深度学习

第八讲

王胤

Embeddings

Post

Text

(word2vec, fastText)

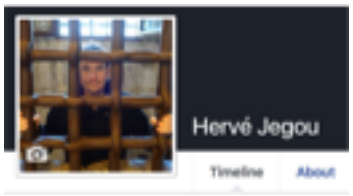
Video

Face

Embedding

User

(filament)

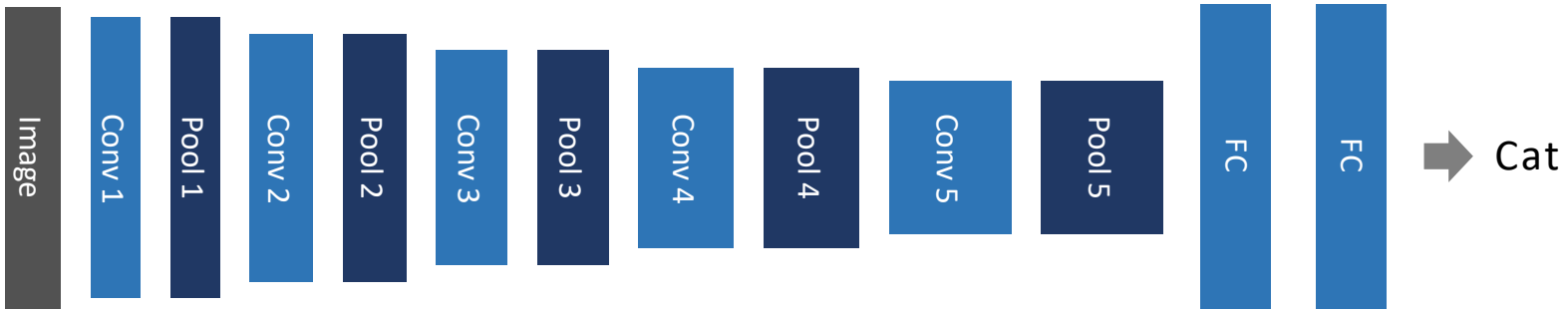


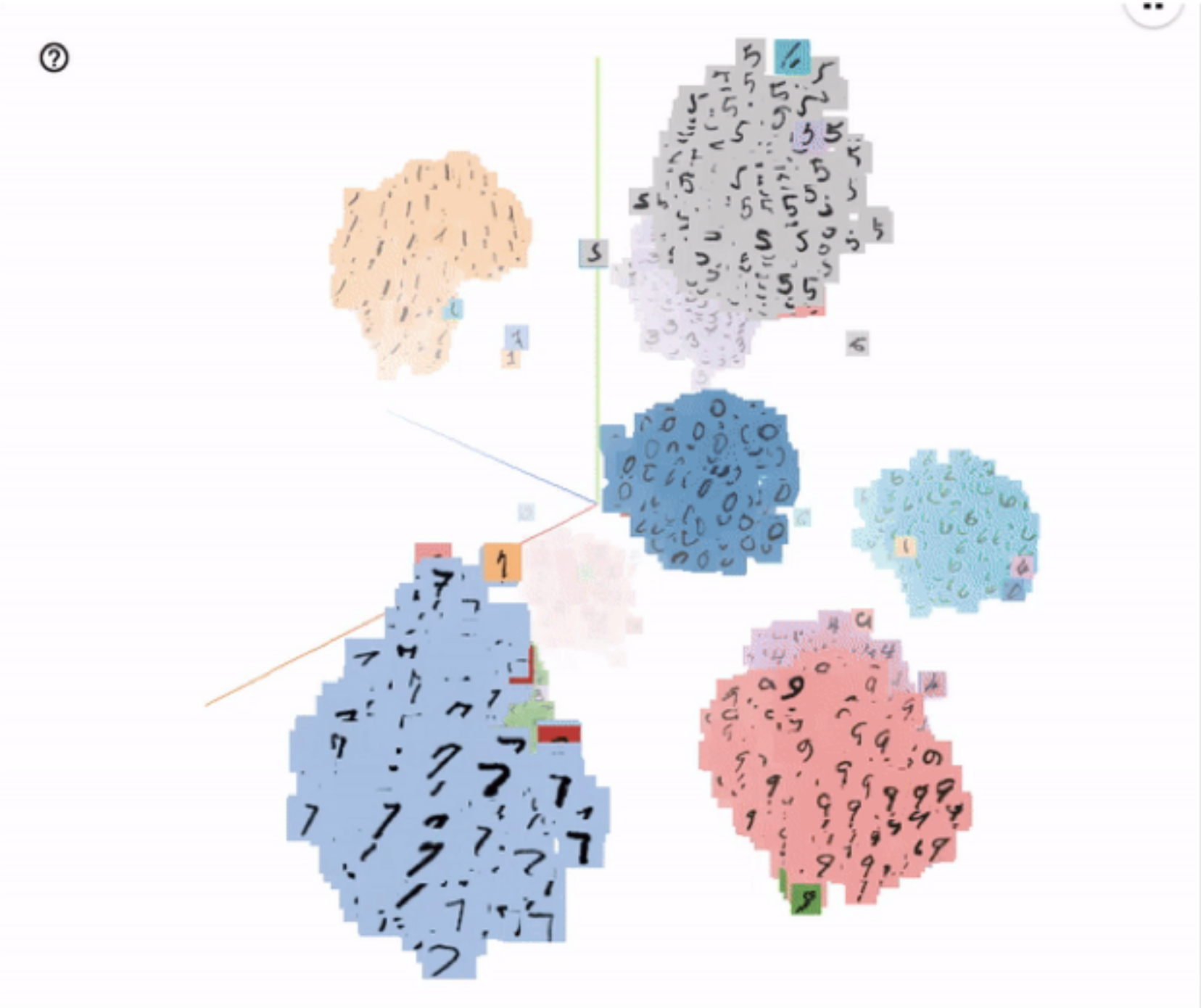
Image

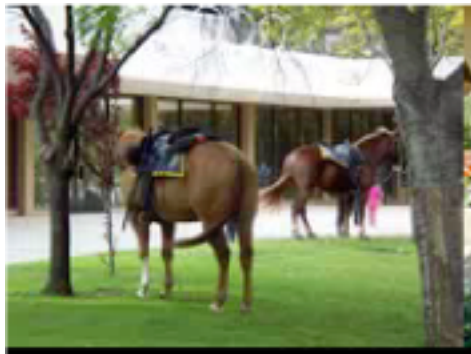
Double Helix

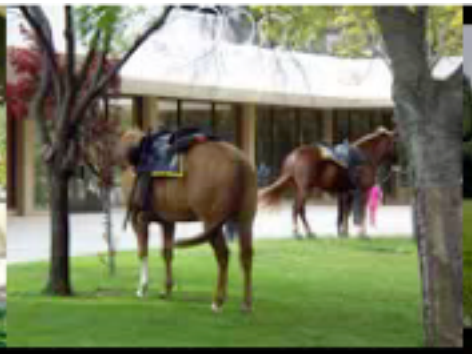
DeepText

Image embeddings

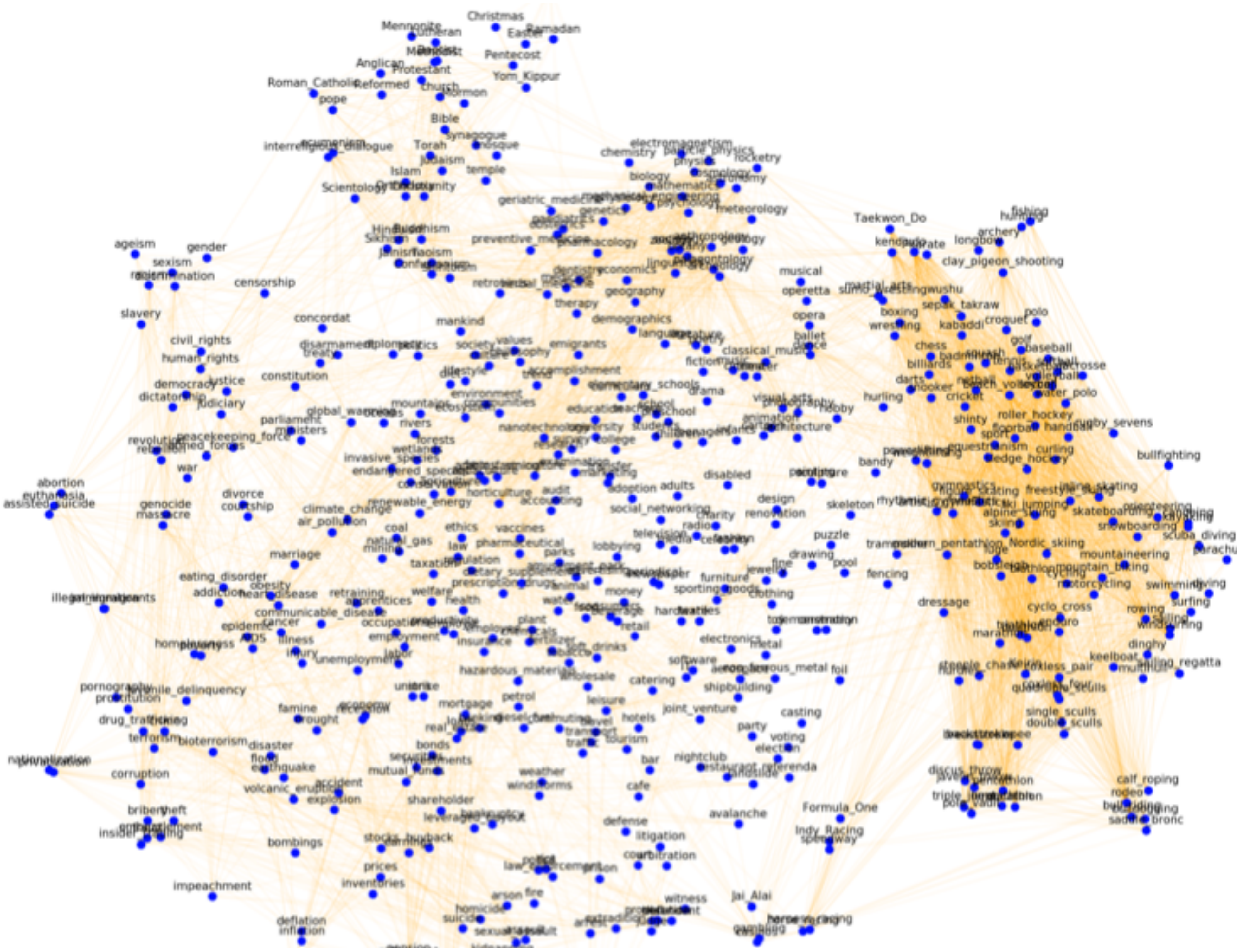












Christmas
Mennonite
Lutheran
Easter
Ramadan
Pentecost
Anglican
Protestant
Yom_Kippur
Roman_Catholic
Reformed
church
Mormon
pope
Bible
synagogue
Torah
mosque
Judaism
temple
Islam
Scientology
Hinduism
Buddhism
Sikhism
Taoism
electromagnetism
chemistry
particle_physics
rocketry
physics
biology
cosmology
geriatric_medicine
meteorology
genetics
preventive_medicine
pharmacology
economics
geography
therapy
demographics
mankind
concordat
censorship
ageism
gender
sexism
discrimination
slavery
civil_rights
human_rights
democracy
justice
dictatorship
judiciary
constitution
parliament
ministers
global_warming
mountains
rivers
forests
wetlands
invasive_species
endangered_species
peacekeeping_force
armed_forces
war
abortion
euthanasia
assisted_suicide
genocide
massacre
divorce
courtship
climate_change
air_pollution
renewable_energy
horticulture
audit
accounting
coal
natural_gas
mining
ethics
vaccines
law
pharmaceutical
regulation
taxation
dietary_supplement
prescription_drugs
marriage
eating_disorder
obesity
addiction
heart_disease
retraining
apprentices
welfare
health
communicable_disease
cancer
epidemic
AIDS
illness
injury
unemployment
labor
employment
insurance
hazardous_material
tobacco
wholesale
petrol
famine
drought
mortgage
recession
economy
juvenile_delinquency
pornography
prostitution
drug_trafficking
terrorism
bioterrorism
disaster
flood
earthquake
corruption
volcanic_eruption
accident
explosion
bribery
theft
embezzlement
insider_trading
bombings
impeachment
prices
inventories
deflation
inflation
shareholder
bankruptcy
leveraged_buyout
stocks
buyback
mutual_funds
investments
bonds
real_estate
weather
windstorms
law_enforcement
prison
arson
fire
homicide
suicide
sexual_assault
witness
extradition
arrest
defense
litigation
arbitration
avalanche
Formula_One
Indy_Racing
speedway
Jai_Alai
gambling
horse_racing
nightclub
restaurant
referenda
landslide
bar
cafe
party
voting
election
casting
leisure
hotels
tourism
traffic
joint_venture
catering
shipbuilding
software
electronics
metal
foil
retail
money
sporting_goods
clothing
furniture
jewellery
drawing
puzzle
pool
television
media
radio
charity
social_networking
design
renovation
adults
adoption
disabled
skeleton
bandy
musical
operetta
opera
ballet
dance
classical_music
fiction
drama
photography
hobby
animation
architecture
visual_arts
education
school
college
university
students
children
teenagers
infants
nanotechnology
emigrants
values
society
philosophy
lifestyle
environment
communities
ecosystem
Taekwon_Do
kendo
karate
longbow
archery
hunting
fishing
clay_pigeon_shooting
martial_arts
sumo_wrestling
wushu
boxing
wrestling
sepak_takraw
kabaddi
croquet
polo
golf
baseball
chess
badminton
tennis
softball
billiards
basketball
lacrosse
darts
snooker
netball
volleyball
hurling
cricket
beach_volleyball
water_polo
shinty
roller_hockey
rugby_sevens
floorball
handball
equestrianism
curling
sledge_hockey
bullfighting
gymnastics
freestyle_skiing
inline_skating
ski_jumping
alpine_skiing
skiing
skateboarding
snowboarding
orienteering
scuba_diving
parachu
modern_pentathlon
Nordic_skiing
luge
bobsleigh
biathlon
mountaineering
mountain_biking
cycling
motorcycling
swimming
diving
surfing
rowing
windsurfing
sailing
fencing
dressage
cyclo_cross
enduro
marathon
triathlon
dinghy
keelboat
multihull
sailing_regatta
steeple_chase
coxless_pair
quadruple_sculls
single_sculls
double_sculls
discus_throw
javelin_throw
pentathlon
triple_jump
heptathlon
pole_vault
calf_roping
rodeo
bull_riding
saddle_bronc

麻辣烫 ——

麻辣拌, 0.884719466741
冒菜, 0.878425403476
麻辣汤, 0.872023557375
杨国福, 0.863779634807
云南米线, 0.855127326166
张亮, 0.833823338064
过桥米线, 0.833199404157
云南过桥米线, 0.825620560846
羊肉米线, 0.824504212524
重庆小面, 0.823935462376

肯德基 ——

KFC, 0.982491910803
kfc, 0.970919196284
麦当劳, 0.94600897271
德克士, 0.945771247431
华莱士, 0.94466343164
汉堡王, 0.938037115861
正新鸡排, 0.920964315301
派乐, 0.91322924147
豪大大, 0.903992194302
正新, 0.903950289626

