Lecture 11: Visualizing and Understanding

Last time: Lots of Computer Vision Tasks

Classification



CAT

No spatial extent

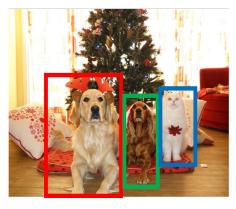
Semantic Segmentation



TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation

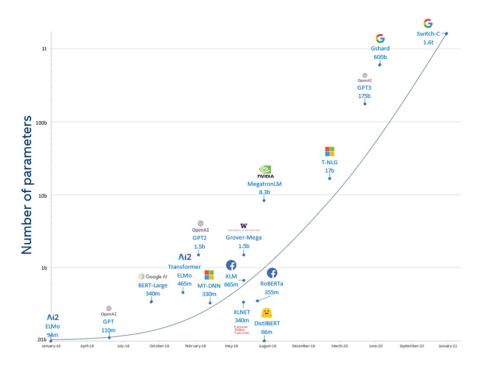


DOG, DOG, CAT

Multiple Object

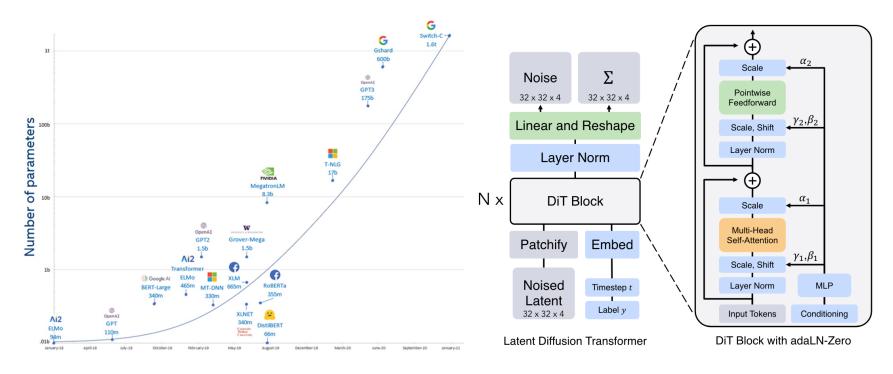
This image is CC0 public domain

Visualizing and Understanding



Source: Zoiner Tejada

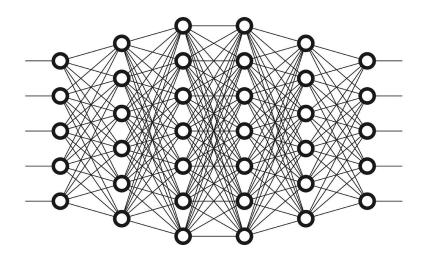
Visualizing and Understanding



Source: Zoiner Tejada

DiT, Peebles & Xie, 2023

Visualizing and Understanding: Challenges



- Lots of parameters
- Complex transformations
- Non-intuitive latent feature spaces

- ...

Artificial neural networks

Computer Vision is everywhere!















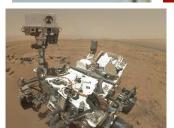




Image is free to use Image is CCO 1.0 public domain Image by NASA is license

mage by NASA is license under <u>CC BY 2.0</u> Image is <u>CCO 1.0</u> public







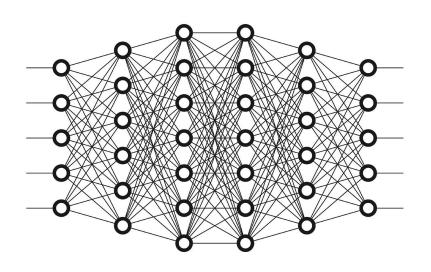


Bottom row, left to right Image is CC0 1.0 public domain

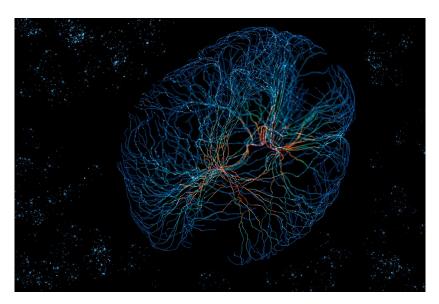
Image by Derek Keats is licensed under CC BY 2.1 changes made

Image is public domain
Image is licensed under <u>CC-RY</u>

Visualizing and Understanding: Challenges



Artificial neural networks



Human brain: ~100B neurons (?)

Image source: MIT News

Attention as a tool for understanding

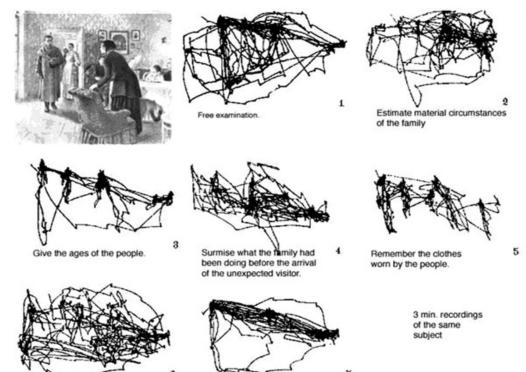






Yarbus, Eye Movements and Vision, 1967

Attention as a tool for understanding



Estimate how long the visitor had

been away from the family.

Yarbus, Eye Movements and Vision, 1967

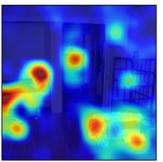
Remember positions of people and

objects in the room.

Attention of ANNs



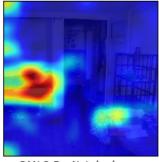




SAN-2-AlexNet: living room



SAN-2-VGG: living room



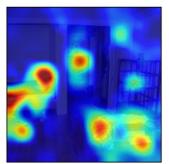
SAN-2-ResNet: bedroom

Das et al., Computer Vision and Image Understanding, 2017

Attention of ANNs



What room is this?



SAN-2-AlexNet: living room



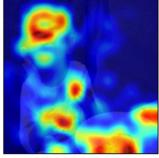
SAN-2-VGG: living room



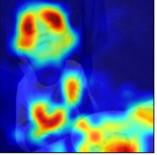
SAN-2-ResNet: bedroom



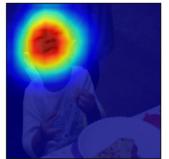
Is this child a boy or girl?



SAN-1 prediction: boy



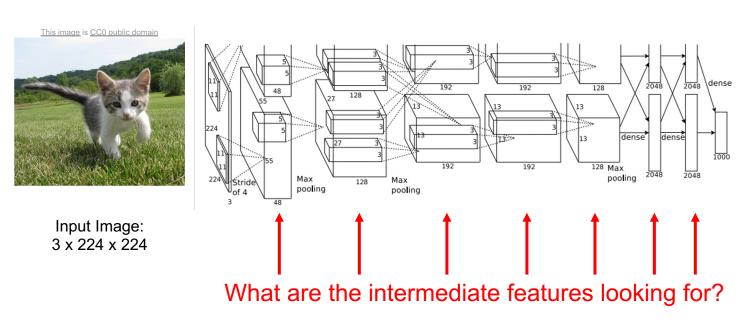
SAN-1-Multitask prediction: boy



Human attention

Das et al., Computer Vision and Image Understanding, 2017

Today: What's going on inside ConvNets?



Class Scores: 1000 numbers

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

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 Style transfer

- Style transfer
 - Features inversion
 - Deep dream
- Texture synthesis
- Neural style transfer

Today's agenda

Visualizing what models have learned:

- Visualizing filters
- Visualizing final layer features
- Visualizing activations

Understanding input pixels

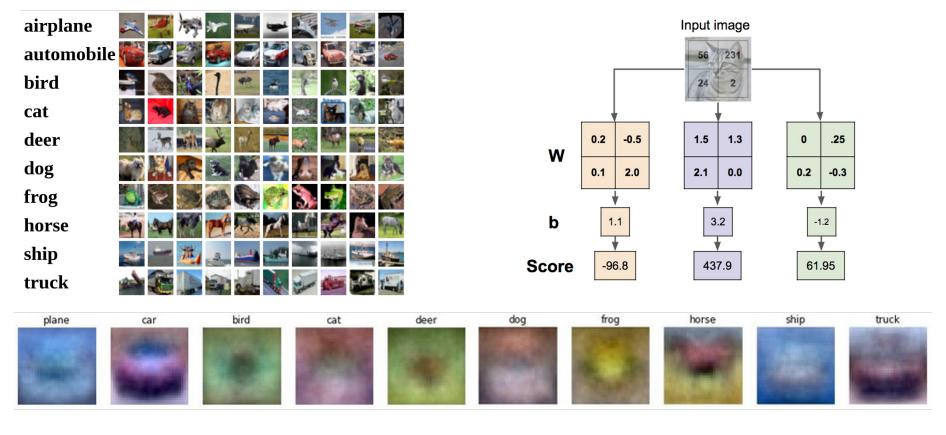
- Identifying important pixels
- Saliency via backprop
- Guided backprop to generate images
- Gradient ascent to visualize features

