

Lecture 8:

Interpretability

Administrative: Assignment 2

Due 1/2 11:59pm

- Multi-layer Neural Networks,
- Image Features,
- Optimizers

Project Proposal

Due Today! 11:59pm

Administrative: Fridays

This Friday

Quantization

Administrative: Midterm #1

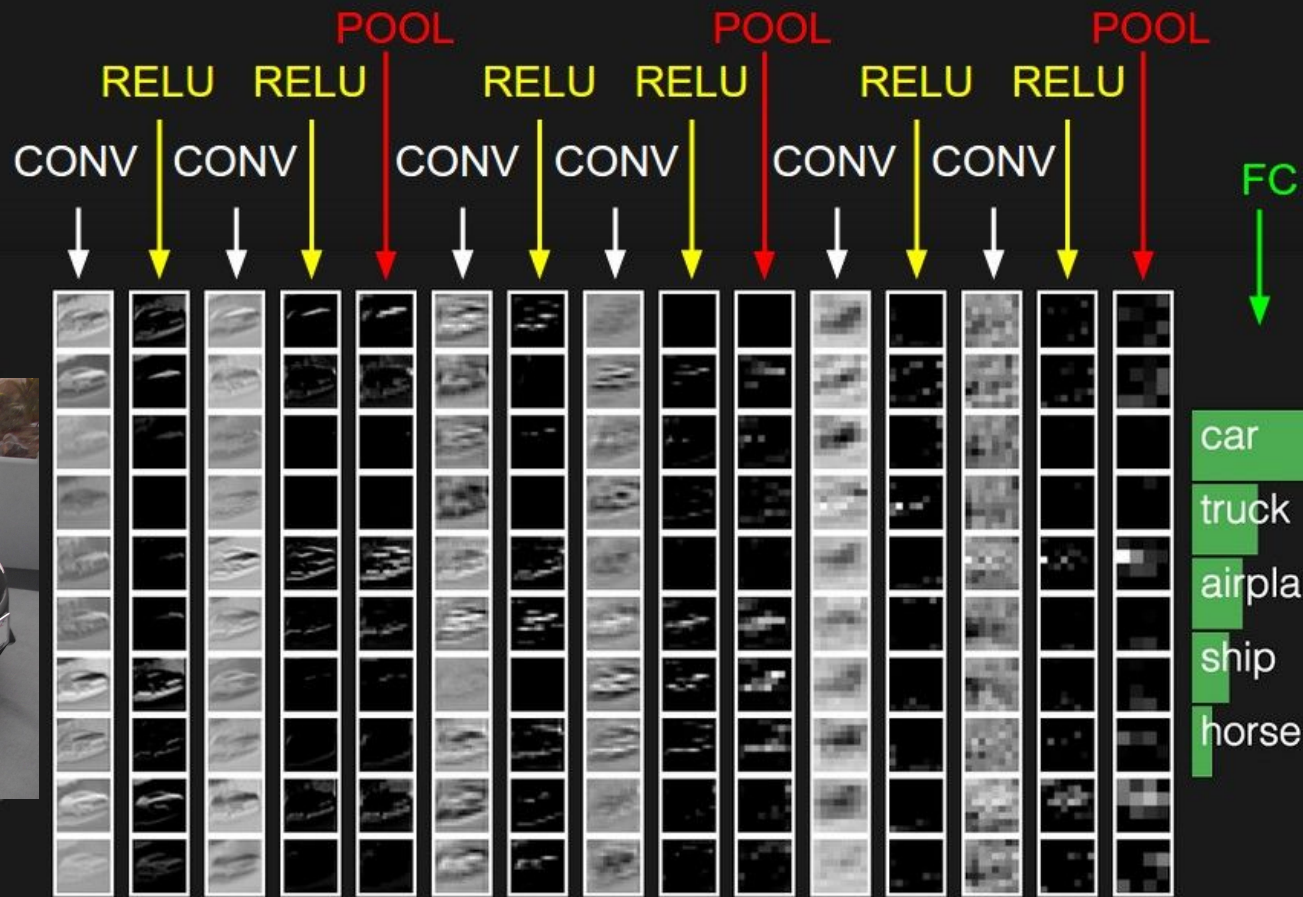
In class, Feb 5

Covers up to lecture 8

Allowed:

- 1 double sided handwritten “cheat sheet”
- 1 non-graphing calculator

Many practice exams in the [Exam Archive](#)



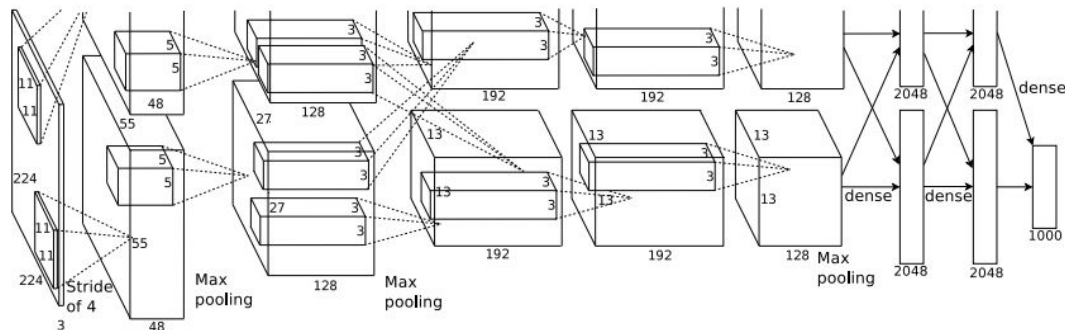
Visualizing and Understanding

Today: 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.

Today's agenda

Visualizing what models have learned:

- Visualizing filters
- Visualizing final layer features
- Visualizing activations

Understanding input pixels

- Identifying important pixels
- Saliency via backprop
- Guided backprop to generate images
- Gradient ascent to visualize features

Adversarial perturbations

Concept Vectors

Today's agenda

Visualizing what models have learned:

- Visualizing filters
- Visualizing final layer features
- Visualizing activations

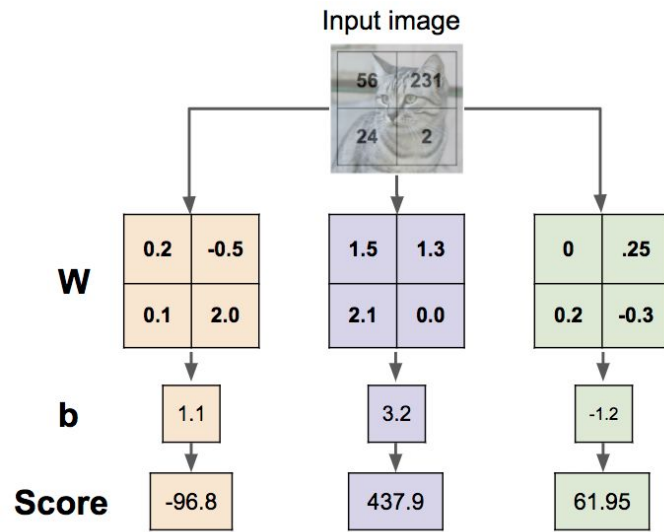
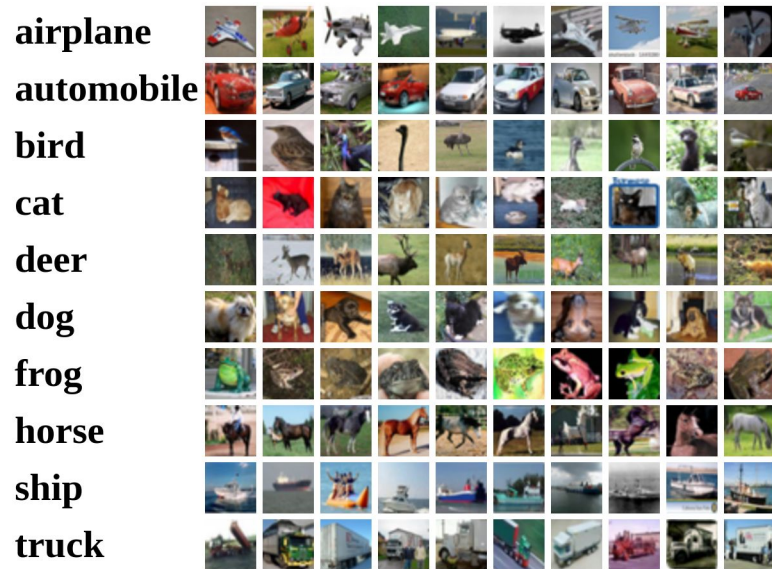
Understanding input pixels

- Identifying important pixels
- Saliency via backprop
- Guided backprop to generate images
- Gradient ascent to visualize features

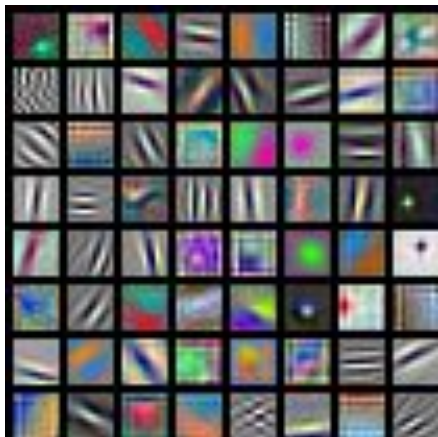
Adversarial perturbations

Concept Vectors

Interpreting a Linear Classifier: Visual Viewpoint

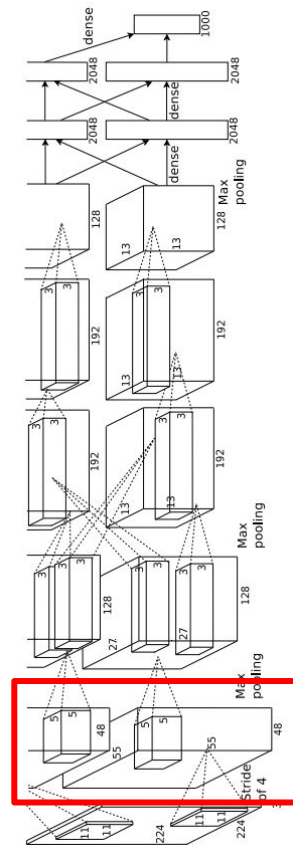


First Layer: Visualize Filters

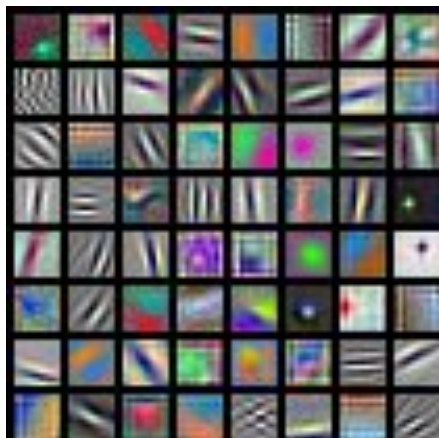


AlexNet:
64 x 3 x 11 x 11

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



First Layer: Visualize Filters



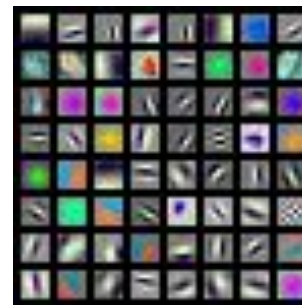
AlexNet:
 $64 \times 3 \times 11 \times 11$



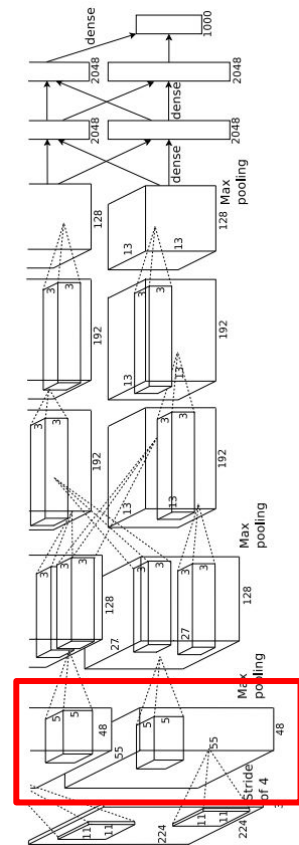
ResNet-18:
 $64 \times 3 \times 7 \times 7$



ResNet-101:
 $64 \times 3 \times 7 \times 7$



DenseNet-121:
 $64 \times 3 \times 7 \times 7$



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

Weights:



layer 1 weights

$16 \times 3 \times 7 \times 7$

Weights:



layer 2 weights

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

Weights:

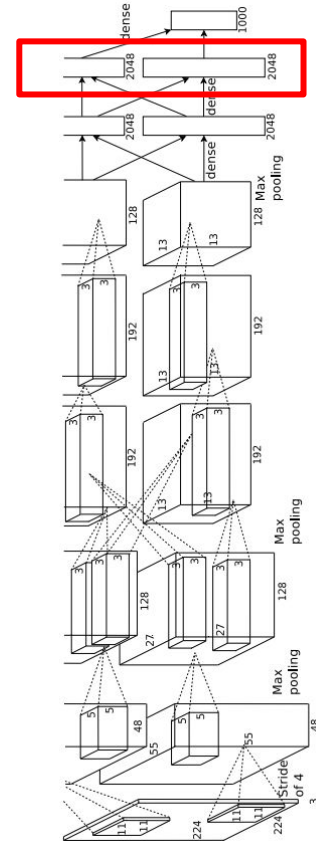


layer 3 weights

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

Last Layer

FC7 layer

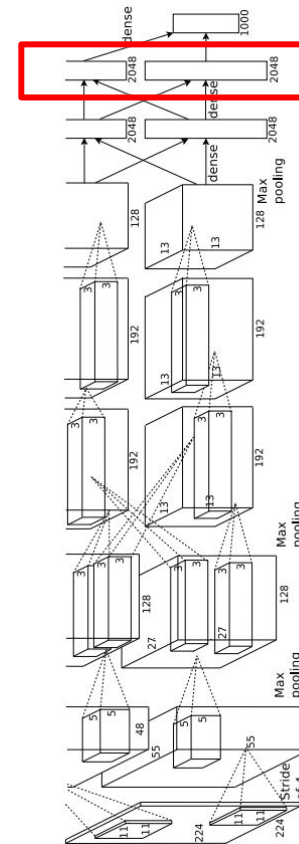


4096-dimensional feature vector for an image
(layer immediately before the classifier)

Run the network on many images, collect the feature vectors

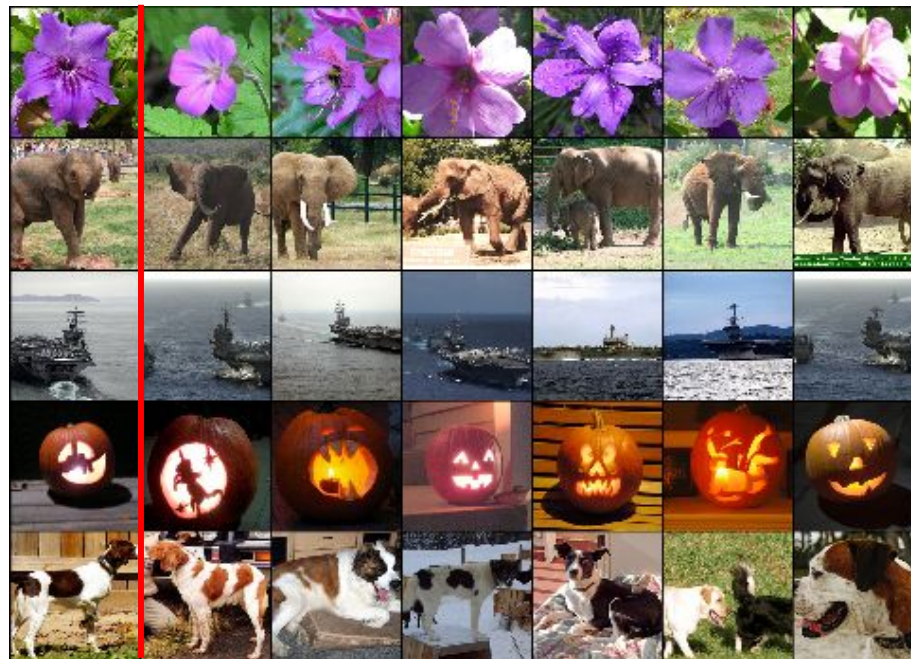
Last Layer: Nearest Neighbors

4096-dim vector



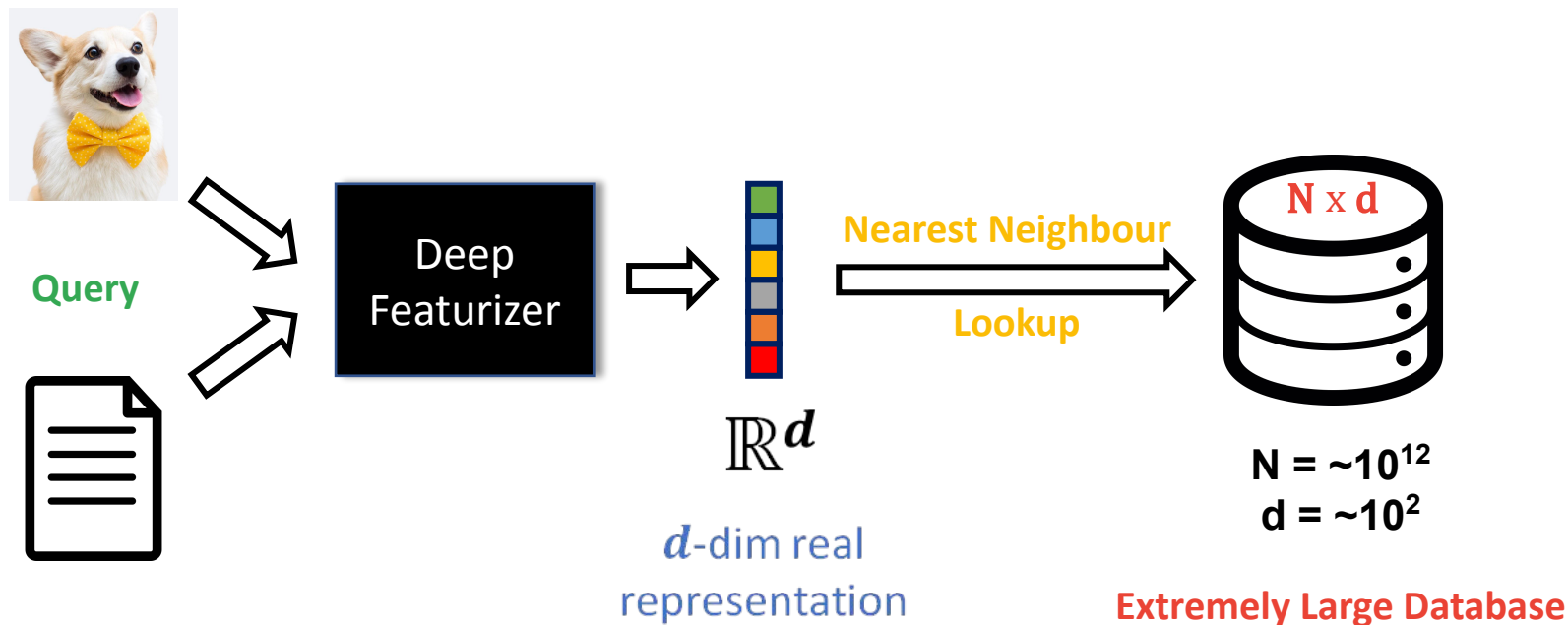
Test image L2 Nearest neighbors in feature space

