CS 4803 / 7643: Deep Learning

Topics:

- Visualizing CNNs

Ramprasaath R. Selvaraju
Salesforce Research

Plan for Today

- What do individual neurons look for in images?
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
- How pixels affect model decisions?
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations
- Do CNNs look at same regions as humans?
  - How to evaluate visualizations?
- How can we synthesize images to see what networks are looking for?
  - Creating “prototypical” images for a class
  - Creating adversarial images
  - Deep dream
  - Feature inversion

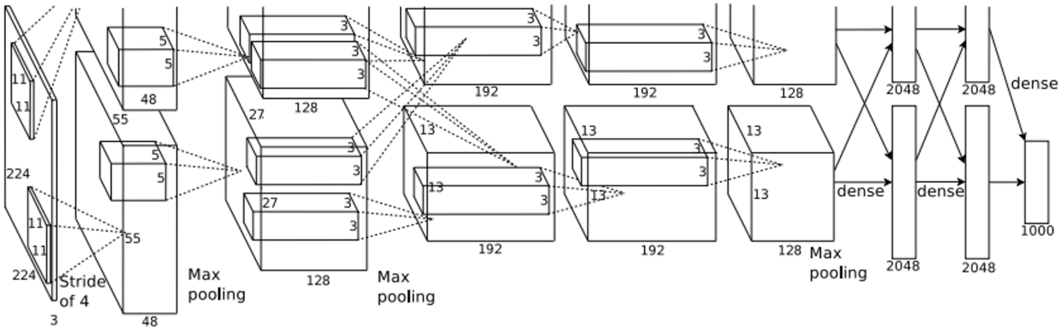
Plan for Today

- **What do individual neurons look for in images?**
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
- How pixels affect decisions?
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations
- Do CNNs look at same regions as humans?
  - How to evaluate visualizations?
- How can we synthesize images to see what networks are looking for?
  - Creating “prototypical” images for a class
  - Creating adversarial images
  - Deep dream
  - Feature inversion

What do individual neurons look for in images?

This image is CC0 public domain

Input Image:
3 x 224 x 224

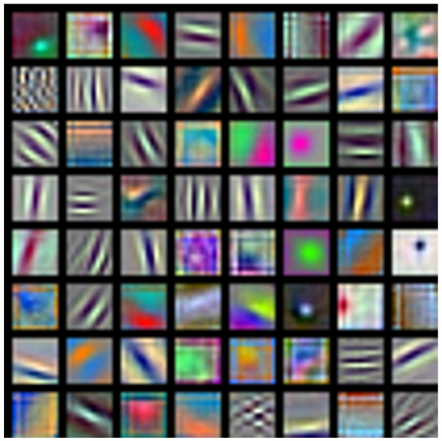


Class Scores:
1000 numbers

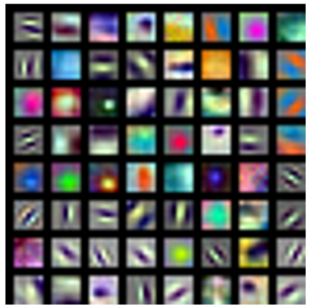
What are the intermediate features looking for?

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

Visualizing filters in first layer

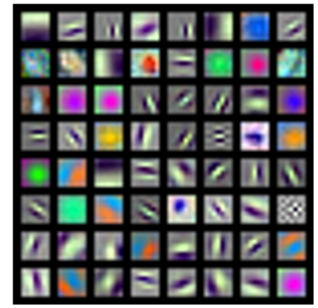


AlexNet:
64 x 3 x 11 x 11

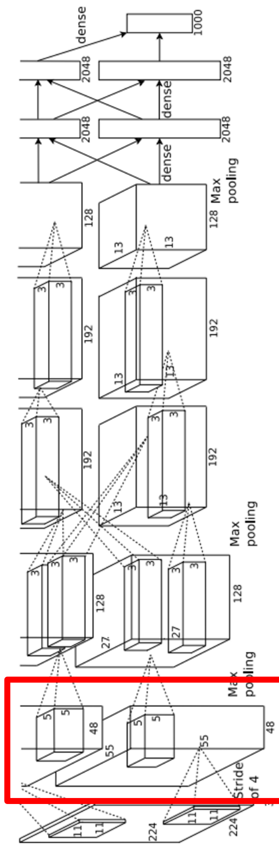


ResNet-18:
64 x 3 x 7 x 7

ResNet-101:
64 x 3 x 7 x 7



DenseNet-121:
64 x 3 x 7 x 7

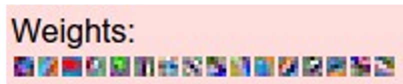


Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014
He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016
Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

Visualizing filters in intermediate layers

Visualize the filters/kernels (raw weights)

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



layer 1 weights

16 x 3 x 7 x 7

layer 2 weights

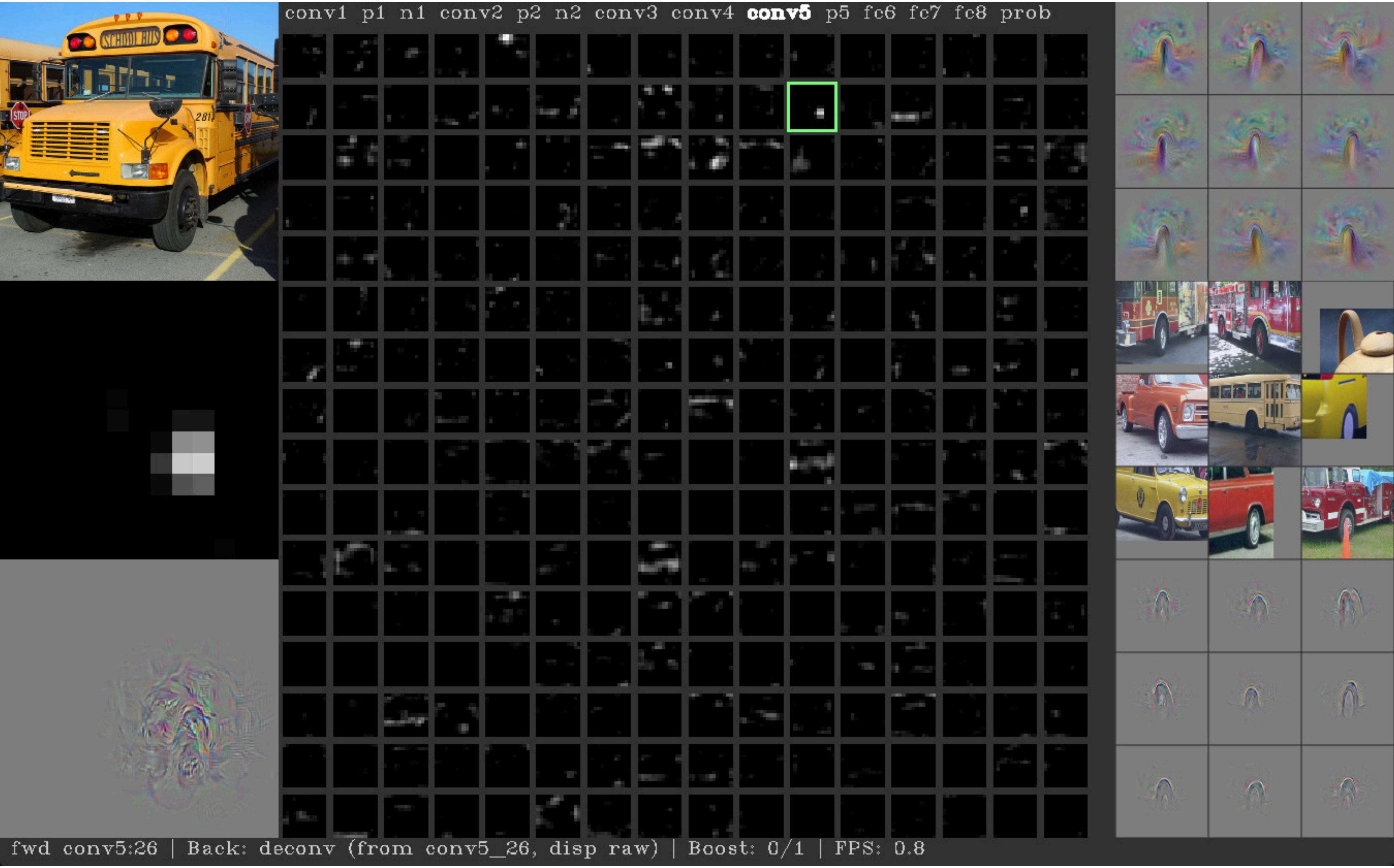
20 x 16 x 7 x 7

layer 3 weights

20 x 20 x 7 x 7

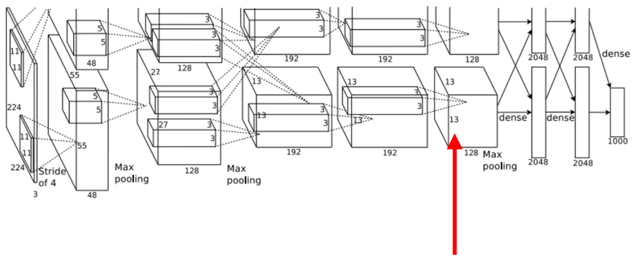
What do neuron activations look like?

Visualizing activations in intermediate layers



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

Maximally Activating Patches



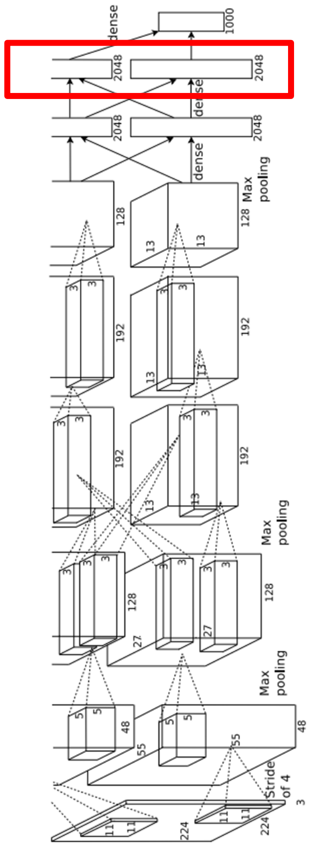
Pick a layer and a channel; e.g. conv5 is 128 x 13 x 13, pick channel 17/128

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

Visualize image patches that correspond to maximal activations

What does the last layer learn?