黄焖鸡 ——

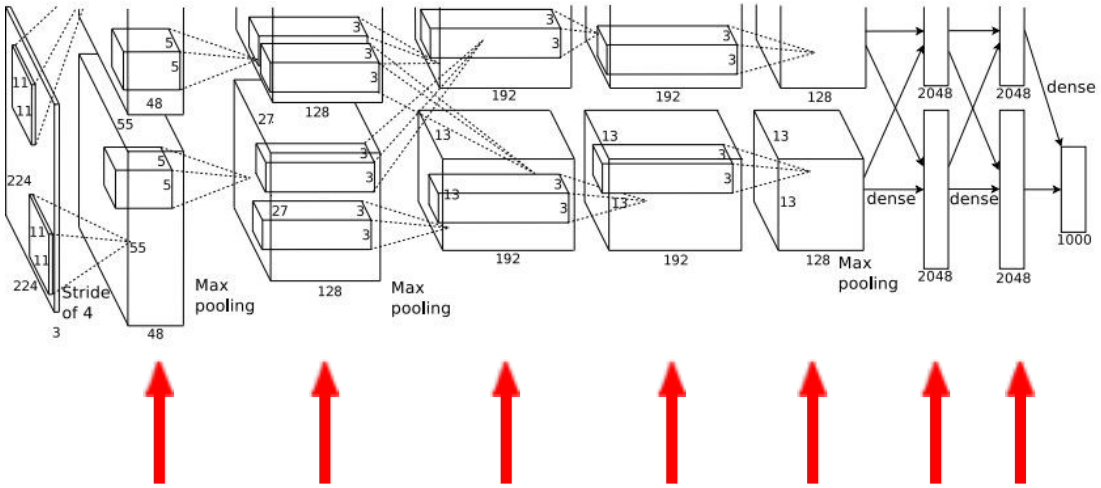
黄焖鸡米饭, 0.952911757759
鸡公煲, 0.91343873243
鸡米饭, 0.90181777612
大盘鸡, 0.876224767056
黄闷鸡, 0.875185069057
麻辣香锅, 0.869952513231
香锅, 0.857016354508
鸡汤泡饭, 0.821301219974
水煮肉, 0.81307743024
炒菜, 0.808240640778

馄饨 ————

混沌, 0.971946557298
馄炖, 0.959630998451
饺子, 0.942945794257
水饺, 0.926106932097
锅贴, 0.923119816702
小混沌, 0.914481422287
生煎, 0.900458398145
汤包, 0.9000372527
小馄饨, 0.897959492742
馄钝, 0.896437234474

What's going on inside ConvNets?

Input Image:
3 x 224 x 224

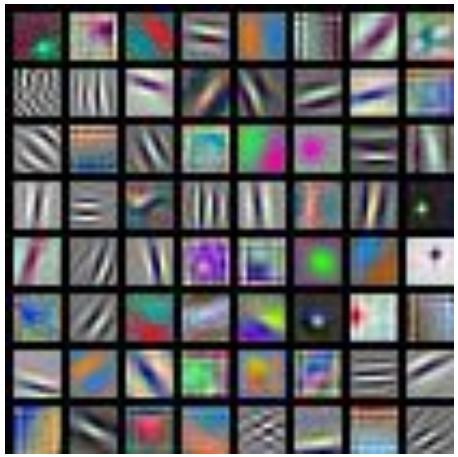


Class Scores:
1000 numbers

What are the intermediate features looking for?

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

First Layer: Visualize Filters

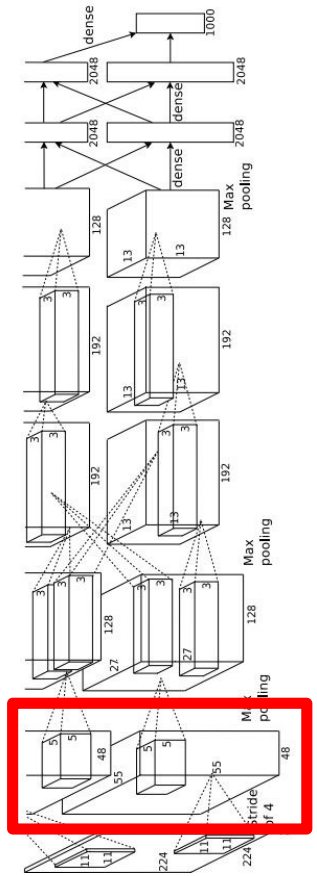


AlexNet:
64 x 3 x 11 x 11

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)

Weights:

layer 1 weights

16 x 3 x 7 x 7

Weights:

layer 2 weights

20 x 16 x 7 x 7

Weights:

layer 3 weights

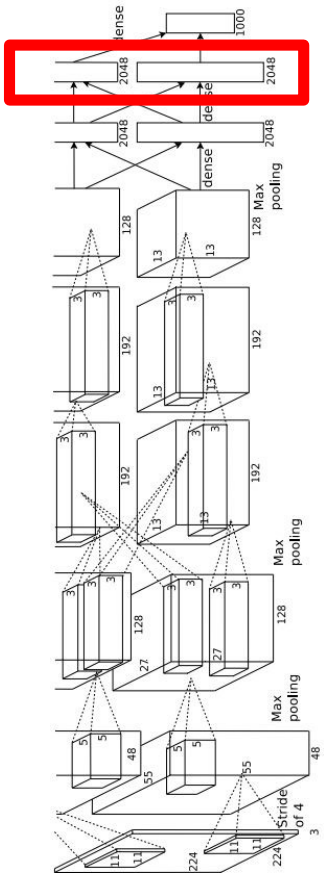
20 x 20 x 7 x 7

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

(from ConvNetJS CIFAR-10 demo)

Last Layer

FC7 layer

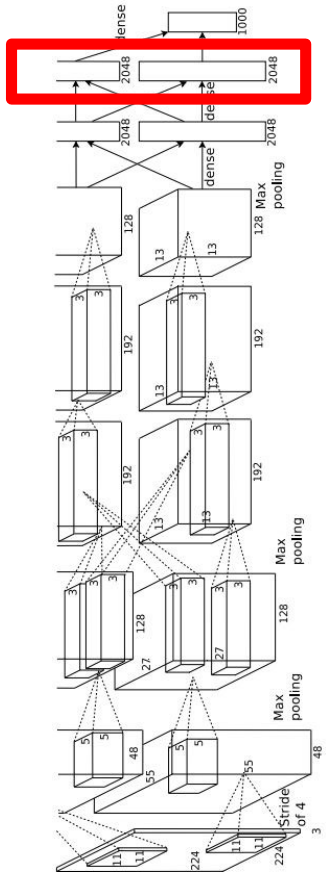


4096-dimensional feature vector for an image (layer immediately before the classifier)

Run the network on many images, collect the feature vectors

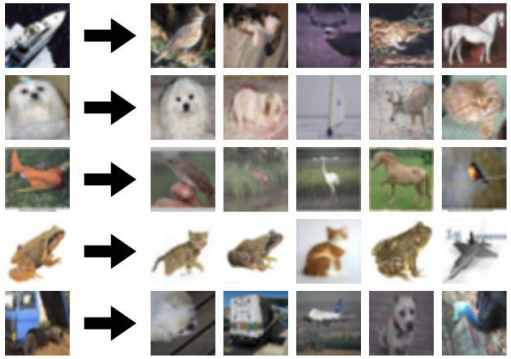
Last Layer: Nearest Neighbors

4096-dim vector



Test image L2 Nearest neighbors in [redacted] space

Recall: Nearest neighbors in [redacted] space



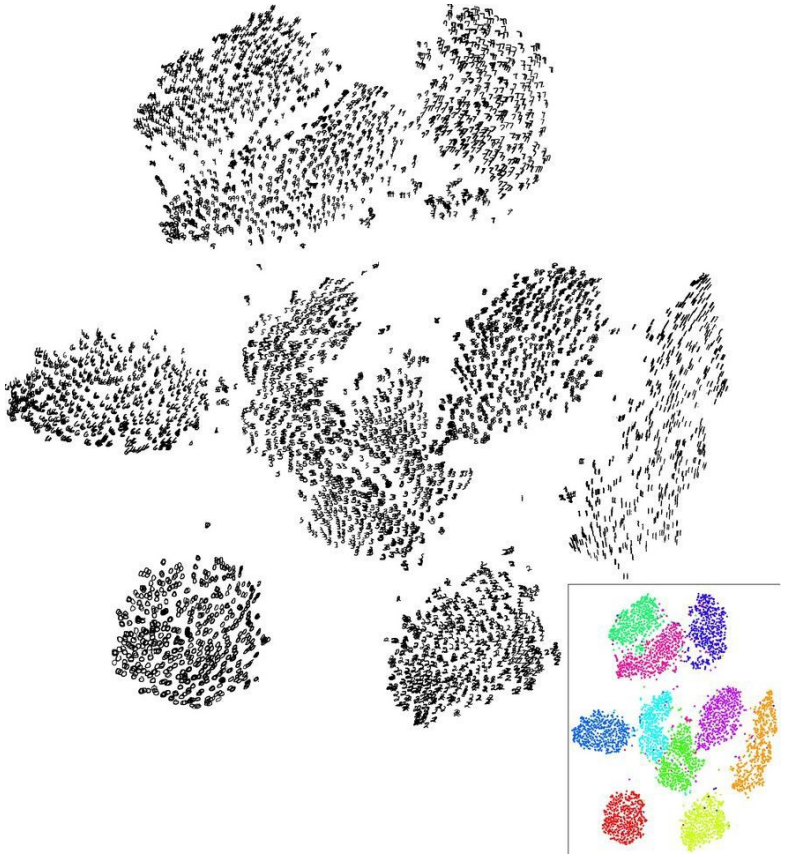
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012.
Figures reproduced with permission.

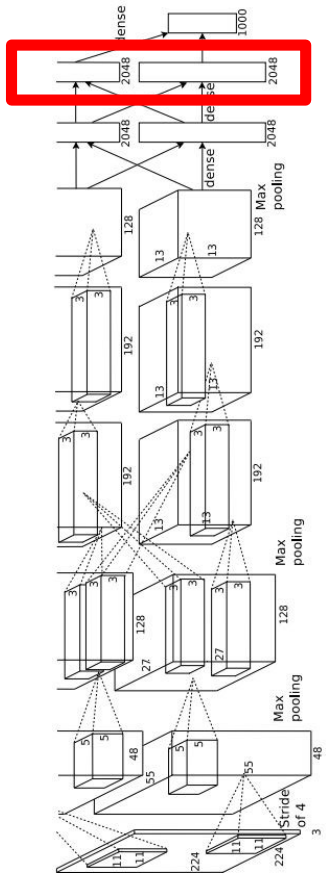
Last Layer: Dimensionality Reduction

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

Simple algorithm: Principal Component Analysis (PCA)

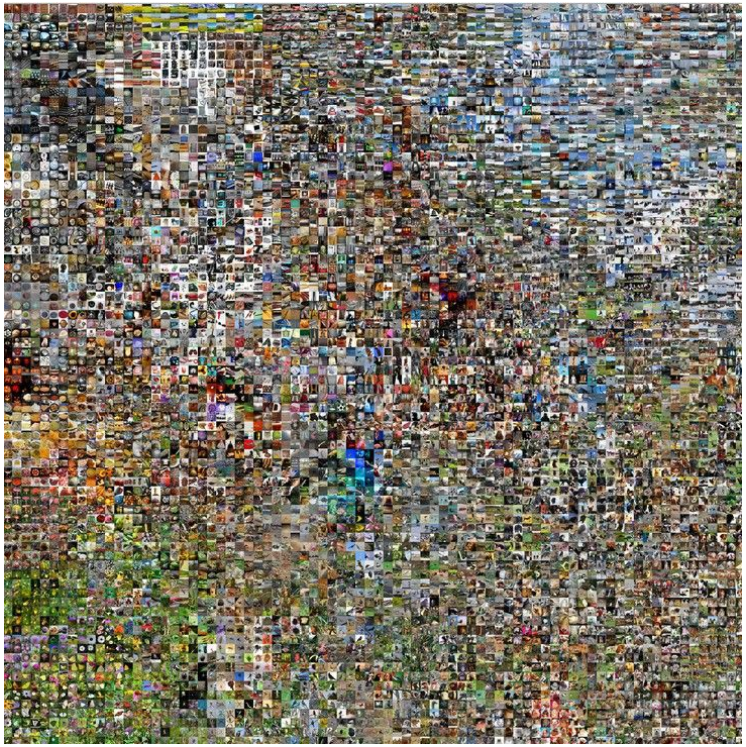
More complex: **t-SNE**

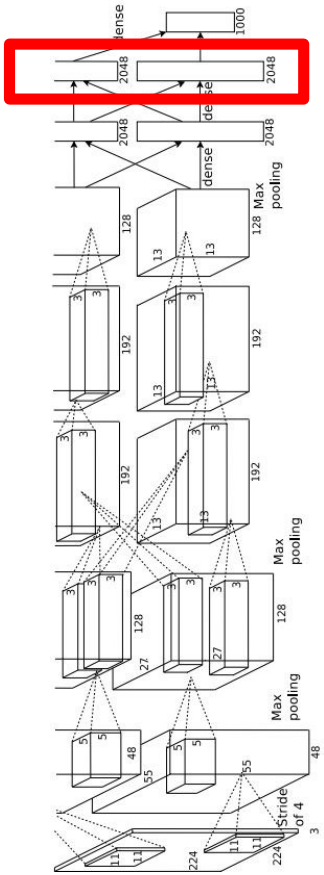




Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008

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

Deep Visualization Toolbox

yosinski.com/deepvis

#deepvis


Jason Yosinski


Jeff Clune


Anh Nguyen


Thomas Fuchs


Hod Lipson

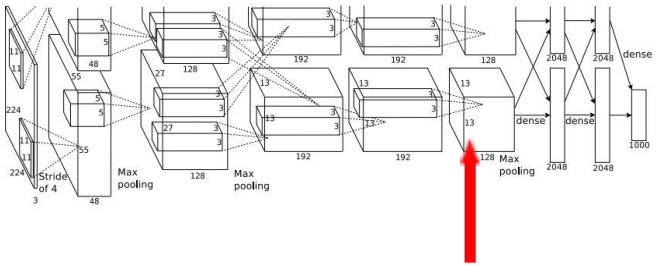
Cornell University

University of Wyoming

NASA Jet Propulsion Laboratory
California Institute of Technology

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

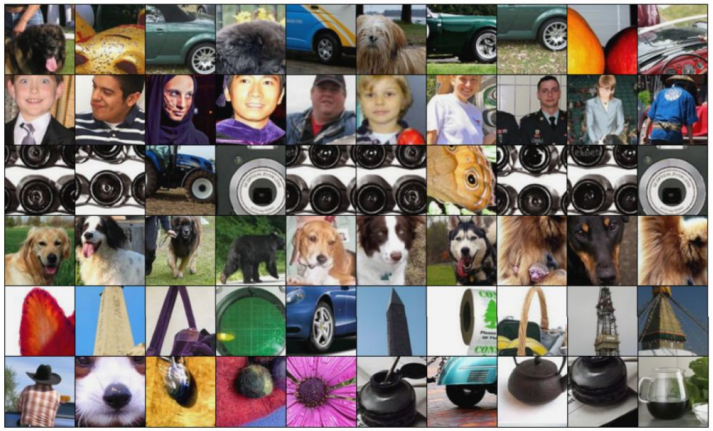
Maximally Activating Patches



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

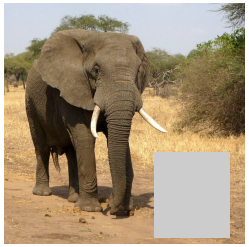
Visualize image patches that correspond to maximal activations