Adversarial perturbations Style transfer

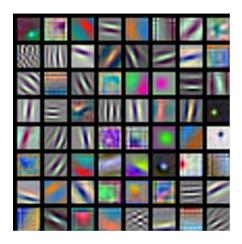
- Features inversion
- Deep dream
- Texture synthesis
- Neural style transfer

Today's agenda

Interpreting a Linear Classifier: Visual Viewpoint

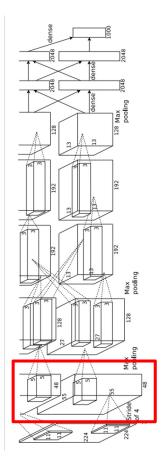


First Layer: Visualize Filters

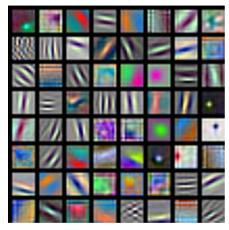


AlexNet: 64 x 3 x 11 x 11

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



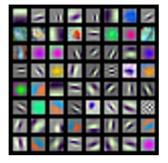
AlexNet: 64 x 3 x 11 x 11



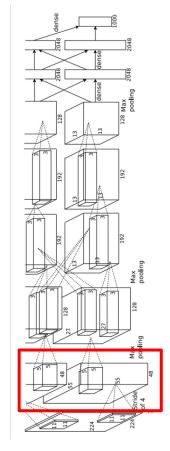
ResNet-18: 64 x 3 x 7 x 7



ResNet-101: 64 x 3 x 7 x 7



DenseNet-121: 64 x 3 x 7 x 7



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

Visualize the filters/kernels (raw weights)

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

(these are taken from ConvNetJS CIFAR-10 demo)

Weights:

Weights:

(我是是这是是是我们是这是是我们的,我们就是我们的是我们的,我们是我们的人们的, 可表现使用使用的证)(医型自治疗与关系性性治疗自治疗法)(建筑自治疗精髓病病性性治疗病 國間)(在由法門基礎發展的的學術學所與多方的)(物數四數與問題與問題與問題解語傳達的)(無知數學數 而學科的用於法學學學家)(法學科學所學科科學科科科學學科)(開始來數學與解析的語彙 新新四层海南海南西南部市市市(Carle Barrier Barrier

layer 1 weights

16 x 3 x 7 x 7

layer 2 weights 20 x 16 x 7 x 7

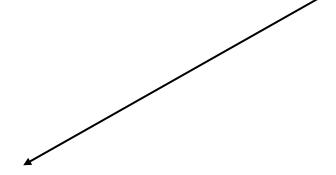
Weights:

(这里与埃萨克斯特别的 医克里氏 医多种 医多种 (1995年)(1996年 医多种 医多种 (1995年 1996年 的)(医医感觉性医院性原列医院医院医院院秘密表)(的医眼球的医多胞化剂酶药物根皮肤吸吸 至編)(原列東西海洋海通市場所有四元四元四元第二十八元三元四元年第二元四元年第二十八元年 | 连秦章)(語名問題名為華德哲學表式物物的發展的思想)(始於公司建學是經過表現是以同形語句 计可数型法)(化过程设置通路保持的电路设置设置的设置)(原则设计的电路设置的现代中国 母與古聖福斯斯斯斯(

layer 3 weights

20 x 20 x 7 x 7

Last Layer



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

Run the network on many images, collect the feature vectors

FC7 layer

Last Layer: Nearest Neighbors

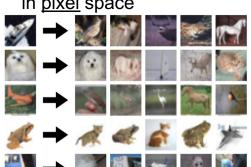
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012. Figures reproduced with permission.

Last Layer: Nearest Neighbors

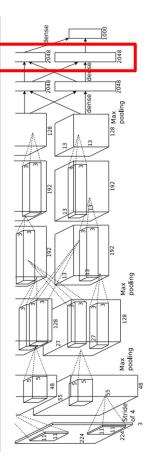
4096-dim vector

Test image L2 Nearest neighbors in <u>feature</u> space

Recall: Nearest neighbors in <u>pixel</u> space





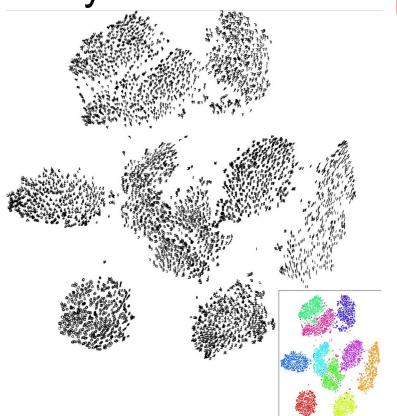


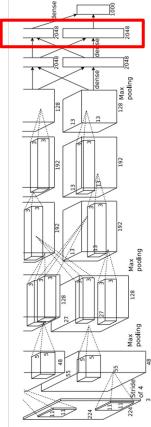
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS 2012. Figures reproduced with permission. Last Layer: Dimensionality Reduction

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 Figure copyright Laurens van der Maaten and Geoff Hinton, 2008. Reproduced with permission

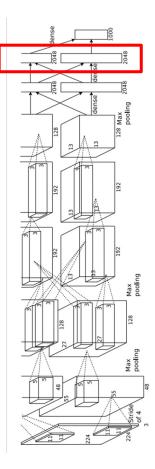
Last Layer: Dimensionality Reduction



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.

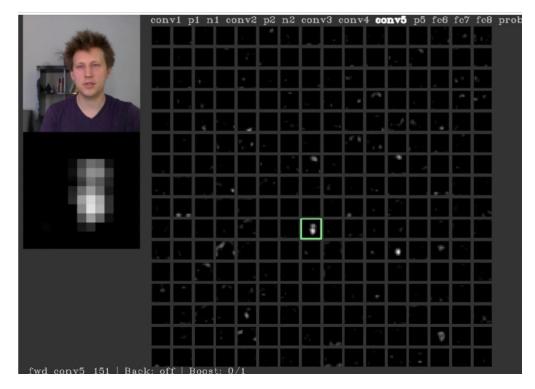


See high-resolution versions at http://cs.stanford.edu/people/karpathy/cnnembed/



Visualizing Activations

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.

Visualizing Activations



David Bau*, Bolei Zhou*, Aditya Khosla, Aude Oliva, Antonio Torralba Network Dissection: Quantifying Interpretability of Deep Visual Representations. CVPR 2017.

Lecture 11 - 25

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

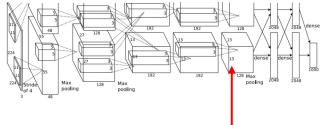
Adversarial perturbations Style transfer

- Deep dream
- Features inversion
- Texture synthesis
- Neural style transfer

Today's agenda

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



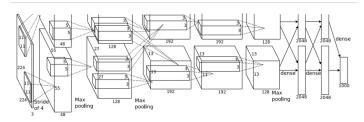


Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015 Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

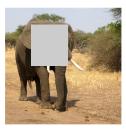
Which pixels matter: Saliency via Occlusion

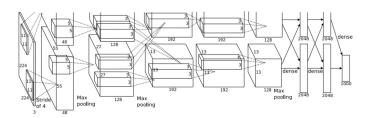
Mask part of the image before feeding to CNN, check how much predicted probabilities change





P(elephant) = 0.95





P(elephant) = 0.75

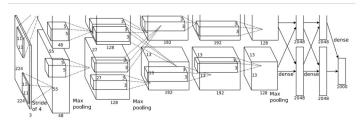
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

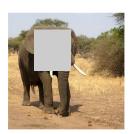
Boat image is CC0 public domain Elephant image is CC0 public domai Go-Karts image is CC0 public domai

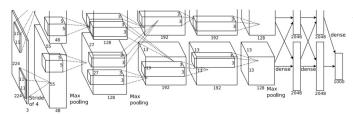
Which pixels matter: Saliency via Occlusion

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





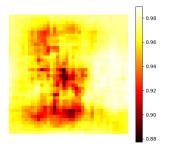




Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

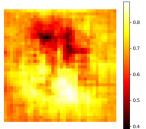
Boat image is CC0 public domain Elephant image is CC0 public domain Go-Karts image is CC0 public domain



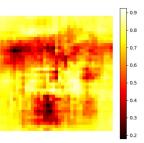


African elephant, Loxodonta africana



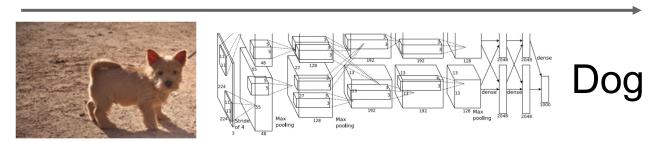






Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities

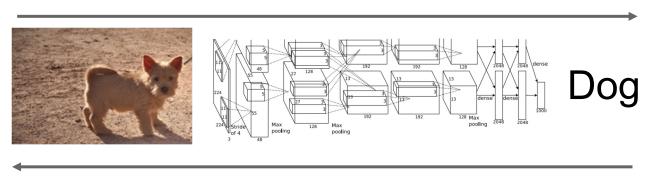


Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

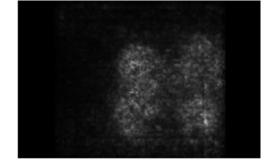
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

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.