Recall: Nearest neighbors
in pixel space

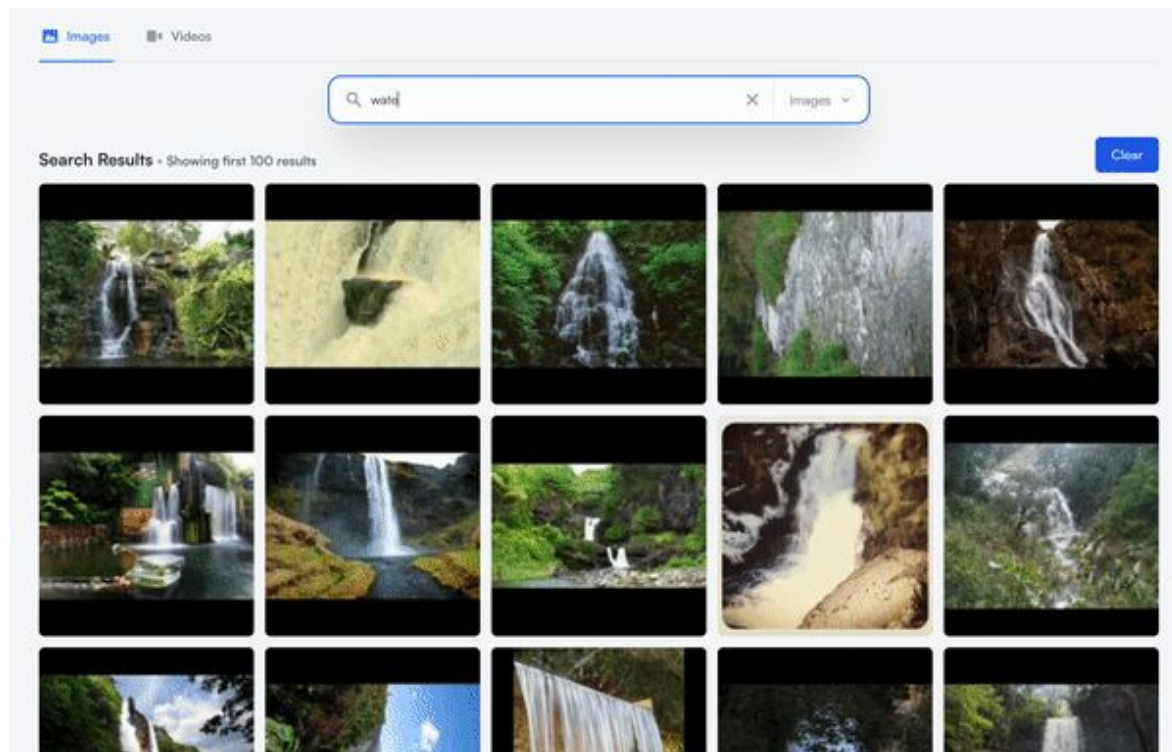


Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figures reproduced with permission.

Last Layer: Learned Metric for “Semantic” Search



Last Layer: Modern Day Search



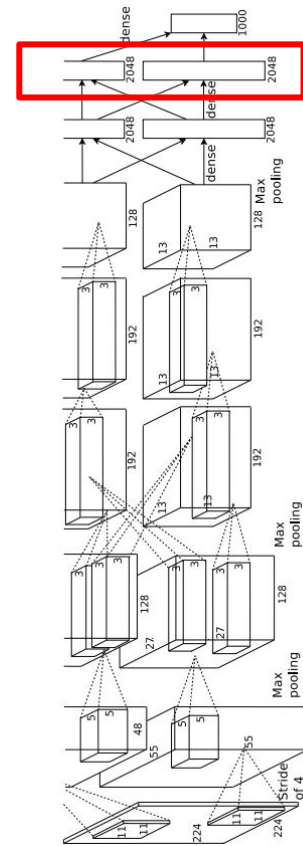
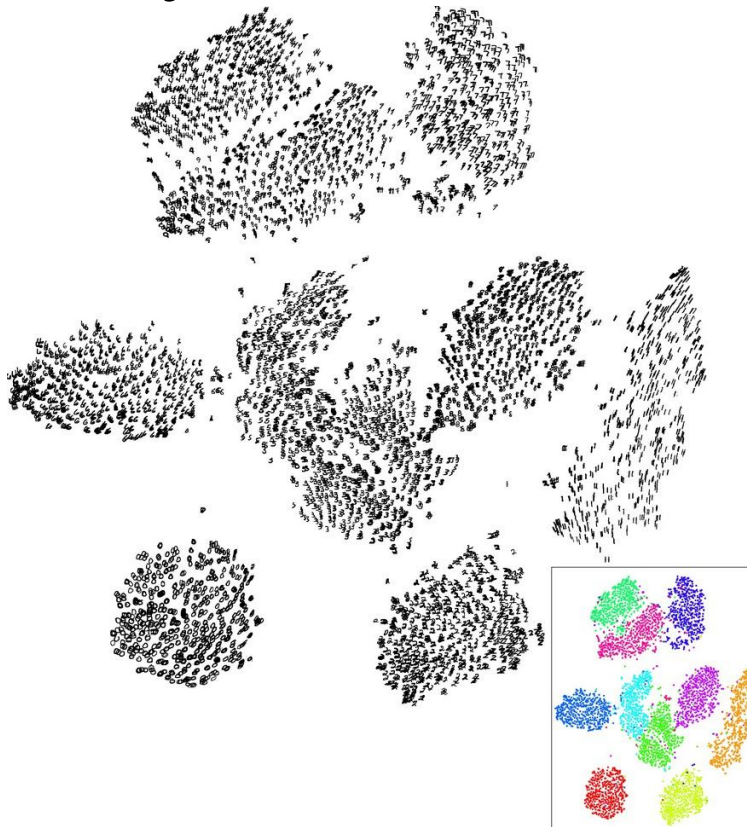
coactive.ai

Last Layer: Dimensionality Reduction

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

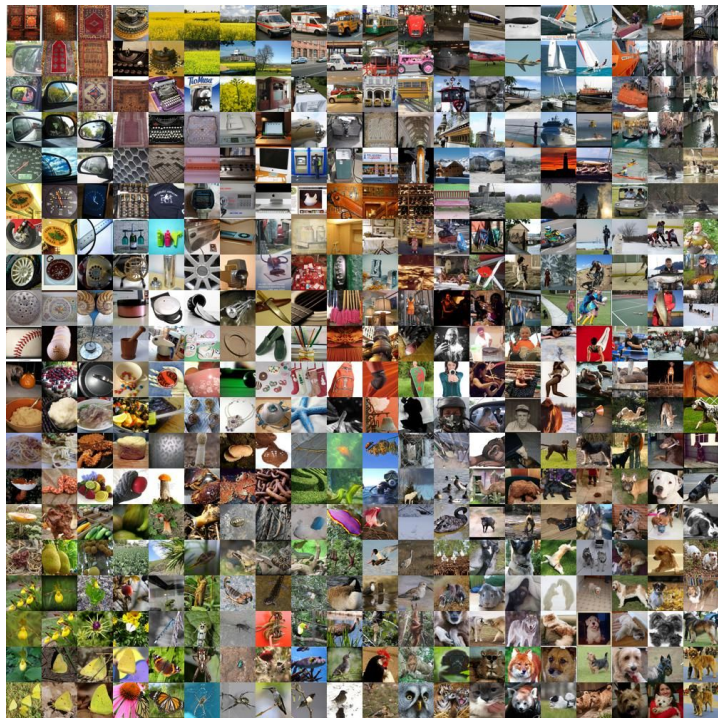
Simple algorithm: Principal Component Analysis (PCA)

More complex: **t-SNE**

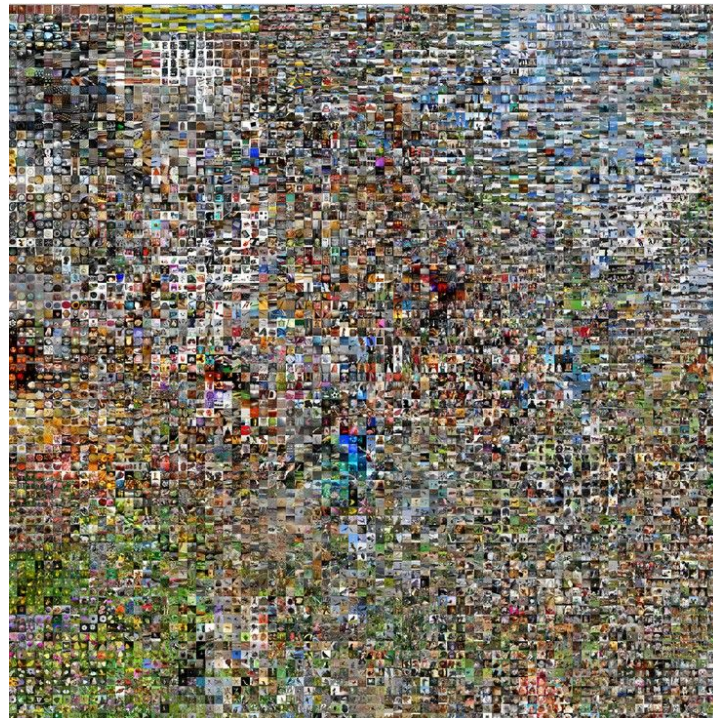


Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008
Figure copyright Laurens van der Maaten and Geoff Hinton, 2008. Reproduced with permission.

Last Layer: Dimensionality Reduction

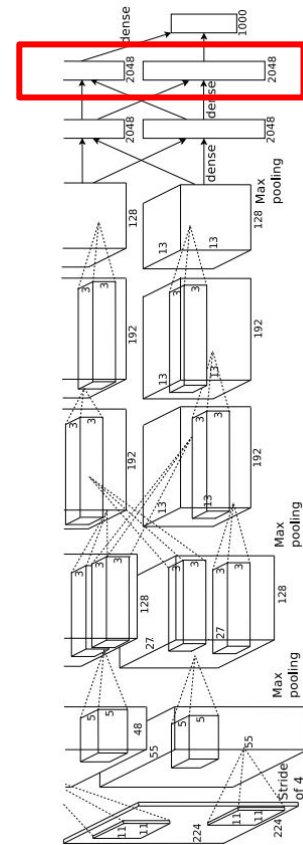


Van der Maaten and Hinton, "Visualizing Data using t-SNE", JMLR 2008
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figure reproduced with permission.



See high-resolution versions at

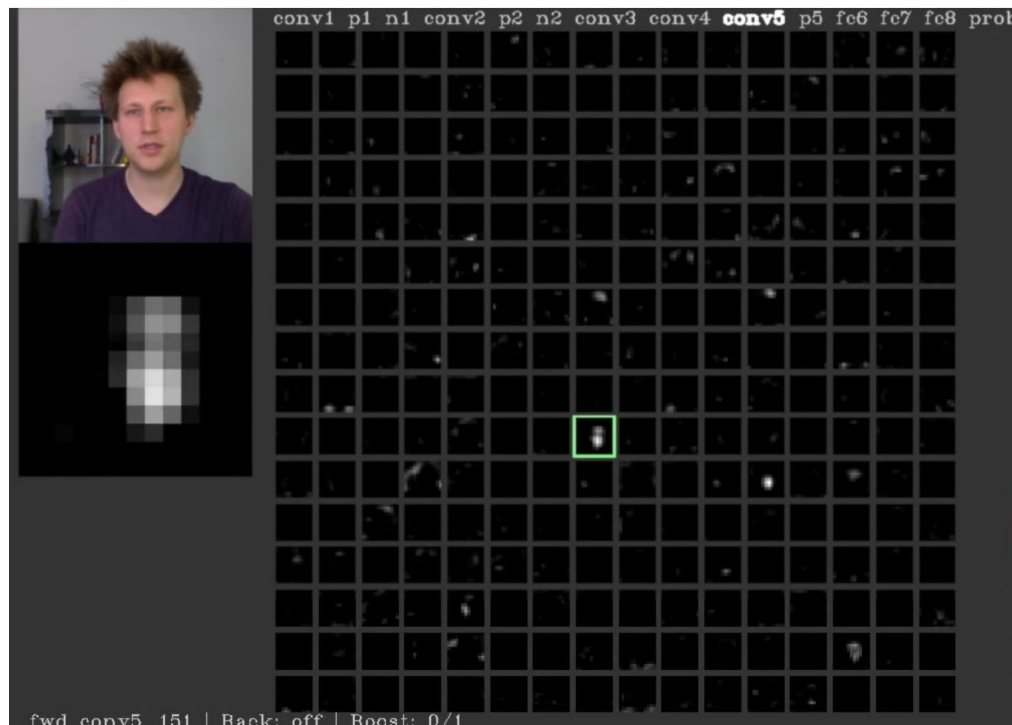
<http://cs.stanford.edu/people/karpathy/cnnembed/>



Visualizing Activations

<https://www.youtube.com/watch?v=AgkflQ4IGaM>

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



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.
Figure copyright Jason Yosinski, 2014. Reproduced with permission.