FC7 layer

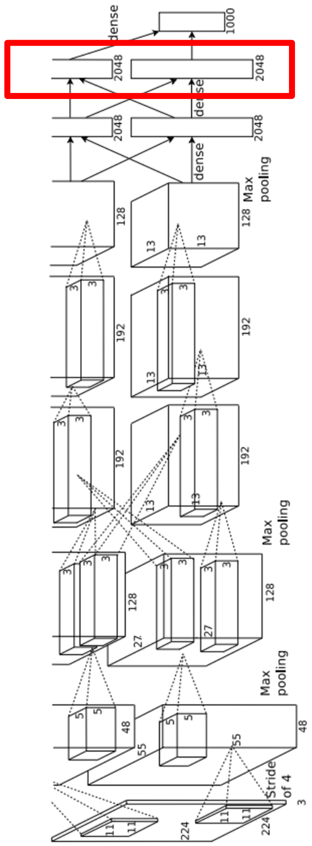


4096-dimensional feature vector for an image
(layer immediately before the classifier)

Run the network on many images, collect the feature vectors

Last Layer: Nearest Neighbors

4096-dim vector



Test image L2 Nearest neighbors in feature space

Recall: Nearest neighbors in pixel space

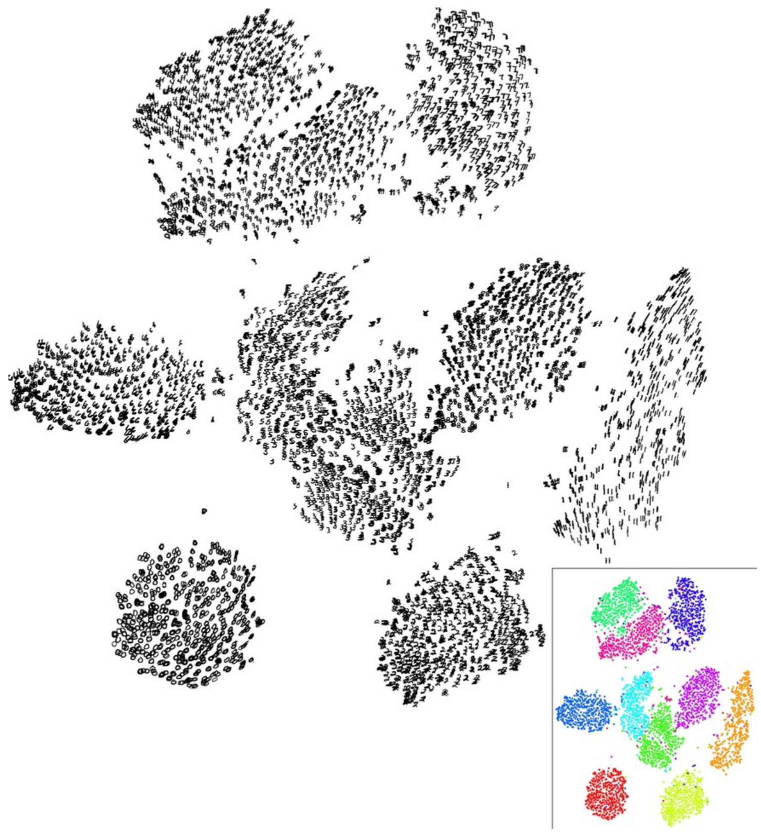
Krizhevsky et al, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012.
Figures reproduced with permission.

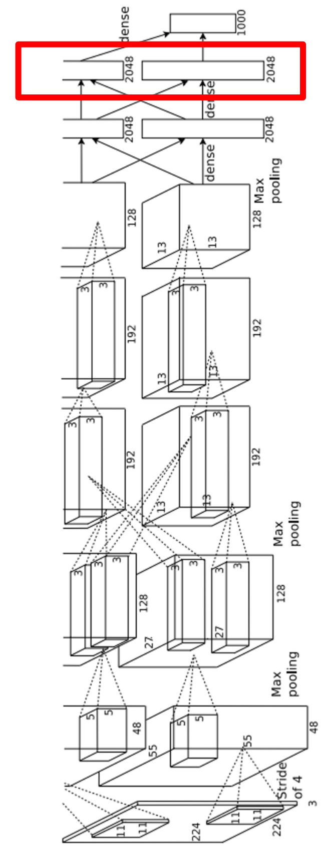
Last Layer: Dimensionality Reduction

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

Simple algorithm: Principal Component Analysis (PCA)

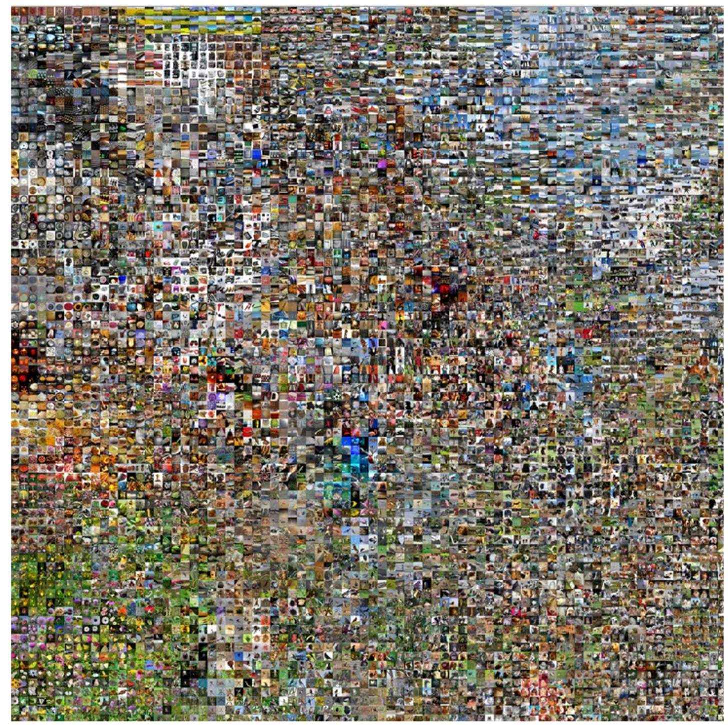
More complex: **t-SNE**

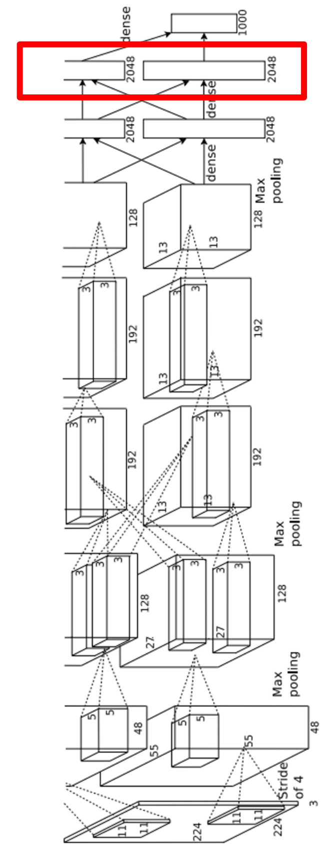




Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008

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/

Plan for Today

- What do individual neurons look for in images?
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
- **How pixels affect decisions from CNNs?**
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations
- Do CNNs look at same regions as humans?
  - How to evaluate visualizations?
- Can we synthesize network-specific images?
  - Creating “prototypical” images for a class
  - Creating adversarial images
  - Deep dream
  - Feature inversion

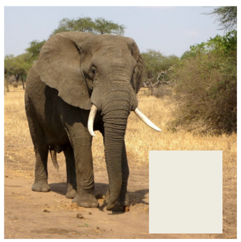
How pixels affect decisions?

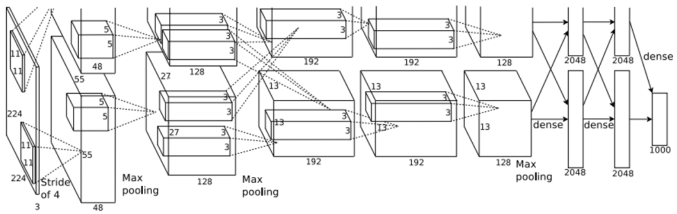
Visual Explanations

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

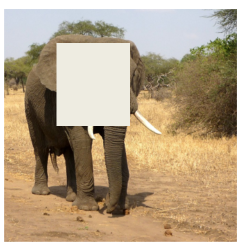
Which pixels matter: Occlusion Maps

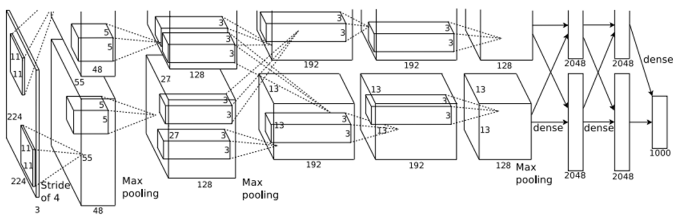
Idea: Mask part of the image before feeding to CNN,
check how much predicted probabilities change





P(elephant) = 0.95



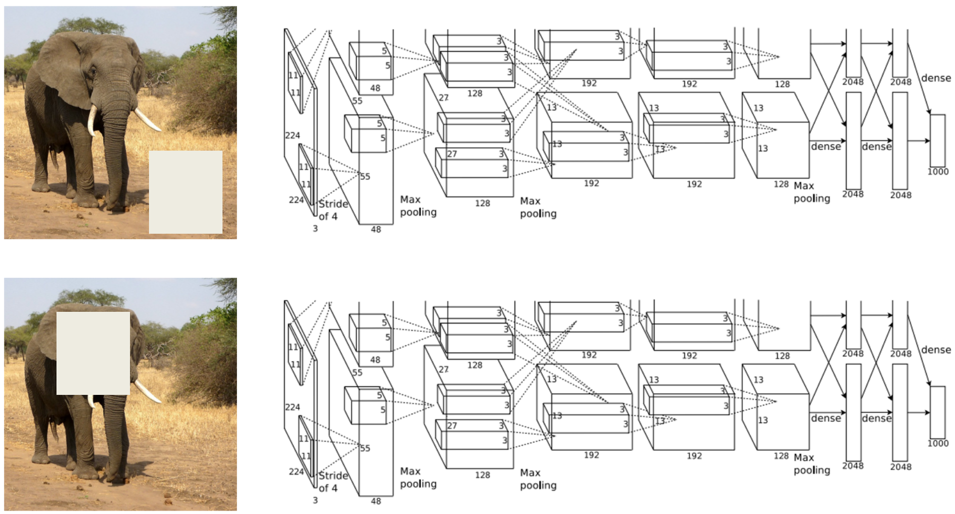


P(elephant) = 0.75

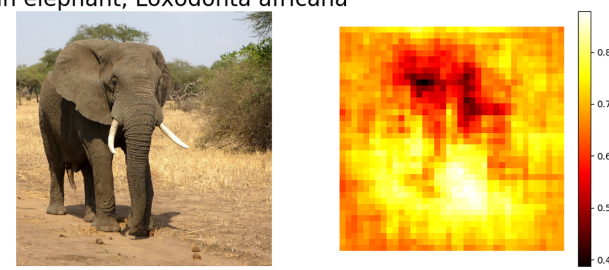
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Which pixels matter: Occlusion Maps

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



African elephant, Loxodonta africana



go-kart

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

Faithful ☺

Very expensive ☹

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

What if our model was linear?

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

What if our model was linear?

