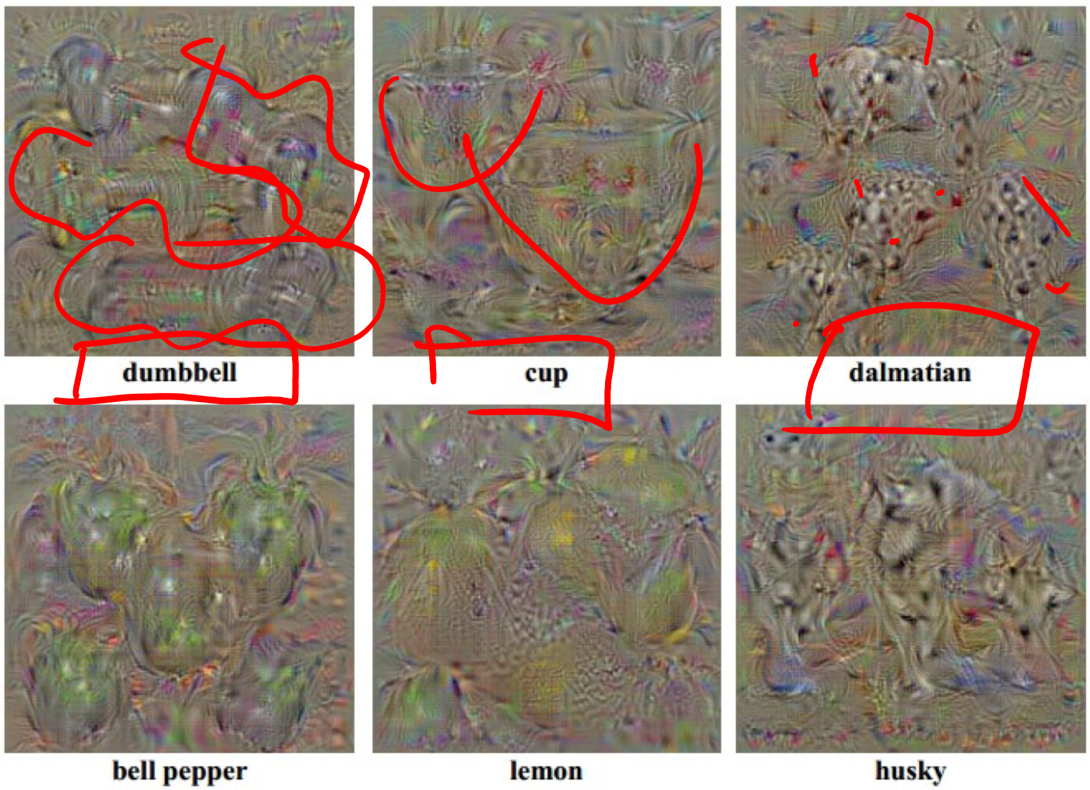
Simple regularizer: Penalize L2 norm of generated image

Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

Visualizing CNN features: Gradient Ascent on Pixels

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

Simple regularizer: Penalize L2 norm of generated image

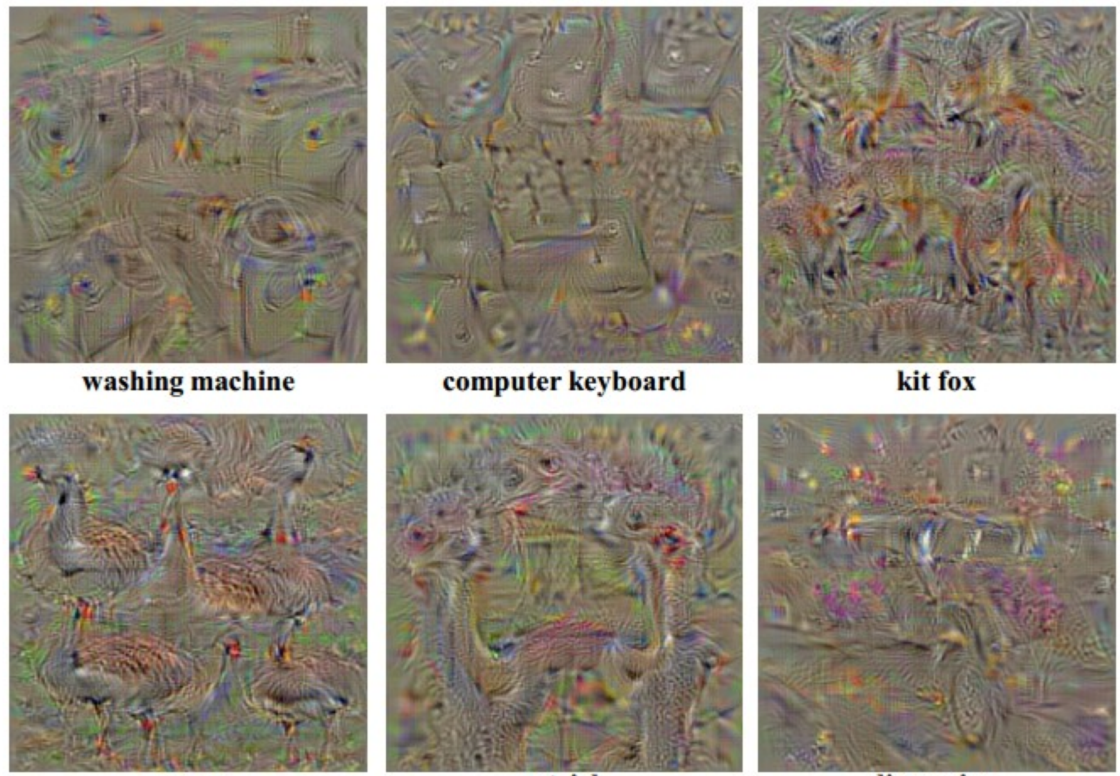


Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

Visualizing CNN features: Gradient Ascent on Pixels

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

Simple regularizer: Penalize L2 norm of generated image



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

Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

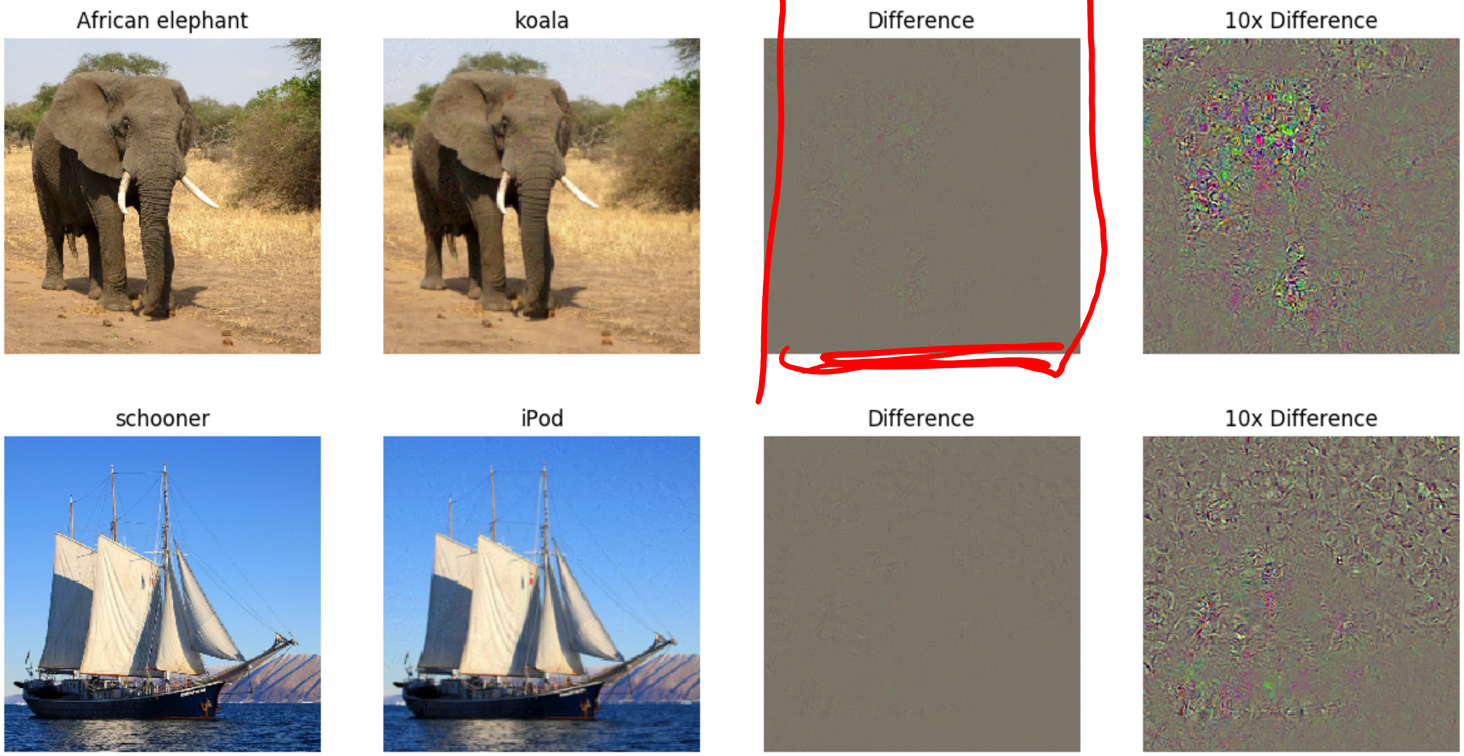
Fooling Images / Adversarial Examples

$$\max_{\Delta x} \left[S_c(x + \Delta x) \right]$$

(1)Start from an arbitrary image
(2)Pick an arbitrary class
(3)Modify the image to maximize the class