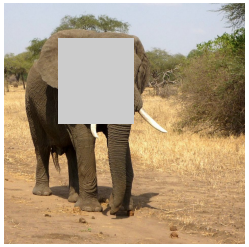


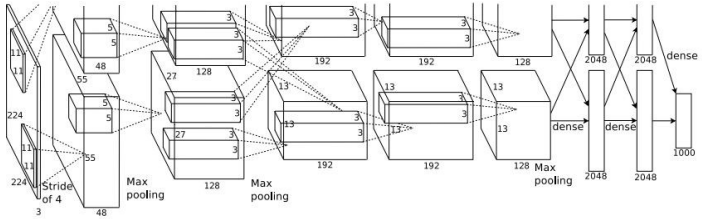
Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015

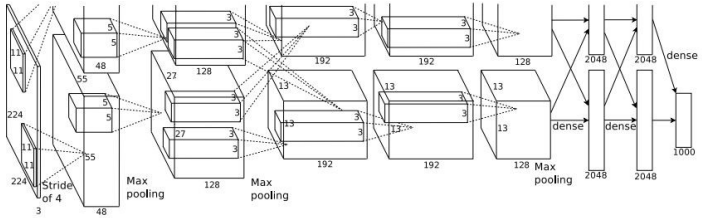
Which pixels matter: Saliency vs Occlusion

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

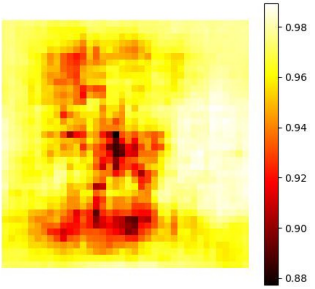




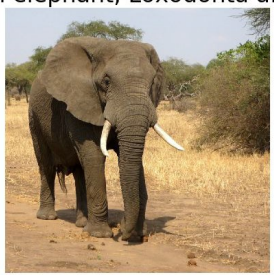


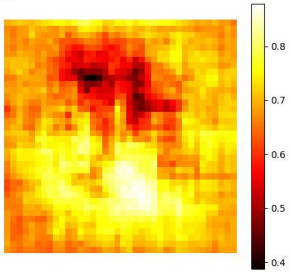


schooner



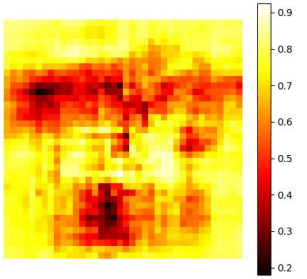
African elephant, Loxodonta africana





P(e

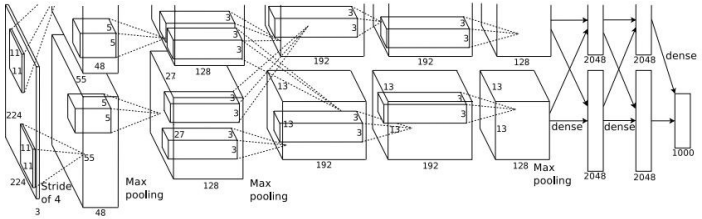
go-kart



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

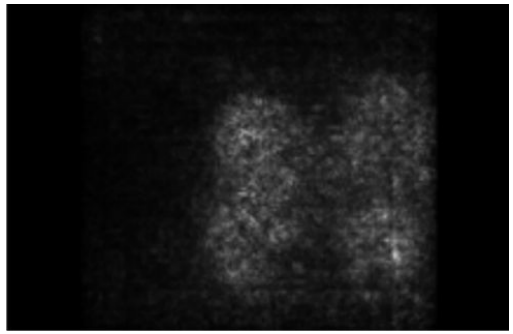
Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities



Dog

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



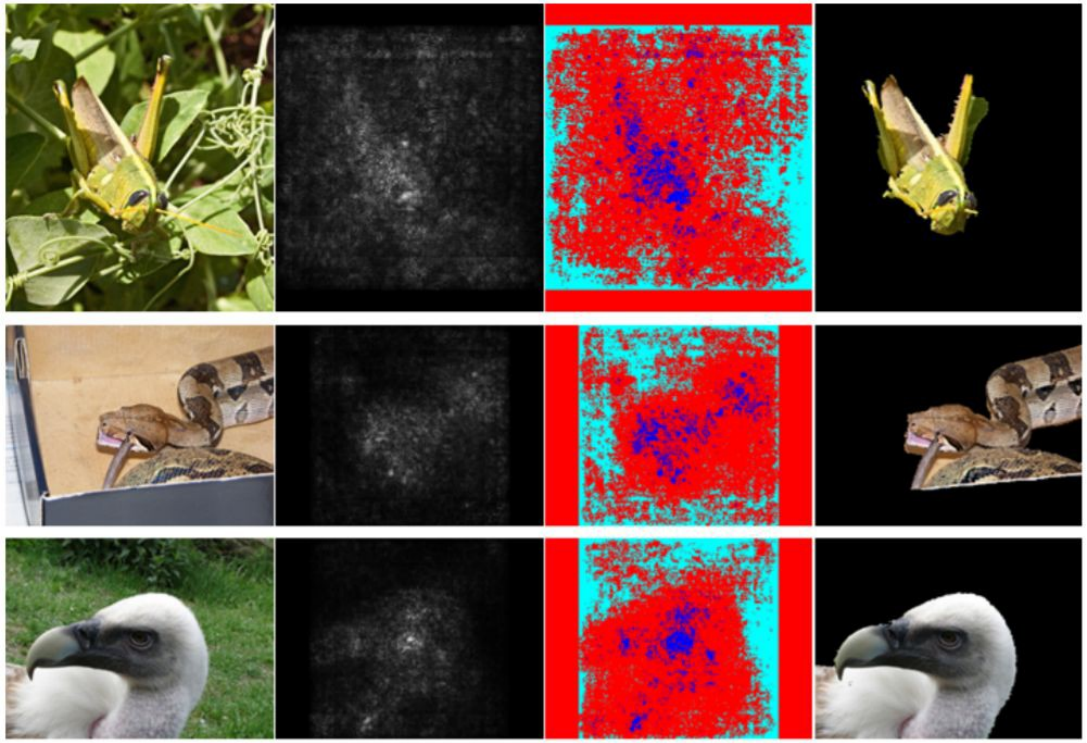
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

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

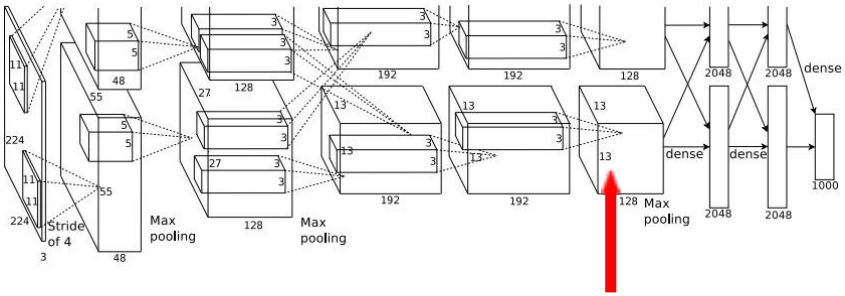
Use GrabCut on saliency map



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

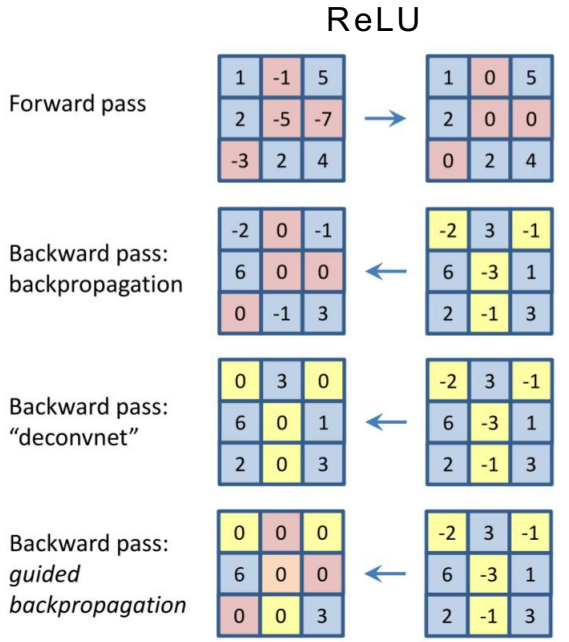
Rother et al, "Grabcut: Interactive foreground extraction using iterated graph cuts", ACM TOG 2004

Intermediate Features via (guided) backprop



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

Compute gradient of neuron value with respect to image pixels



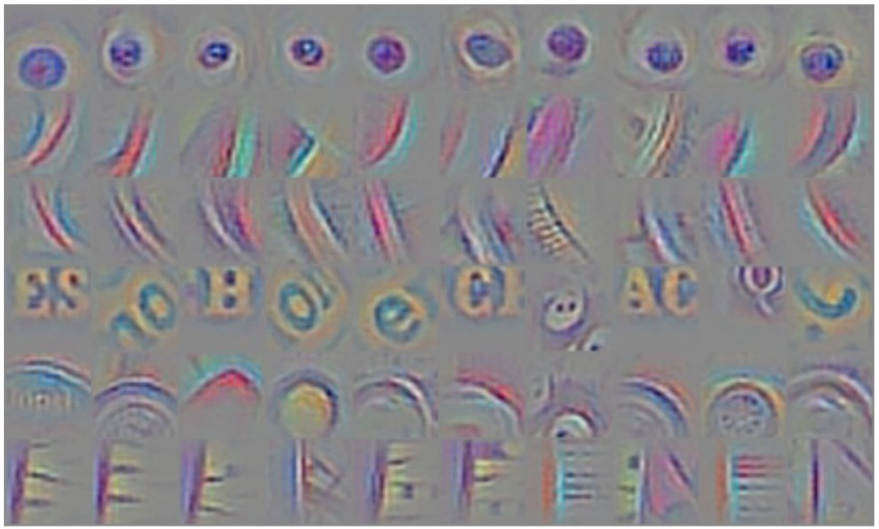
Images come out nicer if you only backprop positive gradients through each ReLU (guided backprop)

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

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

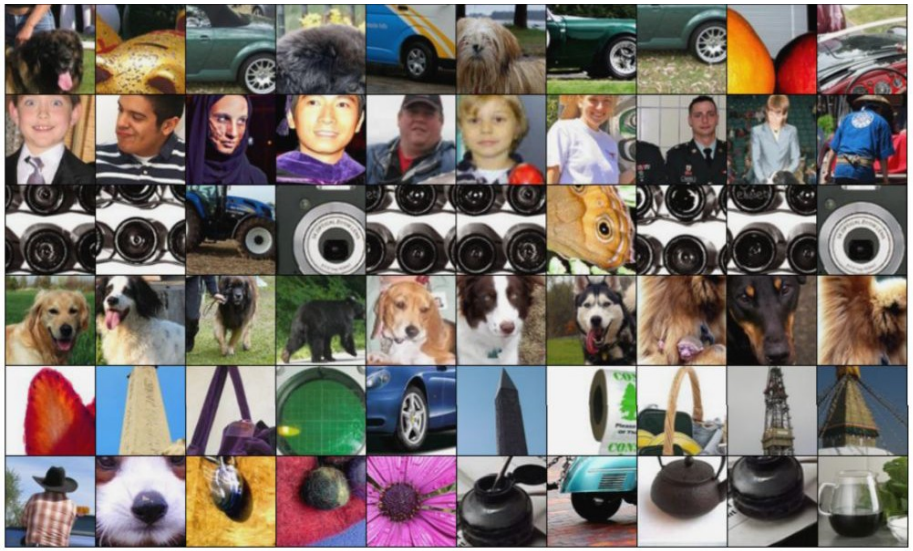
Maximally activating patches
(Each row is a different neuron)



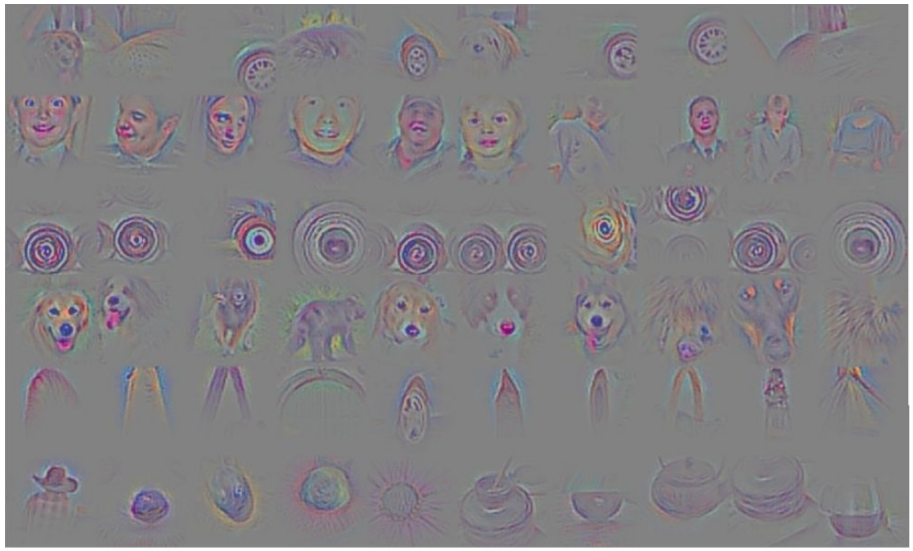
Guided Backprop

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

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 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

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

Neuron value

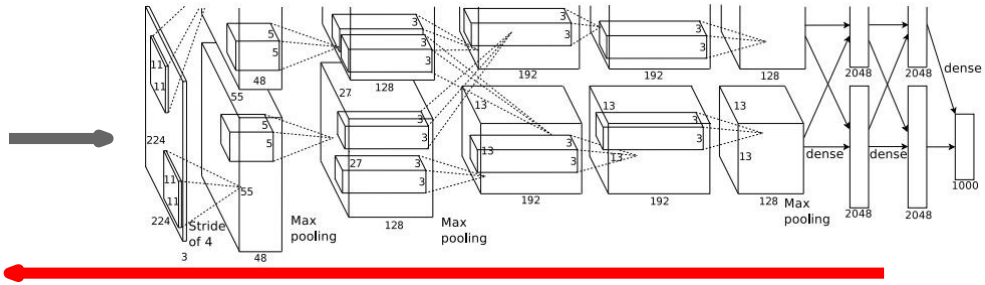
Natural image regularizer

Visualizing CNN features: Gradient Ascent

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

score for class c (before Softmax)