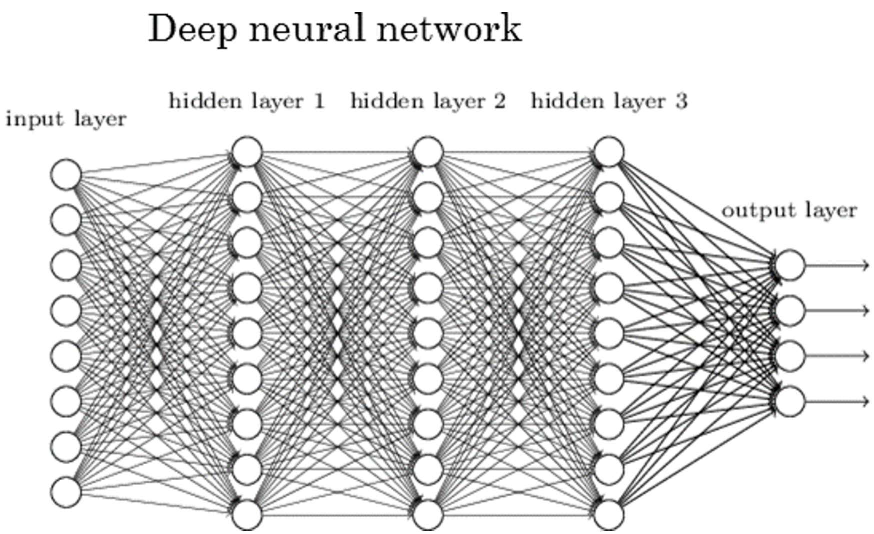
$$\left\langle \begin{bmatrix} \mathbf{100} \\ 0.1 \\ -0.1 \\ \mathbf{510} \\ \mathbf{-200} \end{bmatrix}, \begin{bmatrix} 1 \\ 0.9 \\ -0.2 \\ 0.5 \\ -0.9 \end{bmatrix} \right\rangle + b = S_c(\mathbf{x})$$

But it's not ☹

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

Can we make it linear?

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



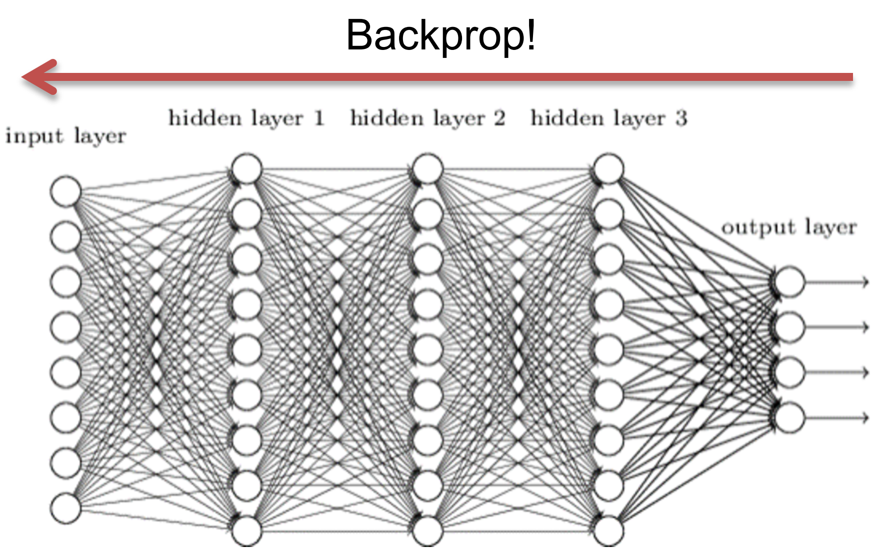
Taylor Series

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0)$$

Feature Importance in Deep Models

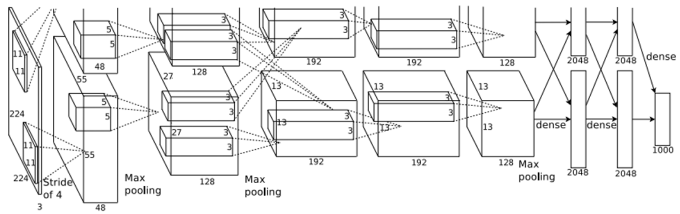
$$\mathbf{w}_c = \left. \frac{\partial S_c}{\partial \mathbf{x}} \right|_{\mathbf{x}_0}$$

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

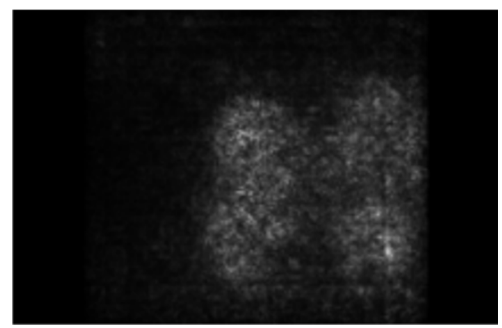


Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities



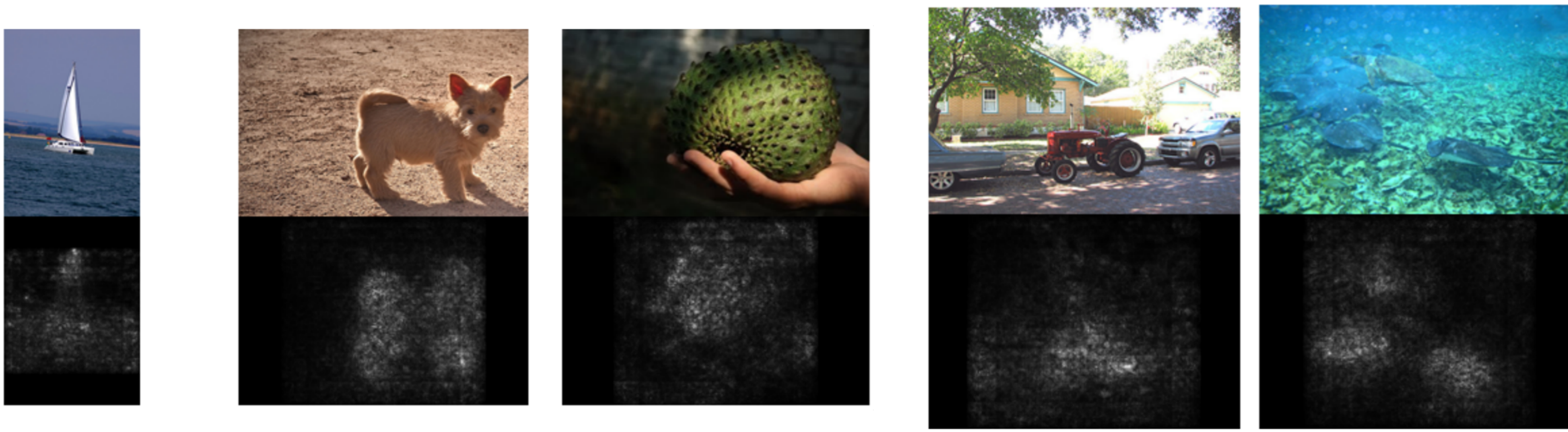
Dog



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

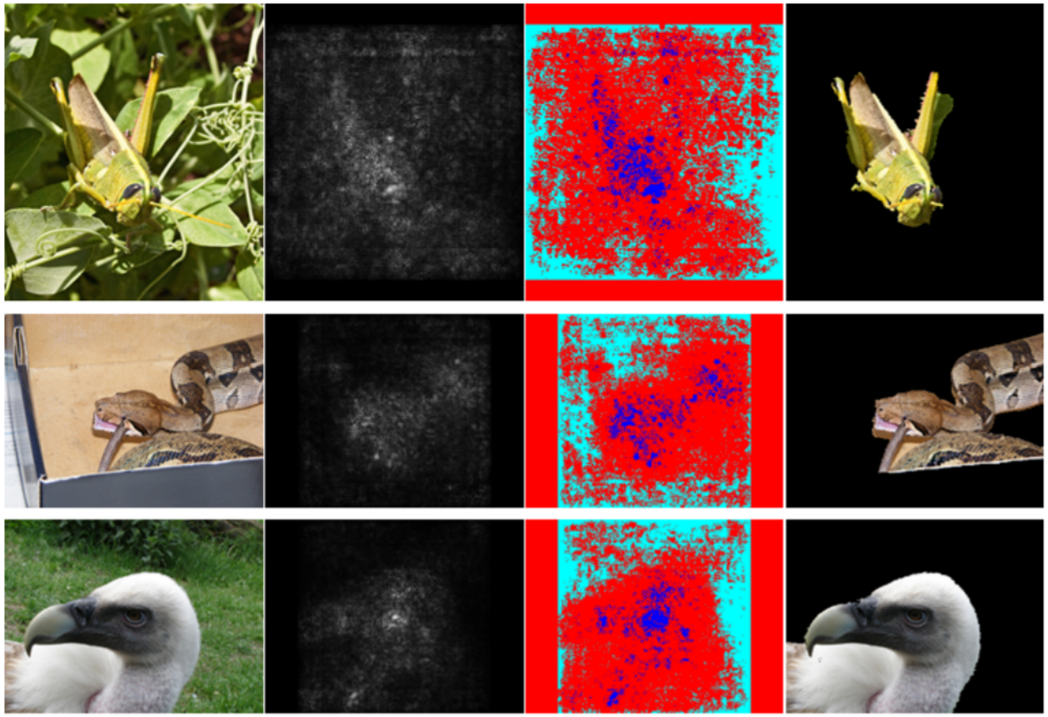
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

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

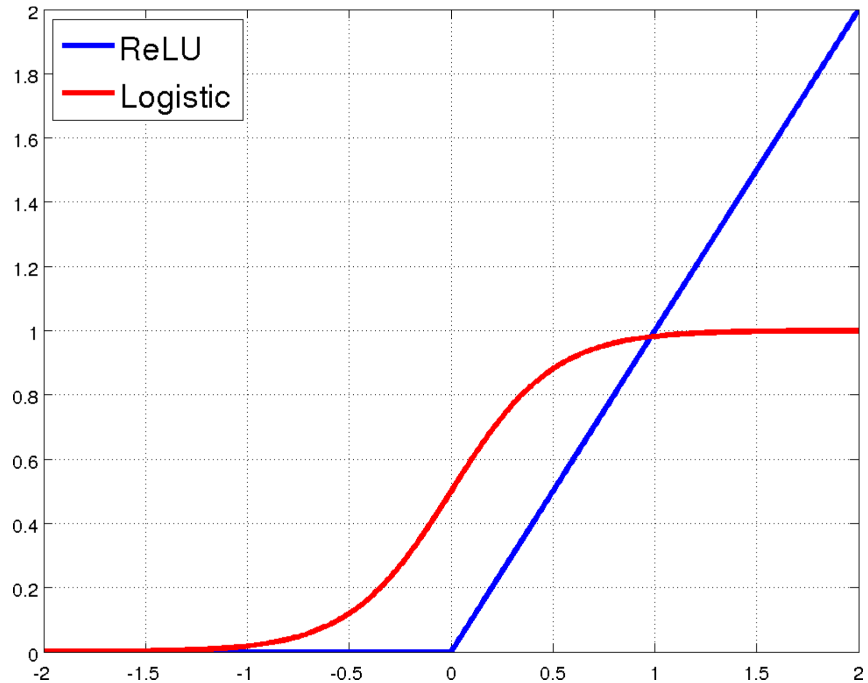
Simonyan, Vedaldi, and Zisserman, “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”, ICLR Workshop 2014.

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

Remember ReLUs?