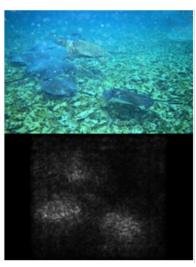
Saliency Maps







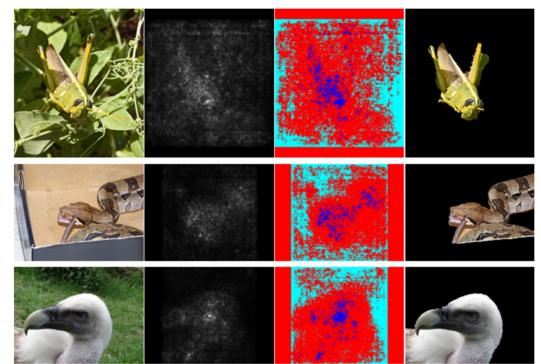




Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

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 nosnow classifier



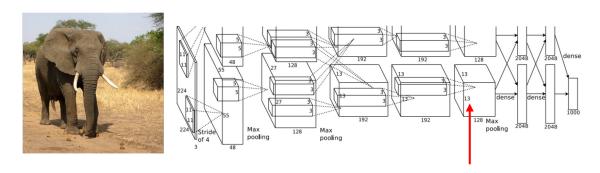
(a) Husky classified as wolf



(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

Intermediate Features via (guided) backprop

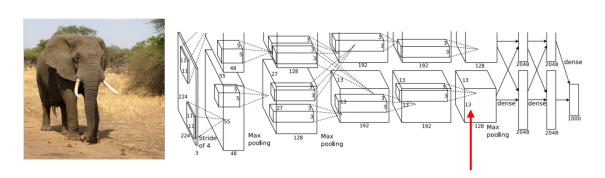


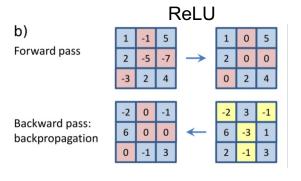
Pick a single intermediate channel, e.g. one value in 128 x 13 x 13 conv5 feature map

Compute gradient of activation value with respect to image pixels

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

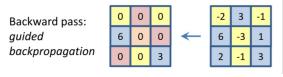
Intermediate Features via (guided) backprop





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)

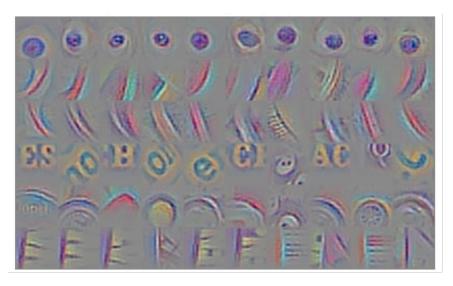
Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission.

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
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.

(Guided) backprop:

Find the part of an image that a neuron responds to

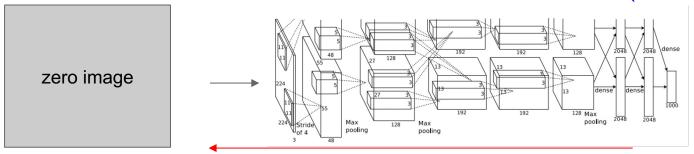
Gradient ascent:

Generate a synthetic image that maximally activates a neuron

1. Initialize image to zeros

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

score for class c (before Softmax)



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

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

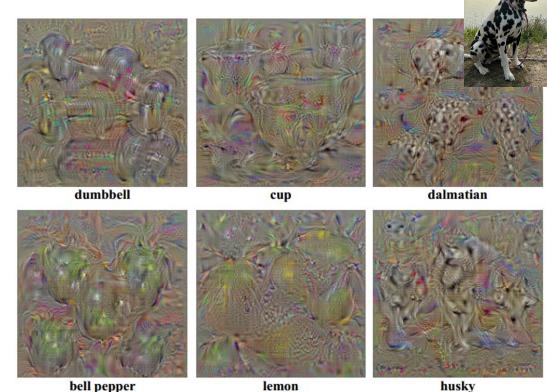
Simple regularizer: Penalize L2 norm of generated image

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.

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

Simple regularizer: Penalize L2 norm of generated image

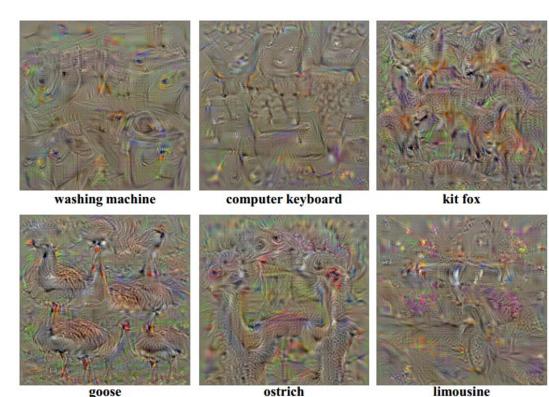


Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014.

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

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

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.

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

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

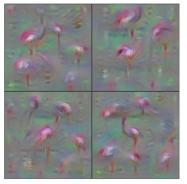
- (1) Gaussian blur image
- (2) Clip pixels with small values to 0
- (3) Clip pixels with small gradients to 0

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

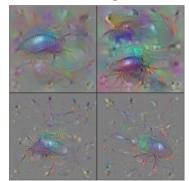
$$\arg\max_{I} S_c(I) - \lambda ||I||_2^2$$

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

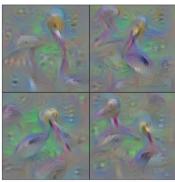
- (1) Gaussian blur image
- (2) Clip pixels with small values to 0
- (3) Clip pixels with small gradients to 0



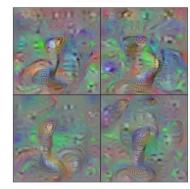
Flamingo



Ground Beetle



Pelican



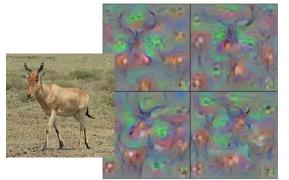
Indian Cobra

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.

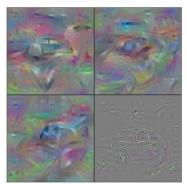
$$\arg\max_{I} S_c(I) - \lambda ||I||_2^2$$

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

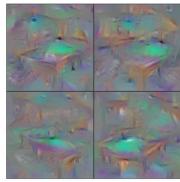
- (1) Gaussian blur image
- (2) Clip pixels with small values to 0
- (3) Clip pixels with small gradients to 0



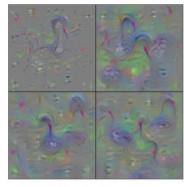
Hartebeest



Station Wagon



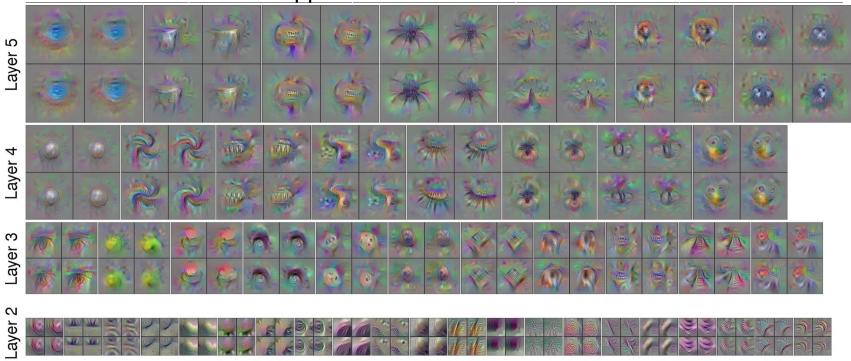
Billiard Table



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.