1. Initialize image to zeros



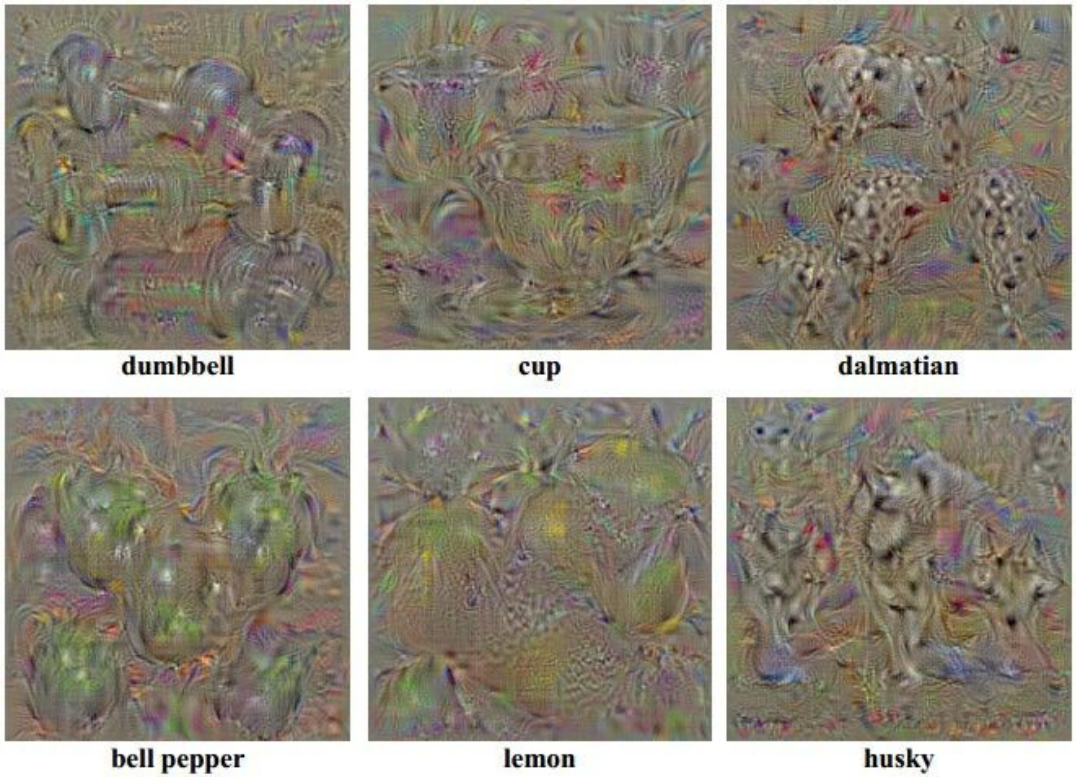
Repeat:

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

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

Simple regularizer: Penalize L2 norm of generated image

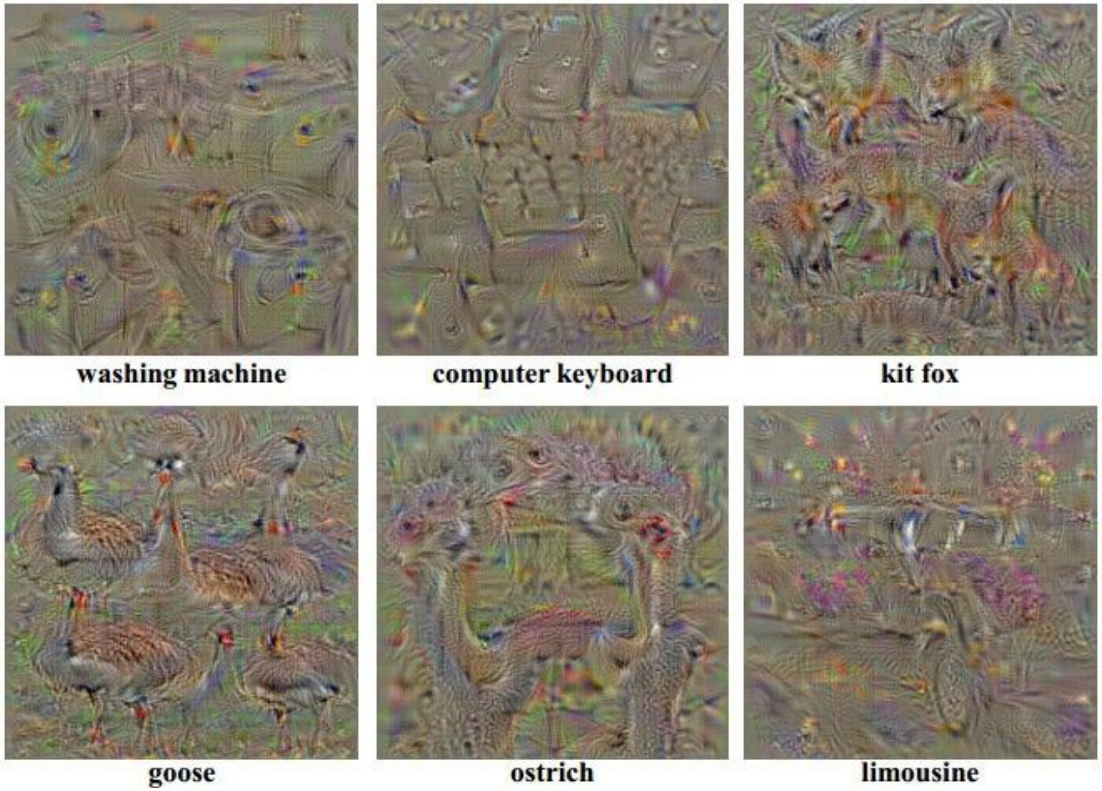


Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

Visualizing CNN features: Gradient Ascent

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

Simple regularizer: Penalize L2 norm of generated image



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.

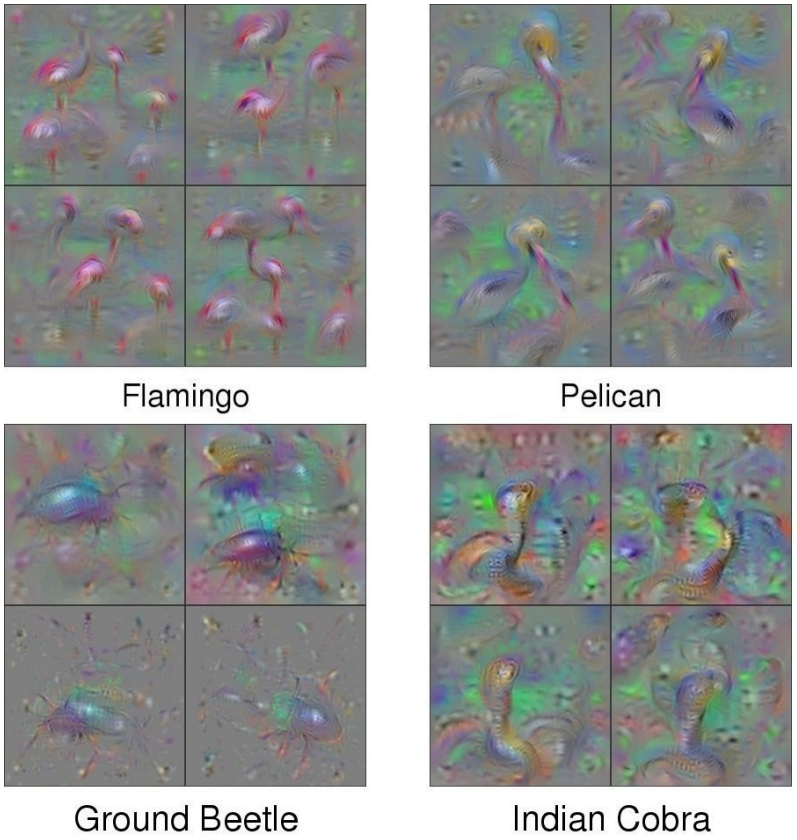
Visualizing CNN features: Gradient Ascent

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

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.

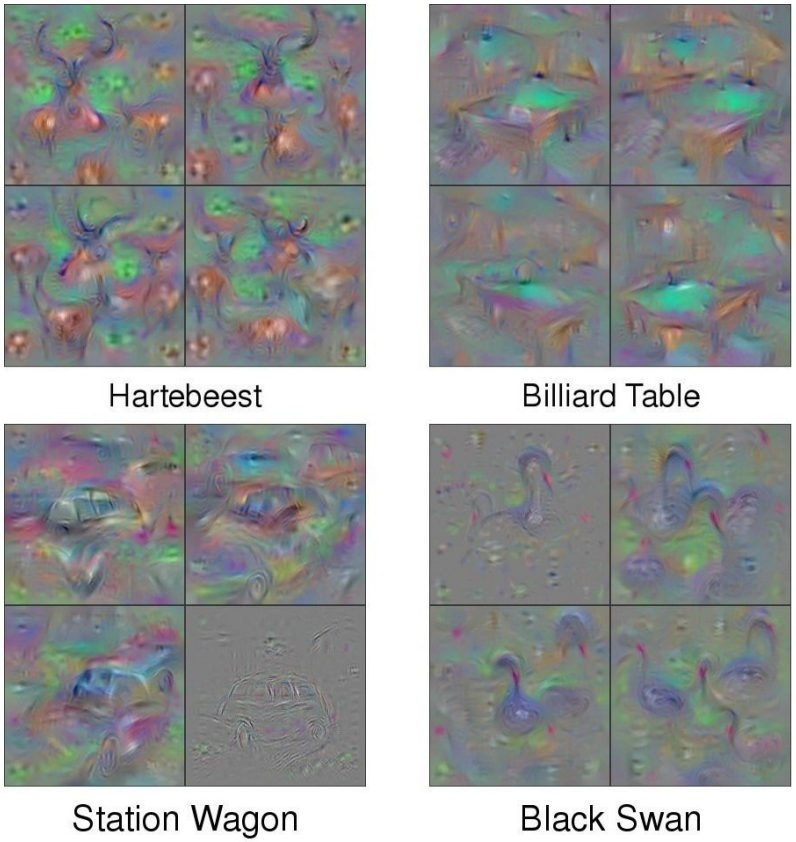
Visualizing CNN features: Gradient Ascent

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

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

periodically

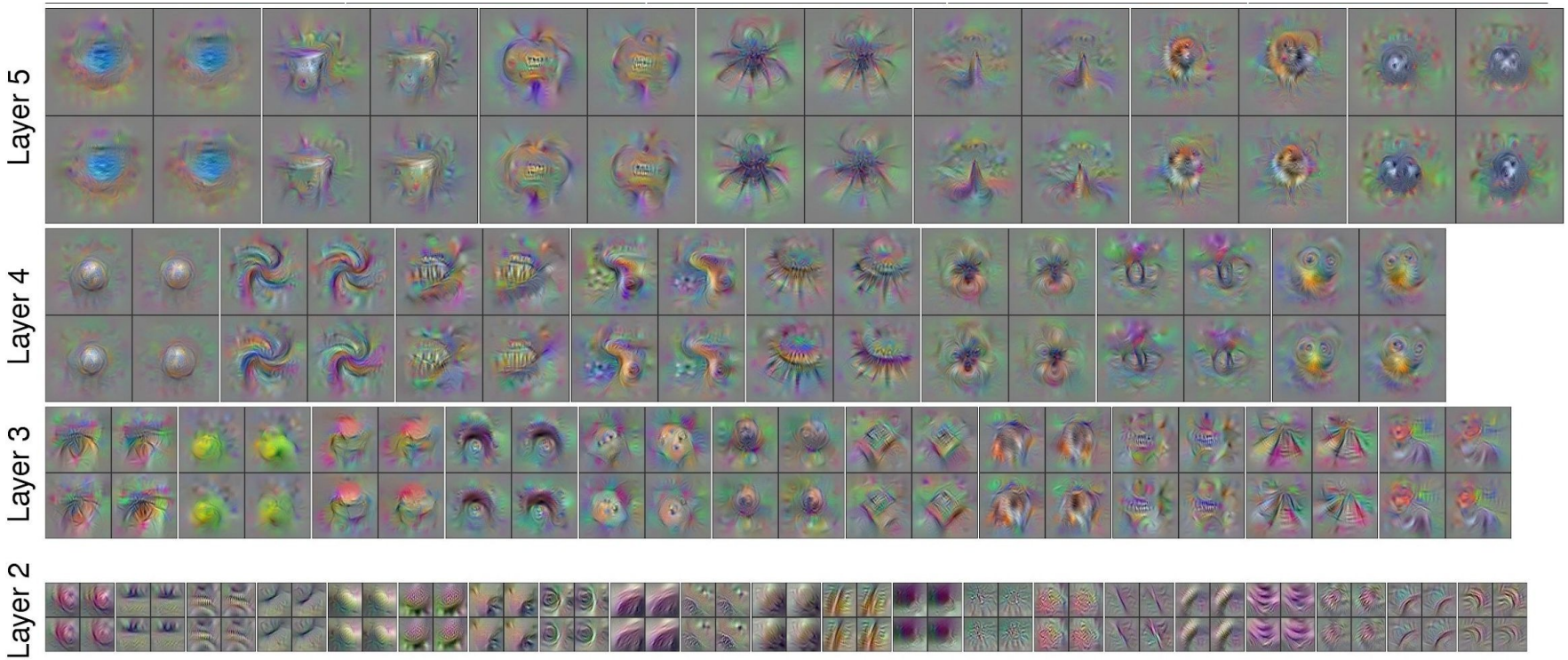
(1) Gaussian blur image
(2) Clip pixels with small values to 0
(3) Clip pixels with small gradients to 0



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

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.

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

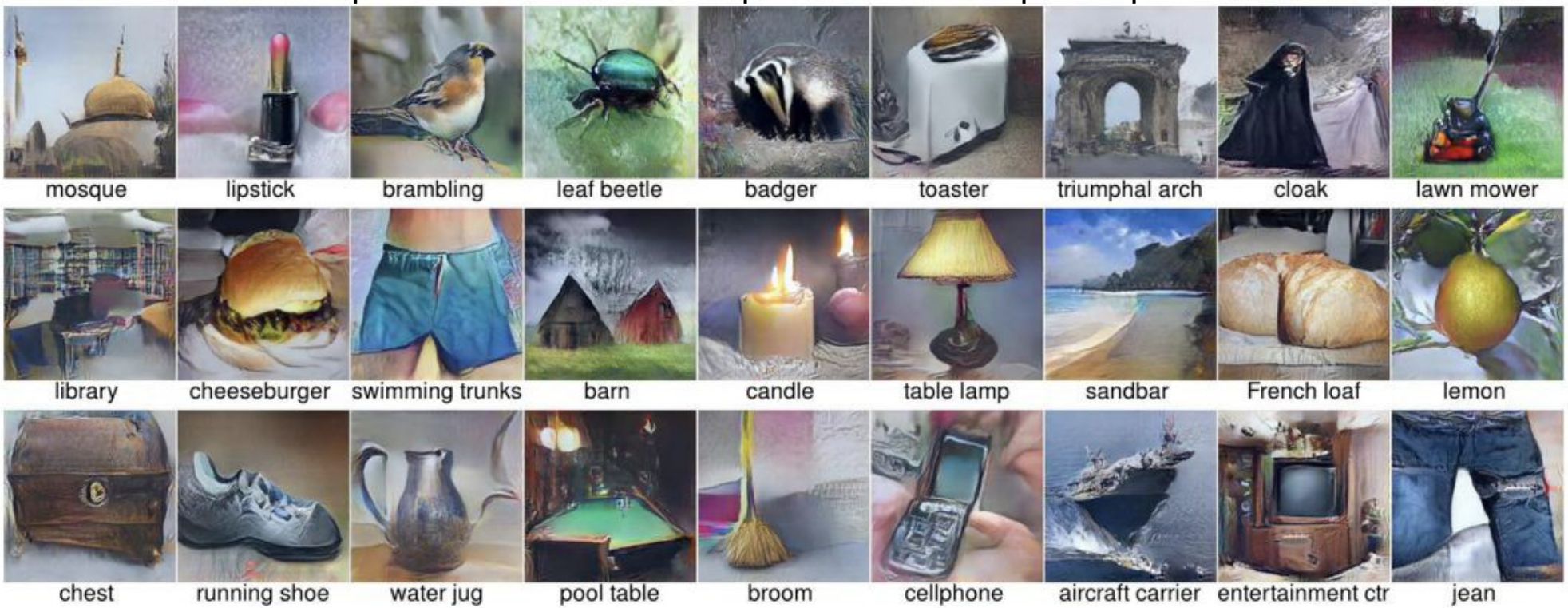
Nguyen et al, “Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks”, ICML Visualization for Deep Learning Workshop 2016.