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

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



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

Forward pass

h^l

1	-1	5
2	-5	-7
-3	2	4

→

h^{l+1}

1	0	5
2	0	0
0	2	4

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

Backward pass: backpropagation

-2	0	-1
6	0	0
0	-1	3

←

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

-2	3	-1
6	-3	1
2	-1	3

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

Backward pass: "deconvnet"

0	3	0
6	0	1
2	0	3

←

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

-2	3	-1
6	-3	1
2	-1	3

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

Backward pass: *guided backpropagation*

0	0	0
6	0	0
0	0	3

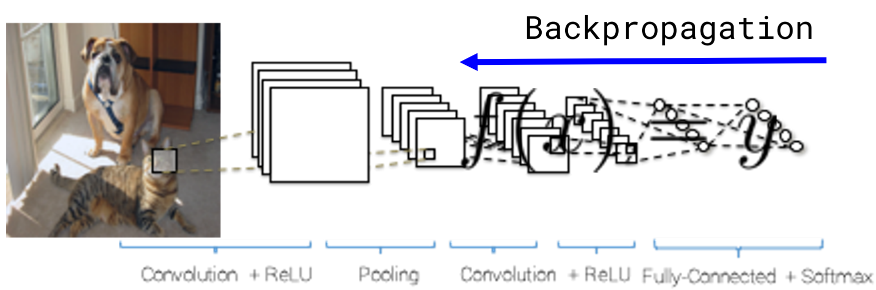
←

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

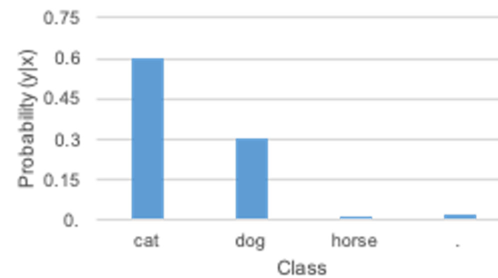
-2	3	-1
6	-3	1
2	-1	3

Gradient-based visualizations

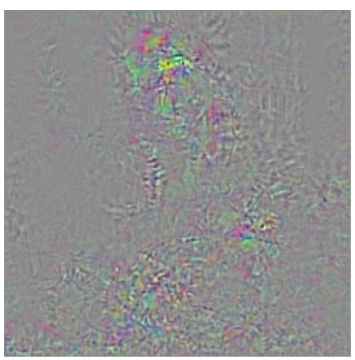
x



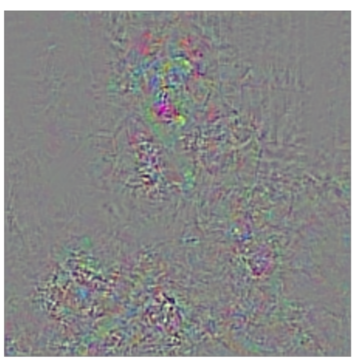
y



$$< w_c, x > + b \approx f(x)$$



Backprop for `cat’



Backprop for `dog’

Noisy

$$w_c = \left.\frac{\partial y_c}{\partial x}\right|_{x=x_0}$$

Guided Backprop for `cat’

Guided Backprop for `dog’

Not Class-Discriminative

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

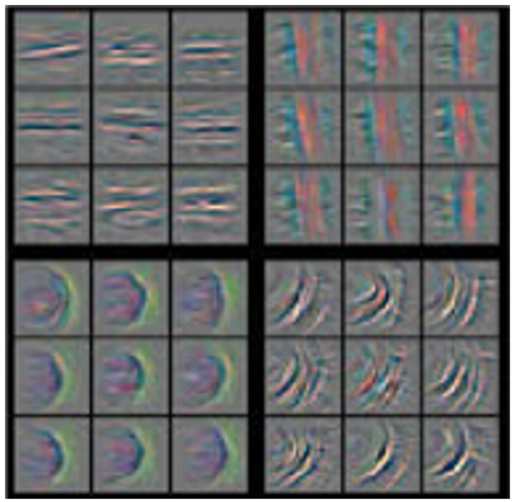
VirginiaTech

facebook research

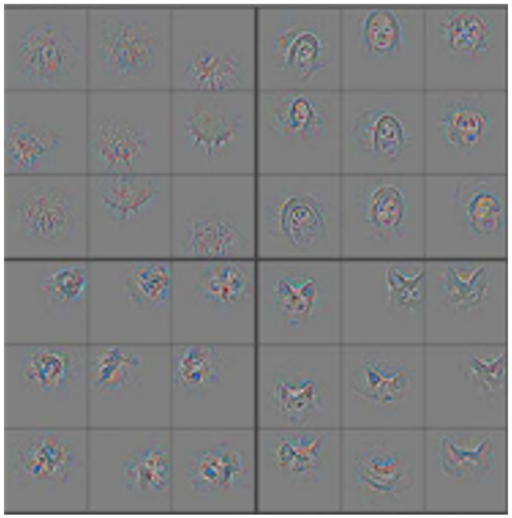
Georgia Institute of Technology

Grad-CAM Motivation

- Perturb semantic neurons in the image and see how it affects the decision



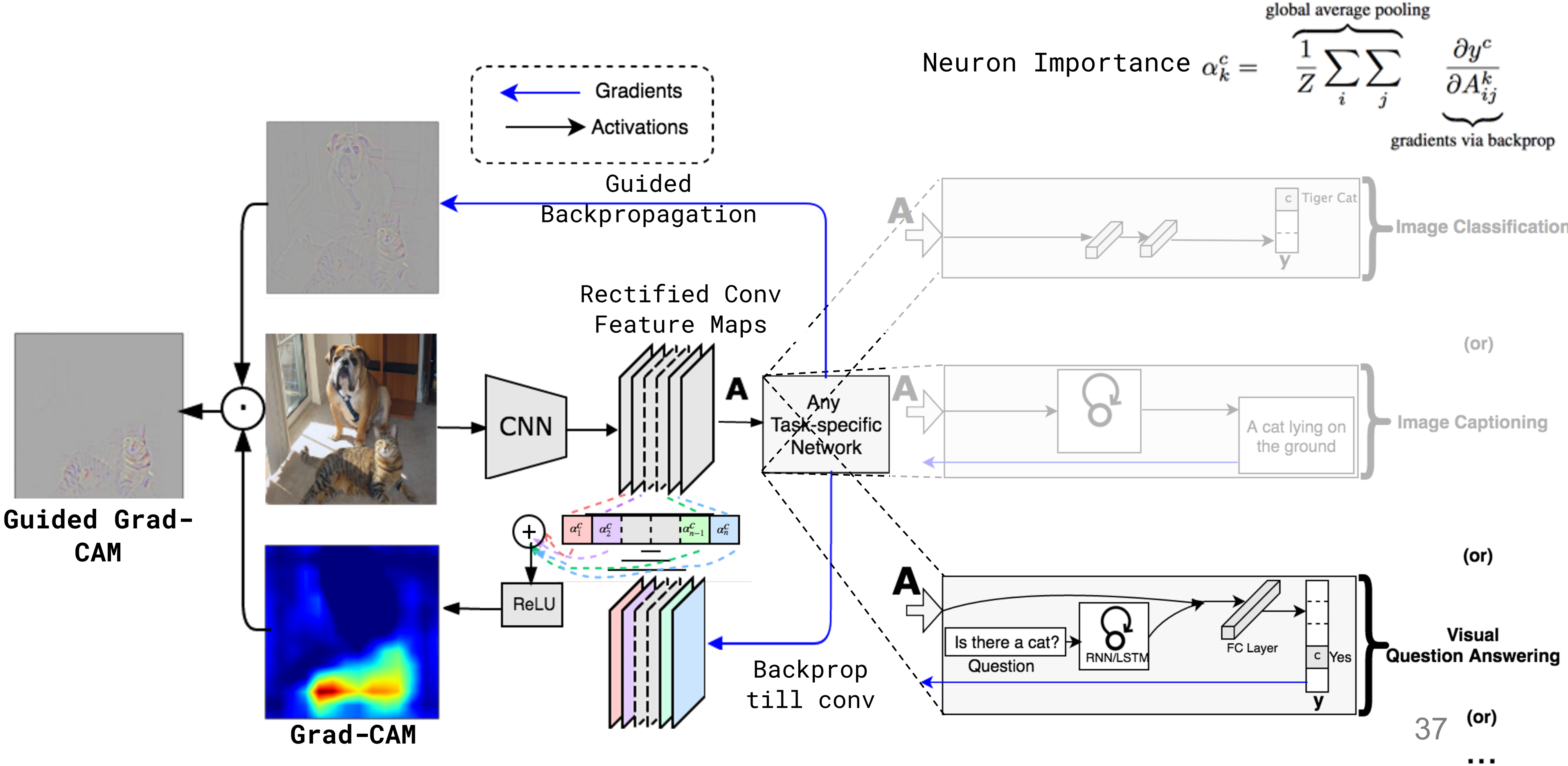
Lower



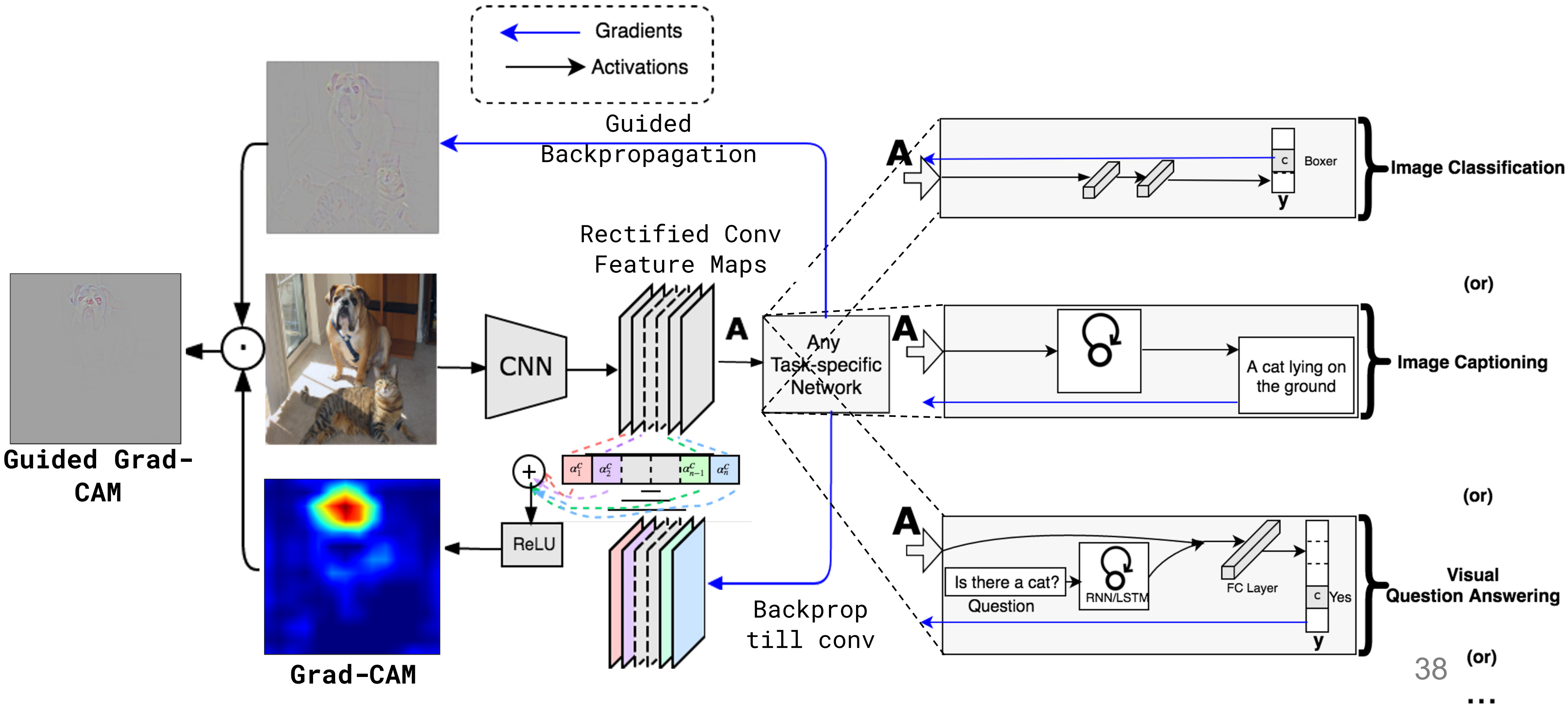
Higher

- Last convolutional layer forms a best compromise between high-level semantics and detailed spatial resolution