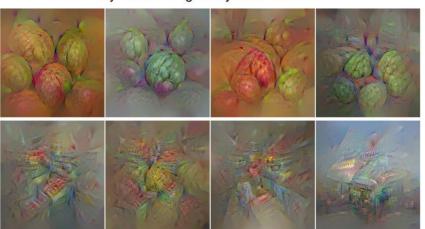
Use the same approach to visualize intermediate features



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.

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

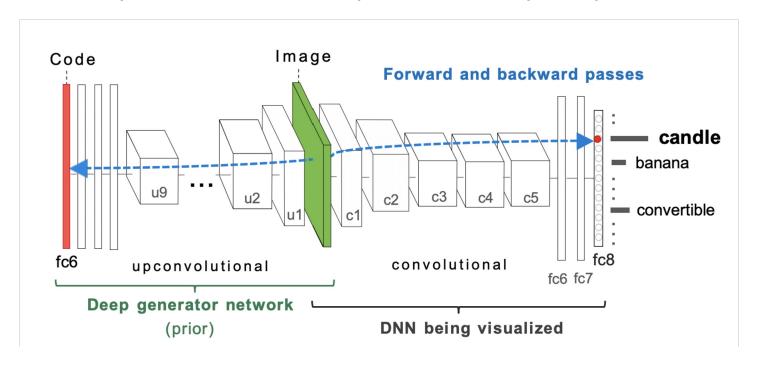


Nguyen et al, "Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks", ICML Visualization for Deep Learning Workshop 2016. Figures copyright Anh Nguyen, Jason Yosinski, and Jeff Clune, 2016; reproduced with permission.



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.

Optimize in FC6 latent space instead of pixel space:



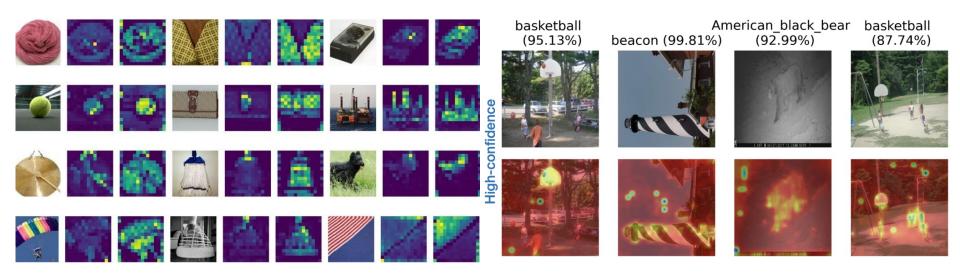
Nguyen et al, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," NIPS 2016 Figure copyright Nguyen et al, 2016; reproduced with permission.

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 ViT features



Chen et al., When Vision Transformers Outperform Resnets Without Pre-training Or Strong Data Augmentations, ICLR 2022; Paul and Chen, Vision Transformers are Robust Learners, AAAI 2022. Reproduced for educational purposes.

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

Style transfer

- Deep dream
- Features inversion
- Texture synthesis
- Neural style transfer

Today's agenda

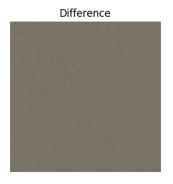
Fooling Images / Adversarial Examples

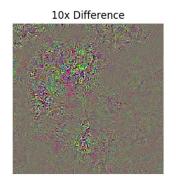
- (1) Start from an arbitrary image
- (2) Pick an arbitrary class
- (3) Modify the image to maximize the class
- (4) Repeat until network is fooled

Fooling Images / Adversarial Examples



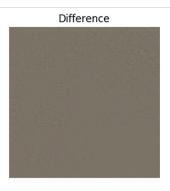


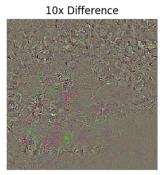












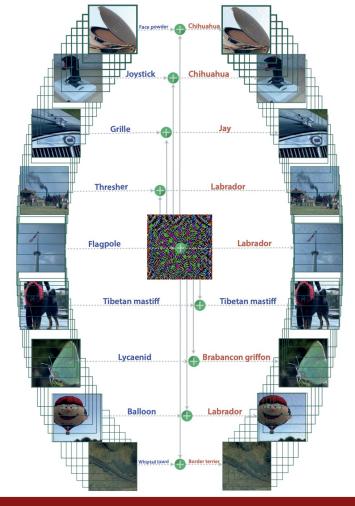
<u>Boat image</u> is <u>CC0 public domain</u> <u>Elephant image</u> is <u>CC0 public domai</u>ı

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



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

Style transfer

- Features inversion
- Deep dream
- Texture synthesis
- Neural style transfer

Today's agenda

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)

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

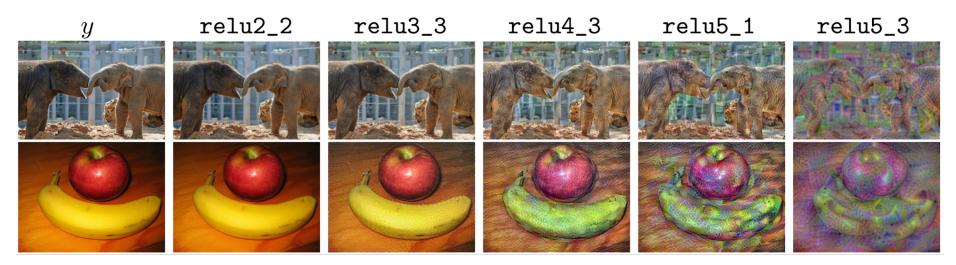
$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\frac{\beta}{2}} \xrightarrow{\text{Total Variation regularizer}}$$
(encourages spatial smoothness)

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

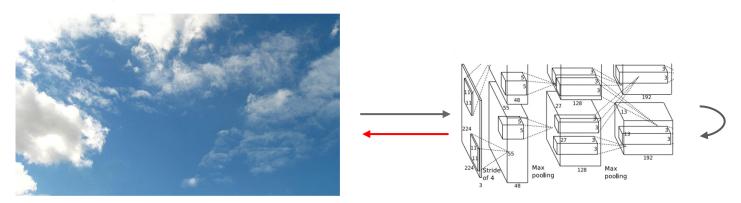
Feature Inversion

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.

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

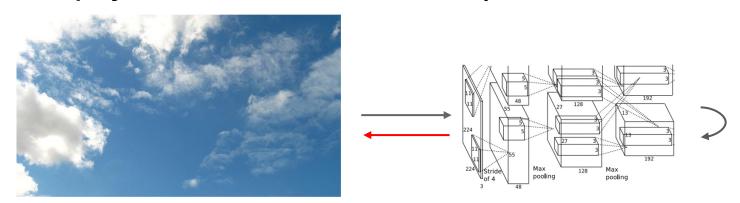


Choose an image and a layer in a CNN; repeat:

- 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

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:

 $_{l}$ I* = arg max_l $\sum_{i} f_{i}(I)^{2}$

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

```
def objective L2(dst):
                                                                                                    Code is very simple but
   dst.diff[:] = dst.data
                                                                                                    it uses a couple tricks:
def make step(net, step size=1.5, end='inception 4c/output',
             jitter=32, clip=True, objective=objective L2):
                                                                                                    (Code is licensed under Apache 2.0)
    '''Basic gradient ascent step.'''
   src = net.blobs['data'] # input image is stored in Net's 'data' blob
   dst = net.blobs[end]
   ox, ov = np.random.randint(-jitter, jitter+1, 2)
                                                                                                    Jitter image
   src.data[0] = np.roll(np.roll(src.data[0], ox, -1), oy, -2) # apply jitter shift
   net.forward(end=end)
   objective(dst) # specify the optimization objective
   net.backward(start=end)
   q = src.diff[0]
   # apply normalized ascent step to the input image
   src.data[:] += step size/np.abs(g).mean() * g
   src.data[0] = np.roll(np.roll(src.data[0], -ox, -1), -oy, -2) # unshift image
   if clip:
       bias = net.transformer.mean['data']
       src.data[:] = np.clip(src.data, -bias, 255-bias)
```

```
def objective L2(dst):
                                                                                                   Code is very simple but
   dst.diff[:] = dst.data
                                                                                                   it uses a couple tricks:
def make step(net, step size=1.5, end='inception 4c/output',
             jitter=32, clip=True, objective=objective L2):
                                                                                                   (Code is licensed under Apache 2.0)
    '''Basic gradient ascent step.'''
   src = net.blobs['data'] # input image is stored in Net's 'data' blob
   dst = net.blobs[end]
   ox, ov = np.random.randint(-jitter, jitter+1, 2)
                                                                                                  Jitter image
   src.data[0] = np.roll(np.roll(src.data[0], ox, -1), oy, -2) # apply jitter shift
   net.forward(end=end)
   objective(dst) # specify the optimization objective
   net.backward(start=end)
   g = src.diff[0]
   # apply normalized ascent step to the input image
                                                                                                 L1 Normalize gradients
   src.data[:] += step size/np.abs(g).mean() * g
   src.data[0] = np.roll(np.roll(src.data[0], -ox, -1), -oy, -2) # unshift image
   if clip:
       bias = net.transformer.mean['data']
       src.data[:] = np.clip(src.data, -bias, 255-bias)
```