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.

Visualizing CNN features: Gradient Ascent

Optimize in FC6 latent space instead of pixel space:

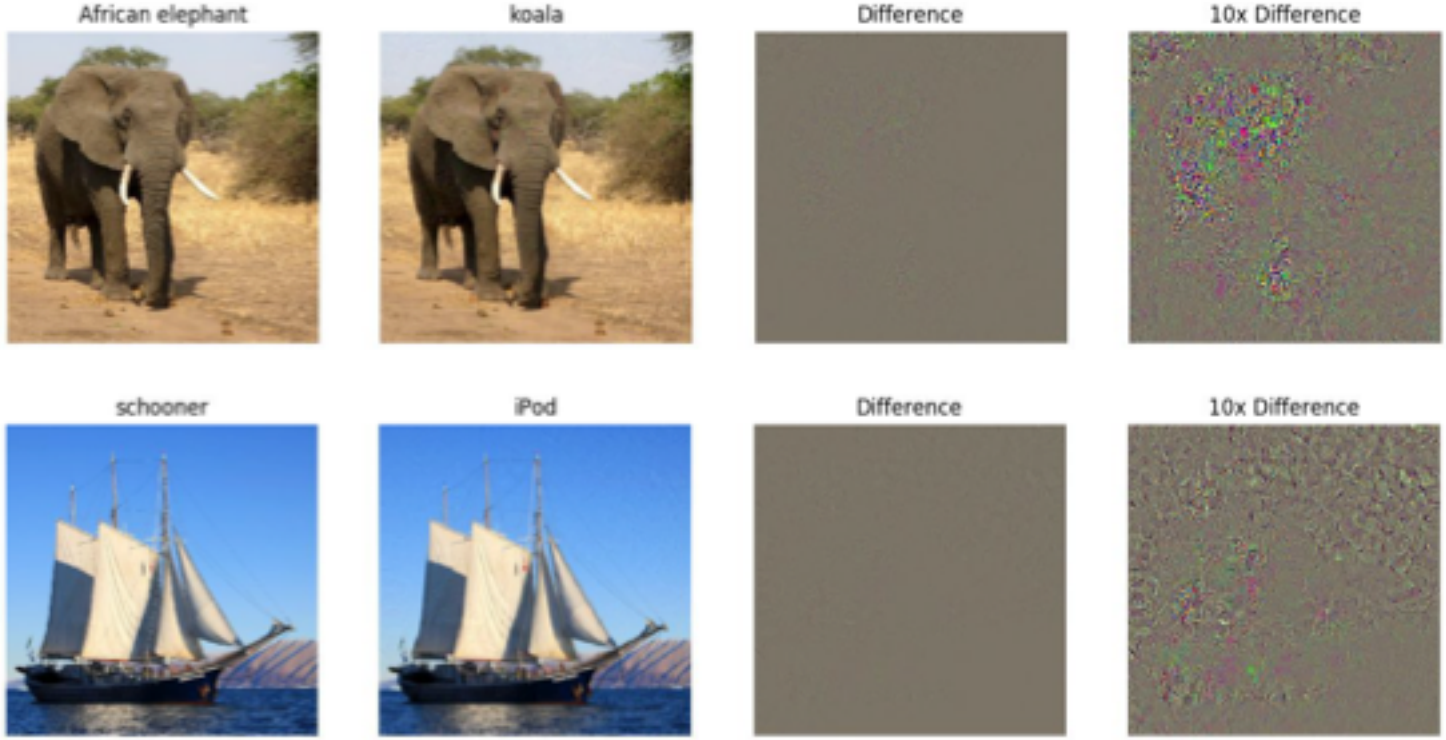


Nguyen et al, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," NIPS 2016

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

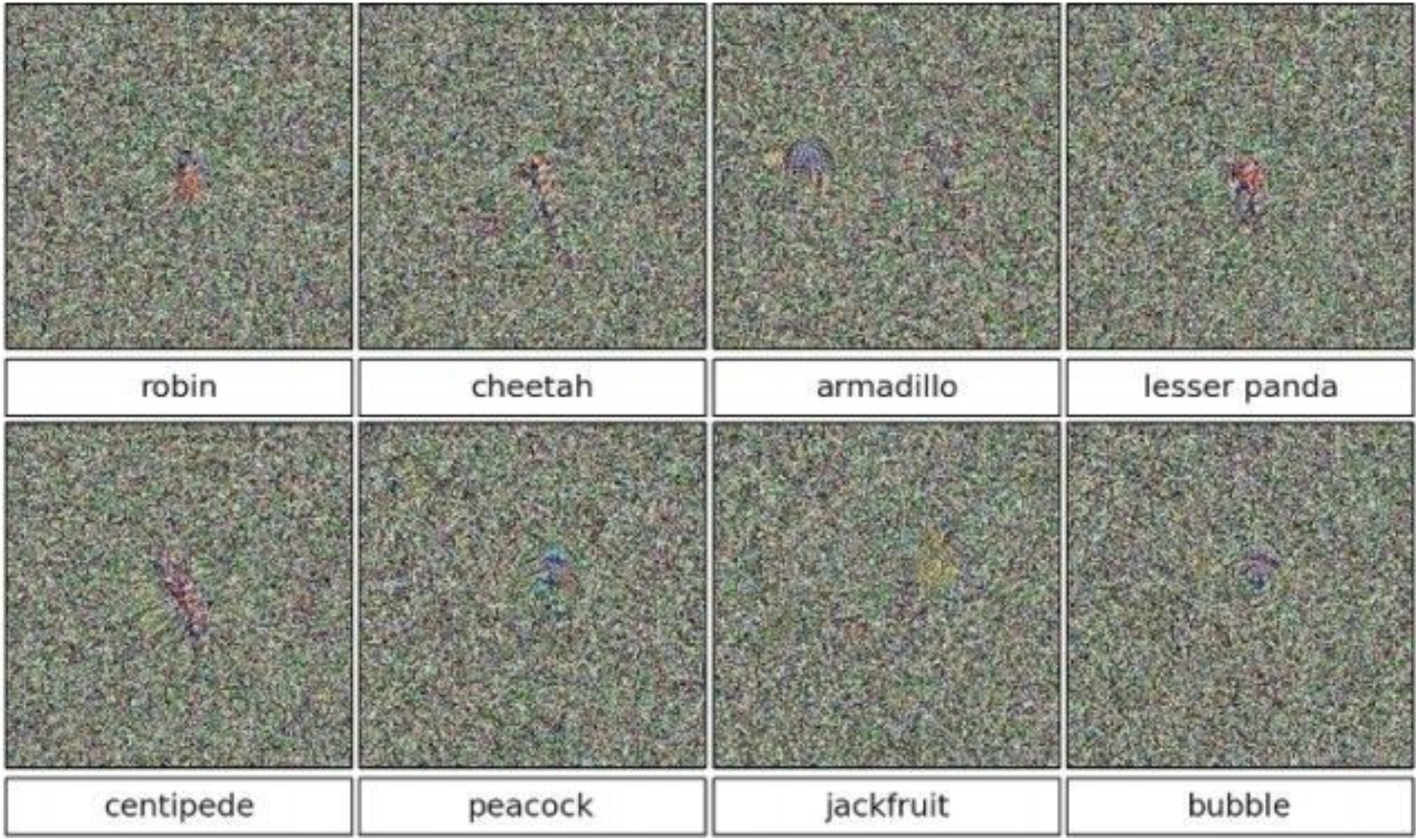
Fooling Images / Adversarial Examples



[Intriguing properties of neural networks, Szegedy et al., 2013]

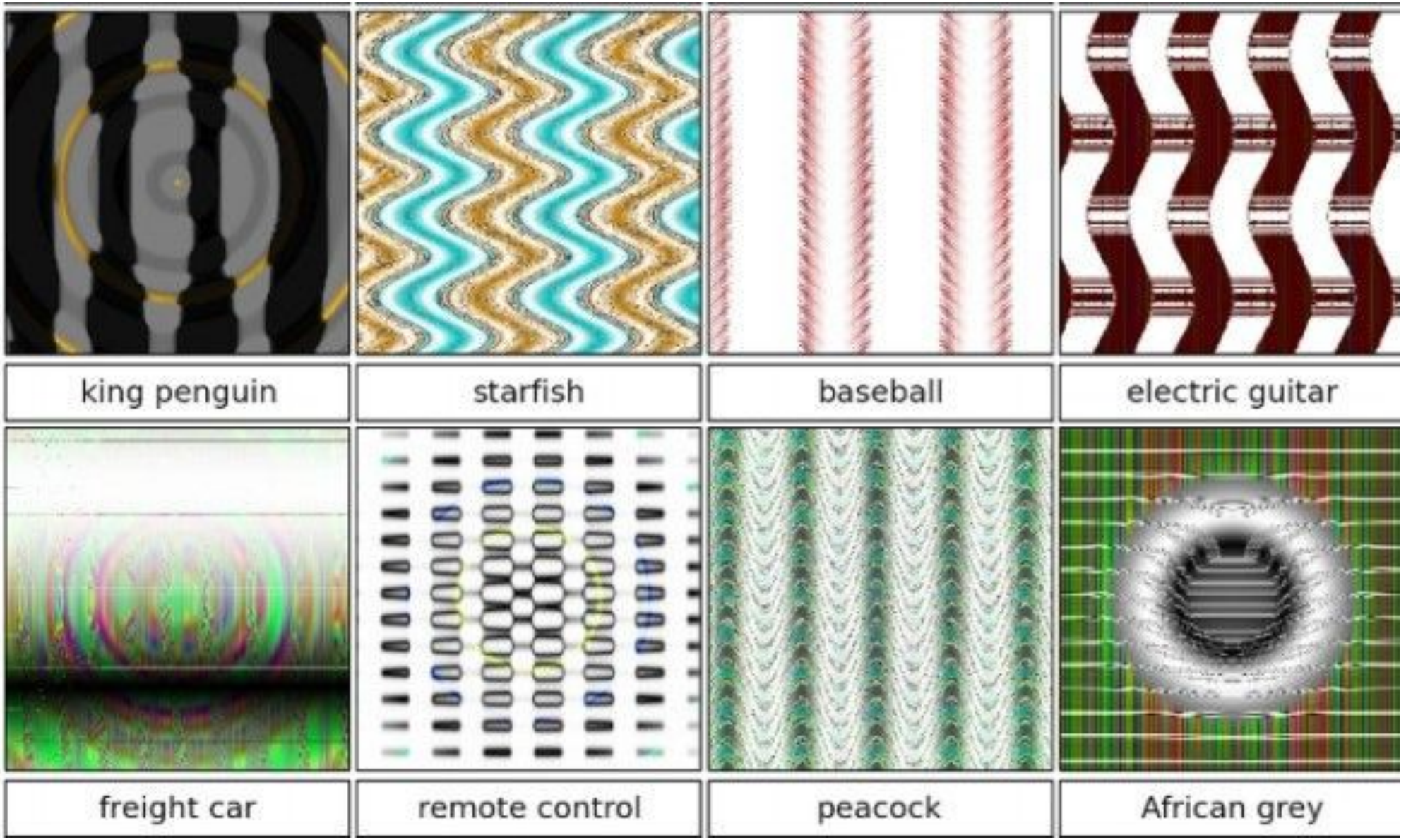
[Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images Nguyen, Yosinski, Clune, 2014]

>99.6% confidences



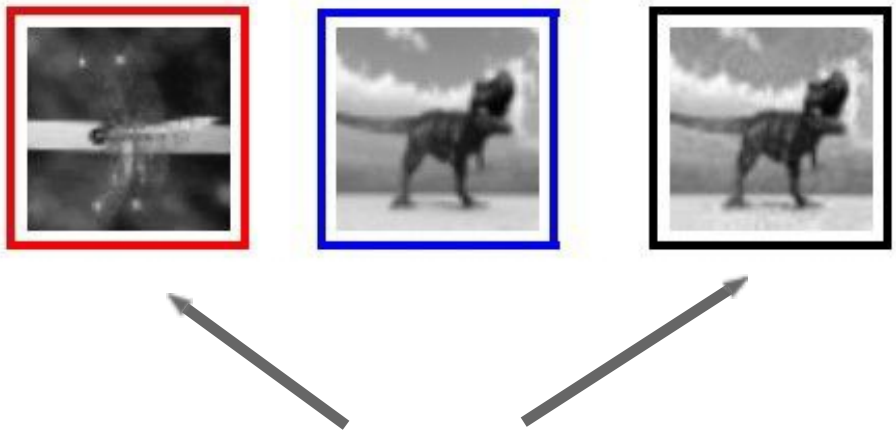
[Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images Nguyen, Yosinski, Clune, 2014]

>99.6% confidences

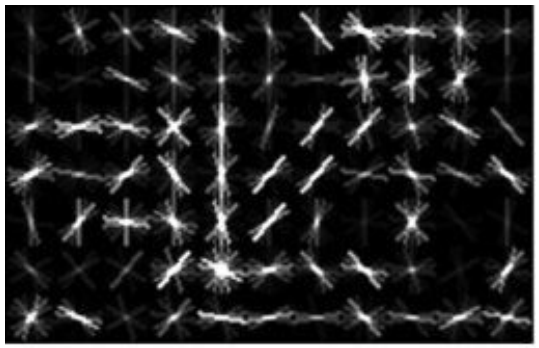


These kinds of results were around even before ConvNets…

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

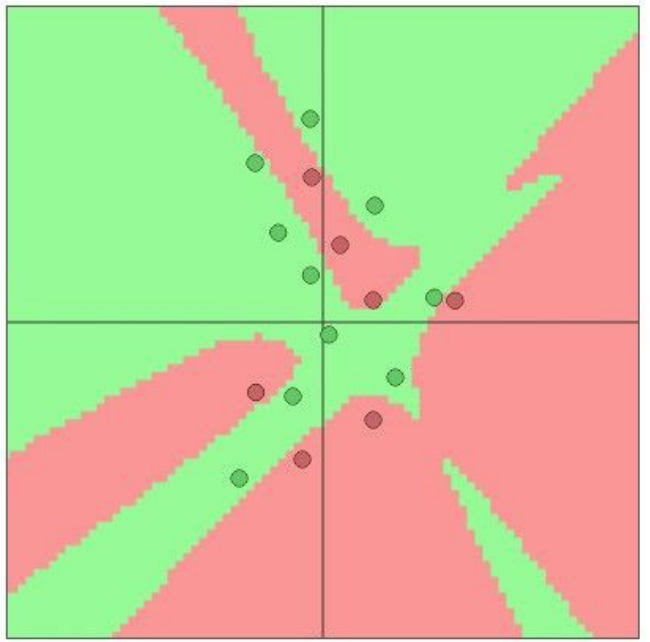


Identical HOG represention



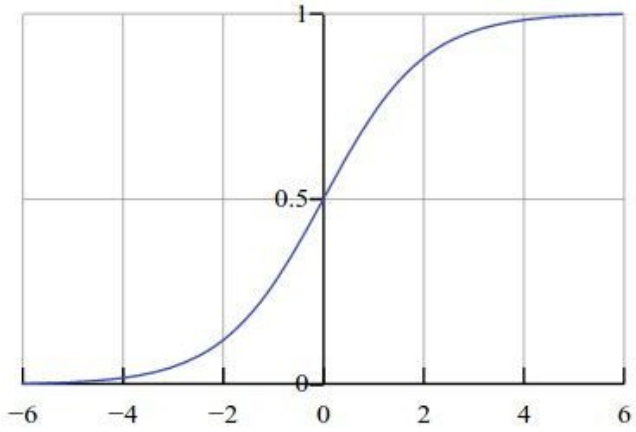
EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES
[Goodfellow, Shlens & Szegedy, 2014]