Guided Grad-CAM



Guided Grad-CAM



Interesting findings with Grad-CAM

- Even simple non-attention based CNN + LSTM models learn to look at appropriate regions

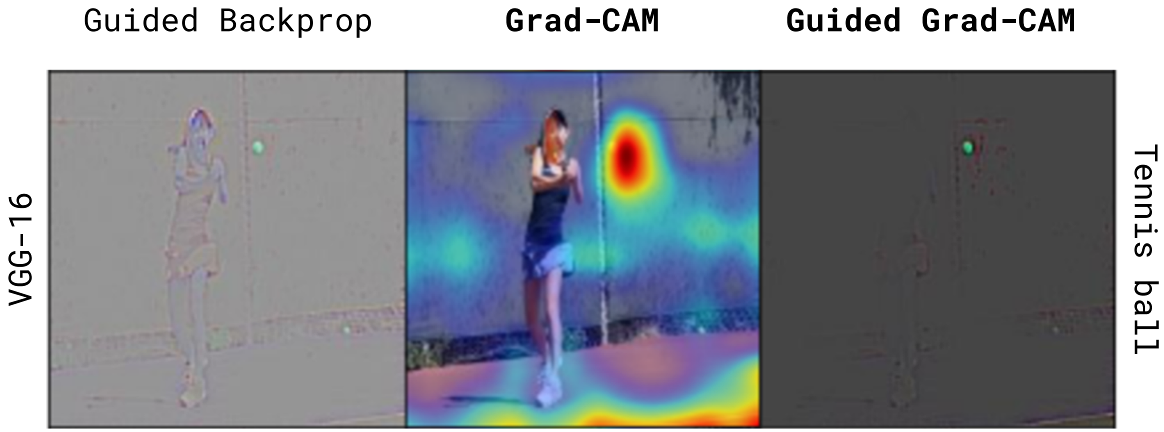
Grad-CAM for captioning

A group of people flying kites on a beach

A man is sitting at a table with a pizza

Grad-CAM for VQA

What is the person hitting?

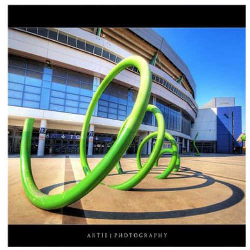


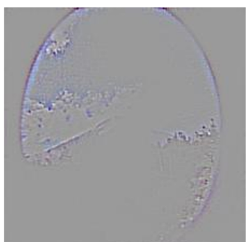
Even simple non-attention based CNN+LSTM models attend to appropriate regions

Interesting findings with Grad-CAM

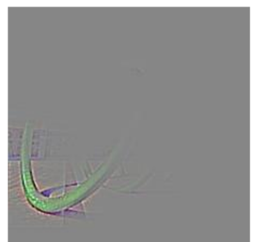
- Even simple non-attention based CNN + LSTM models learn to look at appropriate regions
- Unreasonable predictions often have reasonable explanations

Analyzing Failure modes with Grad-CAM



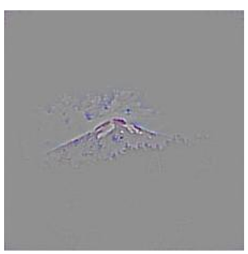


Predicted: *Car mirror*

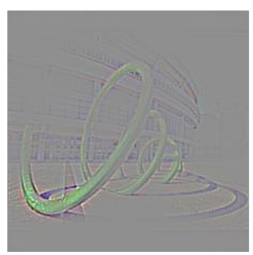


Predicted: *Vine snake*

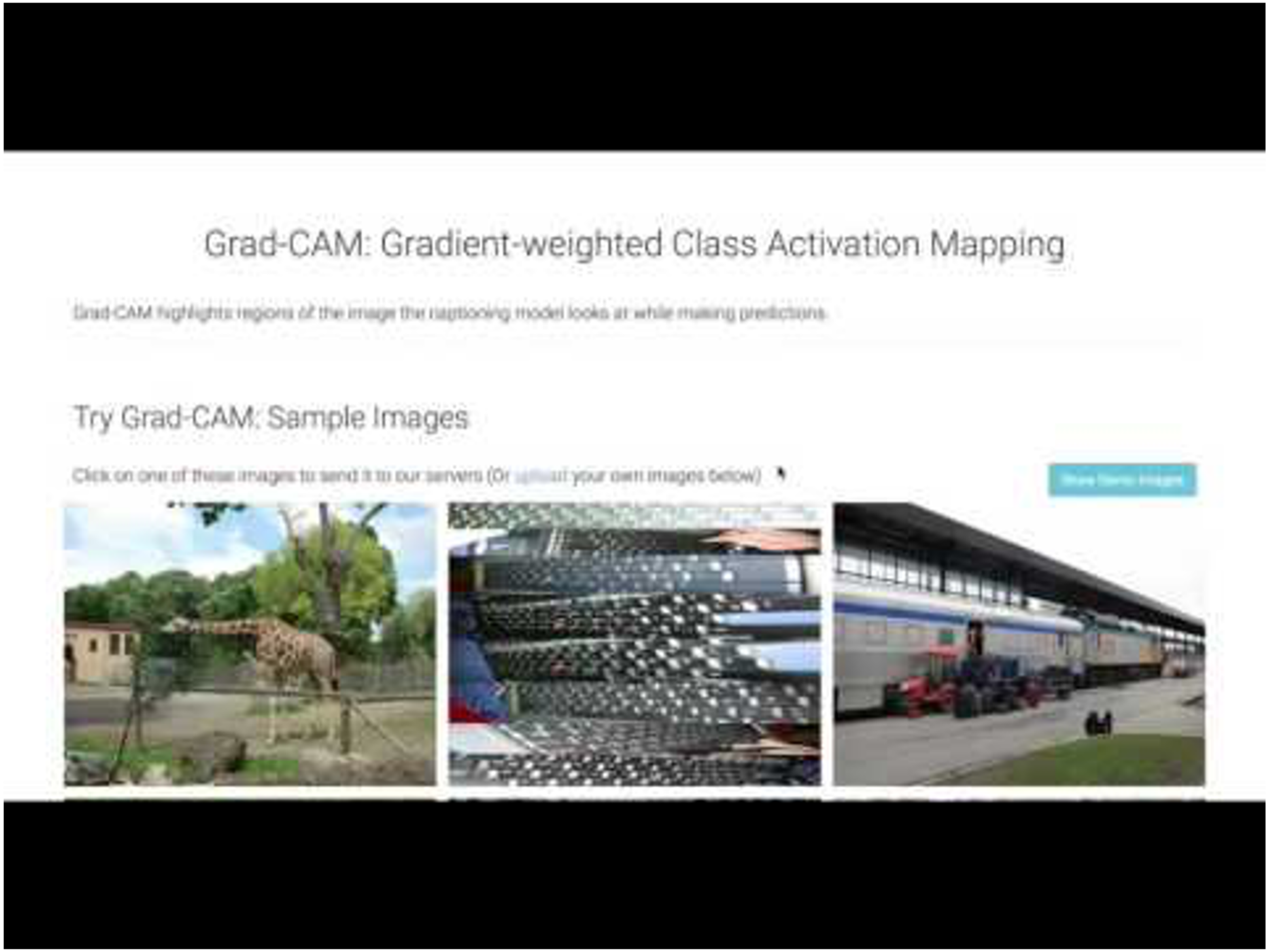
Even unreasonable predictions have reasonable explanations



Ground-truth: *Volcano*



Ground-truth: *coil*


Grad-CAM: Gradient-weighted Class Activation Mapping
Grad-CAM highlights regions of the image the captioning model looks at while making predictions.
Try Grad-CAM: Sample Images
Click on one of these images to send it to our servers (Or upload your own images below)

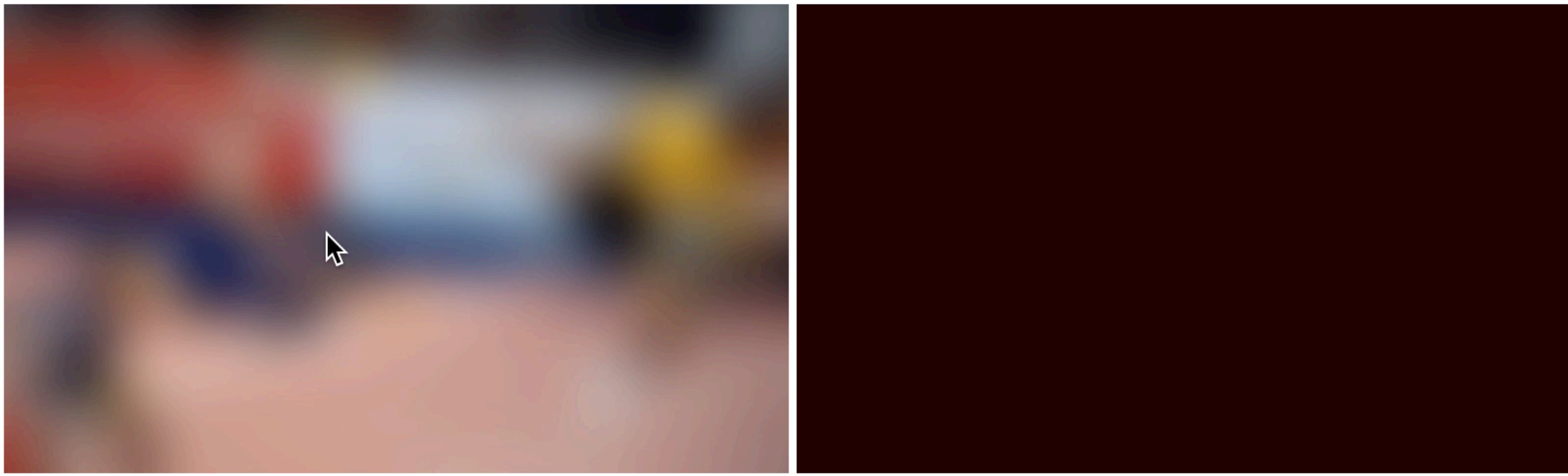
Plan for Today

- What do individual neurons look for in images?
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
- **How pixels affect decisions?**
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations
- **Do CNNs look at same regions as humans?**
  - How to evaluate visualizations?
- Can we synthesize network-specific images?
  - Creating “prototypical” images for a class
  - Creating adversarial images
  - Deep dream
  - Feature inversion

Do CNNs look at same regions as humans?