Today's agenda

Visualizing what models have learned:

- Visualizing filters
- Visualizing final layer features
- Visualizing activations

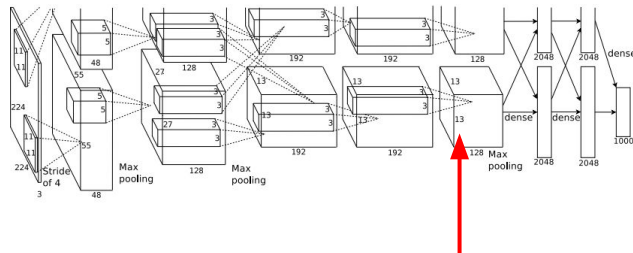
Understanding input pixels

- Identifying important pixels
- Saliency via backprop
- Guided backprop to generate images
- Gradient ascent to visualize features

Adversarial perturbations

Concept Vectors

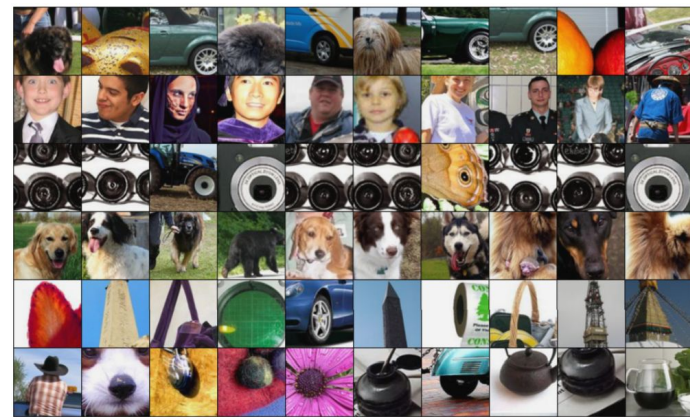
Maximally Activating Patches



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

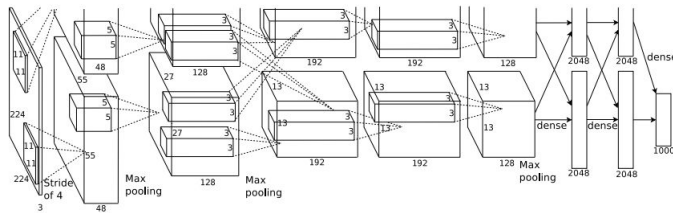
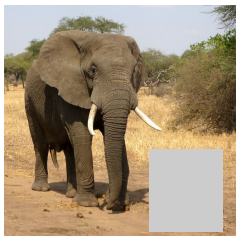
Visualize image patches that correspond to maximal activations



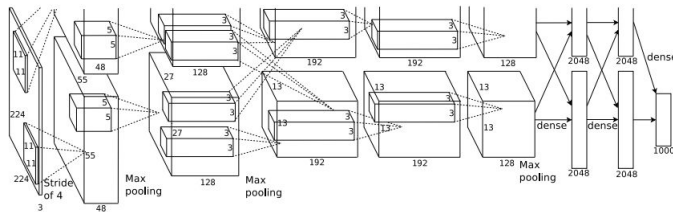
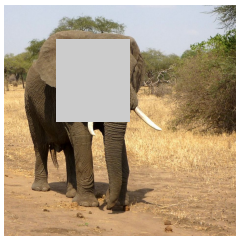
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Which pixels matter: Saliency via Occlusion

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



$$P(\text{elephant}) = 0.95$$



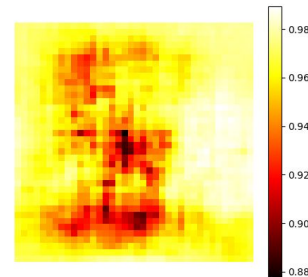
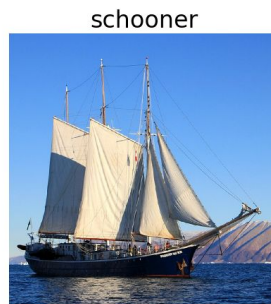
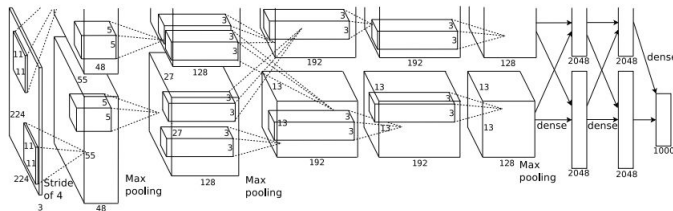
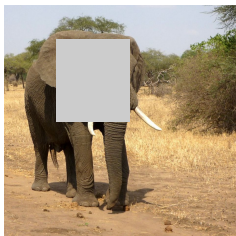
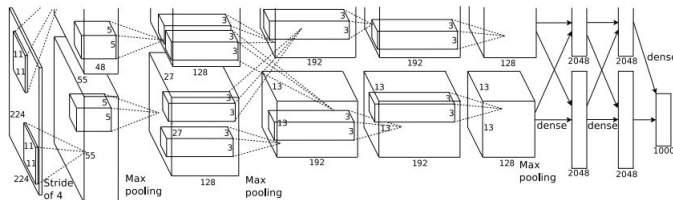
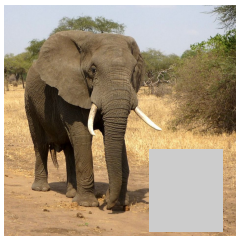
$$P(\text{elephant}) = 0.75$$

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

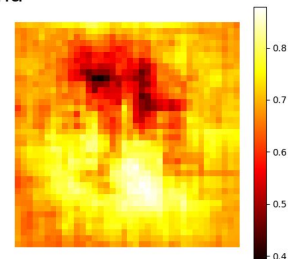
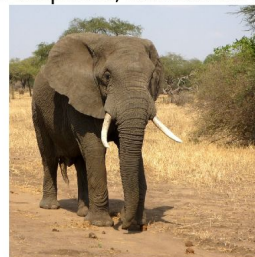
[Boat image](#) is [CC0 public domain](#)
[Elephant image](#) is [CC0 public domain](#)
[Go-Karts image](#) is [CC0 public domain](#)

Which pixels matter: Saliency via Occlusion

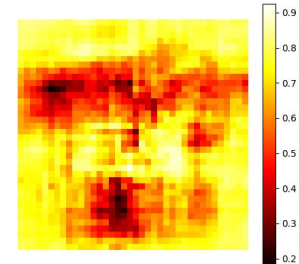
Mask part of the image before feeding to CNN,
check how much predicted probabilities change



African elephant, *Loxodonta africana*



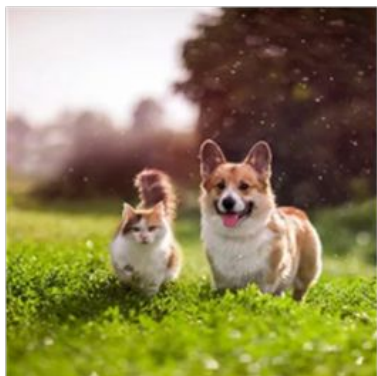
go-kart



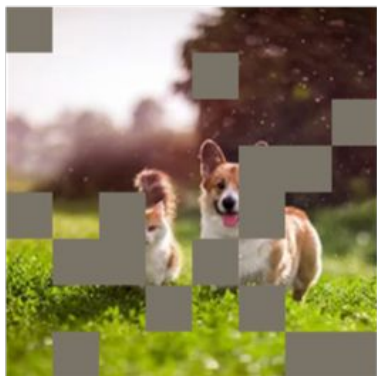
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

[Boat image](#) is [CC0 public domain](#)
[Elephant image](#) is [CC0 public domain](#)
[Go-Karts image](#) is [CC0 public domain](#)

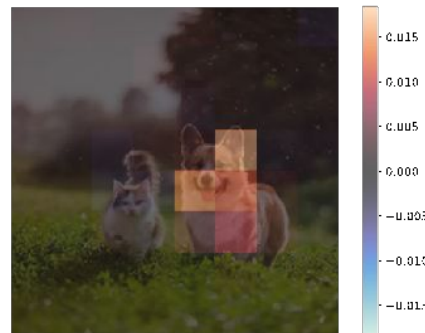
Saliency via Occlusion: Shapley Values



$$P(\text{corgi}) = 0.99$$



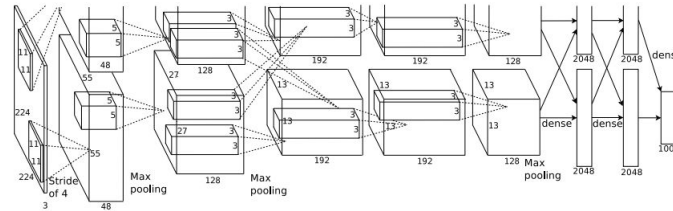
$$P(\text{corgi}) = 0.8$$



Credit: Ian Covert; Lundberg & Lee 2017

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities

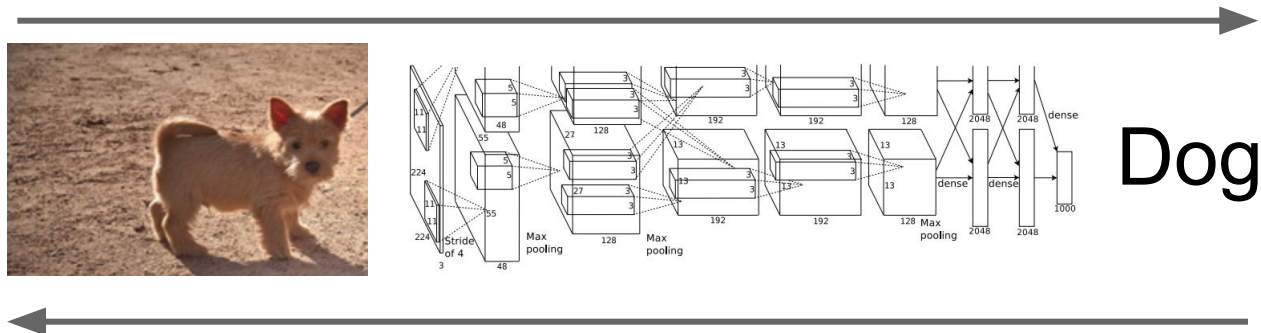


Dog

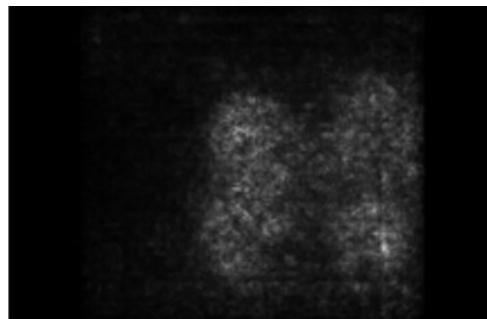
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities

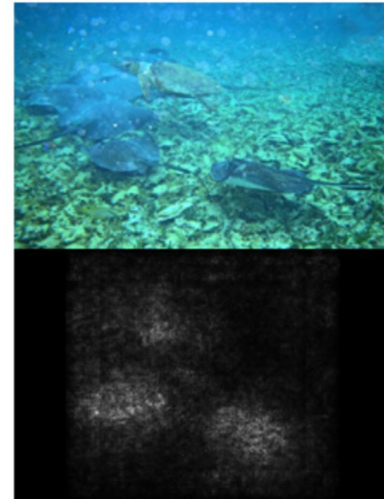
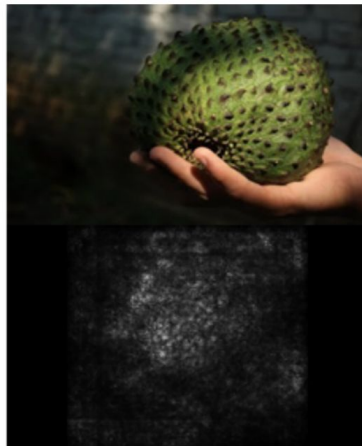
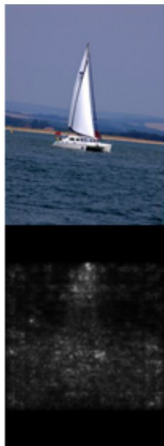


Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels



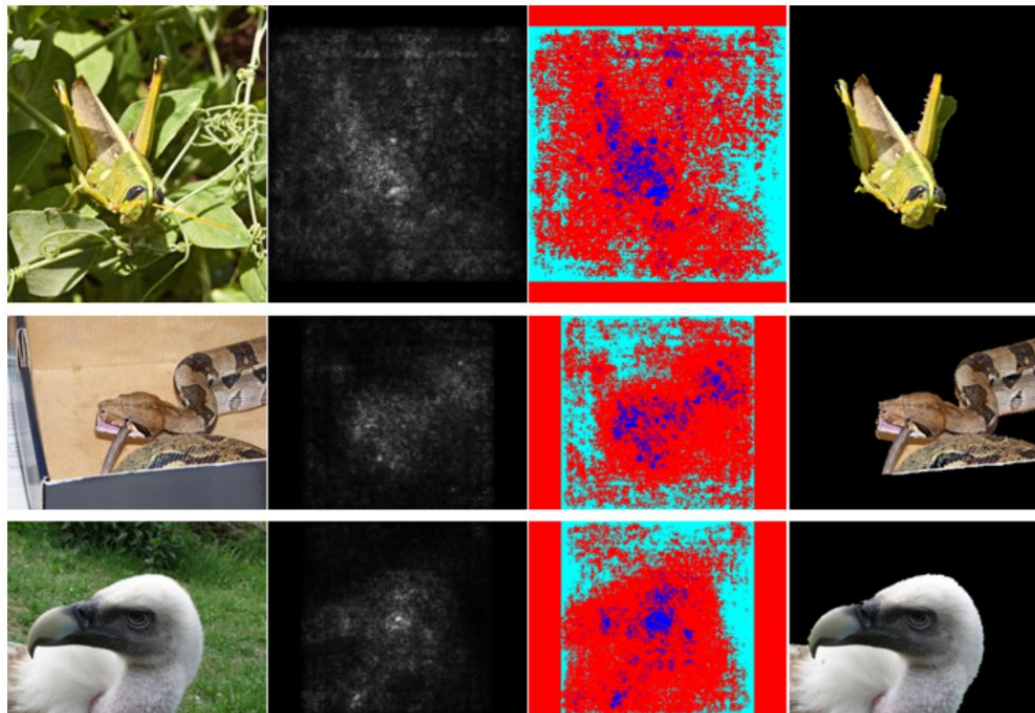
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

Saliency Maps



Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

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.

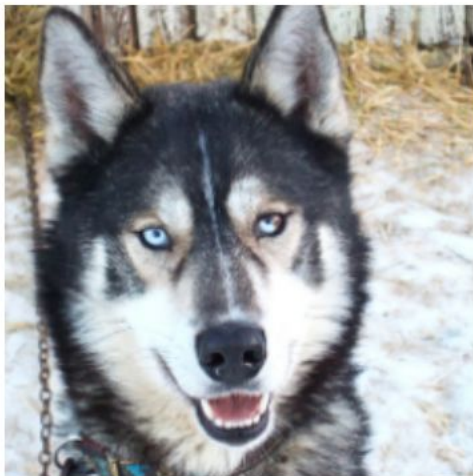
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

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

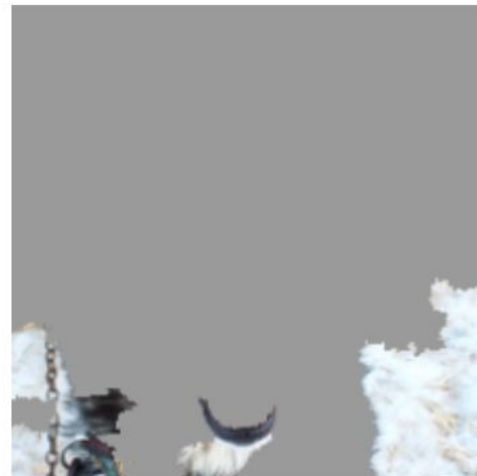
Saliency maps: Uncovers biases

Such methods also find biases

wolf vs dog classifier looks
is actually a snow vs
no-snow classifier



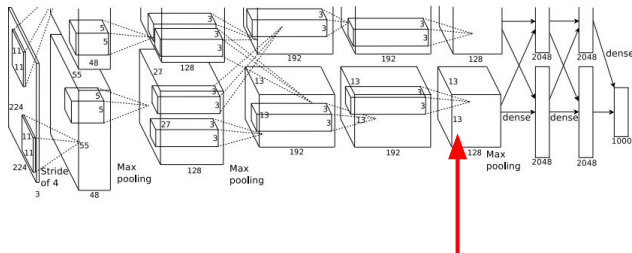
(a) Husky classified as wolf



(b) Explanation

Figures copyright Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, 2016; reproduced with permission.
Ribeiro et al, "Why Should I Trust You?" Explaining the Predictions of Any Classifier", ACM KDD 2016

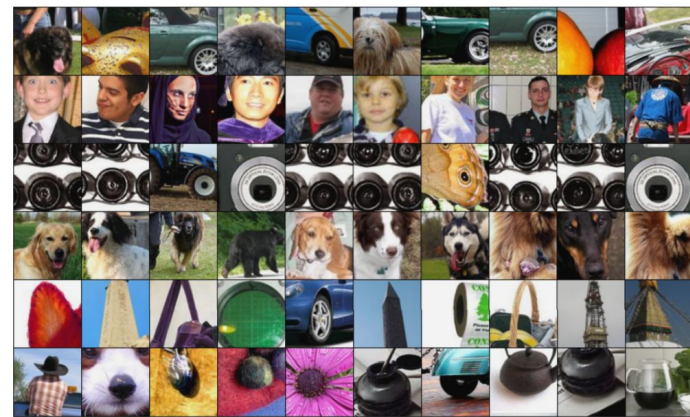
Maximally Activating Patches



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

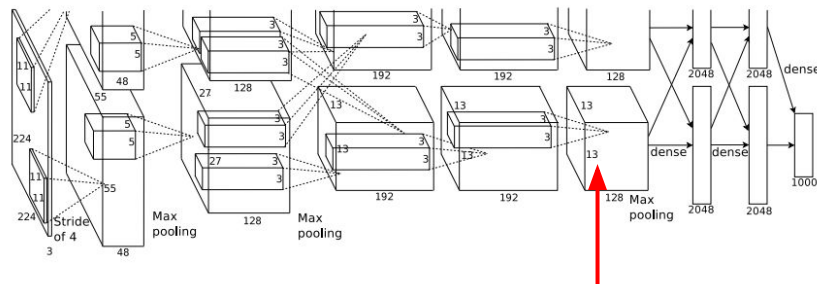
Run many images through the network, record values of chosen channel

Visualize image patches that correspond to maximal activations



Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

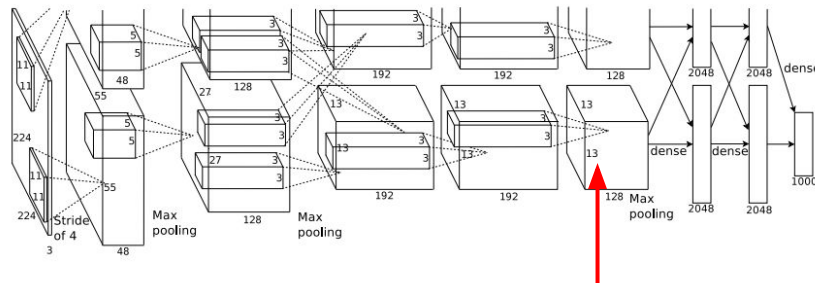
Intermediate Features via (guided) backprop



Pick a single intermediate channel, e.g. one value in 128 x 13 x 13 conv5 feature map