“primary cause of neural networks’ vulnerability to adversarial perturbation is their **linear nature**“



Lets fool a binary linear classifier:
(logistic regression)

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



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

Lets fool a binary linear classifier:

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

input example

weights

class 1 score = dot product:

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

=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

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

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

Lets fool a binary linear classifier:

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

input example

weights

class 1 score = dot product:

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

=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

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

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

Lets fool a binary linear classifier:

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

input example

weights

class 1 score before:

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

=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

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

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

=> probability of class 1 is now 1/(1+e^(-(2))) = 0.88

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

Lets fool a binary linear classifier:

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

input example

weights

class 1 score before:

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

=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

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

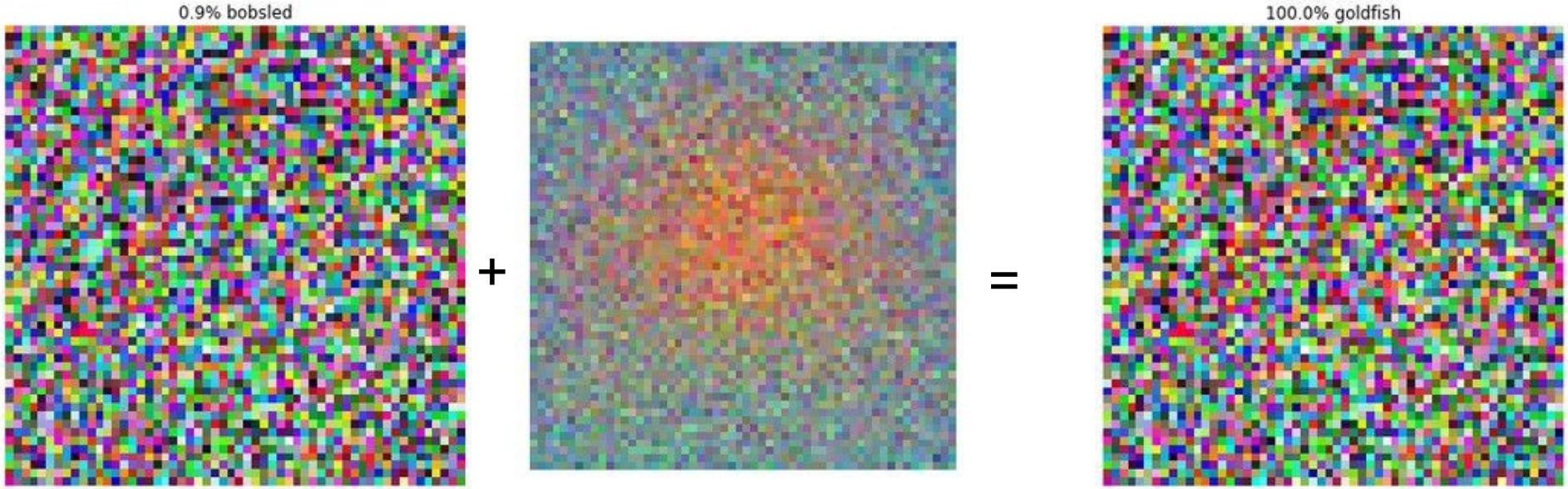
=> probability of class 1 is now 1/(1+e^(-(2))) = 0.88

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

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

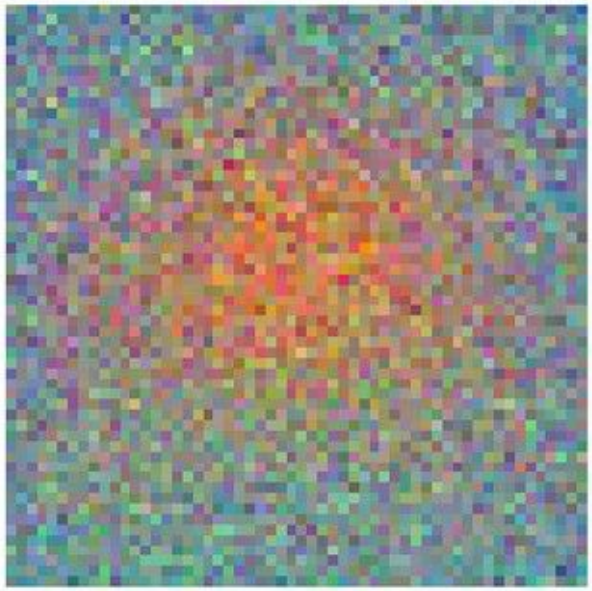
(It's significantly easier with more numbers, need smaller nudge for each)

mix in a tiny bit of Goldfish classifier weights



100% Goldfish

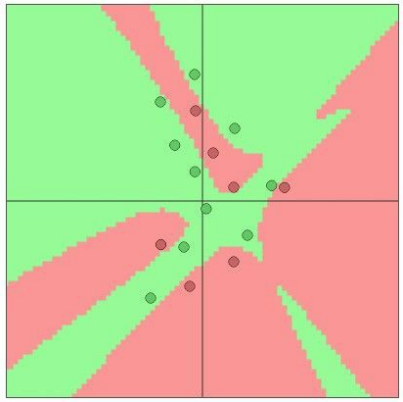

1.0% kit fox




8.0% goldfish

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

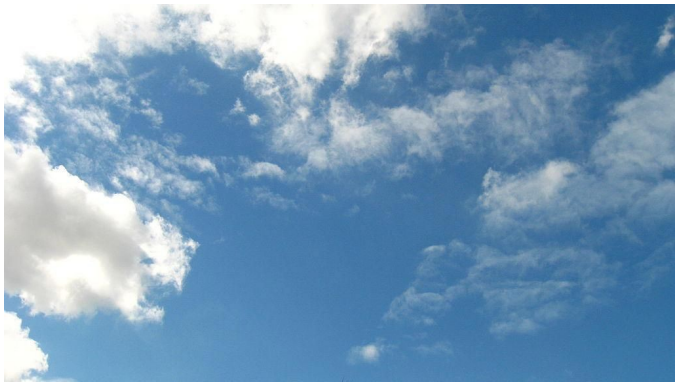
“primary cause of neural networks’ vulnerability to adversarial perturbation is their **linear nature**“
(and very high-dimensional, sparsely-populated input spaces)

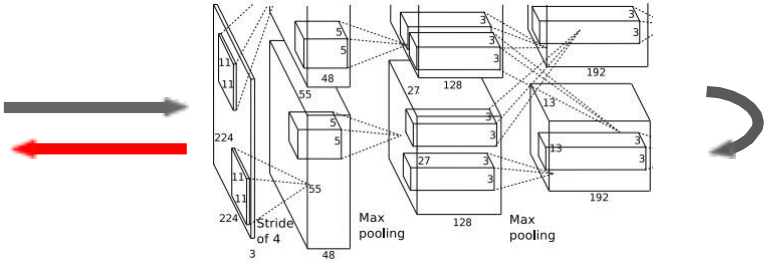


In particular, this is not a problem with Deep Learning, and has little to do with ConvNets specifically. Same issue would come up with Neural Nets in any other modalities.

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



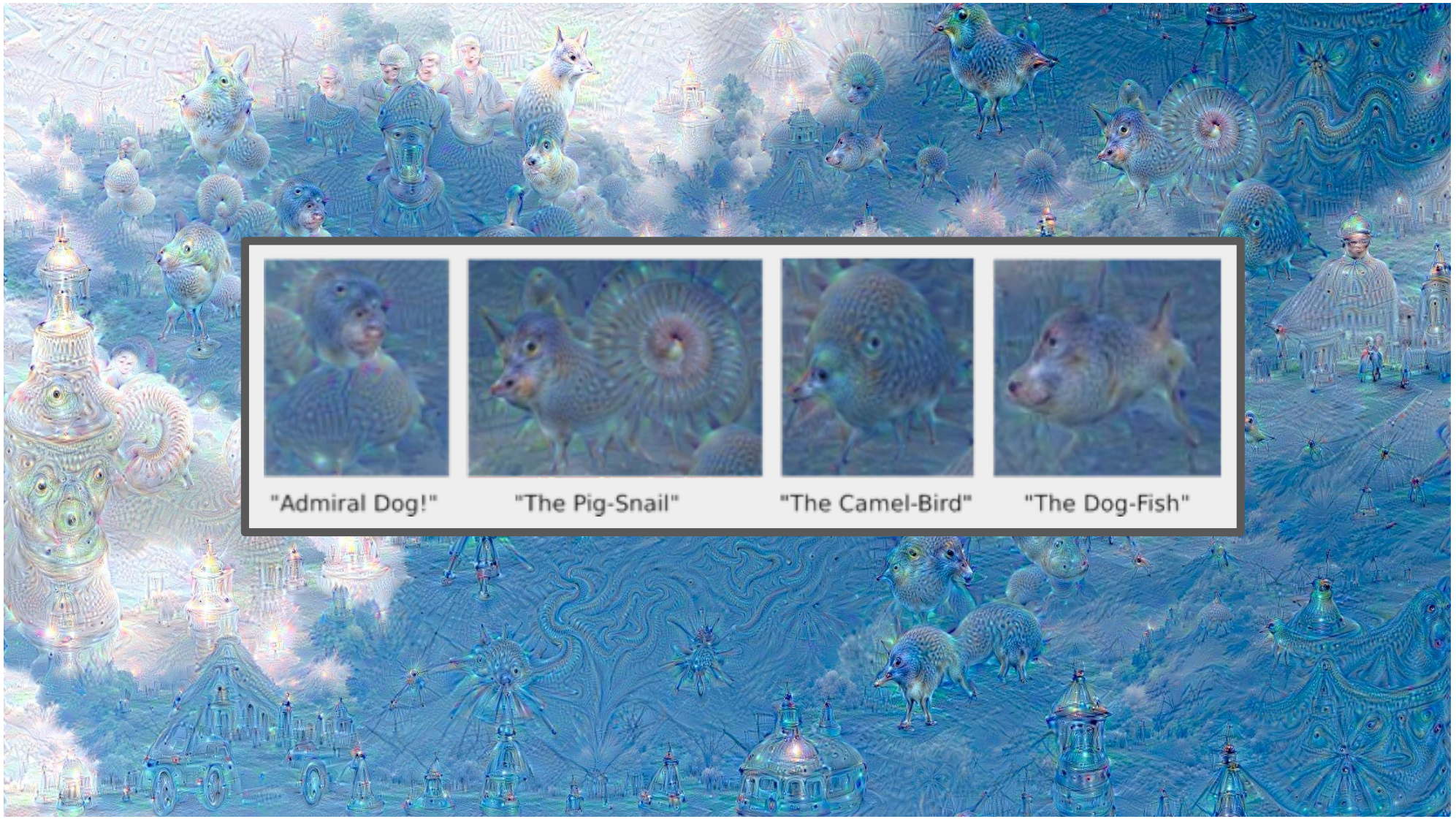


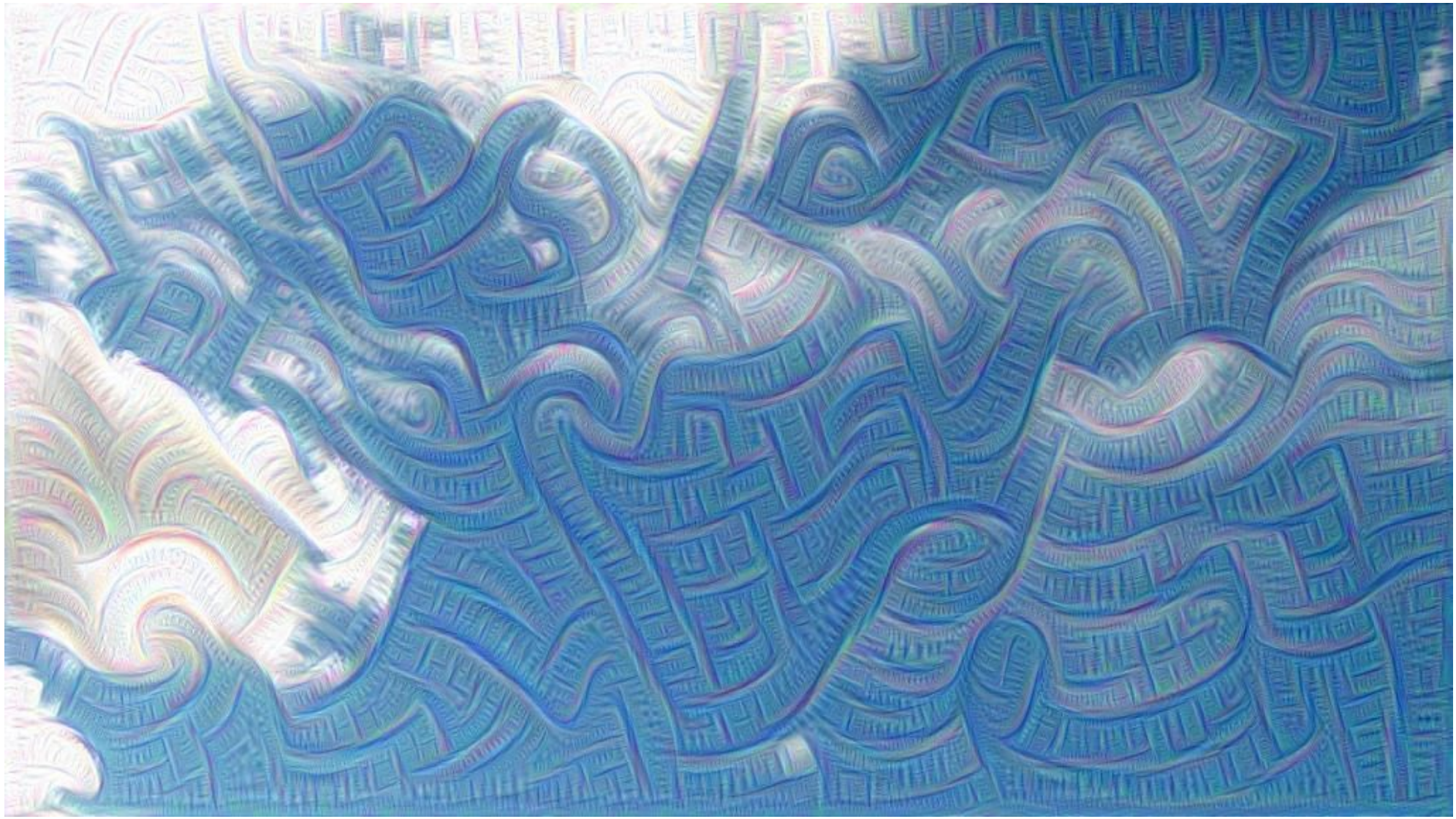
Choose an image and a layer in a CNN; repeat:

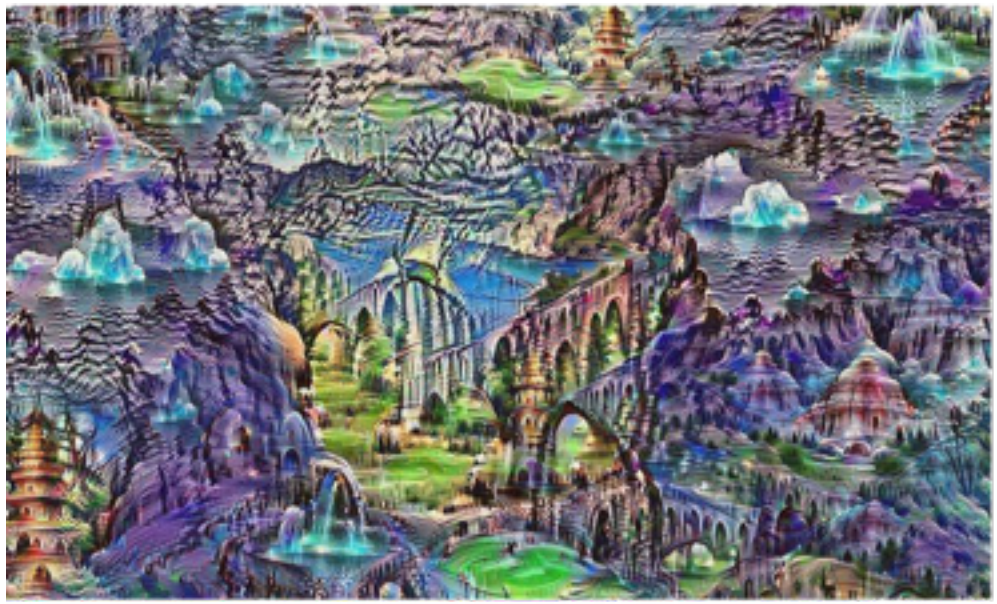
1. Forward: compute activations at chosen layer
2. Set gradient of chosen layer *equal to its activation*
3. Backward: Compute gradient on image
4. Update image

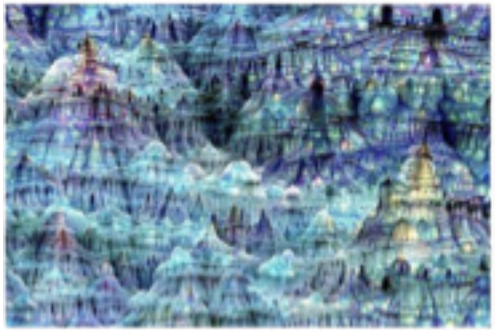
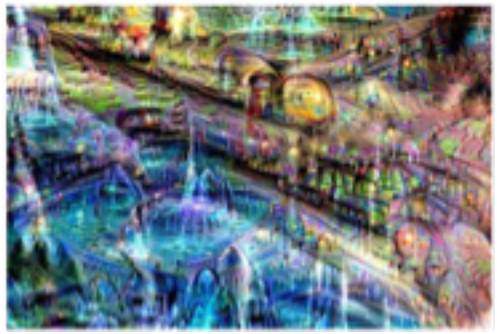
Equivalent to:

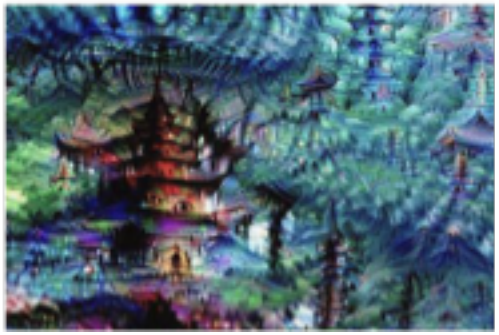
$$I^* = \arg\max_I \sum_i f_i(I)^2$$

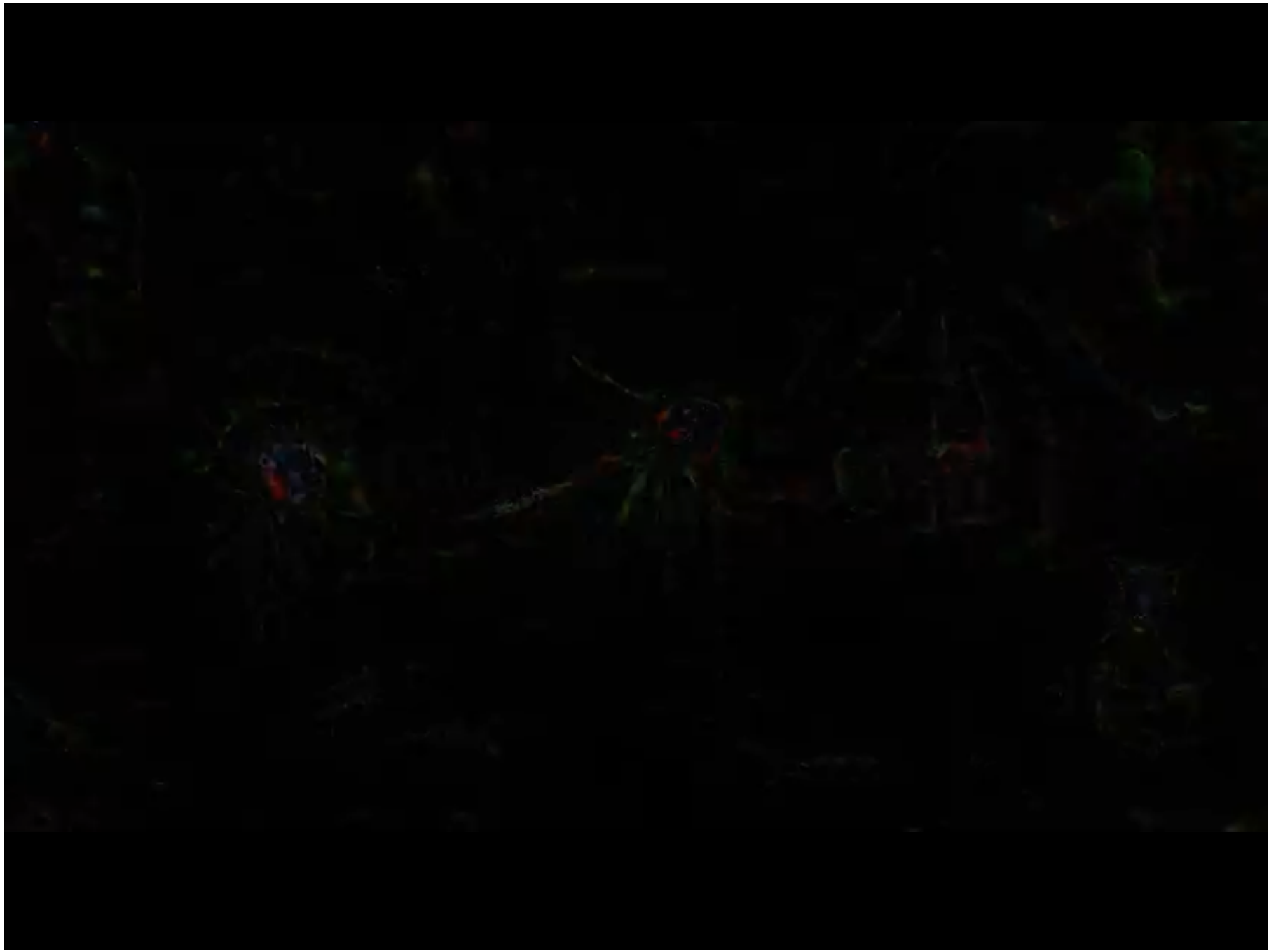

"Admiral Dog!"
"The Pig-Snail"
"The Camel-Bird"
"The Dog-Fish"









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

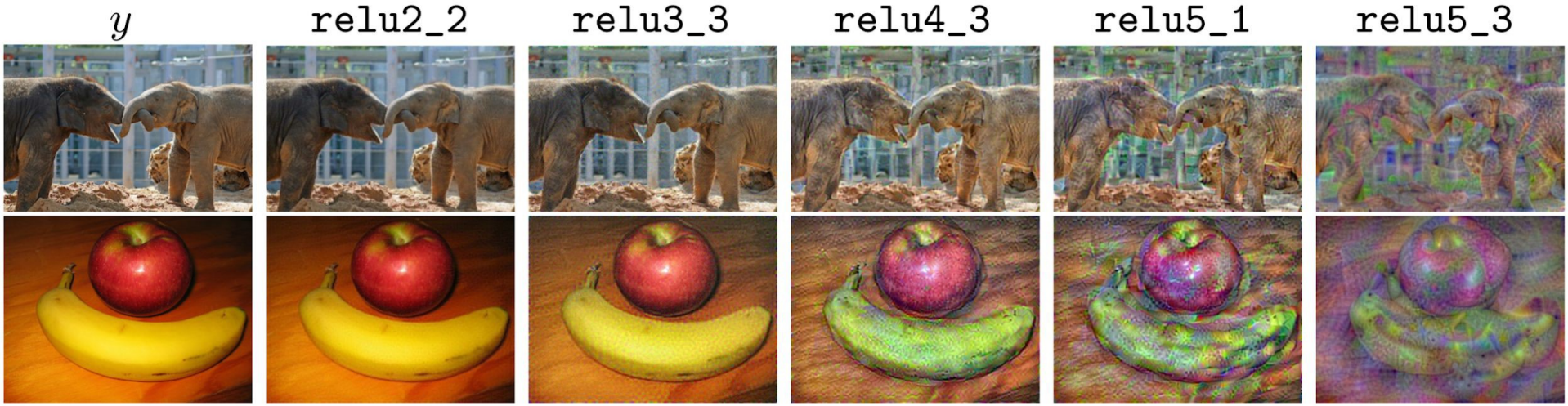
$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

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

Total Variation regularizer
(encourages spatial smoothness)

Feature Inversion

Reconstructing from different layers of VGG-16



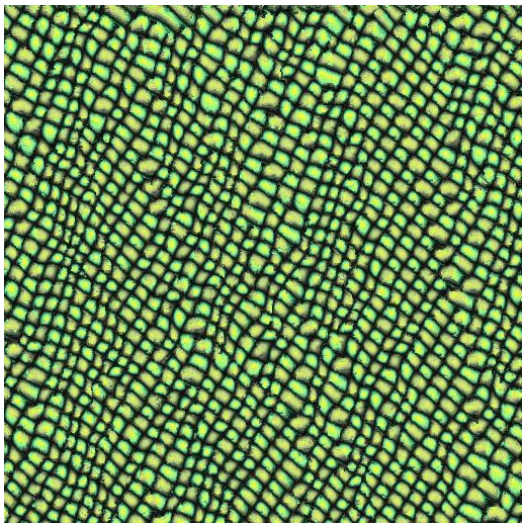
Mahendran and Vedaldi, "Understanding Deep Image Representations by Inverting Them", CVPR 2015

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.

Texture Synthesis

Given a sample patch of some texture, can we generate a bigger image of the same texture?

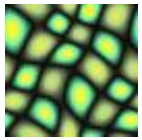
Input

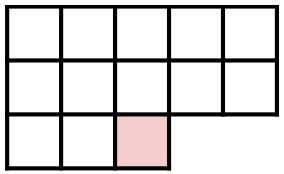


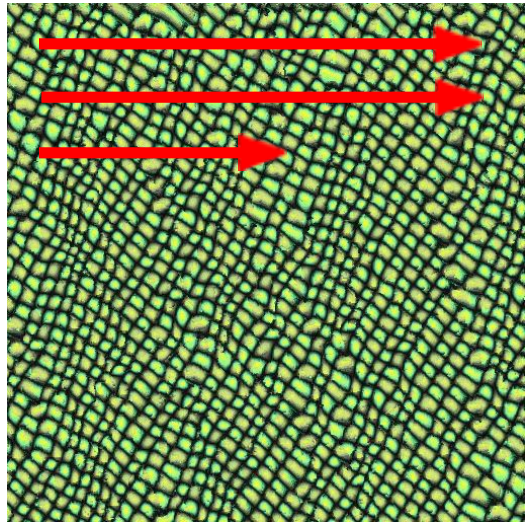
Output

Texture Synthesis: Nearest Neighbor

Generate pixels one at a time in scanline order; form neighborhood of already generated pixels and copy nearest neighbor from input



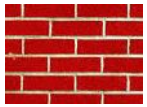


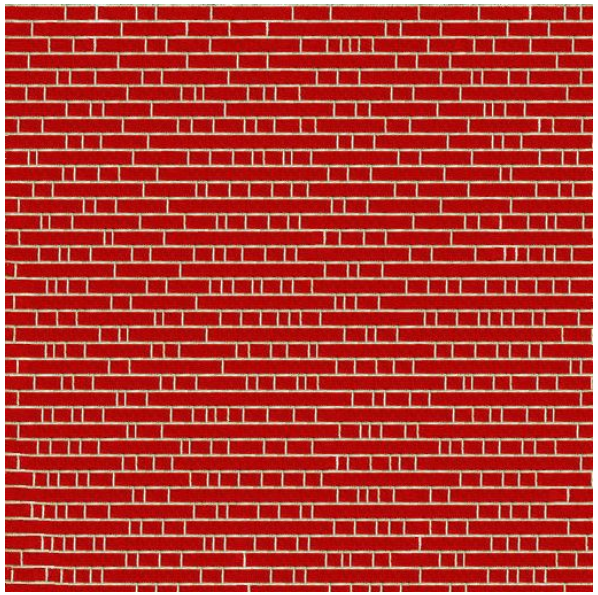


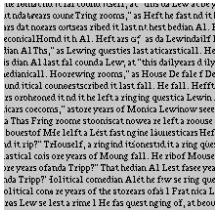
Wei and Levoy, “Fast Texture Synthesis using Tree-structured Vector Quantization”, SIGGRAPH 2000

Efros and Leung, “Texture Synthesis by Non-parametric Sampling”, ICCV 1999

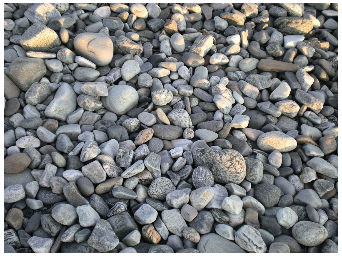
Texture Synthesis: Nearest Neighbor



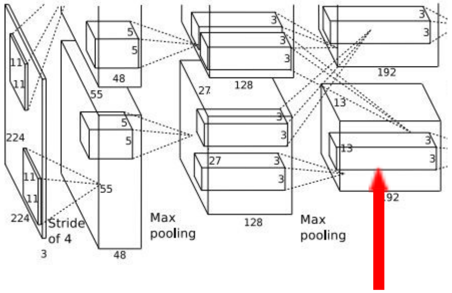




Neural Texture Synthesis: Gram Matrix



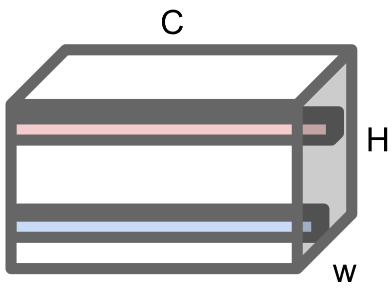
[redacted] is in the public domain.