VQA-HAT (Human ATtention)

Question: How many players are visible in the image?

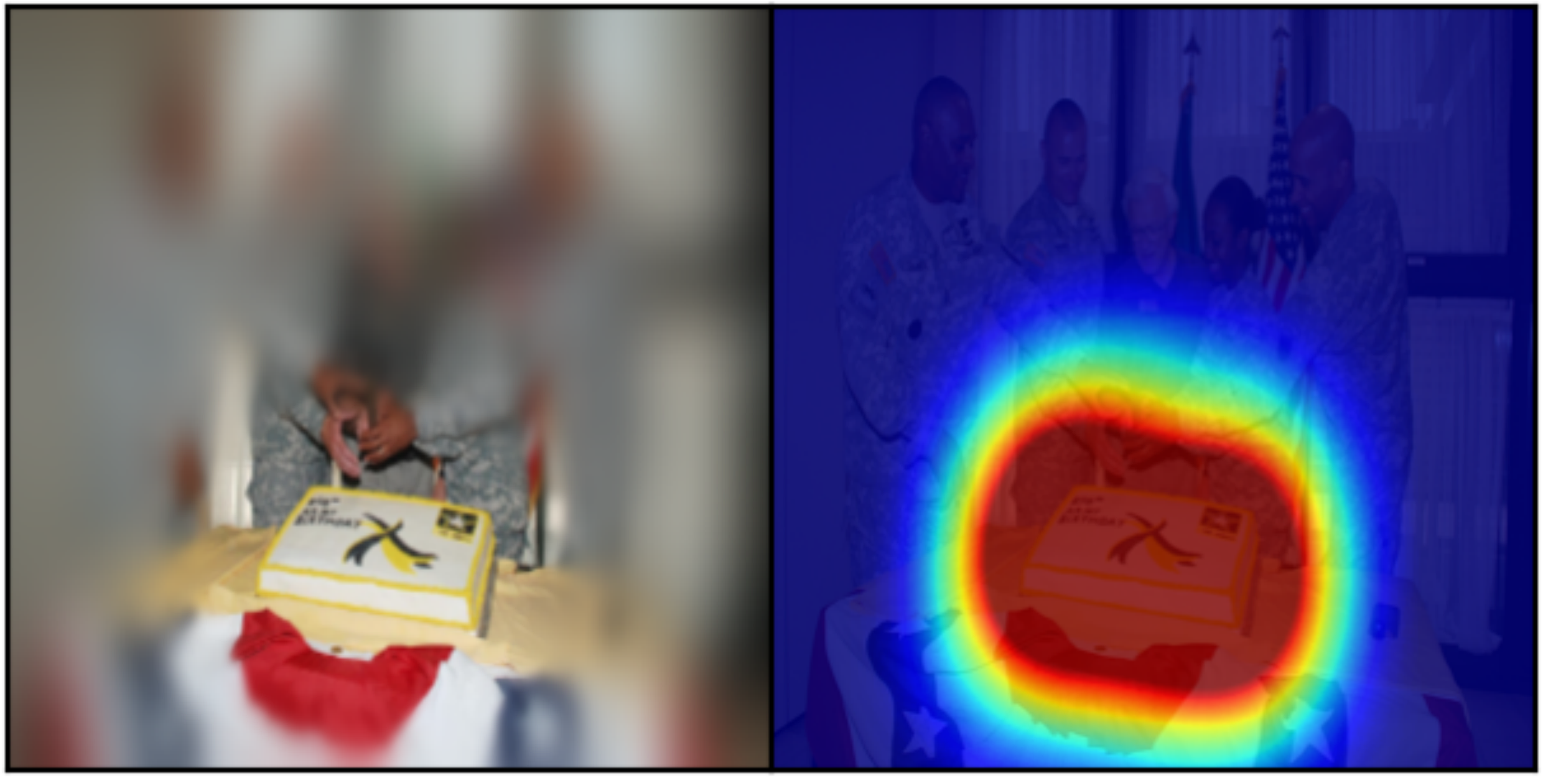


BLUR IMAGE

Answer:

3

VQA-HAT (Human ATtention)

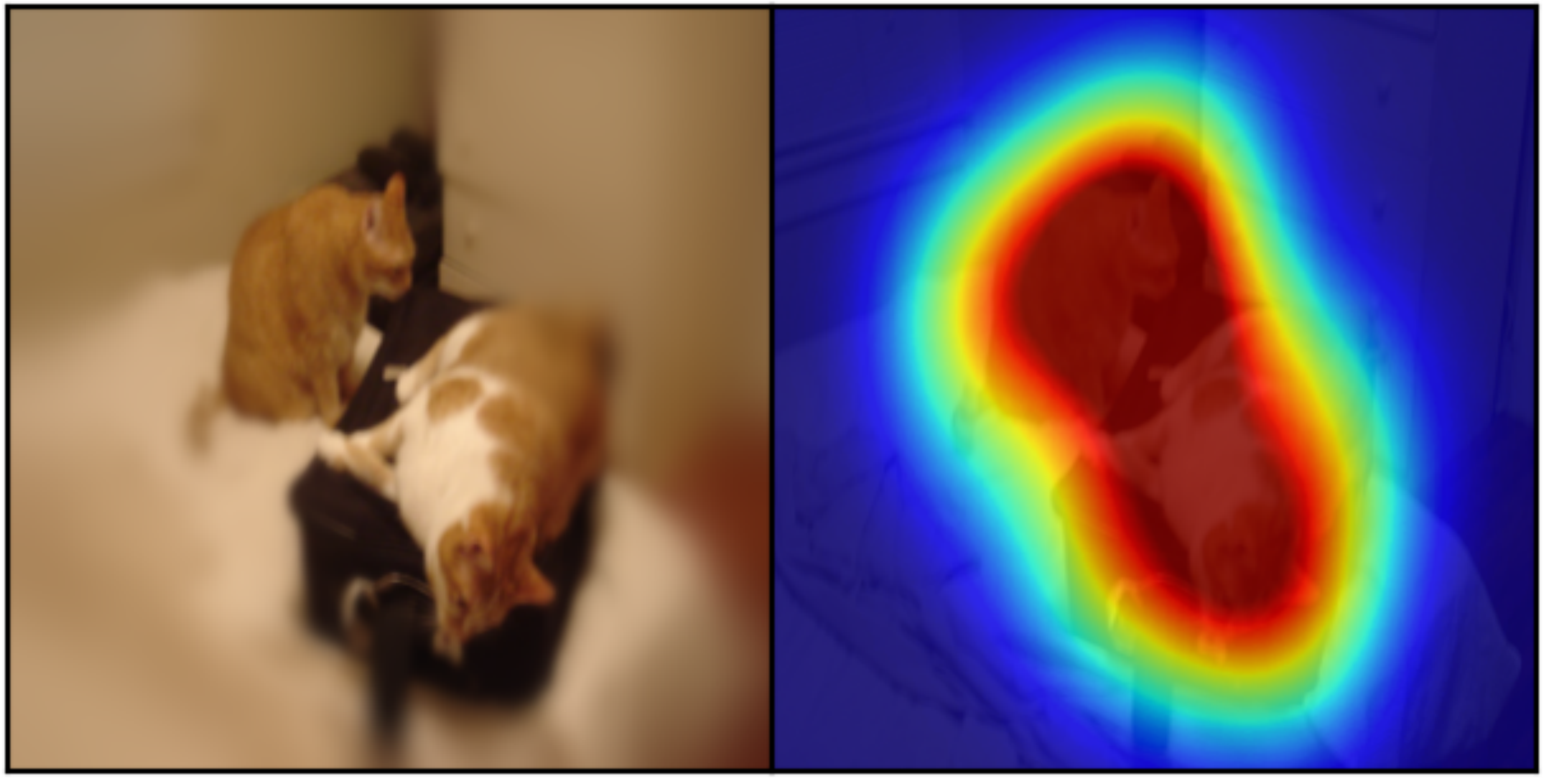


What food is on the table? Cake

VQA-HAT (Human ATtention)

What animal is she riding? Horse

VQA-HAT (Human ATtention)

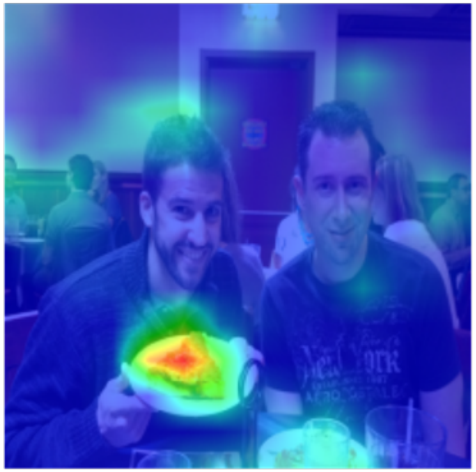


What number of cats are laying on the bed? 2

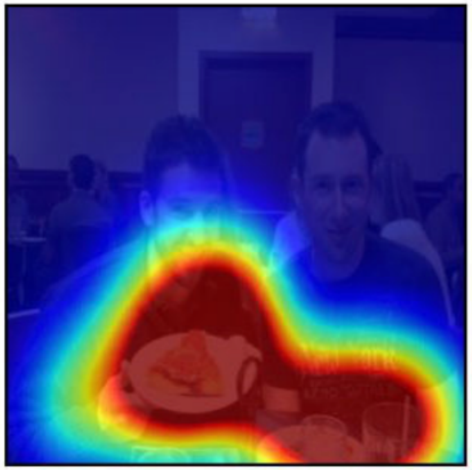
Are Grad-CAM explanations human-like?

- 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

- What do individual neurons look for in images?
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
- How pixels affect decisions?
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations
- Do CNNs look at same regions as humans?
  - How to evaluate visualizations?
- **How can we synthesize images to see what networks are looking for?**
  - Creating “prototypical” images for a class
  - Creating adversarial images
  - Deep dream: amplifying detected features
  - Feature inversion

How can we synthesize images to see what networks are looking for?