(4)Repeat until network is fooled

Fooling Images / Adversarial Examples

$$\underline{f(x)} = W_c^T x + b = \sum_i \underline{w_i}\, \underline{x_i}$$



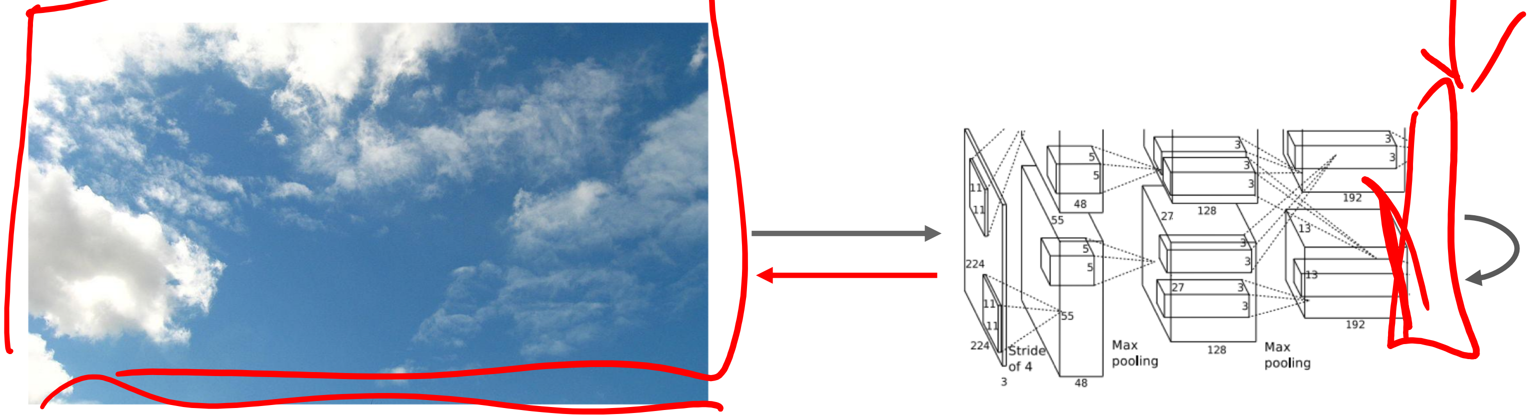
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Plan for Today

- Visualizing CNNs
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
  - Occlusion maps
  - Salient or "important" pixels
    - Gradient-based visualizations
  - How to evaluate visualizations?
  - Creating "prototypical" images for a class
  - Creating adversarial images
  - Deep dream
  - Feature inversion

DeepDream: Amplify existing features

Rather than synthesizing an image to maximize a specific neuron, instead try to **amplify** the neuron activations at some layer in the network



Choose an image and a layer in a CNN; repeat:

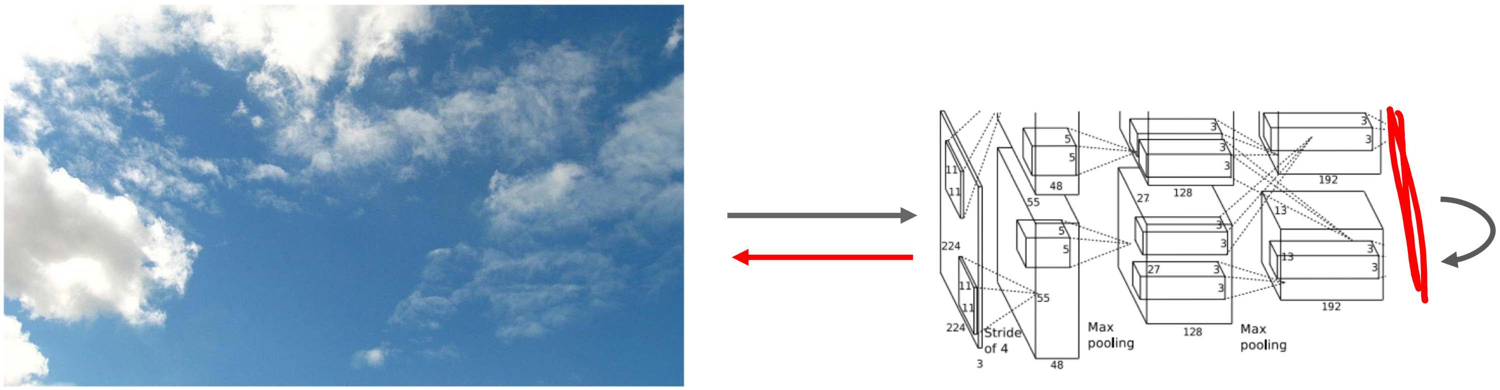
1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer *equal to its activation*
3. Backward: Compute gradient on image
4. Update image

Mordvintsev, Olah, and Tyka, "Inceptionism: Going Deeper into Neural Networks", Google Research Blog. Images are licensed under CC-BY 4.0

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

DeepDream: Amplify existing features

Rather than synthesizing an image to maximize a specific neuron, instead try to **amplify** the neuron activations at some layer in the network



Choose an image and a layer in a CNN; repeat:

1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer *equal to its activation*
3. Backward: Compute gradient on image
4. Update image

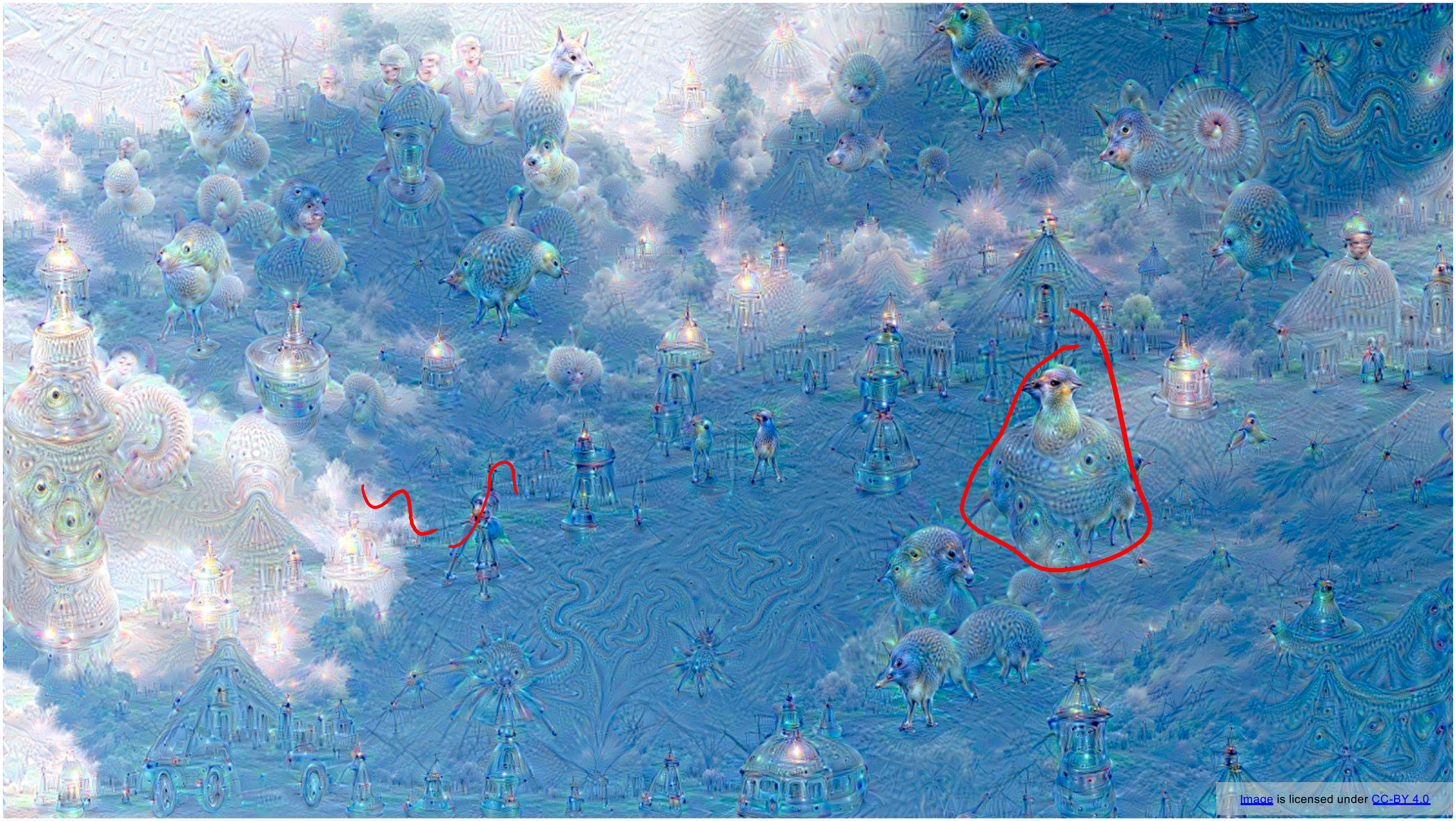
Equivalent to:

$I^* = \arg\max_I \sum_i f_i(I)^2$

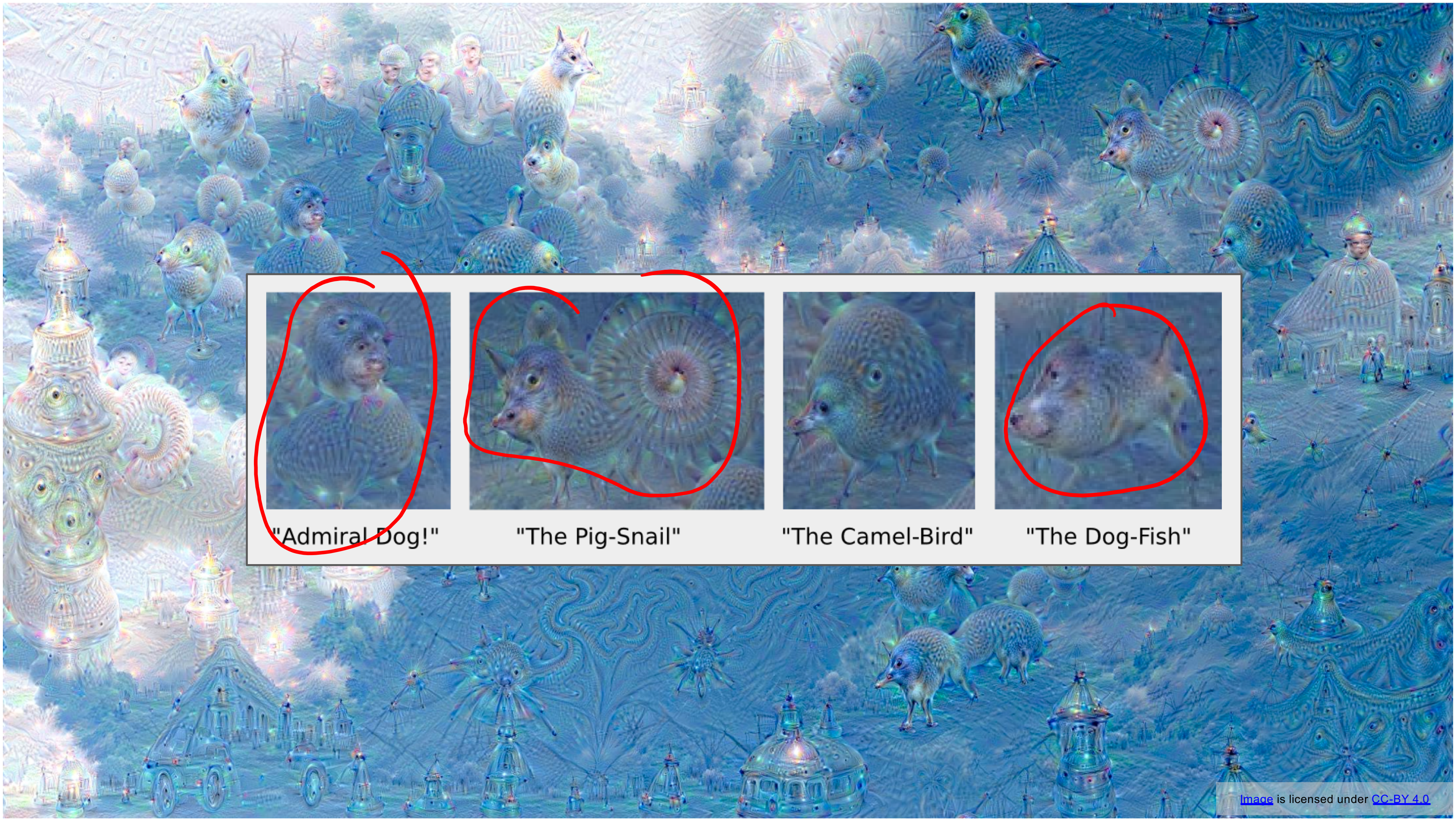
Mordvintsev, Olah, and Tyka, "Inceptionism: Going Deeper into Neural Networks", Google Research Blog. Images are licensed under CC-BY 4.0