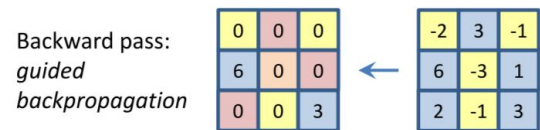
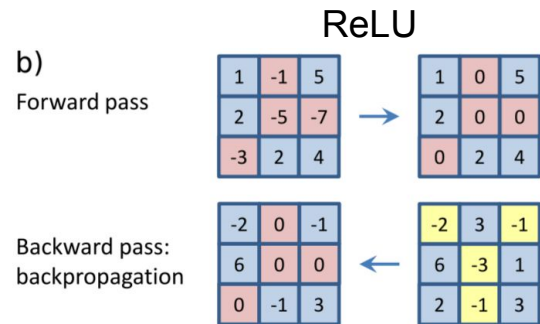
Compute gradient of activation value with respect to image pixels

Intermediate Features via (guided) backprop



Pick a single intermediate channel, e.g. one value in 128 x 13 x 13 conv5 feature map

Compute gradient of activation value with respect to image pixels



Images come out nicer if you only backprop positive gradients through each ReLU (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

Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Intermediate features via (guided) backprop



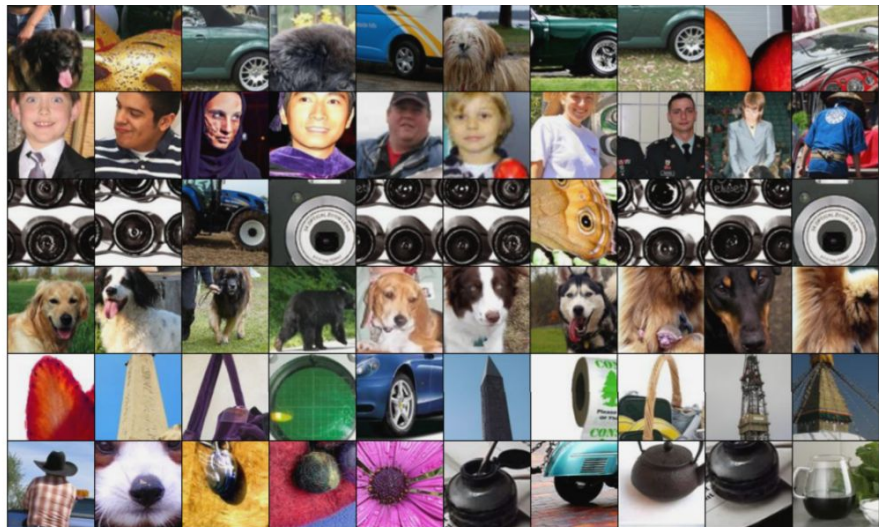
Maximally activating patches
(Each row is a different neuron)



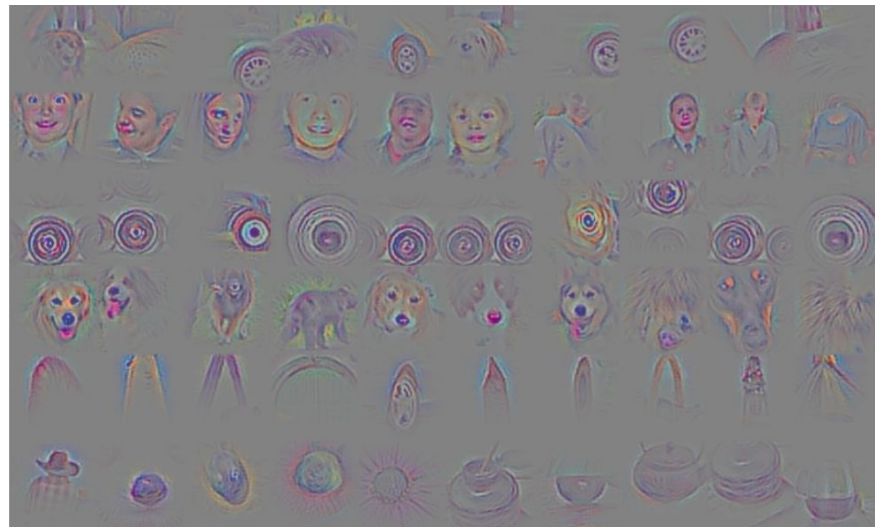
Guided Backprop

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Intermediate features via (guided) backprop



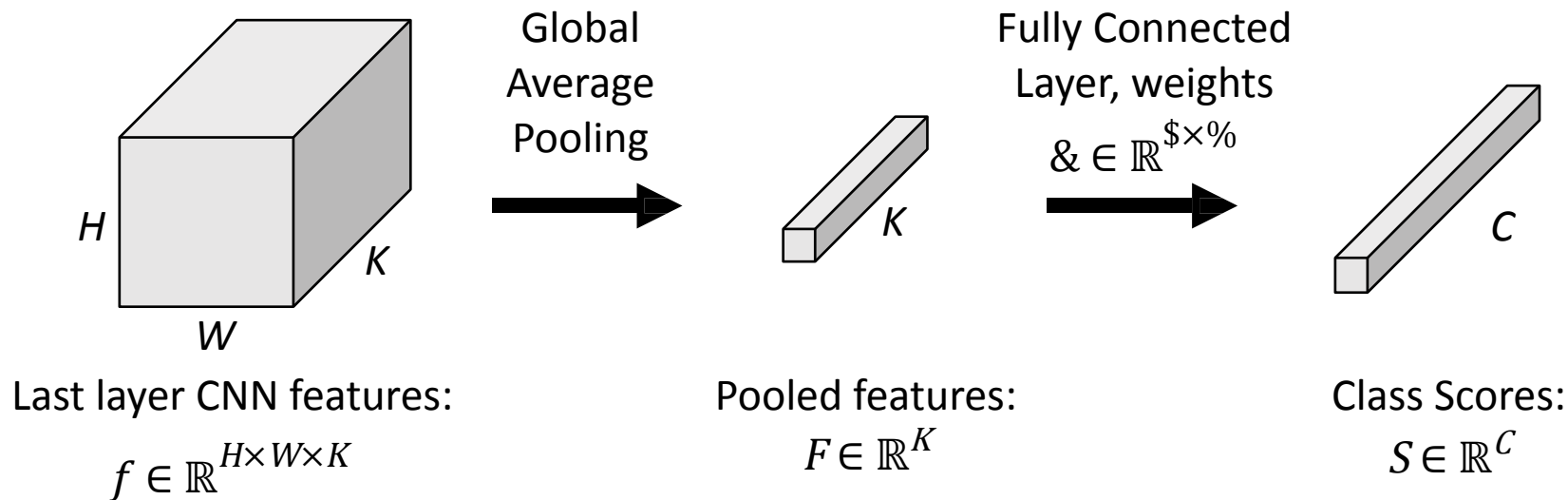
Maximally activating patches
(Each row is a different neuron)



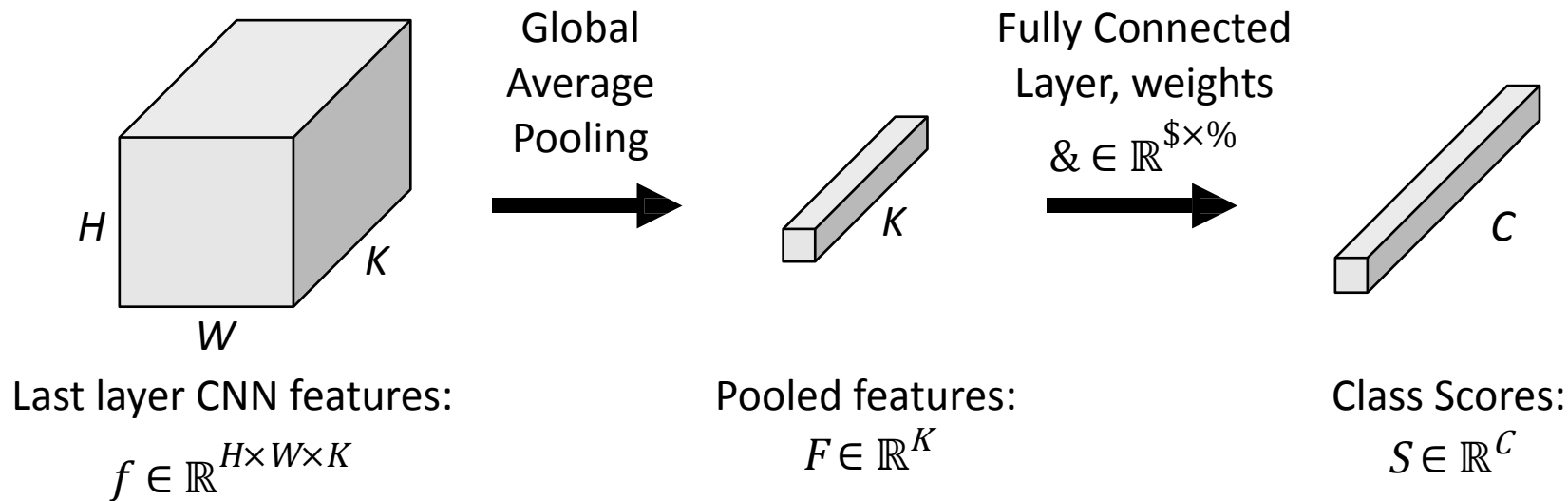
Guided Backprop

Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

Class Activation Mapping (CAM)

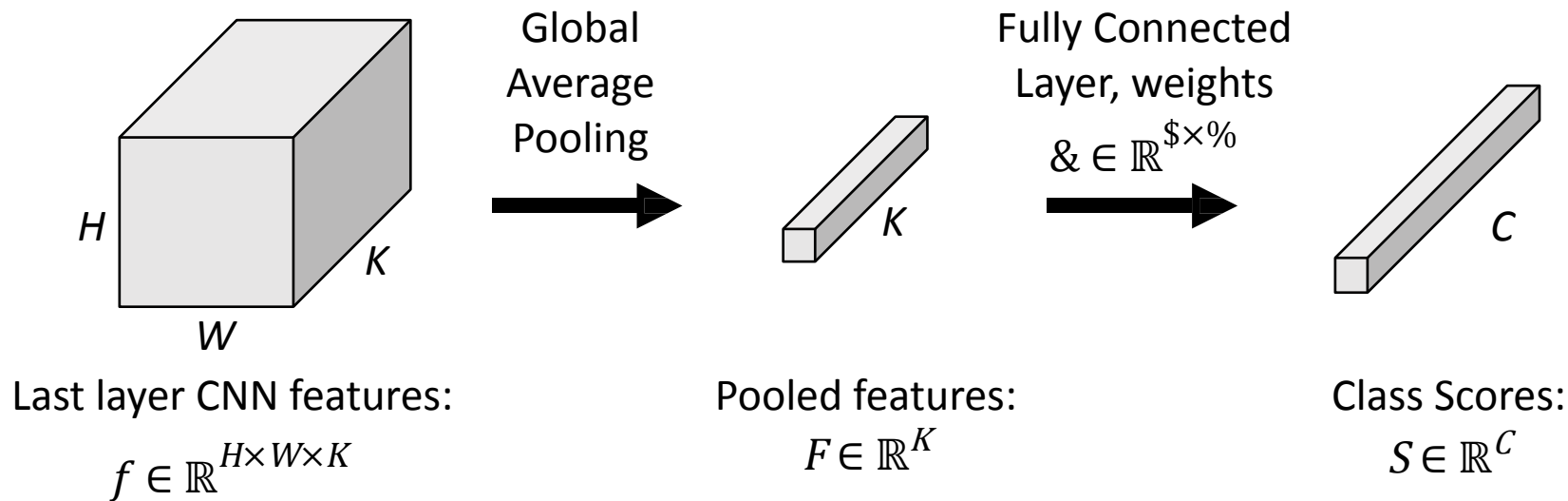


Class Activation Mapping (CAM)



$$F_k = \frac{1}{HW} \sum_{hw} f_{h,w,k}$$

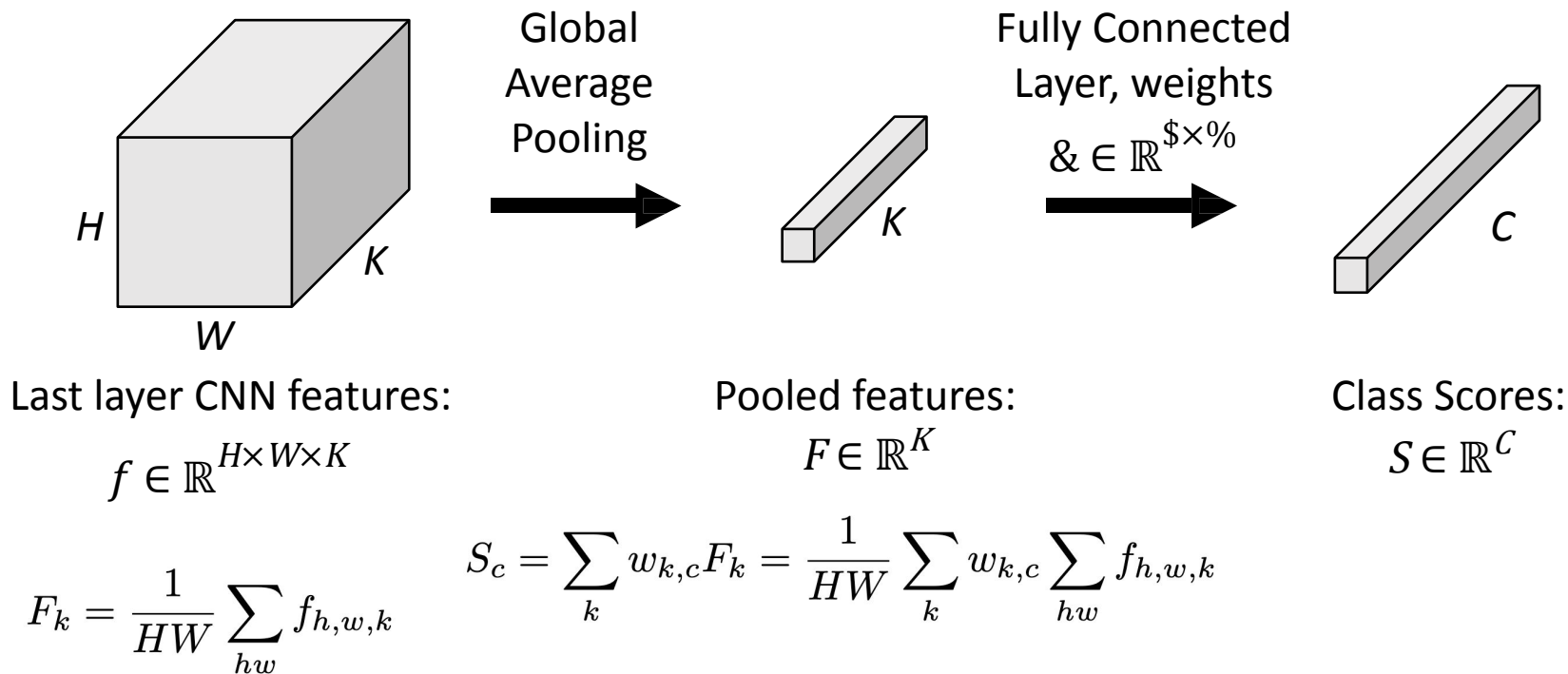
Class Activation Mapping (CAM)



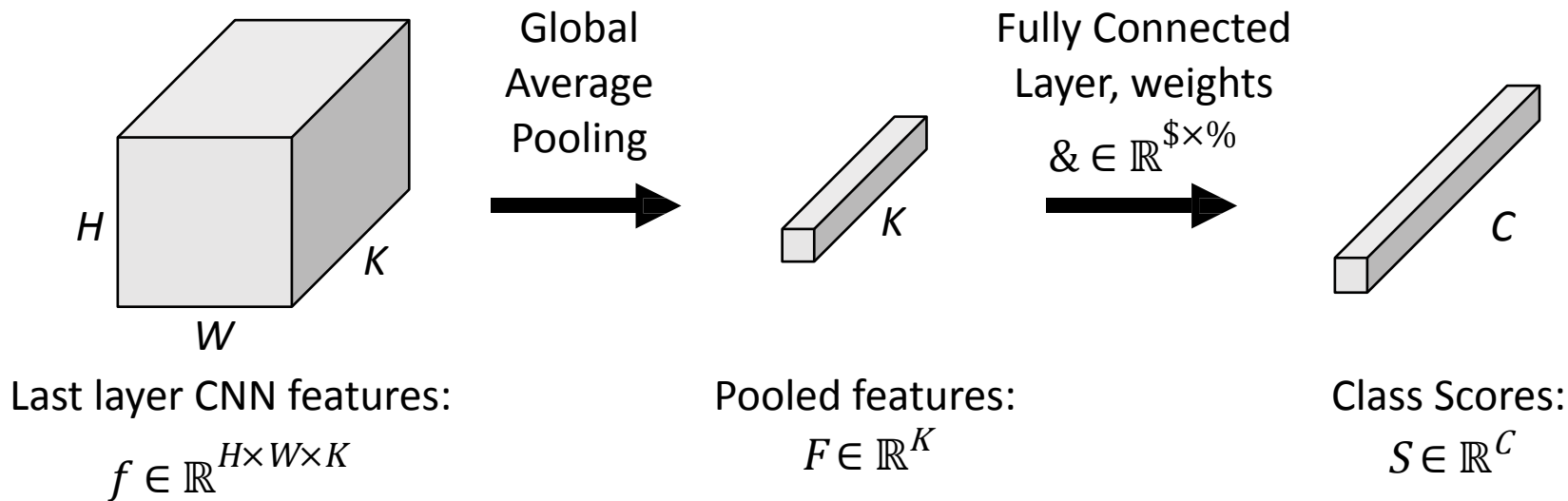
$$F_k = \frac{1}{HW} \sum_{hw} f_{h,w,k}$$

$$S_c = \sum_k w_{k,c} F_k$$

Class Activation Mapping (CAM)