Generating prototypical images for a class

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

$$I^* = \arg\max_I \; f(I) + R(I)$$

Neuron value — $f(I)$

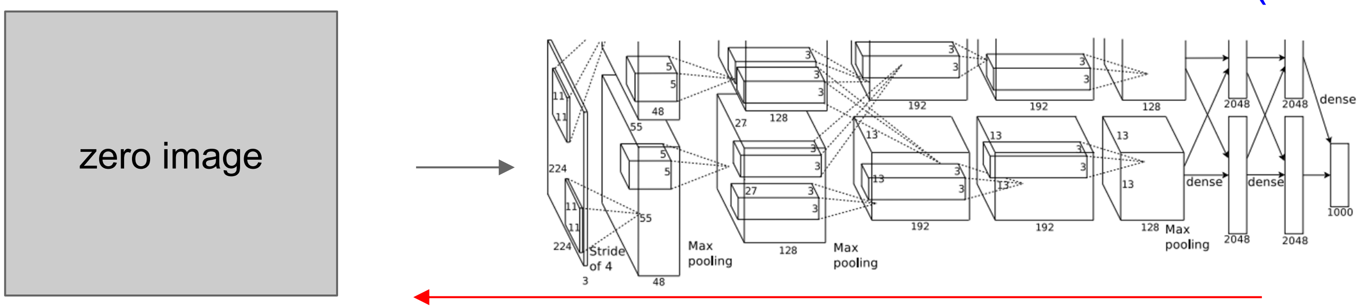
Natural image regularizer — $R(I)$

Visualizing CNN features: Gradient Ascent on Pixels

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

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

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

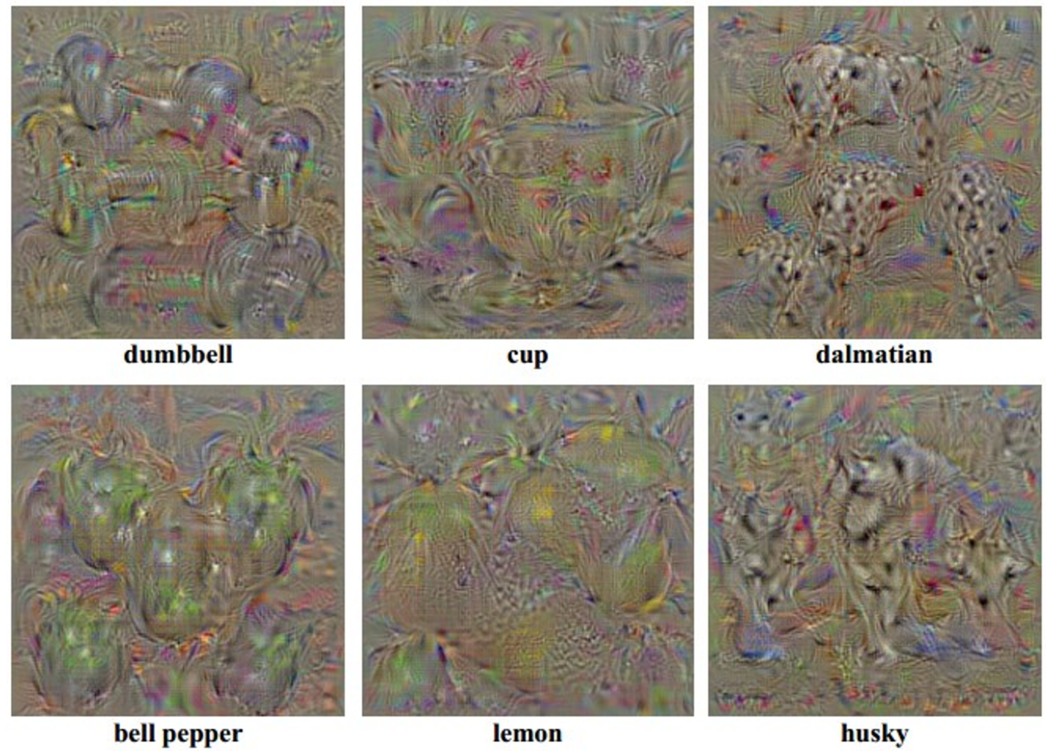
Simple regularizer: Penalize L2
norm of generated image

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

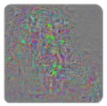
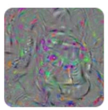
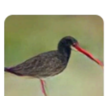
Visualizing CNN features: Gradient Ascent on Pixels

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

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.

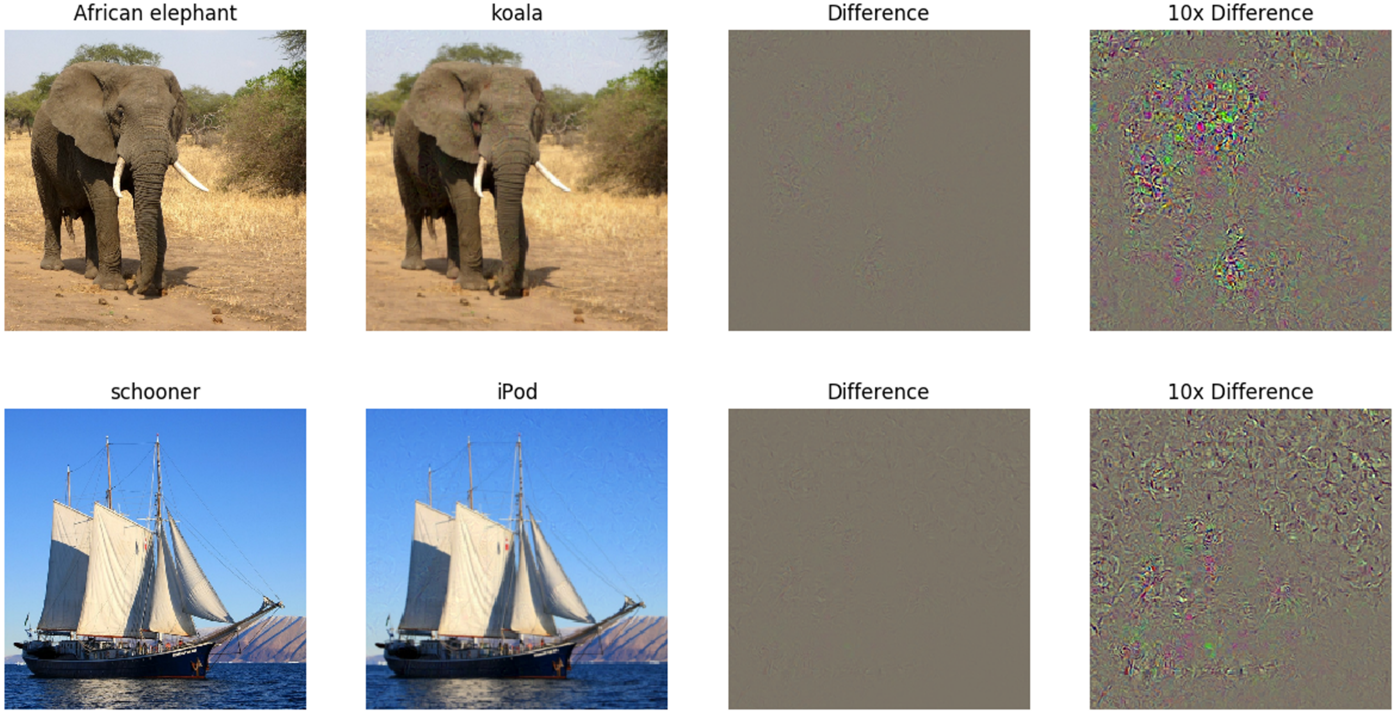
		Weak Regularization avoids misleading correlations, but is less connected to real use.			**Strong Regularization** gives more realistic examples at risk of misleading correlations.	
		Unregularized	**Frequency Penalization**	**Transformation Robustness**	**Learned Prior**	**Dataset Examples**
	**Erhan, *et al.*, 2009** [3] Introduced core idea. Minimal regularization.	■	□	□	□	□
	**Szegedy, *et al.*, 2013** [11] Adversarial examples. Visualizes with dataset examples.	■	□	□	□	■
	**Mahendran & Vedaldi, 2015** [7] Introduces total variation regularizer. Reconstructs input from representation.	□	■	□	□	□
	**Nguyen, *et al.*, 2015** [14] Explores counterexamples. Introduces image blurring.	□	■	□	□	□
	**Mordvintsev, *et al.*, 2015** [4] Introduced jitter & multi-scale. Explored GMM priors for classes.	□	□	■	■	□
	**Øygard, *et al.*, 2015** [15] Introduces gradient blurring. (Also uses jitter.)	□	■	■	□	□
	**Tyka, *et al.*, 2016** [16] Regularizes with bilateral filters. (Also uses jitter.)	□	■	■	□	□
	**Mordvintsev, *et al.*, 2016** [17] Normalizes gradient frequencies. (Also uses jitter.)	□	■	■	□	□
	**Nguyen, *et al.*, 2016** [18] Paramaterizes images with GAN generator.	□	□	□	■	□
	**Nguyen, *et al.*, 2016** [10] Uses denoising autoencoder prior to make a generative model.	□	□	□	■	□

Can neural networks be fooled?

Fooling Images / Adversarial Examples

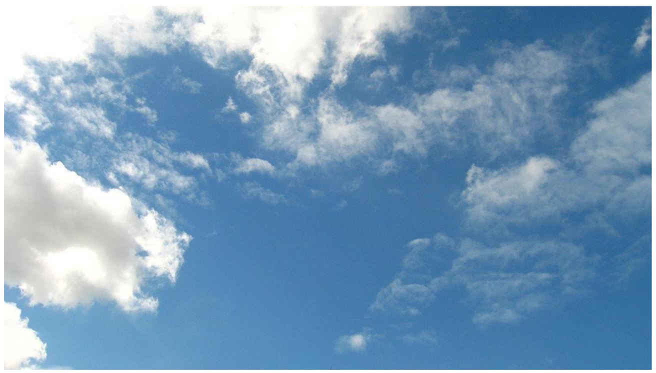
(1)Start from an arbitrary image
(2)Pick an arbitrary class
(3)Modify the image to maximize the class
(4)Repeat until network is fooled

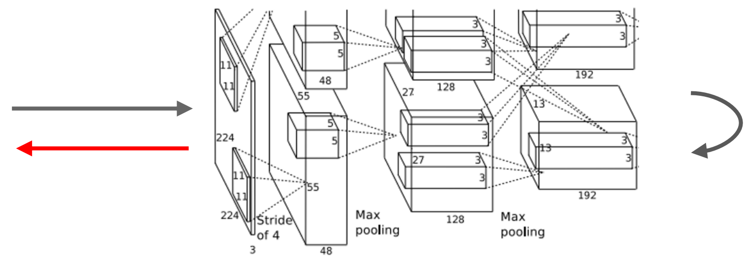
Fooling Images / Adversarial Examples



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





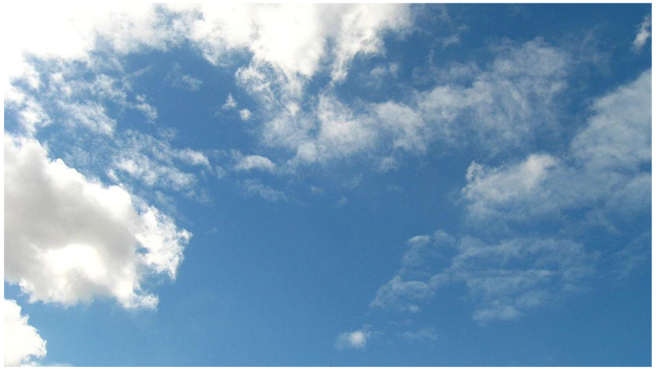
Choose an image and a layer in a CNN; repeat:

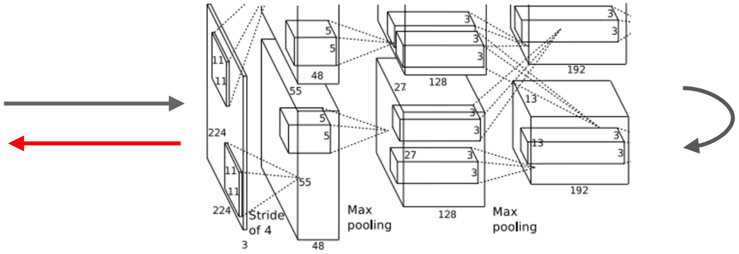
1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer *equal to its activation*
3. Backward: Compute gradient on image
4. Update image

Mordvintsev, Olah, and Tyka, "Inceptionism: Going Deeper into Neural Networks", Google Research Blog.