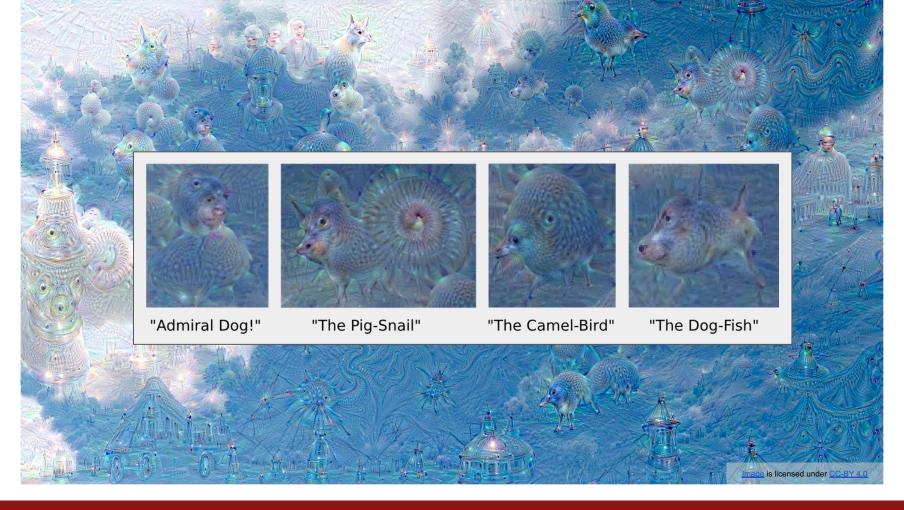
```
def objective L2(dst):
                                                                                                  Code is very simple but
   dst.diff[:] = dst.data
                                                                                                  it uses a couple tricks:
def make step(net, step size=1.5, end='inception 4c/output',
             jitter=32, clip=True, objective=objective L2):
                                                                                                  (Code is licensed under Apache 2.0)
    '''Basic gradient ascent step.'''
   src = net.blobs['data'] # input image is stored in Net's 'data' blob
   dst = net.blobs[end]
   ox, ov = np.random.randint(-jitter, jitter+1, 2)
                                                                                                 Jitter image
   src.data[0] = np.roll(np.roll(src.data[0], ox, -1), oy, -2) # apply jitter shift
   net.forward(end=end)
   objective(dst) # specify the optimization objective
   net.backward(start=end)
   g = src.diff[0]
   # apply normalized ascent step to the input image
                                                                                                 L1 Normalize gradients
   src.data[:] += step size/np.abs(g).mean() * g
   src.data[0] = np.roll(np.roll(src.data[0], -ox, -1), -oy, -2) # unshift image
       bias = net.transformer.mean['data']
                                                                                                Clip pixel values
       src.data[:] = np.clip(src.data, -bias, 255-bias)
```

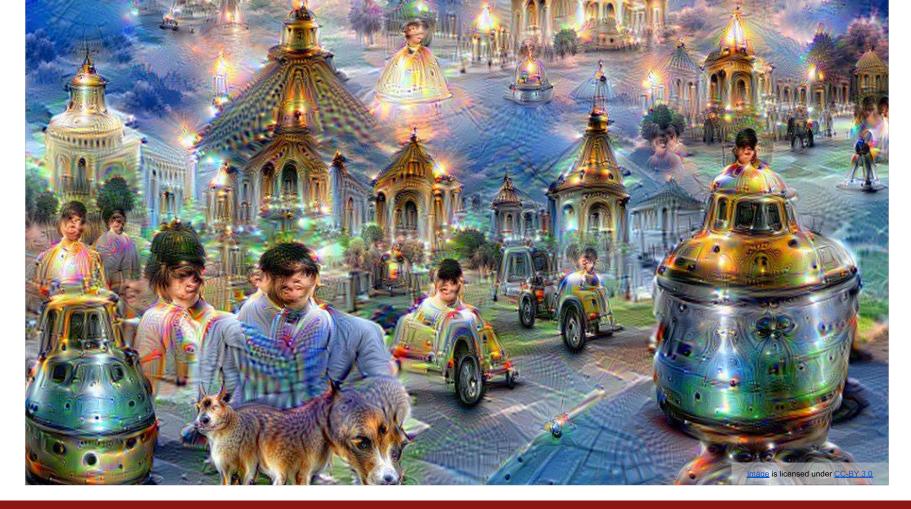
Also uses multiscale processing for a fractal effect (not shown)















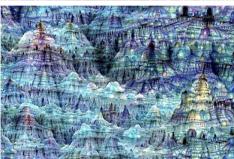




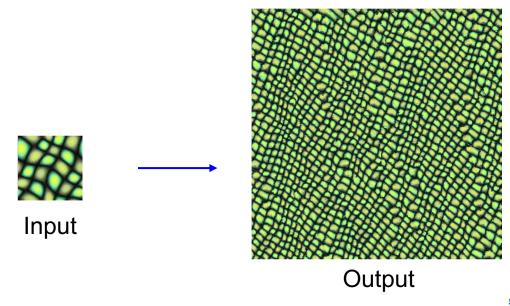




Image is licensed under CC-BY 4.0

Texture Synthesis

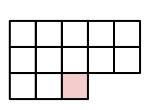
Given a sample patch of some texture, can we generate a bigger image of the same texture?



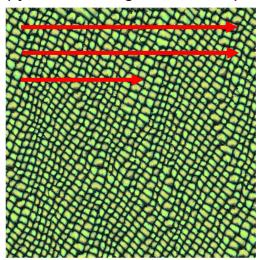
Output image is licensed under the MIT license

Texture Synthesis: Nearest Neighbor



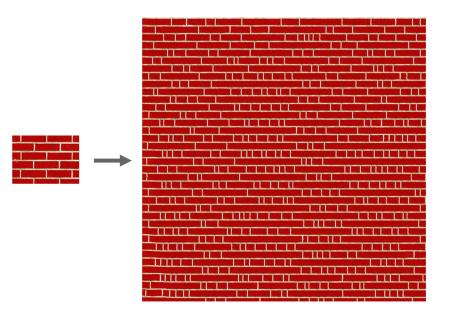


Generate pixels one at a time in scanline order; form neighborhood of already generated pixels and copy nearest neighbor from input



Wei and Levoy, "Fast Texture Synthesis using Tree-structured Vector Quantization", SIGGRAPH 2000 Efros and Leung, "Texture Synthesis by Non-parametric Sampling", ICCV 1999