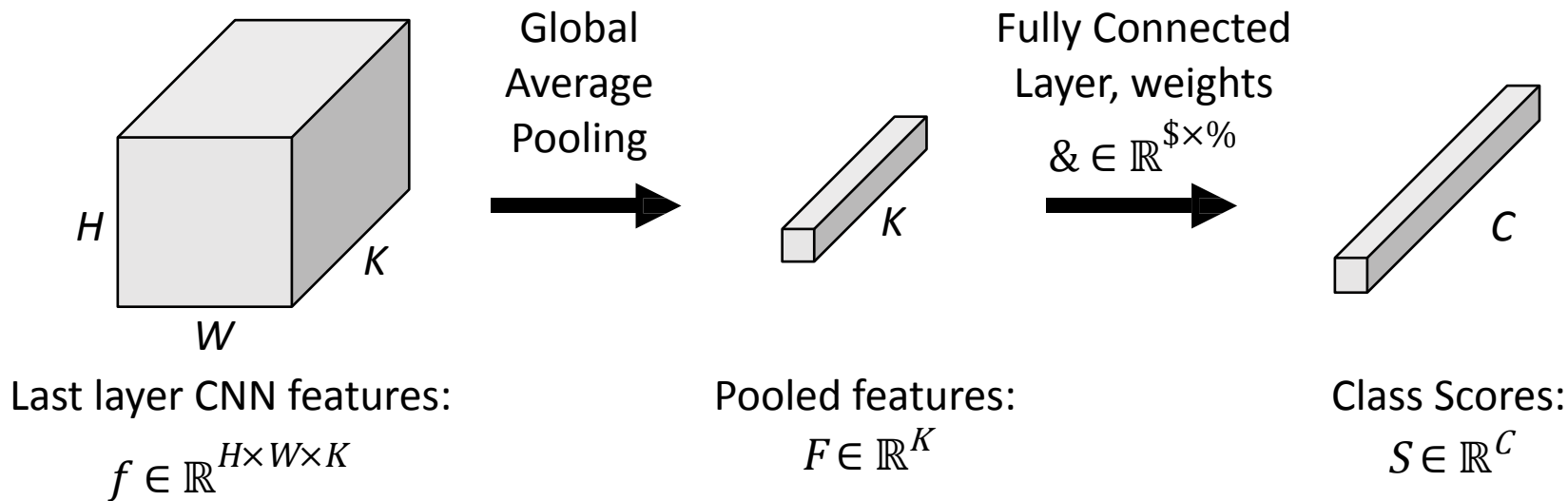
Class Activation Mapping (CAM)



$$F_k = \frac{1}{HW} \sum_{hw} f_{h,w,k}$$

$$\begin{aligned} S_c &= \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{hw} f_{h,w,k} \\ &= \frac{1}{HW} \sum_{hw} \sum_k w_{k,c} f_{h,w,k} \end{aligned}$$

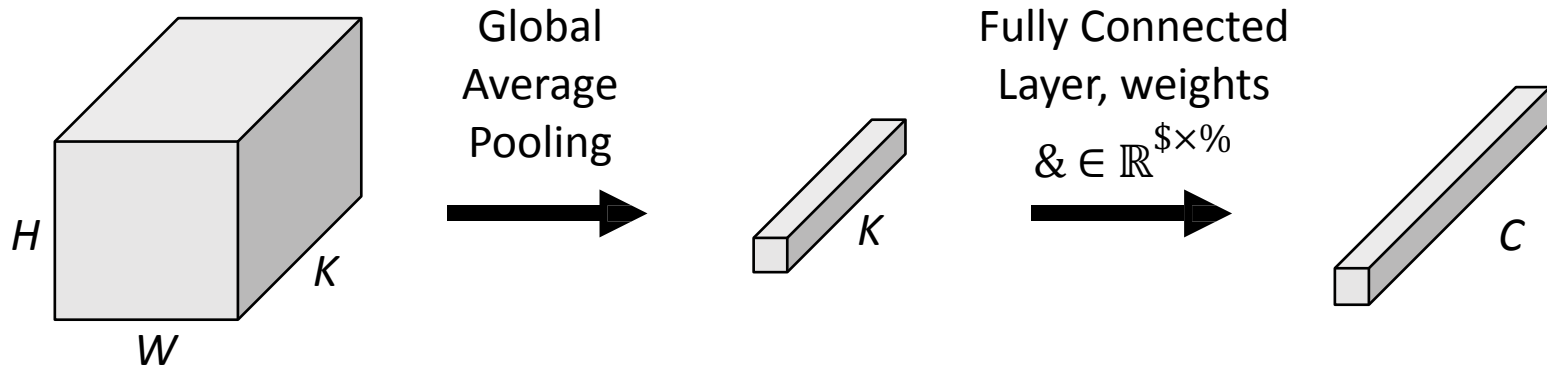
Class Activation Mapping (CAM)



$$F_k = \frac{1}{HW} \sum_{hw} f_{h,w,k}$$

$$\begin{aligned} S_c &= \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{hw} f_{h,w,k} \\ &= \frac{1}{HW} \sum_{hw} \sum_k w_{k,c} f_{h,w,k} \end{aligned}$$

Class Activation Mapping (CAM)



Last layer CNN features:

$$f \in \mathbb{R}^{H \times W \times K}$$

Pooled features:

$$F \in \mathbb{R}^K$$

Class Scores:

$$S \in \mathbb{R}^C$$

$$F_k = \frac{1}{HW} \sum_{hw} f_{h,w,k}$$

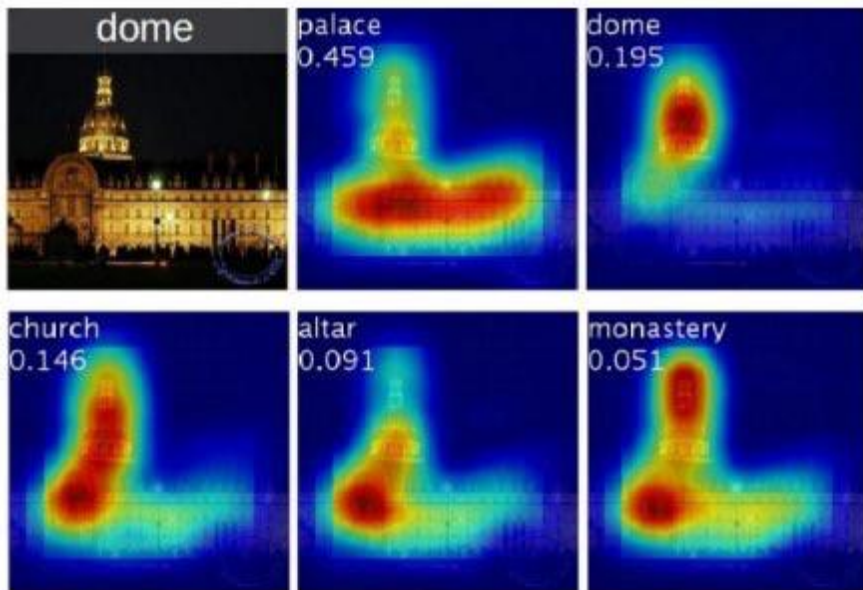
$$\begin{aligned} S_c &= \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{hw} f_{h,w,k} \\ &= \frac{1}{HW} \sum_{hw} \sum_k w_{k,c} f_{h,w,k} \end{aligned}$$

CAMs

$$M \in \mathbb{R}^{C, H, W}$$

$$M = \sum_k w_{k,c} f_{h,w,k}$$

Class Activation Mapping (CAM)



Class activation maps of top 5 predictions

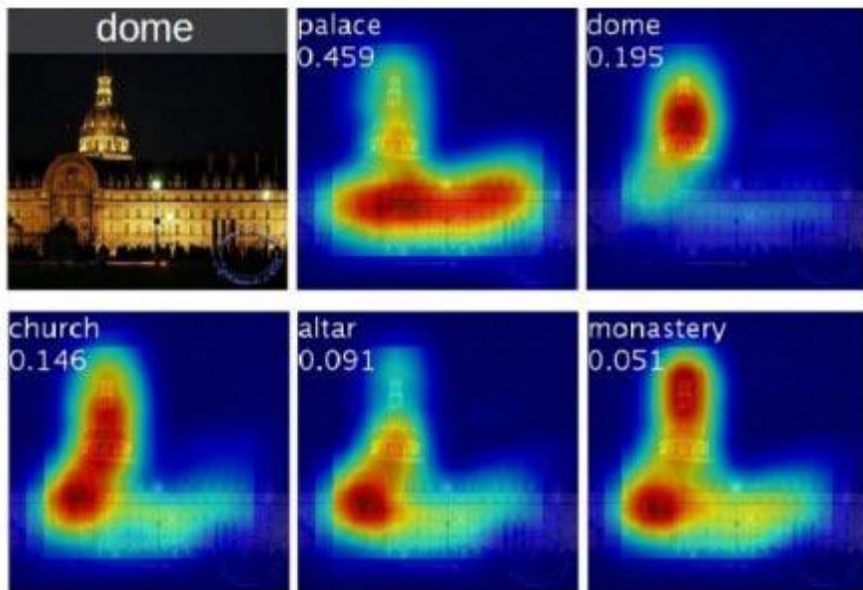


Class activation maps for one object class

Zhou et al, "Learning Deep Features for Discriminative Localization", CVPR 2016

Class Activation Mapping (CAM)

Problem: Can only
apply to last conv
layer



Class activation maps of top 5 predictions



Class activation maps for one object class

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$
2. Compute gradient of class score S_c with respect to A :

$$\frac{\partial S_c}{\partial A} \in \mathbb{R}^{H \times W \times K}$$

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$
2. Compute gradient of class score S_c with respect to A :

$$\frac{\partial S_c}{\partial A} \in \mathbb{R}^{H \times W \times K}$$

3. Global Average Pool the gradients to get weights $\alpha \in \mathbb{R}^K$:

$$\alpha_k = \frac{1}{HW} \sum_{h,w} \frac{\partial S_c}{\partial A_{h,w,k}}$$

Gradient-Weighted Class Activation Mapping (Grad-CAM)

1. Pick any layer, with activations $A \in \mathbb{R}^{H \times W \times K}$
2. Compute gradient of class score S_c with respect to A :

$$\frac{\partial S_c}{\partial A} \in \mathbb{R}^{H \times W \times K}$$

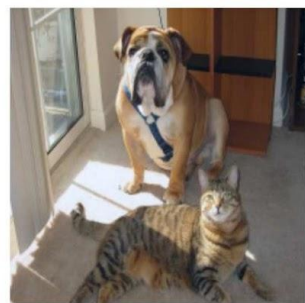
3. Global Average Pool the gradients to get weights $\alpha \in \mathbb{R}^K$:

$$\alpha_k = \frac{1}{HW} \sum_{h,w} \frac{\partial S_c}{\partial A_{h,w,k}}$$

4. Compute activation map $M^c \in \mathbb{R}^{H,W}$:

$$M_{h,w}^c = \text{ReLU} \left(\sum_k \alpha_k A_{h,w,k} \right)$$

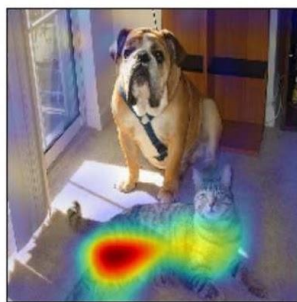
Gradient-Weighted Class Activation Mapping (Grad-CAM)



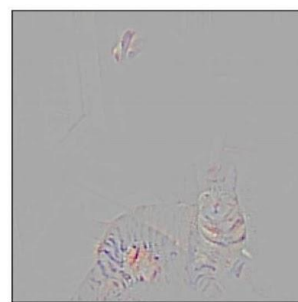
(a) Original Image



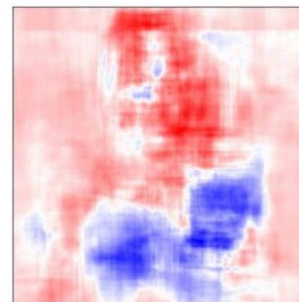
(b) Guided Backprop 'Cat'



(c) Grad-CAM 'Cat'



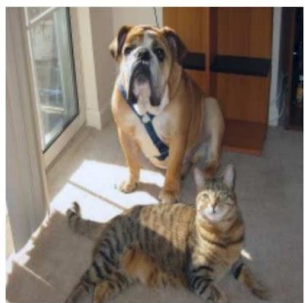
(d) Guided Grad-CAM 'Cat'



(e) Occlusion map for 'Cat'



(f) ResNet Grad-CAM 'Cat'



(g) Original Image



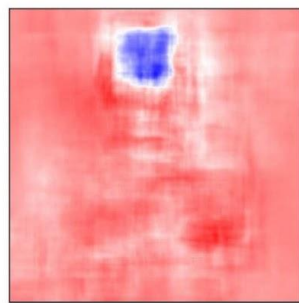
(h) Guided Backprop 'Dog'



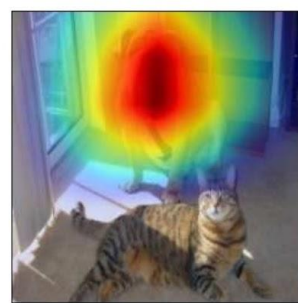
(i) Grad-CAM 'Dog'



(j) Guided Grad-CAM 'Dog'



(k) Occlusion map for 'Dog'



(l) ResNet Grad-CAM 'Dog'

Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Gradient-Weighted Class Activation Mapping (Grad-CAM)

Can also be applied beyond classification models, e.g. image captioning

Grad-CAM



A group of people flying kites on a beach

Grad-CAM



A man is sitting at a table with a pizza

Selvaraju et al, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization", CVPR 2017

Visualizing CNN features: Gradient Ascent

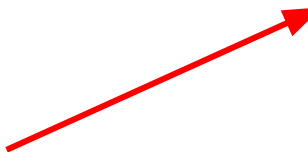
(Guided) backprop:

Find the part of an image that a neuron responds to

Gradient ascent:

Generate a synthetic image that maximally activates a neuron

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



Neuron value



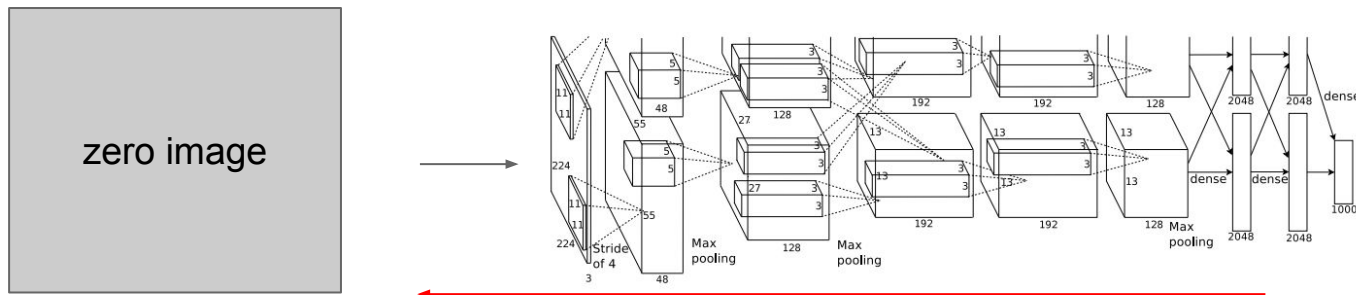
Natural image regularizer

Visualizing CNN features: Gradient Ascent

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

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

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

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.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

Visualizing CNN features: Gradient Ascent

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

Simple regularizer: Penalize L2 norm of generated image



dumbbell



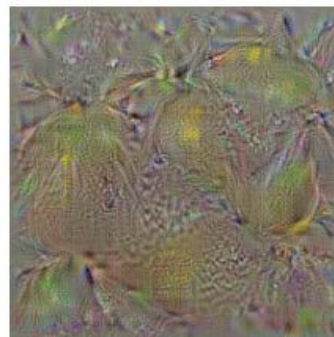
cup



dalmatian



bell pepper



lemon



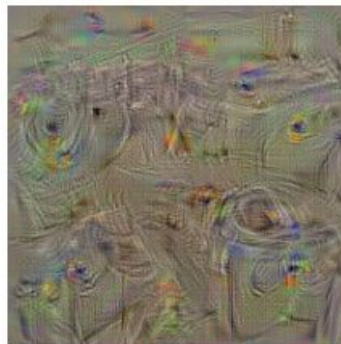
husky

Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

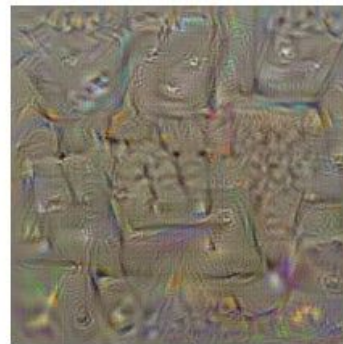
Visualizing CNN features: Gradient Ascent

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

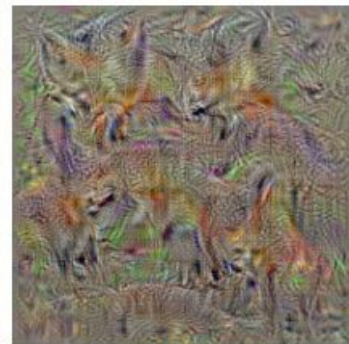
Simple regularizer: Penalize L2 norm of generated image



washing machine



computer keyboard



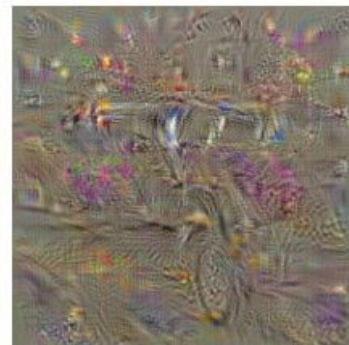
kit fox



goose



ostrich



limousine

Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.
Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reproduced with permission.