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n



Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Feature Inversion

Given a CNN feature vector for an image, find a new image that:

- Matches the given feature vector
- “looks natural” (image prior regularization)

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

Given feature vector

Features of new image

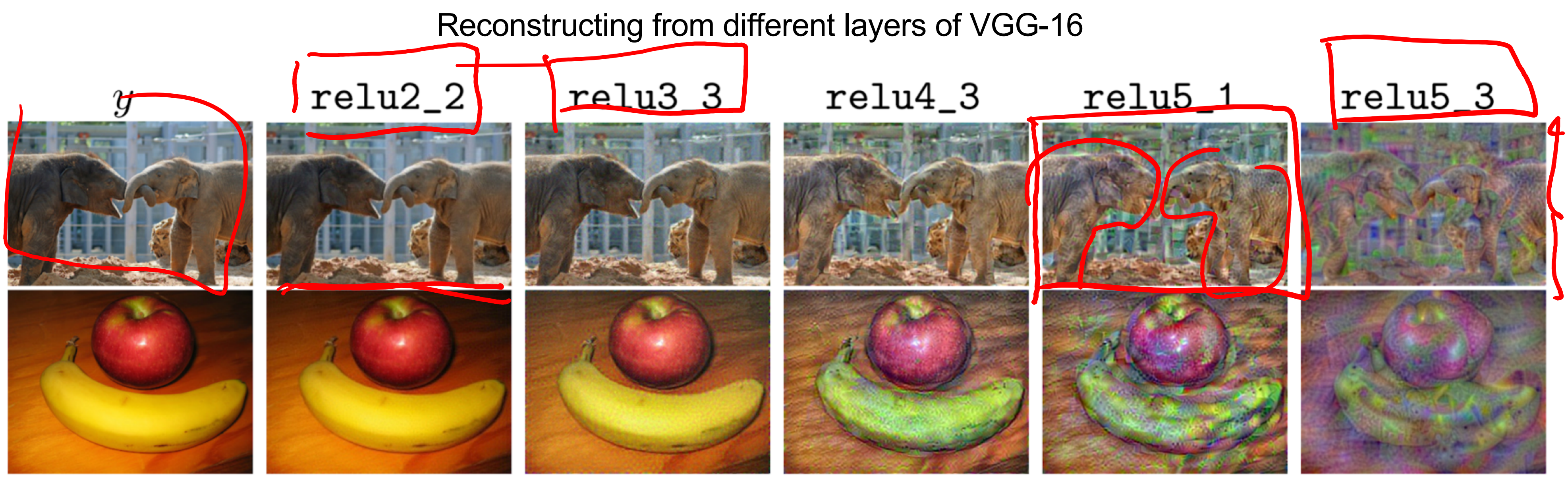
$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\frac{\beta}{2}}$$

Total Variation regularizer
(encourages spatial smoothness)

Mahendran and Vedaldi, “Understanding Deep Image Representations by Inverting Them”, CVPR 2015

Feature Inversion



Mahendran and Vedaldi, “Understanding Deep Image Representations by Inverting Them”, CVPR 2015
Figure from Johnson, Alahi, and Fei-Fei, “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”, ECCV 2016. Copyright Springer, 2016.
Reproduced for educational purposes.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n