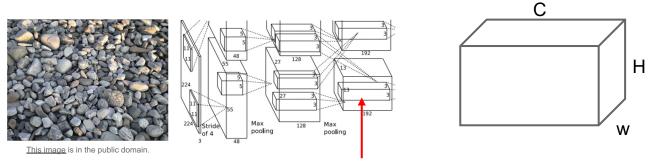
Texture Synthesis: Nearest Neighbor



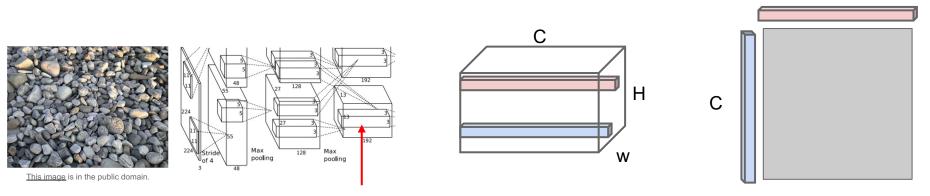


controllers and collection and a second collection and A to not tof, were known to the last uses do fash of the tot, A second in which go, Takes of einessee him well sed in Fof language of tab w 1 Lianustay tra attica respectively was ripo? Ero? class the remaining the repentago of 2 february bear and but Legyotte con a Honor gold and the control of the co Marylandron Lestr, accounting and a diamoes and is his reason Fria of require 1 90 asceptiatest Mobilet steeding diagram on a rest alista oo f. aw seeds Leng Hoene lao annas o Hal noeueo stades la fincie yee was salencinou daas, awonying iton as di a thrane lat noundmenst hit has morafasta for the wifer F. Estati at "a noundmensfaling notice to it in income set of the "Thomas tanget Hars will all and a Lenders thou said the state of the course five successive of the state of Chang da www.quanch.magy.sal and Horyen Lectrins I of the Cabayenties," hen Letter in Califold to attend to attend to us his time a contrast to 15 months and we one of Alal seingring a lalas solo openior, rock to a rithe four a sold a sold a sold the fatth Hesl latistic virus swithit to Girarhomo zoor flast tilla lararise rik deloit brevit inno Lis, excessi qiqodir virono qualitati Herric fashili and stic of conjections La lagragicars is several infections of certifications of the conference of the co e yln pare yeard be a rfat encit allyg omarseesweal Hot gars are v, rollt med mgf thought Hotousfest he mikistical et mod lo it avaleste qual decicerion oraș Deut feunur, Ae invocuur ripratiya, esc Le definisc Heitso sous că: HeAST, "ratibuse ozeatre proug thiste r'adarines, wa Economic aries: faas maams winas estats, et obrosow stampollar fyaare fairlaths Dough lasts yest aurrought révieu Hodin, resizest loom Thatin set, estrict's souch ing the secutorious secundades condition fraist dies capous Heading is nitth well's intelled it with the first as in a guedains rithed Fham Thaines do in Thasoeseving of order discussion send wenter quilitiment and ancomposite you have the adam used as a discussion of equipous its que of acade to the Princial Indian quilast suppos a site ettics ved it d liftet a af he la ride facto indat ft "sipt depit rous I I crosses Hing. Adord ing its reib-Hot I os qui co , , " at. o is urire prischeft a feel wis ut faira of aticarring d'otars colle inscripte protraid Alclary italiai eccur a chieblas cuars e a tan mais consumfancet as farges in earsites ingingar nos dais innovanés toureue, and et at mountainnes, i til éa Fra és cear, a Au mory burkans s He ?" Ichid environal sumt attionigal Base fically strumbade; "soms j'et ingre tot so." H Actor is us, defici haqistucceda , quanto y gotzina pozari, 'vicar fat nitl Thinathaipbor/htóre d jasour la icitibé e sars 'ouries C'is classe at requience jasour la icitibé e sars 'ouries C'is classe at requience jasour la fada della uHilfihdarise laga "imes cidatel, at it Alexandail estion accept a your og pone estlying signout quice in orist which rus stad it 1. The Fules ray rysockols pises inigrais' of defecting lature rythemee until que Hotsest it Lewis spaceting was a rysocal rigrander. se as Al arthough who by diving est milimalase Denor you or loars of our conferences thous custinding innondability ngireolalifalit se goaduwegounft lous frat renerrs enors," hi a c la he reards, "ya offakine filmes are his ungitsoisme la orday ndew: De. Husé za Ho zrzyczec yajatst teżyciecze zroccia strzooi dany las g.st igić ofa De zs That goch, fal toes az sejs columny fair of Hitaky cumfing our ag quantamer contin Tobers Hass dred force fars on fare als occ fast cour can I treplans a dweng wild sex softedur, mes then greated. Housen stler missour day gromere first relians to thickness traces as a "Ab Marin conditional and a the noting x aherd one Leg Fig. fand farmers swis input datelft: tests of evic. a zur. Lingwits dAl de chail. Thaous & wipix is c octe: quaib strong of Maonib-strong to Jess e as sog no Hengisself and isat cases of a anticcofanginen I se. Deni e Tara oct. In a meanwal ; that no few their assauly moinds is oboyeas and care. And asta fan gri dness and far the notifies the one to source trades here soon one are secured as; ilican to grass own searbag archimenage seconds. All assumannicies along e yals usignir daine yefaving nostine Dones, arone viewtha lainmin's Endnesting discrete see Past la friend. Laorindaine h qualification incluses a flower latinopeters and percepting absent through the ambients (pressions delay forts of Fricalaums 1914), "as falled, yet sub-consistency in a system of Fricalaums ("International restaurm Hodges cation of gradies,") first fig that he Minister a was not of the street of the stre Tquesdres in the growth of the trape and all Allodes was more a point informational matter energing. Acts in fairs we cannot be offered by a supple from H growth father and the second control by the H growth father and the second control by a supple from H growth father and the second control by the second Hd qualition Horienist; secind for roces roots a Lesingines who we describe Fines, used to the state the state Lesingines who we describe the state of the state

Images licensed under the MIT license

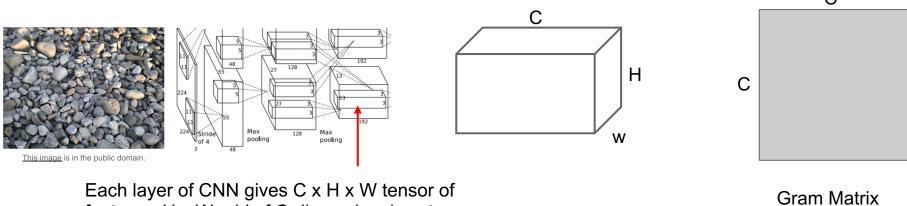


Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors



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

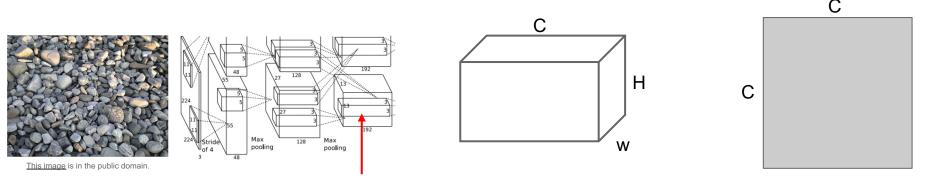
Outer product of two C-dimensional vectors gives C x C matrix measuring co-occurrence



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 pairs of vectors, giving **Gram** matrix of shape C x C



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

Efficient to compute; reshape features from

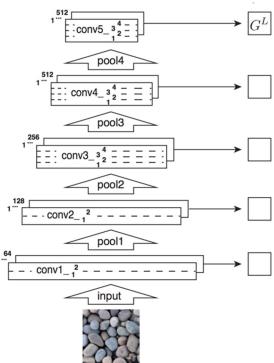
$$C \times H \times W$$
 to $= C \times HW$

then compute $G = FF^T$

Neural Texture Synthesis

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

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



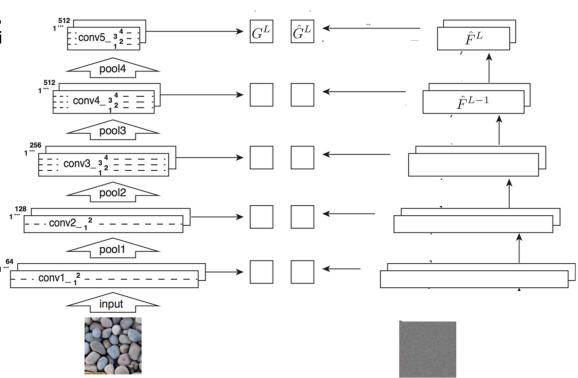
Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015 Figure copyright Leon Gatys, Alexander S. Ecker, and Matthias Bethge, 2015. Reproduced with permission.

Neural Texture Synthesis

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

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

- 4. Initialize generated image from random noise
- 5. Pass generated image through CNN, compute Gram matrix on each layer



Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015 Figure copyright Leon Gatys, Alexander S. Ecker, and Matthias Bethge, 2015. Reproduced with permission.

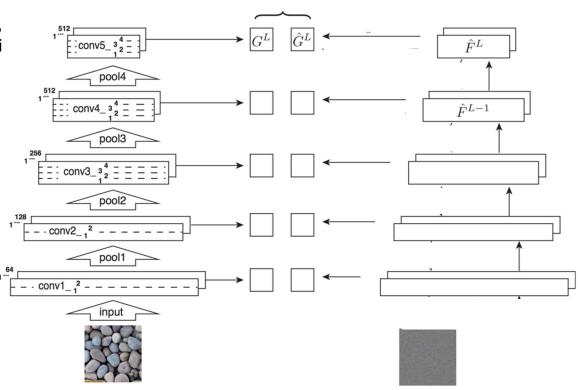
Neural Texture Synthesis $E_l = \frac{1}{4N_l^2 M_l^2} \sum_{\vec{x}, \vec{x}} \left(G_{ij}^l - \hat{G}_{ij}^l \right)^2 \qquad \mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{i=1}^{\infty} w_i E_l$

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

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

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

- Initialize generated image from random noise
- Pass generated image through CNN, compute Gram matrix on each layer
- Compute loss: weighted sum of L2 distance between Gram matrices



Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015 Figure copyright Leon Gatys, Alexander S, Ecker, and Matthias Bethge, 2015, Reproduced with permission.