Visualizing CNN features: Gradient Ascent

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

Better regularizer: Penalize L2 norm of image; also during optimization periodically

- (1) Gaussian blur image
- (2) Clip pixels with small values to 0
- (3) Clip pixels with small gradients to 0

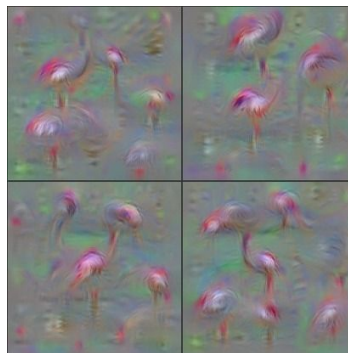
Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Visualizing CNN features: Gradient Ascent

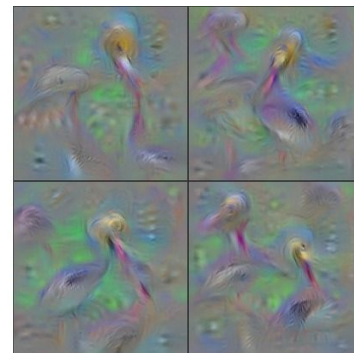
$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

Better regularizer: Penalize L2 norm of image; also during optimization periodically

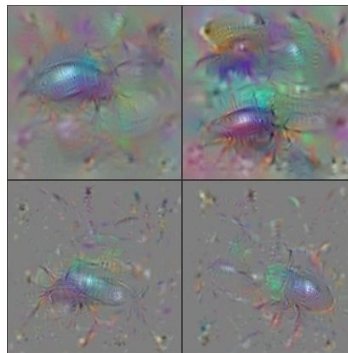
- (1) Gaussian blur image
- (2) Clip pixels with small values to 0
- (3) Clip pixels with small gradients to 0



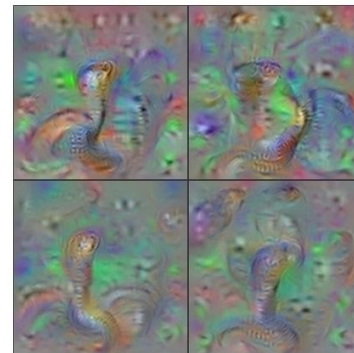
Flamingo



Pelican



Ground Beetle



Indian Cobra

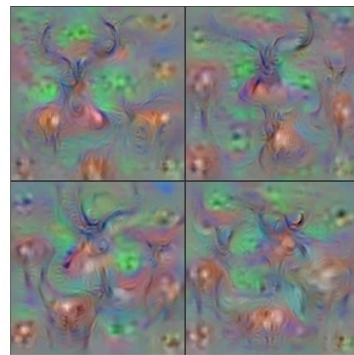
Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.
Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reproduced with permission.

Visualizing CNN features: Gradient Ascent

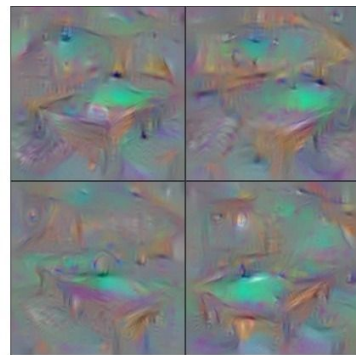
$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

Better regularizer: Penalize L2 norm of image; also during optimization periodically

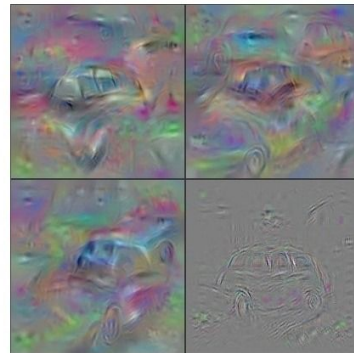
- (1) Gaussian blur image
- (2) Clip pixels with small values to 0
- (3) Clip pixels with small gradients to 0



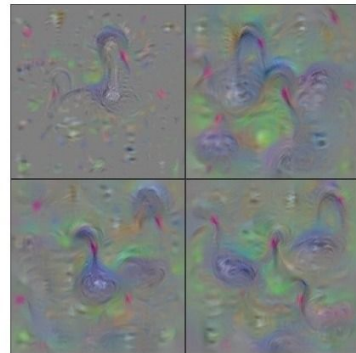
Hartebeest



Billiard Table



Station Wagon

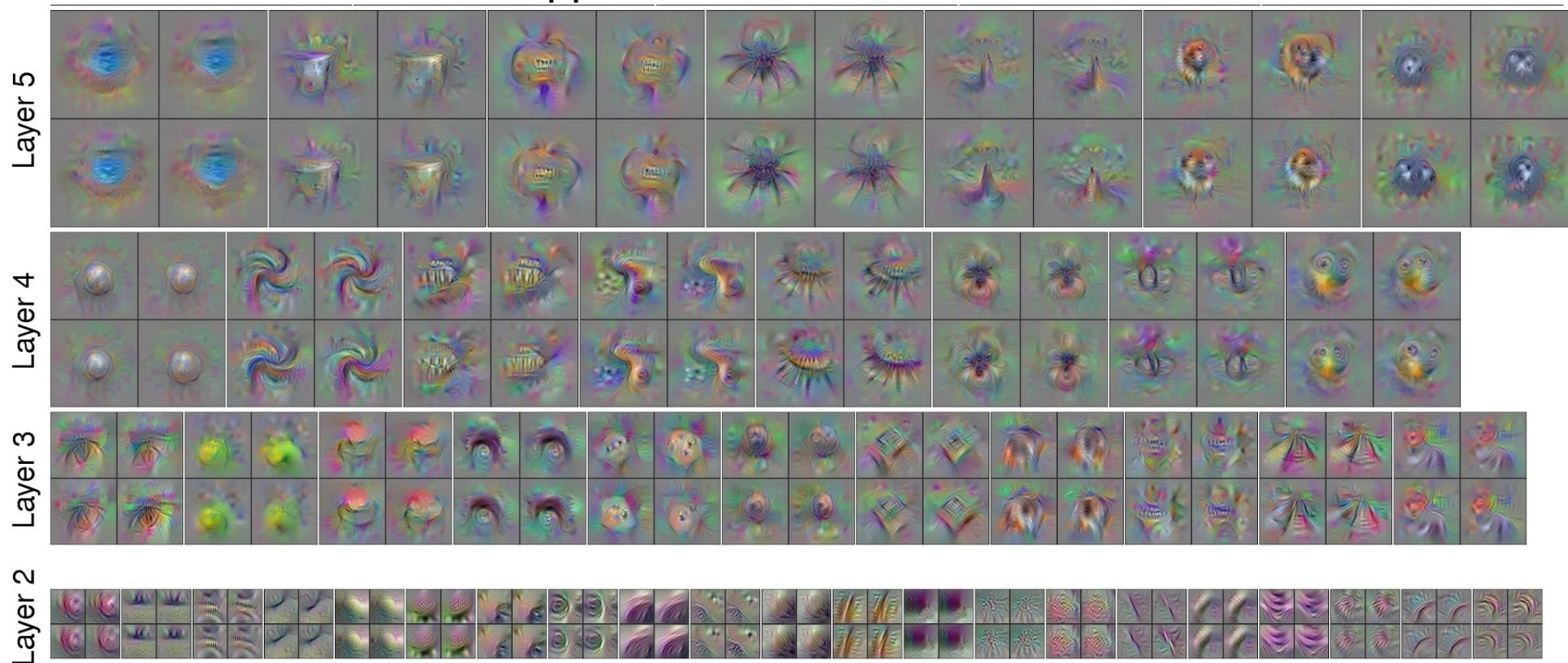


Black Swan

Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.
Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reproduced with permission.

Visualizing CNN features: Gradient Ascent

Use the same approach to visualize intermediate features



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

Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reproduced with permission.

Visualizing CNN features: Gradient Ascent

Adding “multi-faceted” visualization gives even nicer results:
(Plus more careful regularization, center-bias)

Reconstructions of multiple feature types (facets) recognized
by the same “grocery store” neuron



Corresponding example training set images recognized
by the same neuron as in the “grocery store” class



Nguyen et al, “Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks”, ICML Visualization for Deep Learning Workshop 2016.
Figures copyright Anh Nguyen, Jason Yosinski, and Jeff Clune, 2016; reproduced with permission.

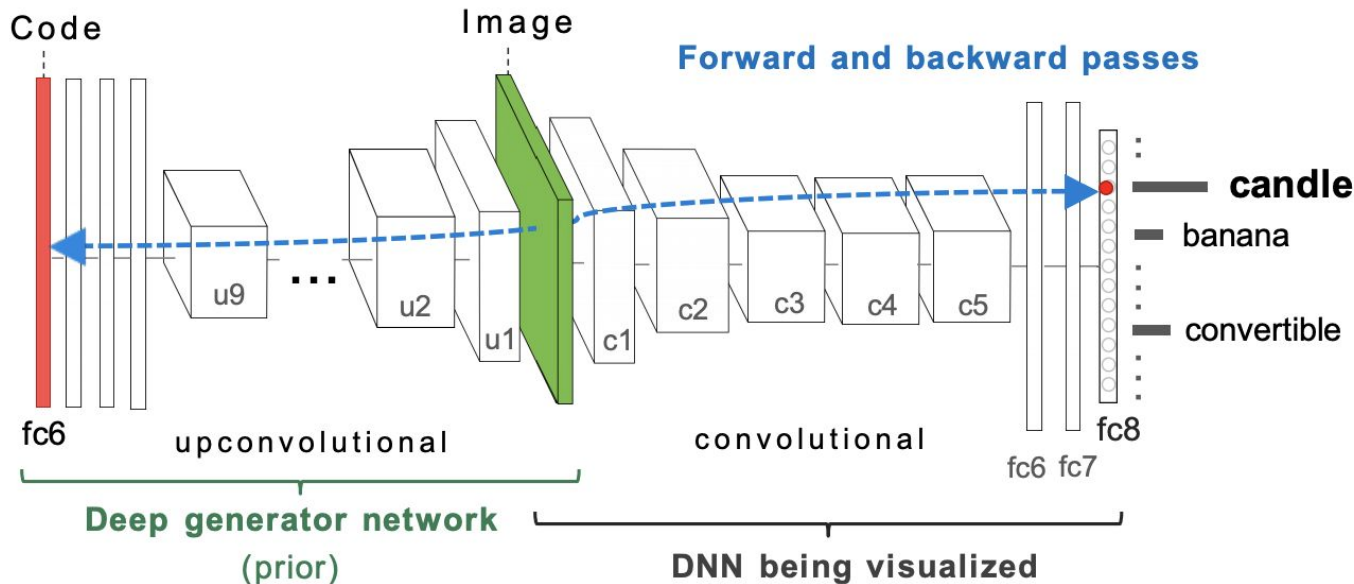
Visualizing CNN features: Gradient Ascent



Nguyen et al, "Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks", ICML Visualization for Deep Learning Workshop 2016.
Figures copyright Anh Nguyen, Jason Yosinski, and Jeff Clune, 2016; reproduced with permission.

Visualizing CNN features: Gradient Ascent

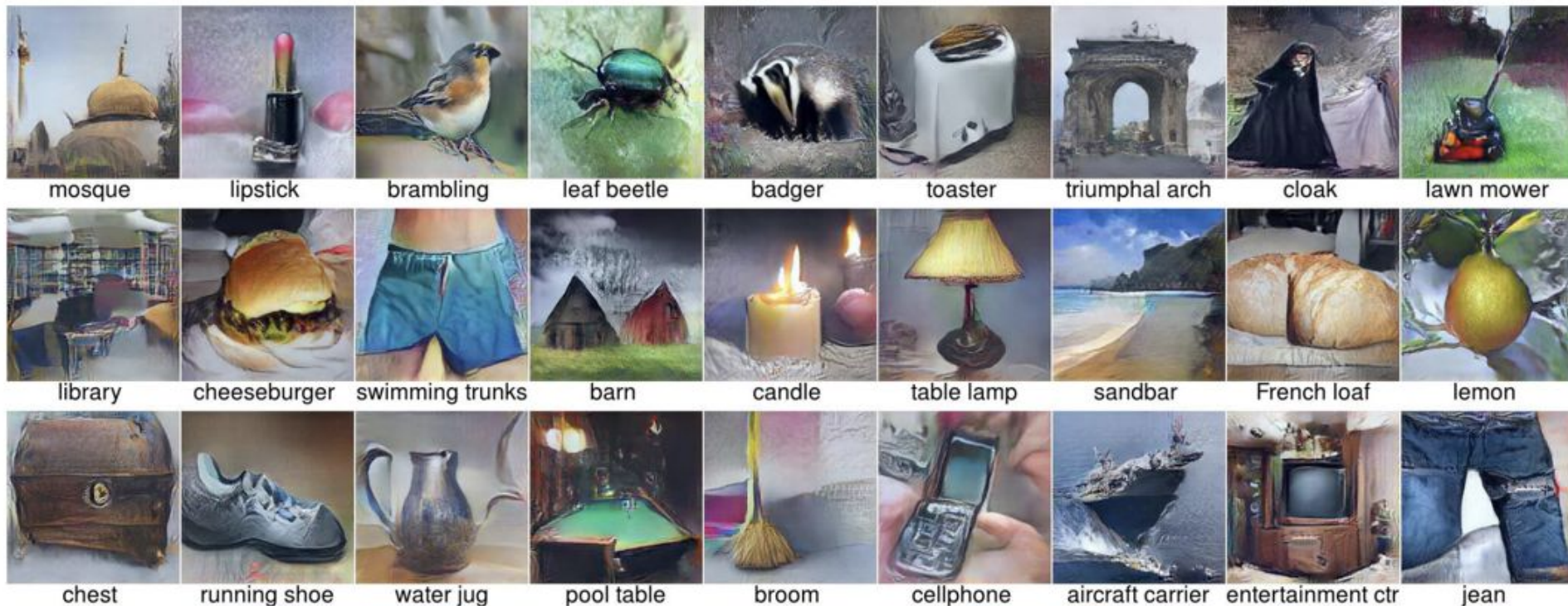
Optimize in FC6 latent space instead of pixel space:



Nguyen et al, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," NIPS 2016
Figure copyright Nguyen et al. 2016; reproduced with permission.

Visualizing CNN features: Gradient Ascent

Optimize in FC6 latent space instead of pixel space:



Nguyen et al, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," NIPS 2016
Figure copyright Nguyen et al, 2016; reproduced with permission.