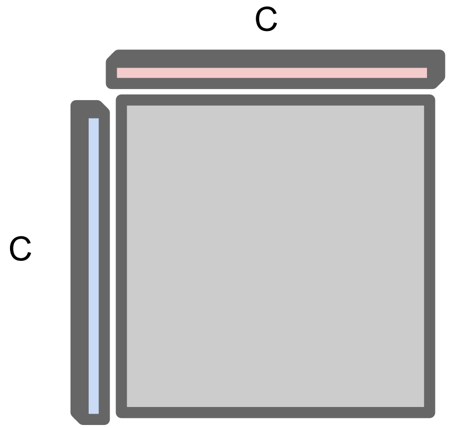


Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

Outer product of two C-dimensional vectors gives C x C matrix measuring co-occurrence

Average over all HW pairs of vectors, giving **Gram matrix** of shape C x C





Gram Matrix

Efficient to compute; reshape features from

C x H x W to =C x HW

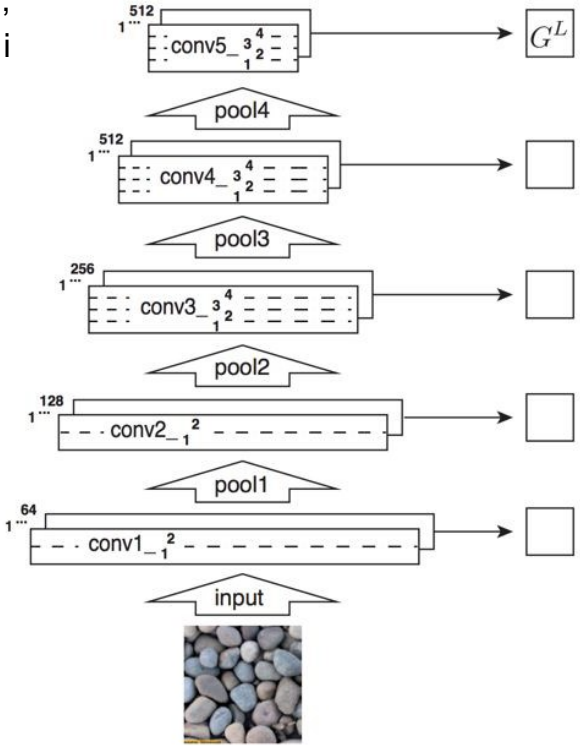
then compute $G = FF^T$

Neural Texture Synthesis

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - \hat{G}_{ij}^l \right)^2 \qquad \mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$$

1. Pretrain a CNN on ImageNet (VGG-19)
2. Run input texture forward through CNN, record activations on every layer; layer i gives feature map of shape $C_i \times H_i \times W_i$
3. At each layer compute the *Gram matrix* giving outer product of features:

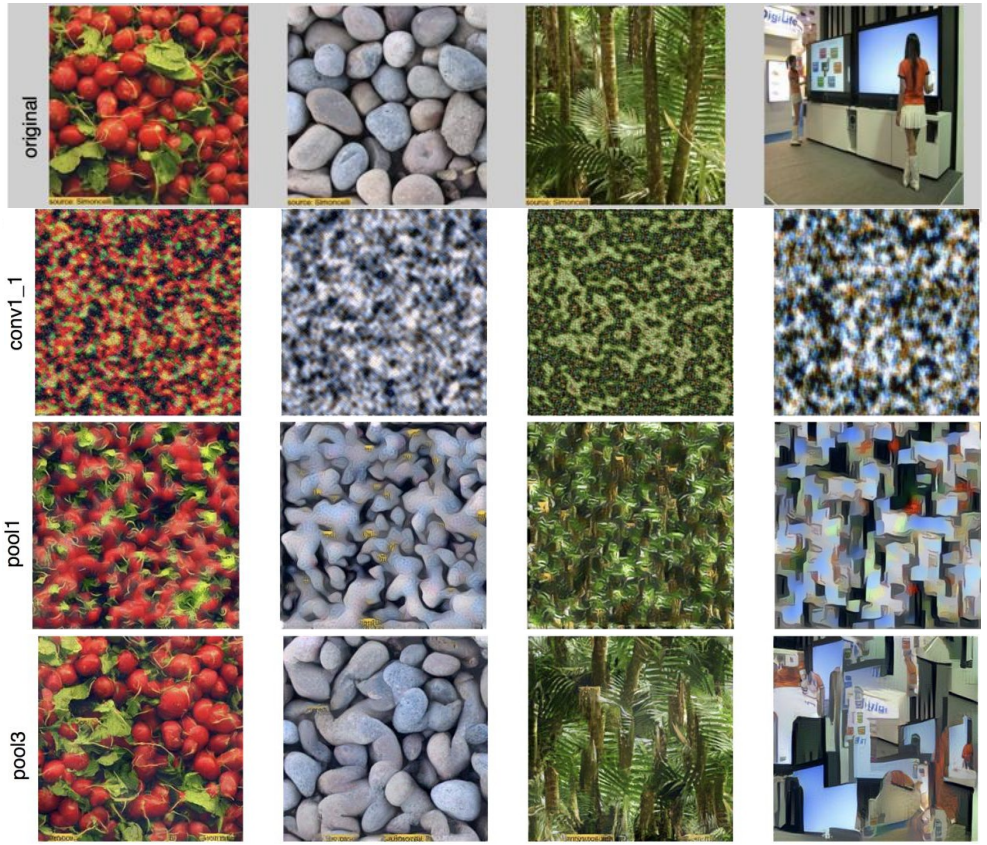
$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \text{ (shape } C_i \times C_i)$$



Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015

Neural Texture Synthesis

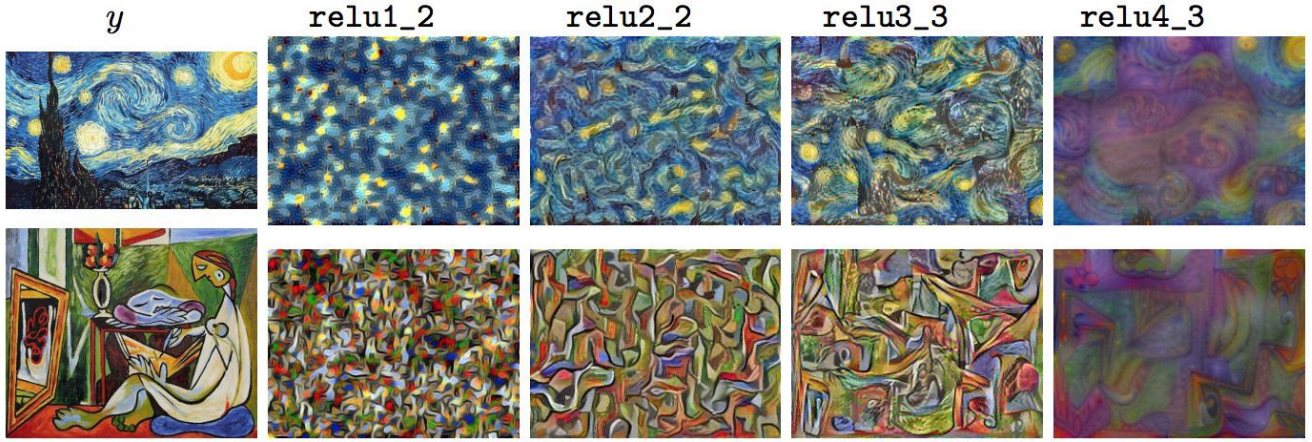
Reconstructing texture from higher layers recovers larger features from the input texture



Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015

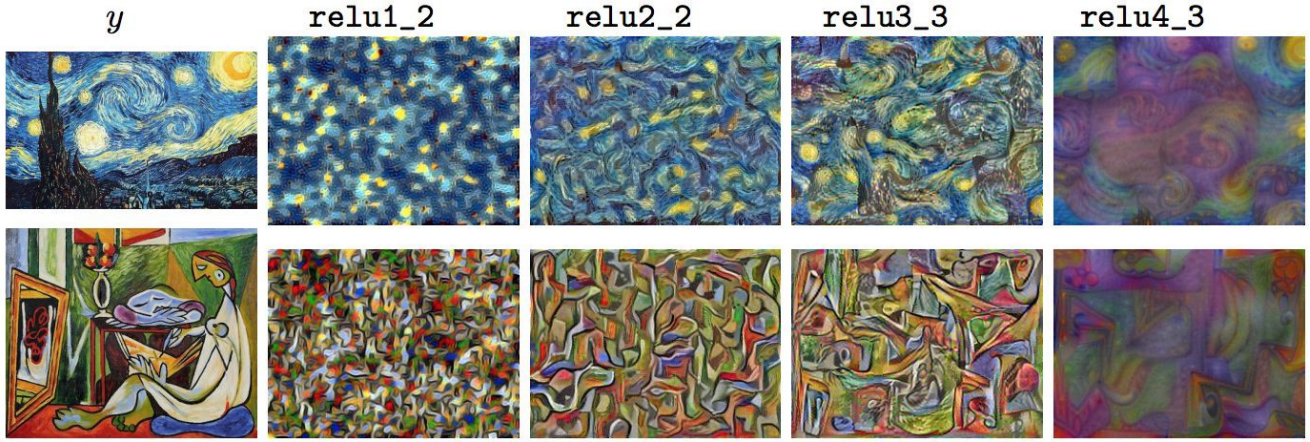
Neural Texture Synthesis: Texture = Artwork

Texture synthesis (Gram reconstruction)

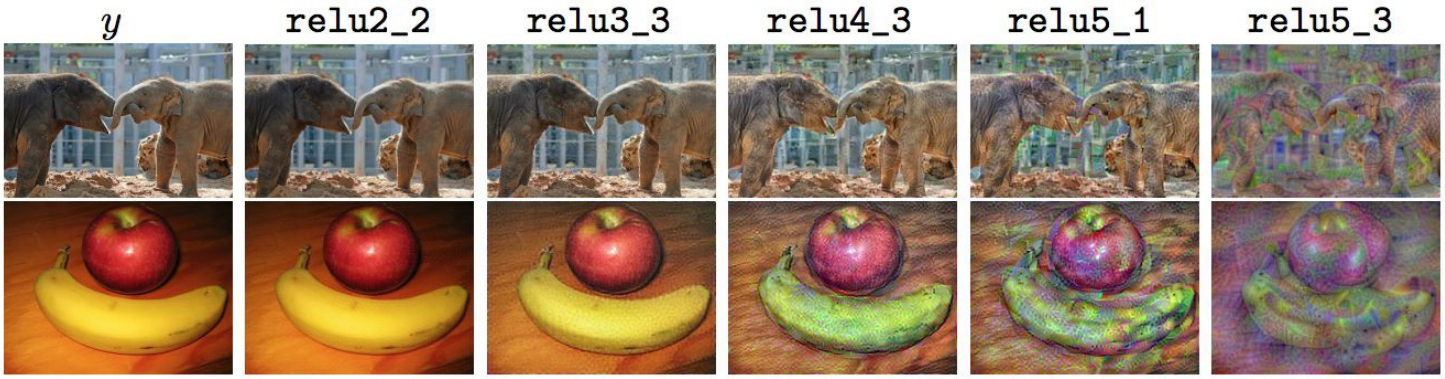


Neural Style Transfer: Feature + Gram Reconstruction

Texture synthesis (Gram reconstruction)



Feature reconstruction



Neural Style Transfer

Content Image

[redacted] is licensed under CC-BY 3.0

+

Style Image

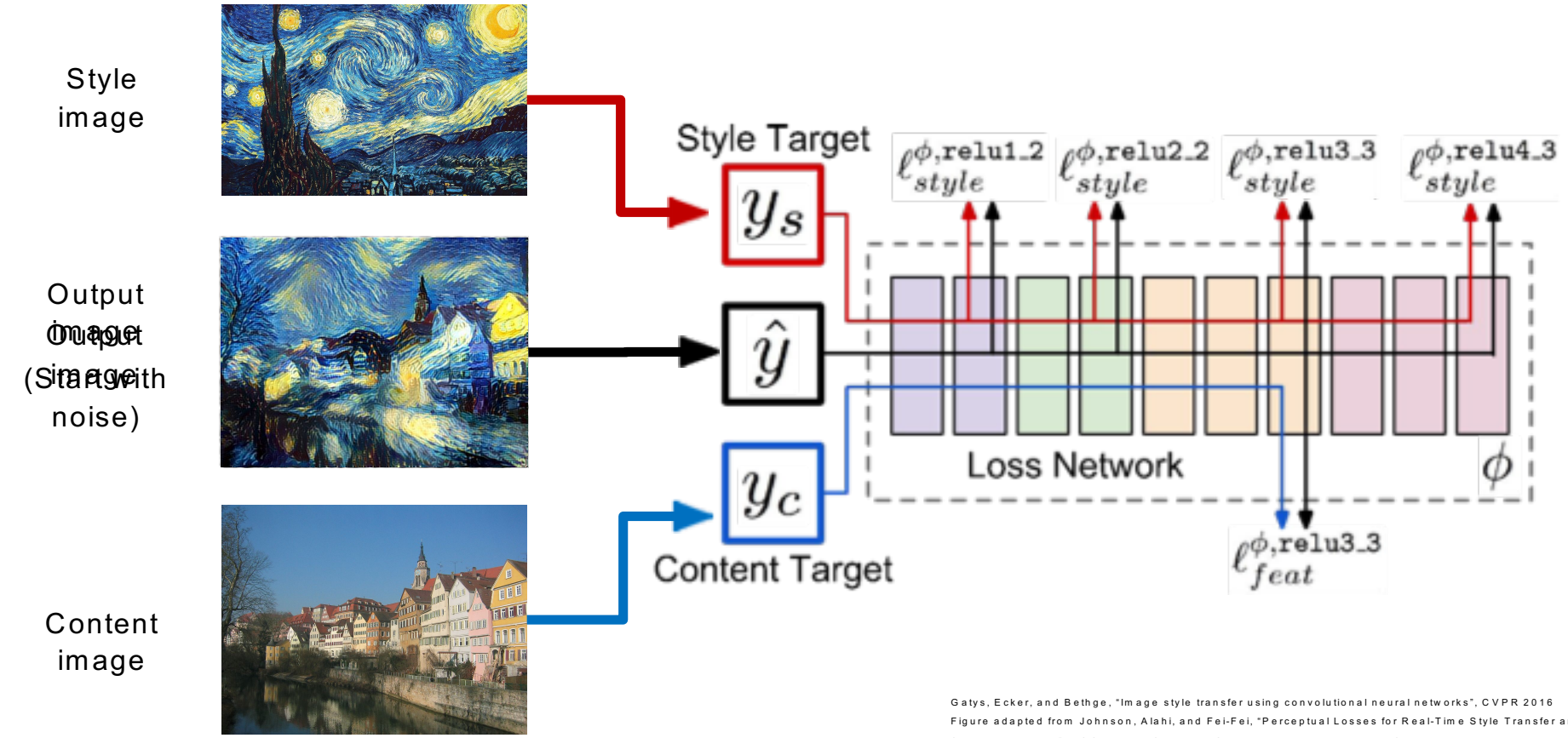
[redacted]yright by Van Gogh is in the public domain

=

Style Transfer!

[redacted] copyright Justin Johnson, 2015. Reproduced with permission.

Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016
Figure adapted 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.

Neural Style Transfer

Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016