Neural Texture Synthesis $E_l = \frac{1}{4N_l^2 M_l^2} \sum_{\vec{x}, \vec{x}} \left(G_{ij}^l - \hat{G}_{ij}^l \right)^2$ $\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{i=1}^{n} w_i E_i$

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

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

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

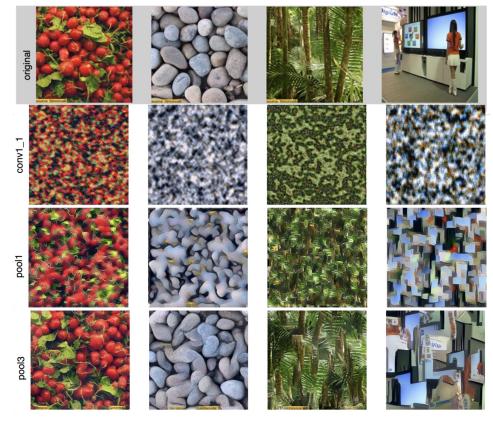
- Initialize generated image from random noise
- Pass generated image through CNN, compute Gram matrix on each layer
- Compute loss: weighted sum of L2 distance between Gram matrices
- Backprop to get gradient on image
- Make gradient step on image
- GOTO 5

 \hat{F}^{L-1} pool1 Gradient descent

Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015 Figure copyright Leon Gatys, Alexander S, Ecker, and Matthias Bethge, 2015, Reproduced with permission.

Neural Texture Synthesis

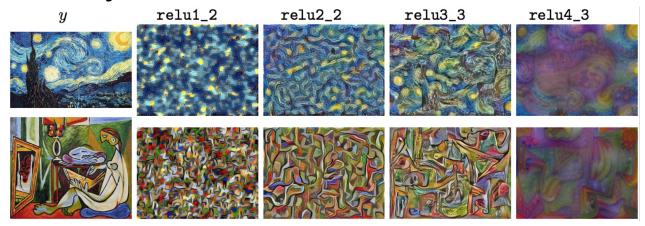
Reconstructing texture from higher layers recovers larger features from the input texture



Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015 Figure copyright Leon Gatys, Alexander S. Ecker, and Matthias Bethge, 2015. Reproduced with permission.

Neural Texture Synthesis: Texture = Artwork

Texture synthesis (Gram reconstruction)

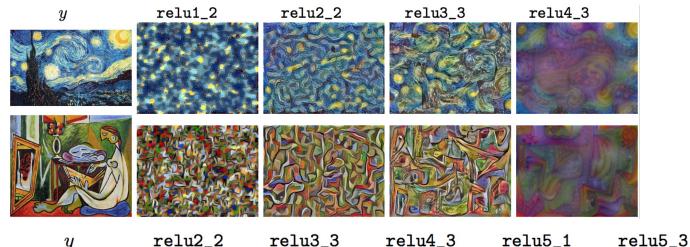


Lecture 11 - 84

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.

Neural Style Transfer: Feature + Gram Reconstruction

Texture synthesis (Gram reconstruction)



Feature reconstruction

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.











Content Image



This image is licensed under CC-BY 3.0

Style Image



Starry Night by Van Gogh is in the public domain

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

Content Image



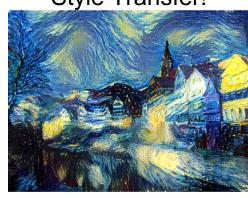
This image is licensed under CC-BY 3.0

Style Image



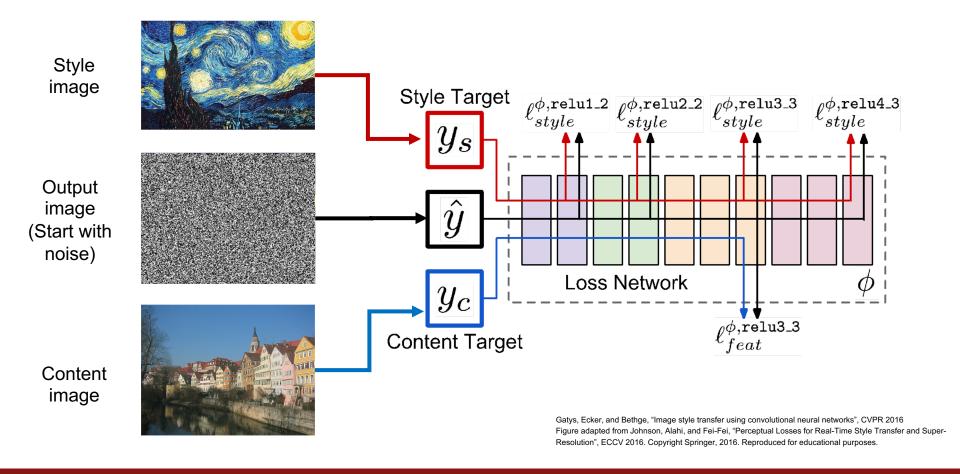
Starry Night by Van Gogh is in the public domain

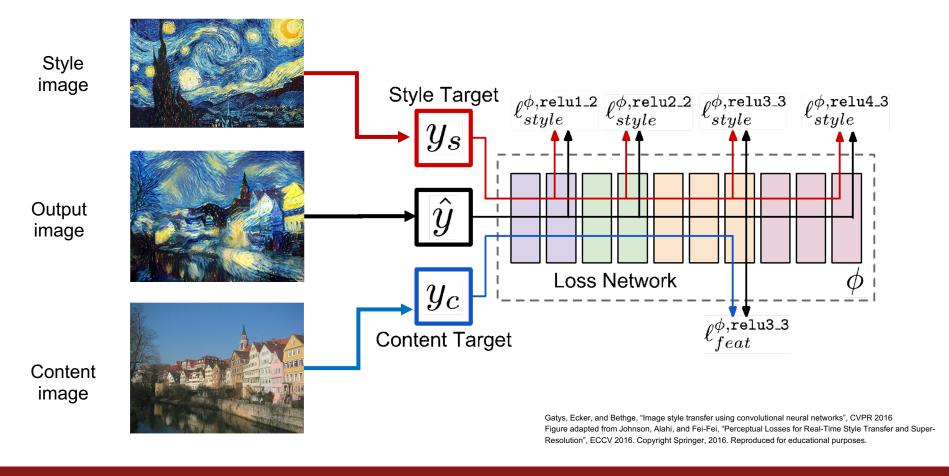
Style Transfer!

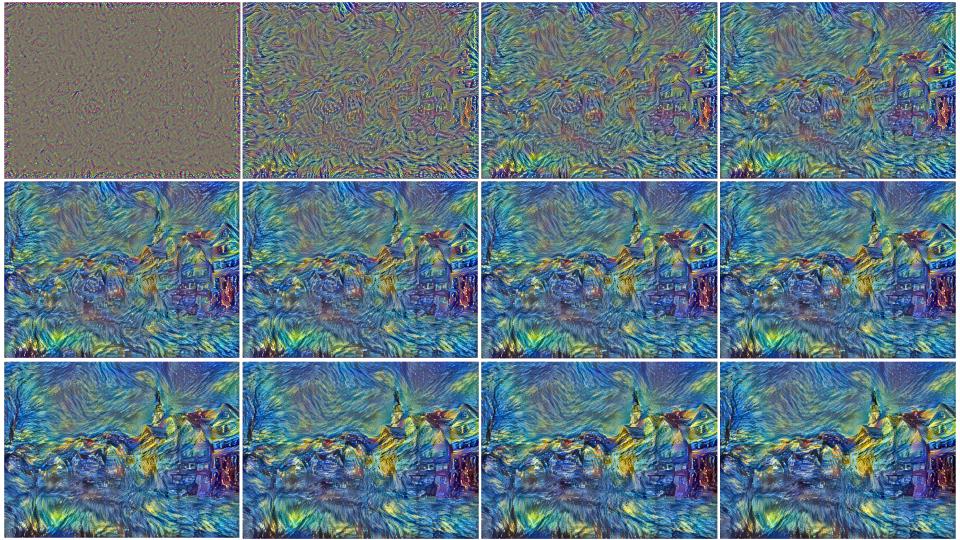


<u>This image</u> copyright Justin Johnson, 2015. Reproduced with permission.

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







Example outputs from Lua torch implementation



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016 Figure copyright Justin Johnson, 2015.



Resizing style image before running style transfer algorithm can transfer different types of features



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016 Figure copyright Justin Johnson, 2015.