Today's agenda

Visualizing what models have learned:

- Visualizing filters
- Visualizing final layer features
- Visualizing activations

Understanding input pixels

- Identifying important pixels
- Saliency via backprop
- Guided backprop to generate images
- Gradient ascent to visualize features

Adversarial perturbations

Concept Vectors

Fooling Images / Adversarial Examples

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

Fooling Images / Adversarial Examples

African elephant



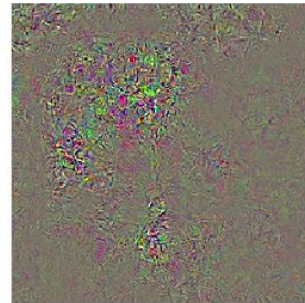
koala



Difference



10x Difference



schooner



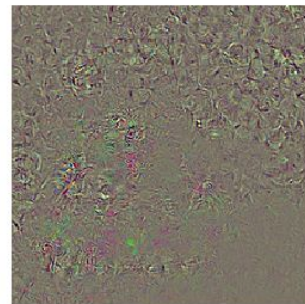
iPod



Difference

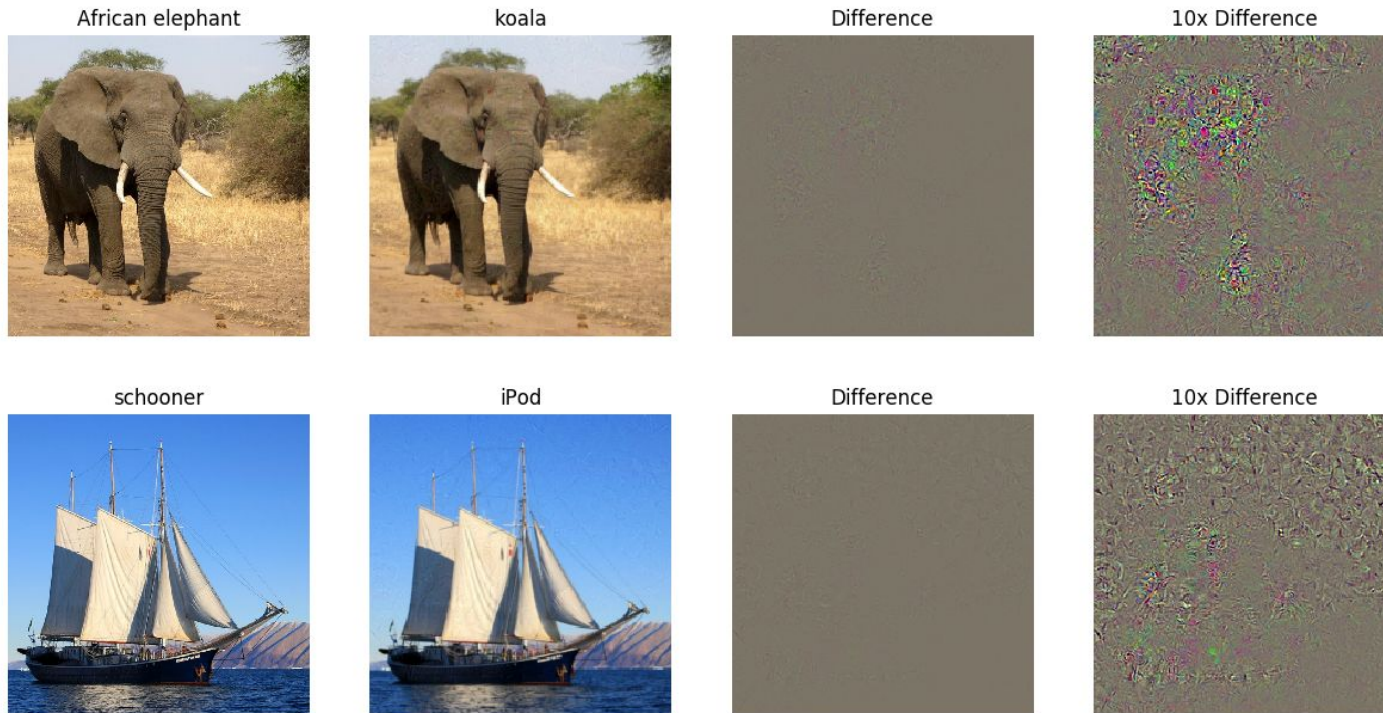


10x Difference



Boat image is [CC0 public domain](#)
Elephant image is [CC0 public domain](#)

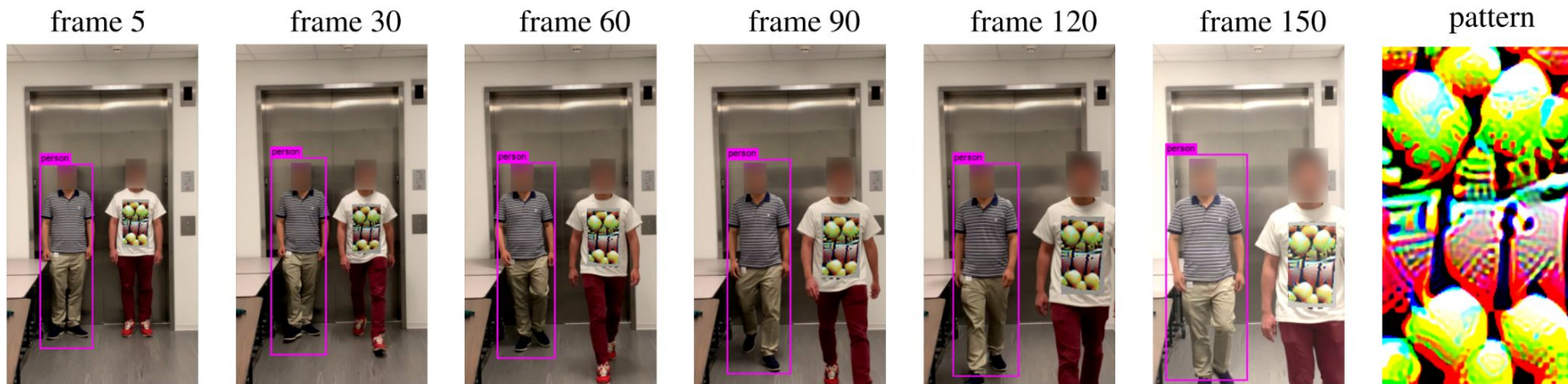
Fooling Images / Adversarial Examples



Check out [Ian Goodfellow's lecture](#) from 2017

Boat image is [CC0 public domain](#)
Elephant image is [CC0 public domain](#)

Fooling Person Detectors and Self-driving Cars

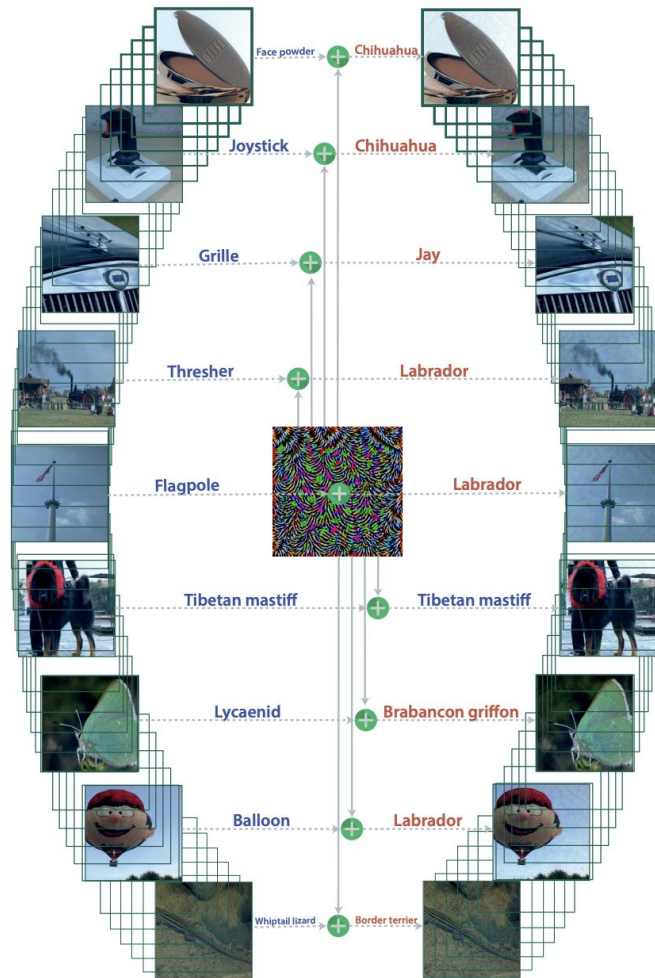


Xu et al., 2019; Eykholt et al., 2018

Fooling Images / Adversarial Examples

Universal perturbations

Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
Figure reproduced with permission



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)

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

Given feature vector

Features of new image

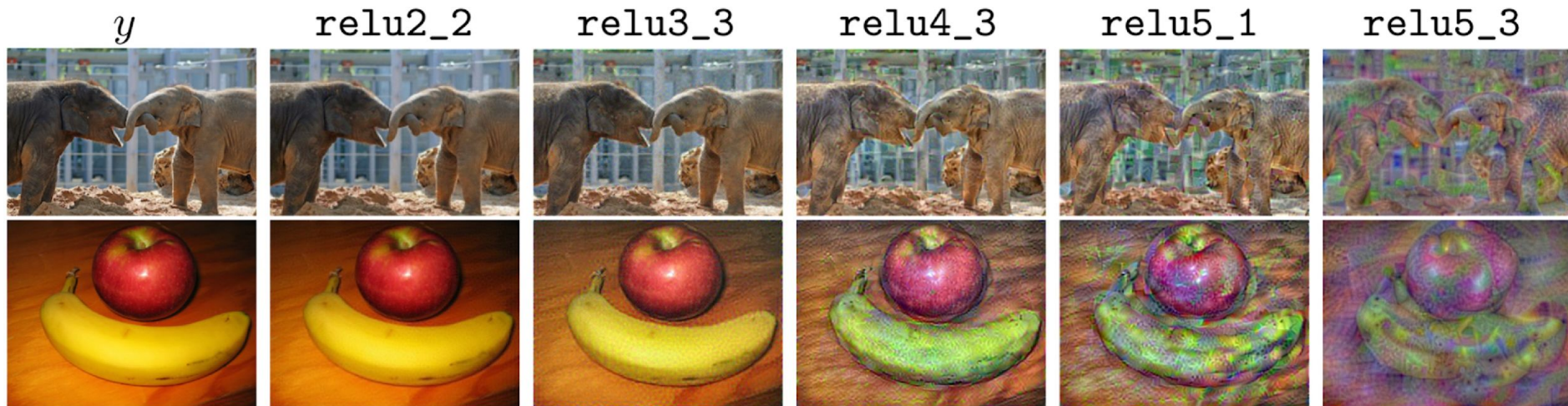
$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$
$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\frac{\beta}{2}}$$

Total Variation regularizer (encourages spatial smoothness)

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

Feature Inversion

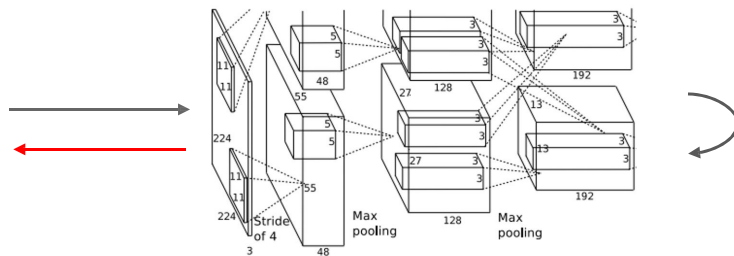
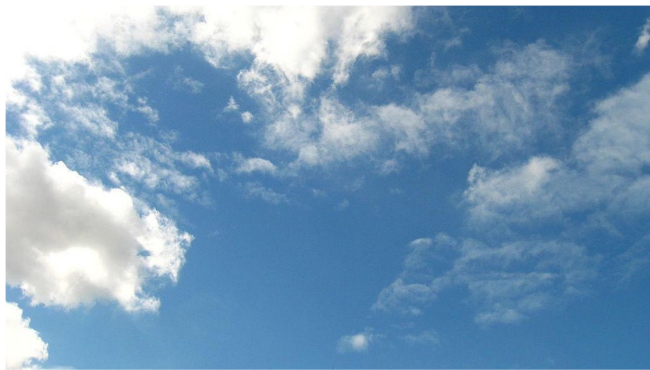
Reconstructing from different layers of VGG-16



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

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



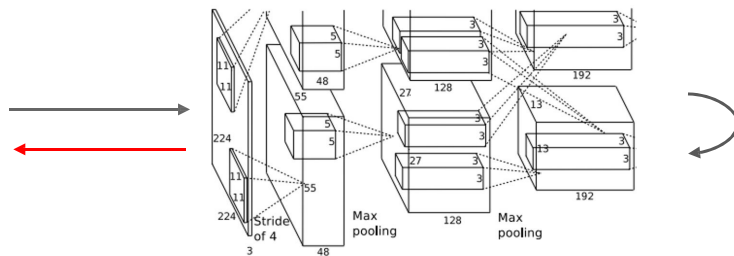
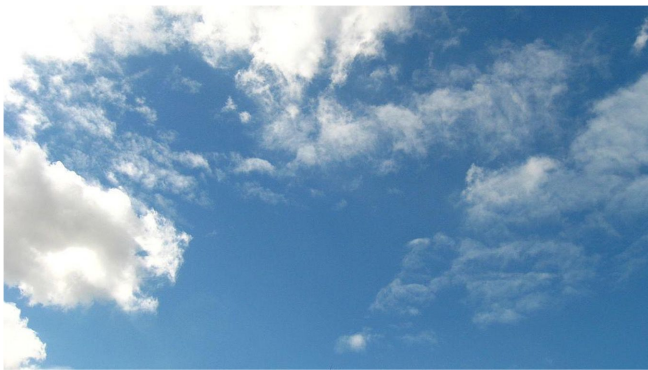
Choose an image and a layer in a CNN; repeat:

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](#)

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

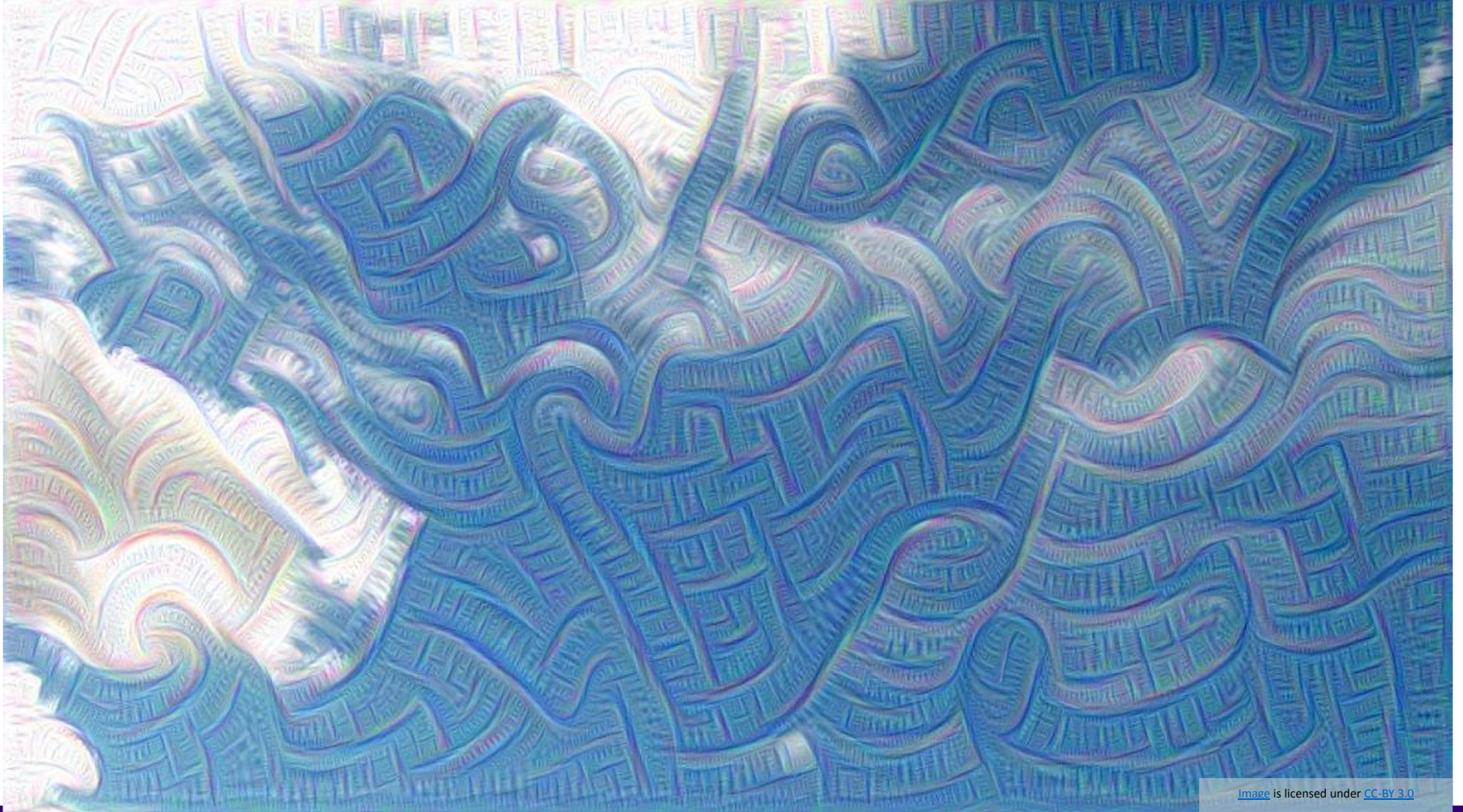
Equivalent to:

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

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



[Sky image](#) is licensed under [CC-BY SA 3.0](#)



[Image](#) is licensed under [CC-BY 3.0](#)