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

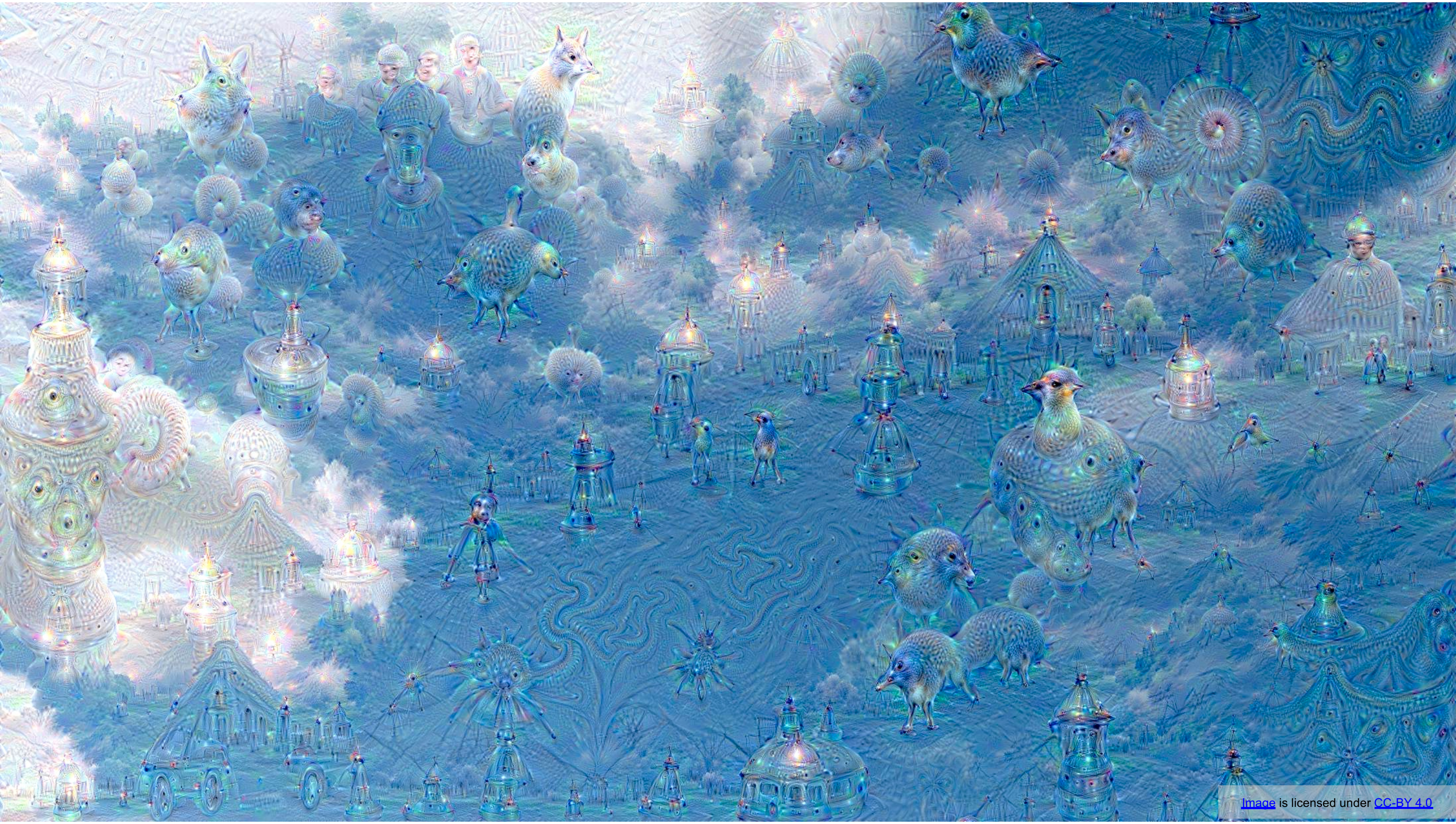
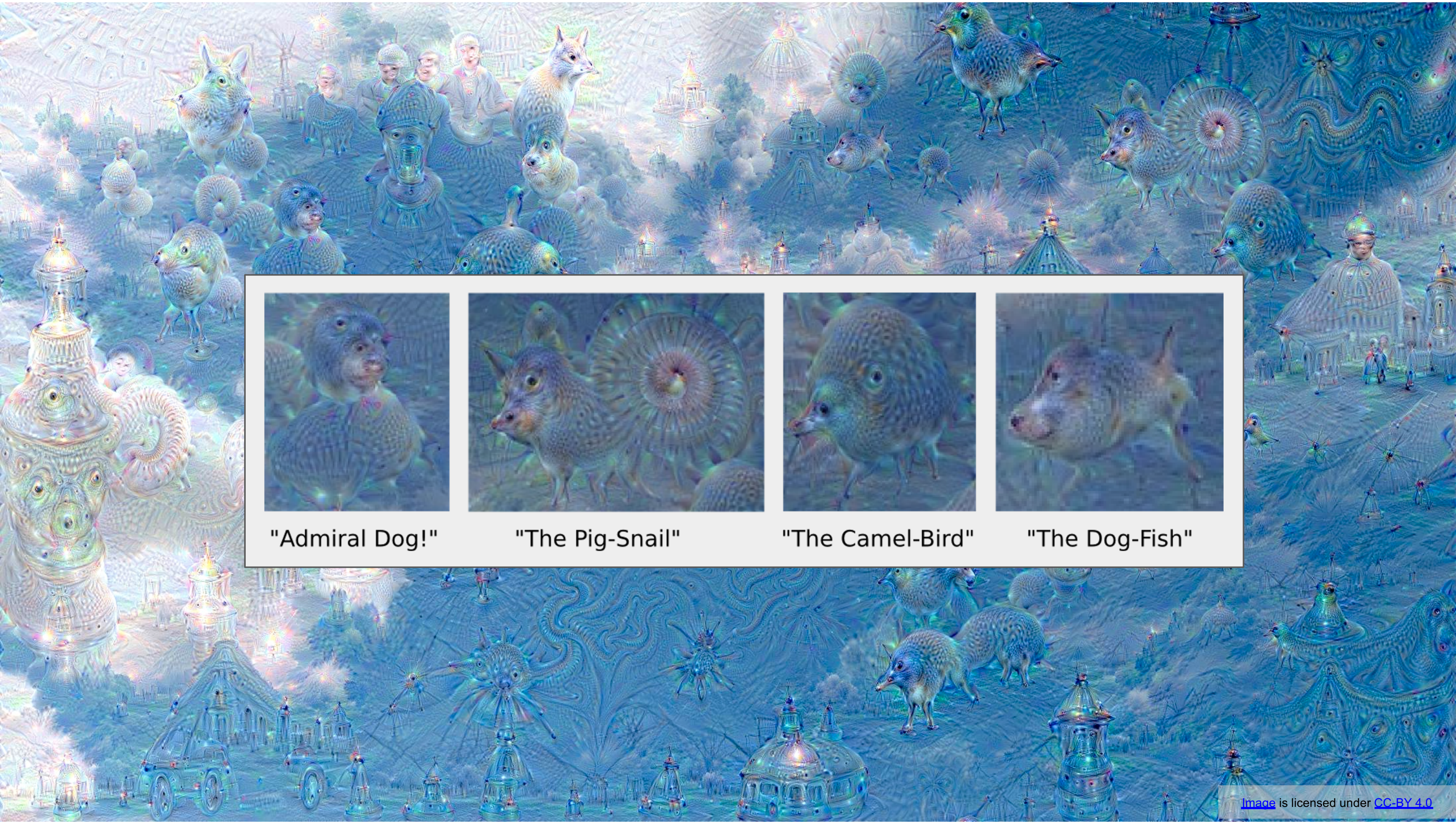


Image is licensed under CC-BY 4.0


"Admiral Dog!"
"The Pig-Snail"
"The Camel-Bird"
"The Dog-Fish"

Given the feature vector can you reconstruct the image?

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 (Φ_0)

Features of new image ($\Phi(\mathbf{x})$)

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

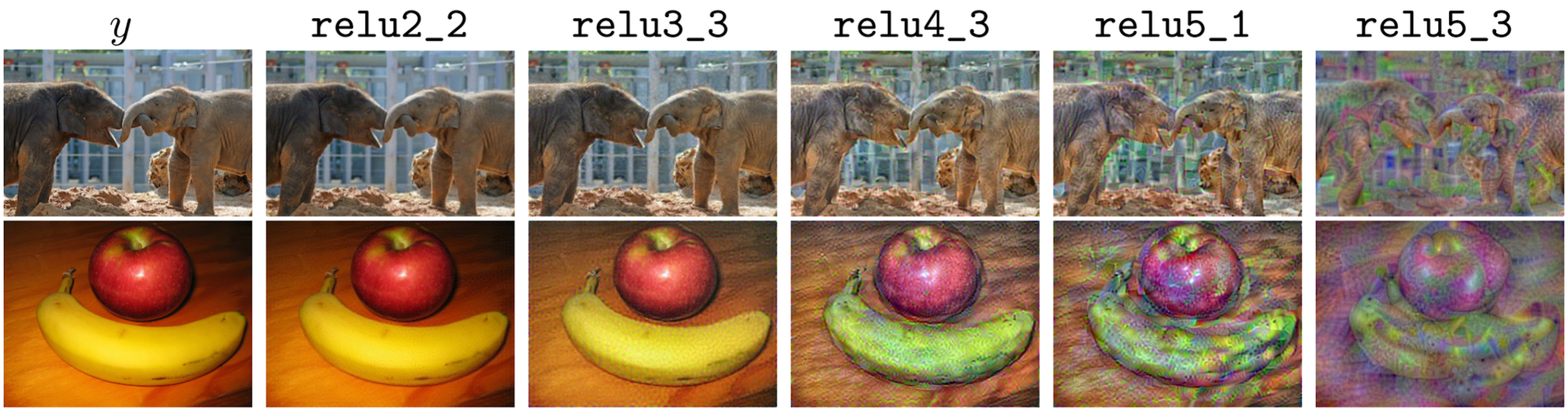
$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j}\left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2\right)^{\frac{\beta}{2}}$$

Total Variation regularizer (encourages spatial smoothness)

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

Feature Inversion

Reconstructing from different layers of VGG-16



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.

What we covered today

- What do individual neurons look for in images?
  - Visualizing filters
  - Last layer embeddings
  - Visualizing activations
  - Maximally activating patches
- How pixels affect model decisions?
  - Occlusion maps
  - Salient or “important” pixels
    - Gradient-based visualizations
- Do CNNs look at same regions as humans?
  - How to evaluate visualizations?
- How can we synthesize images to see what networks are looking for?
  - Creating “prototypical” images for a class
  - Creating adversarial images
  - Deep dream
  - Feature inversion

Thank you

ramprs.github.io

rselvaraju@salesforce.com