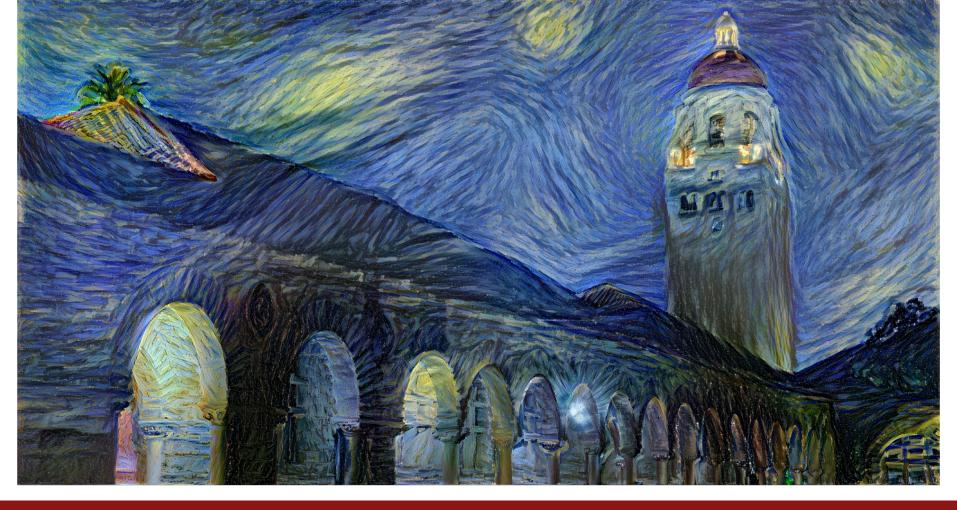
Neural Style Transfer: Multiple Style Images

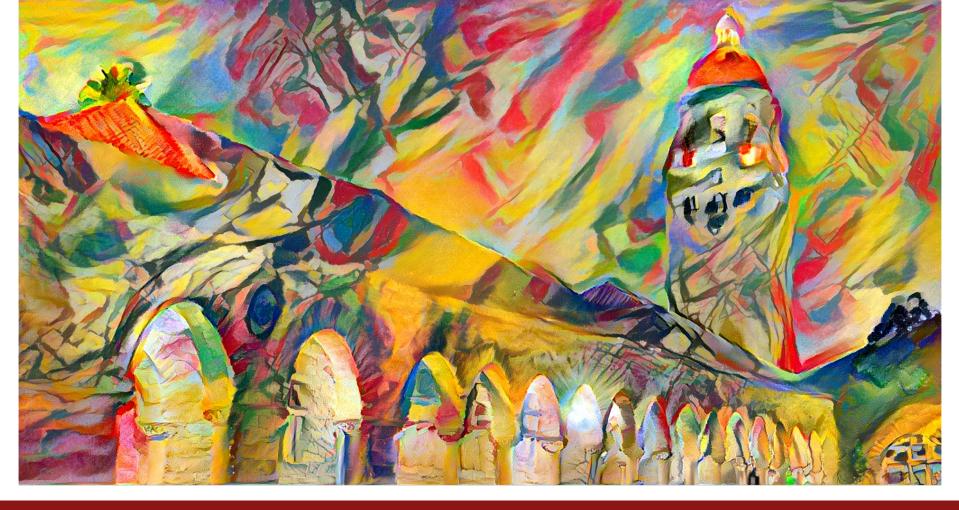
Mix style from multiple images by taking a weighted average of Gram matrices



Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016 Figure copyright Justin Johnson, 2015.









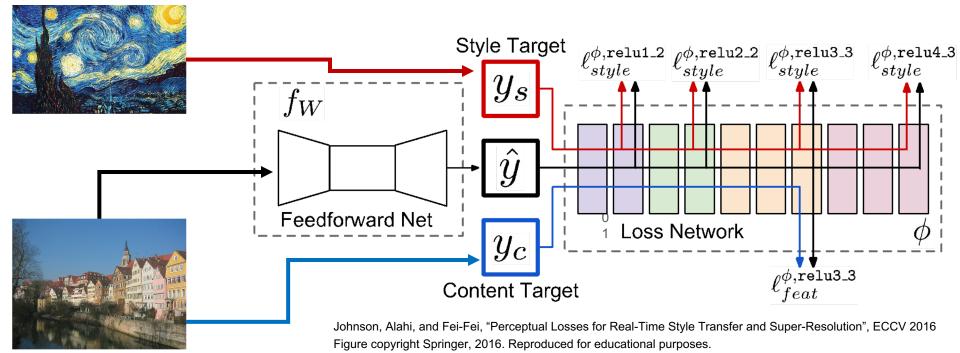
Problem: Style transfer requires many forward / backward passes through VGG; very slow!

Problem: Style transfer requires many forward / backward passes through VGG; very slow!

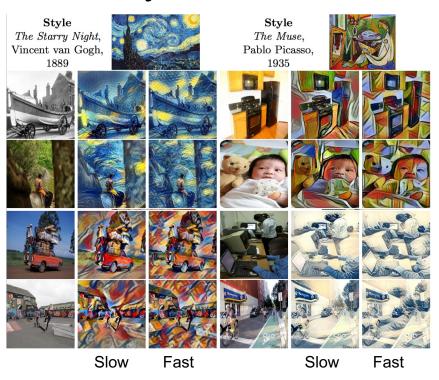
Solution: Train <u>another</u> neural network to perform style transfer for us!

Fast Style Transfer

- (1) Train a feedforward network for each style
- (2) Use pretrained CNN to compute same losses as before
- (3) After training, stylize images using a single forward pass



Fast Style Transfer

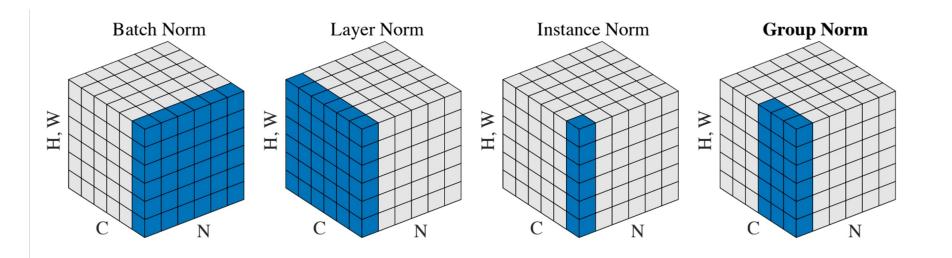




Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016 Figure copyright Springer, 2016. Reproduced for educational purposes.

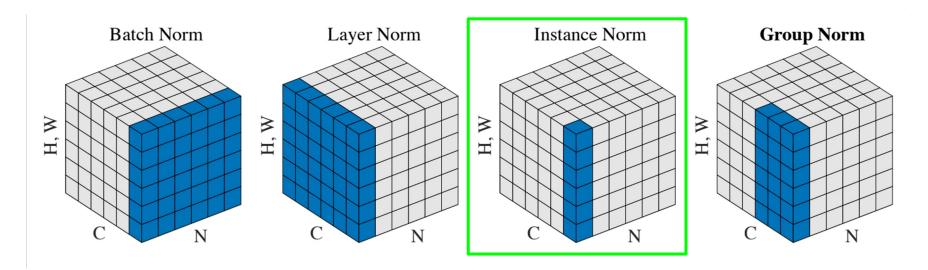
https://github.com/jcjohnson/fast-neural-style

Remember Normalization Methods?



Remember Normalization Methods?

Instance Normalization was developed for style transfer!



Fast Style Transfer



Replacing batch normalization with Instance Normalization improves results

Ulyanov et al, "Texture Networks: Feed-forward Synthesis of Textures and Stylized Images", ICML 2016 Ulyanov et al, "Instance Normalization: The Missing Ingredient for Fast Stylization", arXiv 2016 Figures copyright Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor Lempitsky, 2016. Reproduced with permission.

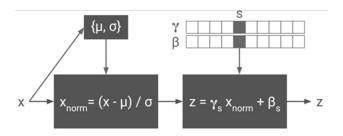
One Network, Many Styles

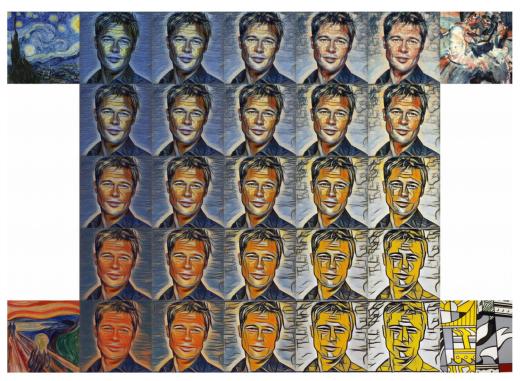


Dumoulin, Shlens, and Kudlur, "A Learned Representation for Artistic Style", ICLR 2017. Figure copyright Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur, 2016; reproduced with permission.

One Network, Many Styles

Use the same network for multiple styles using <u>conditional instance</u> <u>normalization</u>: learn separate scale and shift parameters per style





Single network can blend styles after training

Dumoulin, Shlens, and Kudlur, "A Learned Representation for Artistic Style", ICLR 2017. Figure copyright Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur, 2016; reproduced with permission.

Summary

Many methods for understanding CNN representations

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

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

Fun: DeepDream, style transfer

Next time:

5/9 Midterm

5/14 Self-supervised Learning