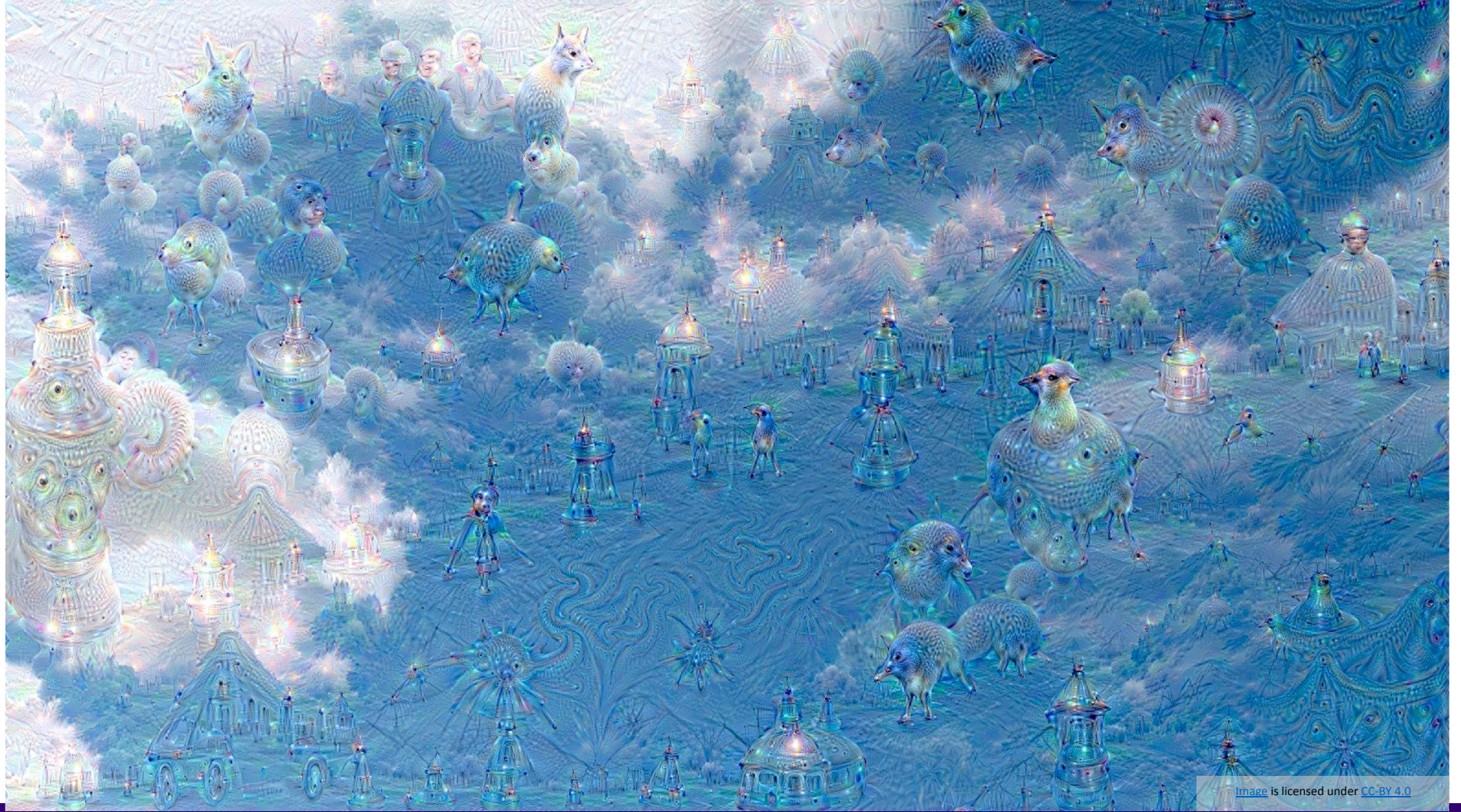
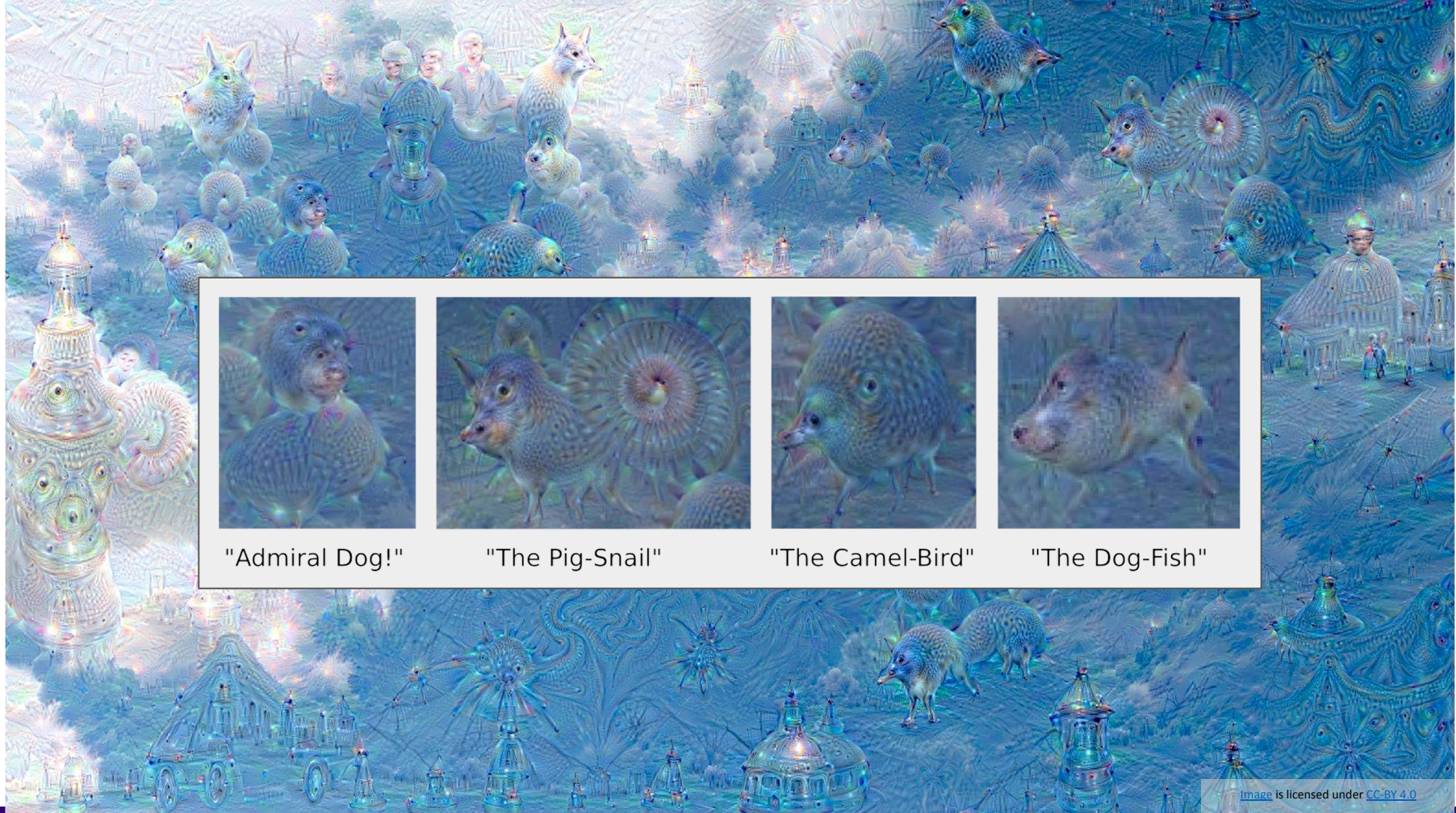


Image is licensed under [CC-BY 4.0](#)



"Admiral Dog!"



"The Pig-Snail"



"The Camel-Bird"

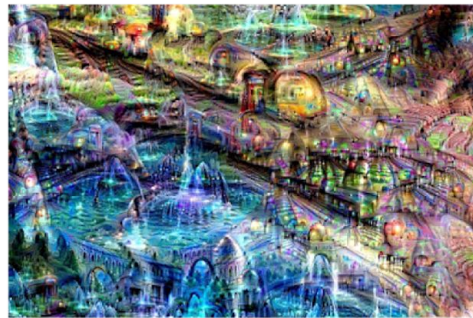
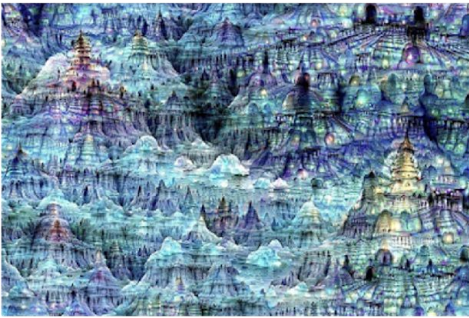
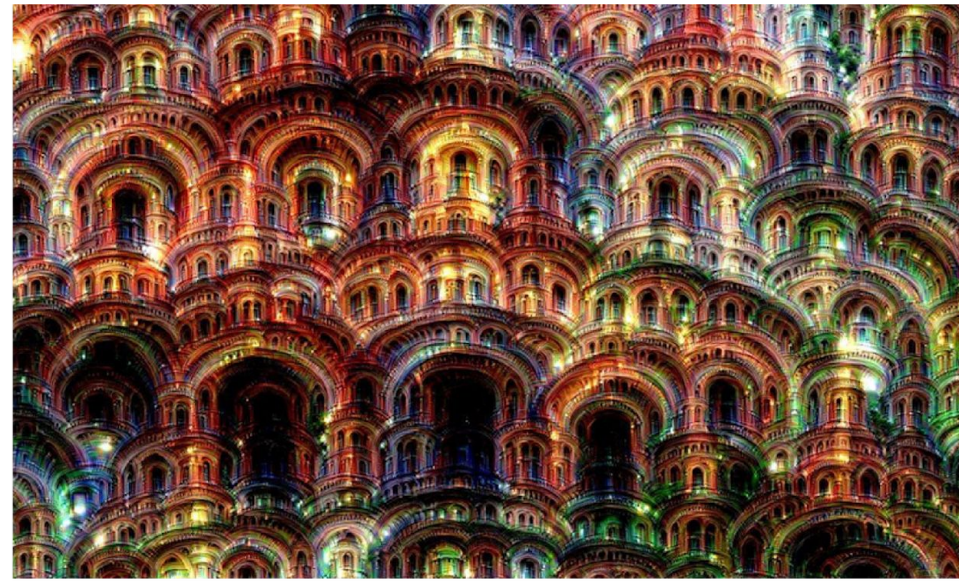
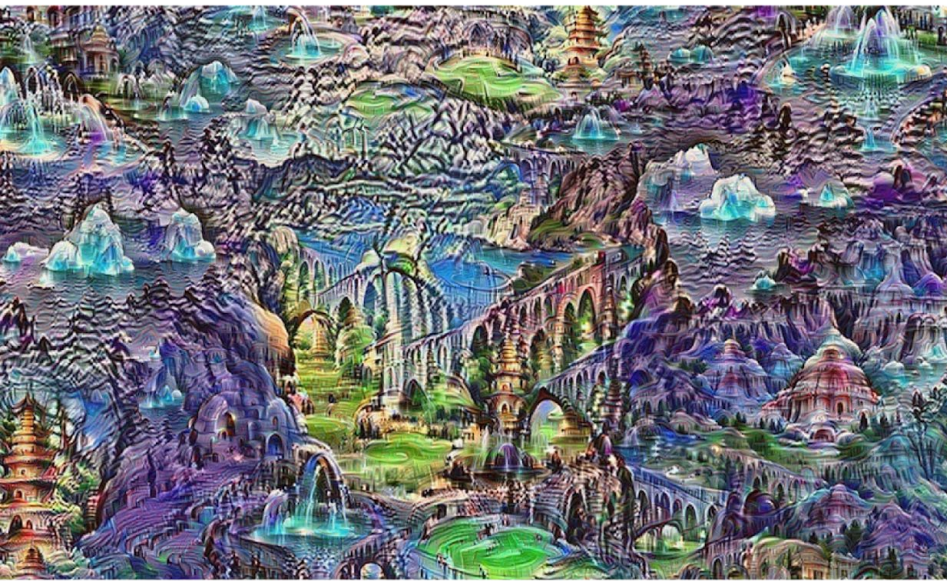


"The Dog-Fish"

Image is licensed under [CC-BY 4.0](#)



Image is licensed under [CC-BY 3.0](#)



[Image](#) is licensed under [CC-BY 4.0](#)

Today's agenda

Visualizing what models have learned:

- Visualizing filters
- Visualizing final layer features
- Visualizing activations

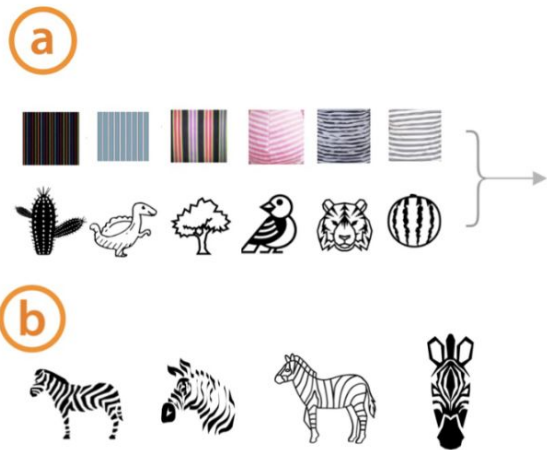
Understanding input pixels

- Identifying important pixels
- Saliency via backprop
- Guided backprop to generate images
- Gradient ascent to visualize features

Adversarial perturbations

Concept Vectors

Concept activation vectors



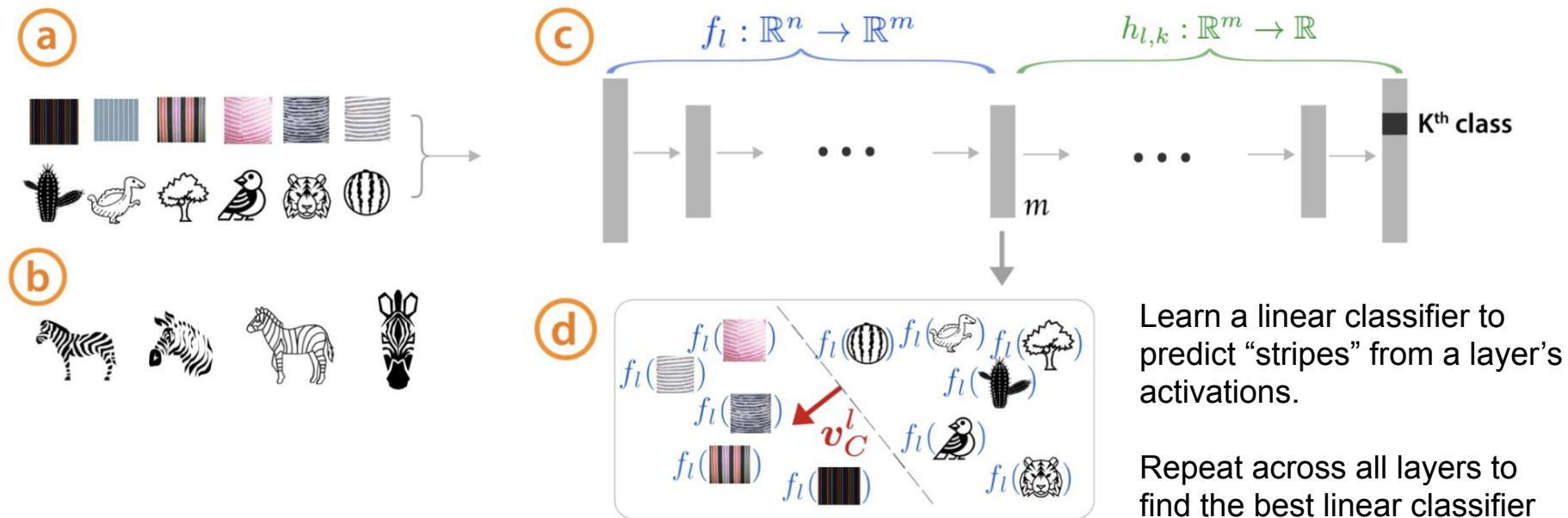
Let's see if a neural network has learned a specific concept and uses it effectively.

Example use case:

Q1. Has it learnt what stripes are?

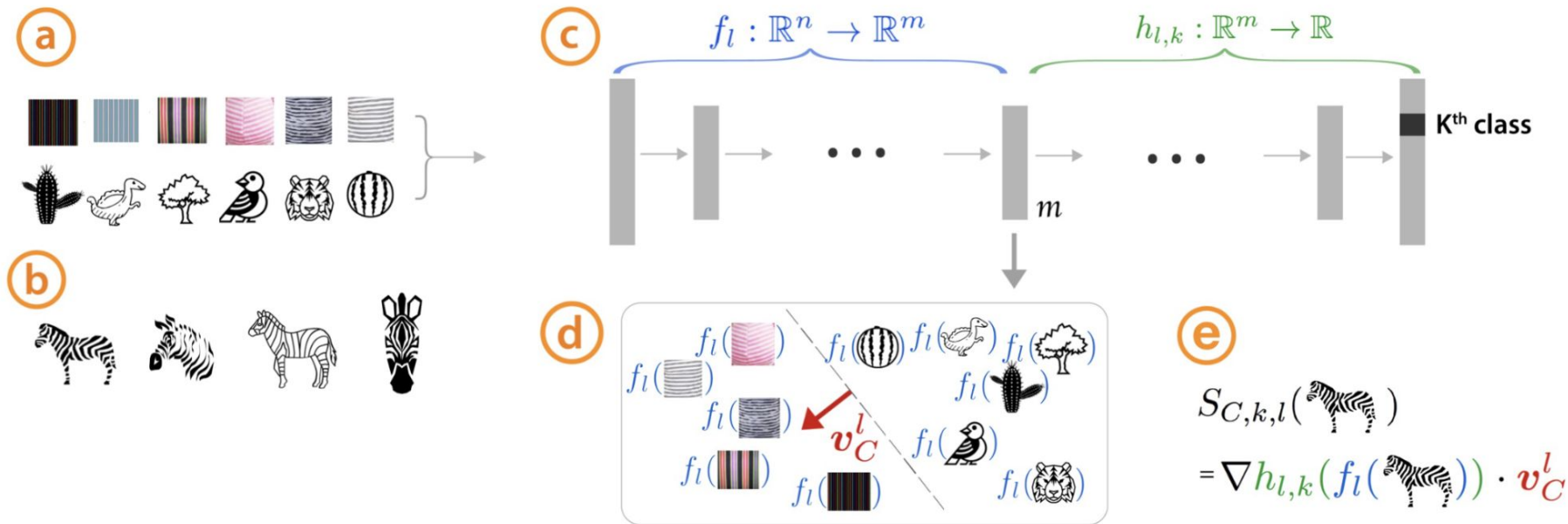
Q2. Can it identify the category "zebra" by using the concept "stripes"

Concept activation vectors



Kim, Been, et al. "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)." ICML, 2018.

Calculate if the gradient for that layer when predicting “zebra” matches the classifier



Kim, Been, et al. "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav)." ICML, 2018.

Summary

Many methods for understanding CNN representations

Activations: Nearest neighbors, Dimensionality reduction, maximal patches, occlusion

Gradients: Saliency maps, class visualization, fooling images, feature inversion

Adversarial Examples: To confuse the models

Concept Vectors: Human interpretable probing method

Next time:

Introduction to Language