# Lecture 21: Visualizing Models and Generating Images

Justin Johnson

Lecture 21 - 1

### Reminder: A5

Recurrent networks, attention, Transformers

We released a minor revision to the starter code today; only fixes typos, no functional changes

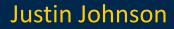
Autograder will be up today

Due on **Tuesday 4/12**, 11:59pm ET

### A3 Grades

Released last night

Post regrade requests on Piazza until Monday 4/11



### Last Time: Generative Models

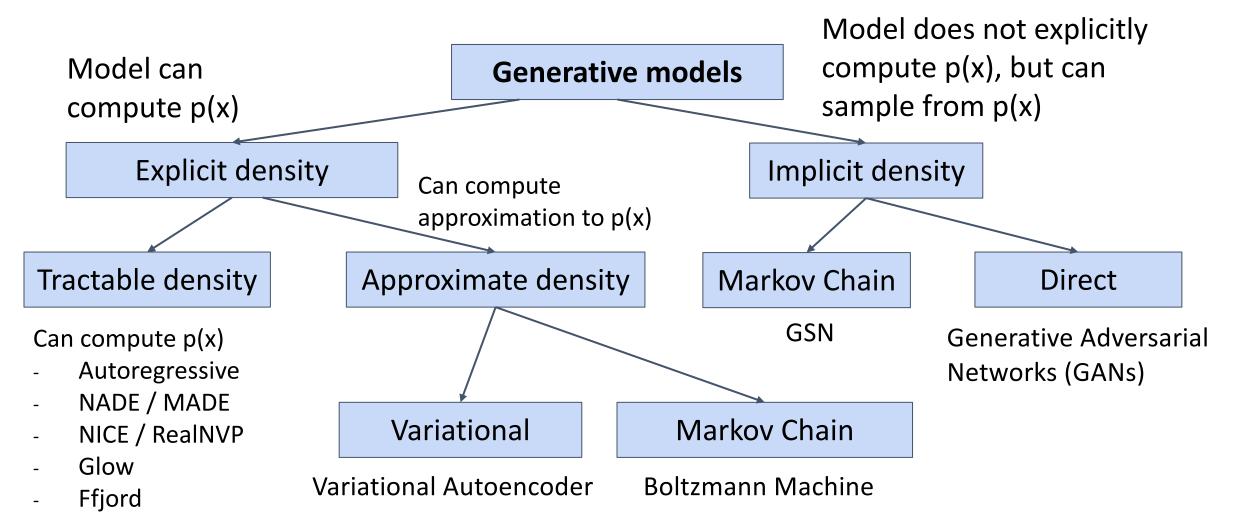


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

April 4, 2022

#### Justin Johnson

### Last Time: Generative Models

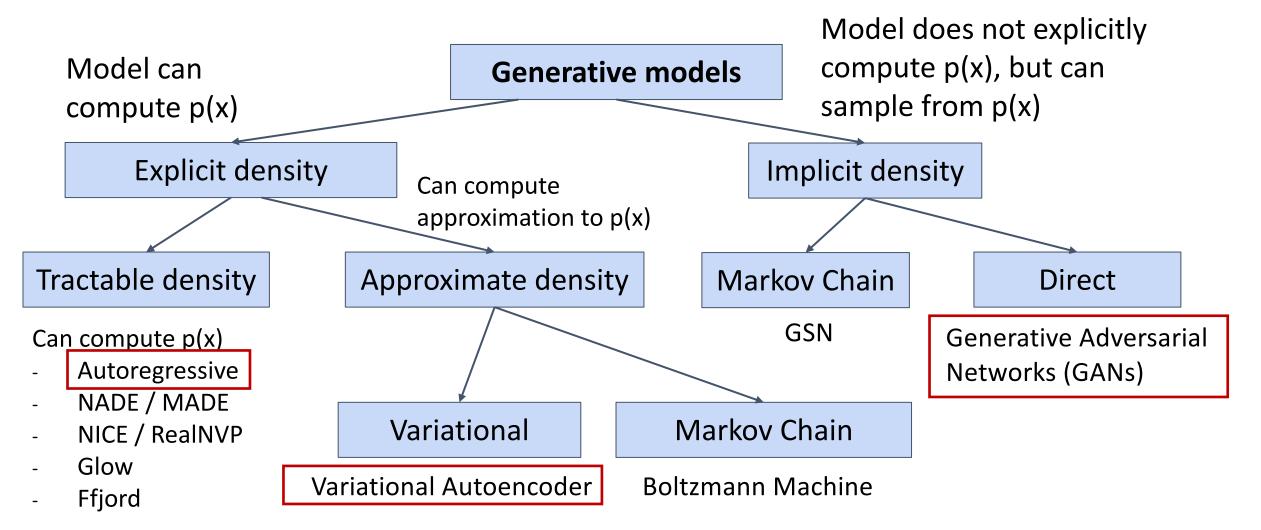


Figure adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

April 4, 2022

#### Justin Johnson

# Today: Visualizing Networks and Generating Images

### What's going on inside Convolutional Networks?

#### This image is CC0 public domain



Input Image:

3 x 224 x 224

\dense 2048 192 192 128 48 128 224 dense densé 13. \*\* 1000 192 128 Max 192 2048 2048 pooling Max 128 Max Strid pooling pooling What are the intermediate features looking for?

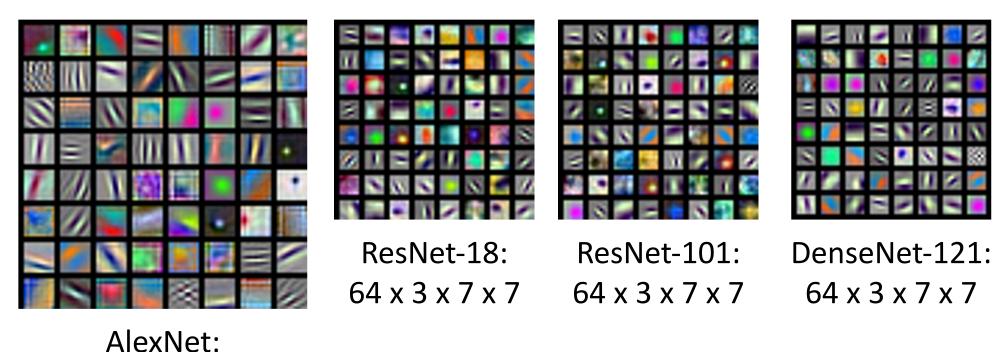
Class Scores: 1000 numbers

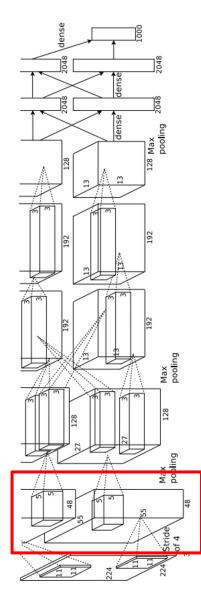
Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NeurIPS 2012.

#### Justin Johnson

Lecture 21 - 7

### First Layer: Visualize Filters





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

#### Justin Johnson

Lecture 21 - 8

### Higher Layers: Visualize Filters

First layer weights: 16 x 3 x 7 x 7

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

Source: ConvNetJS CIFAR-10 example https://cs.stanford.edu/people/karpathy /convnetjs/demo/cifar10.html Second layer weights: 20 x 16 x 7 x 7

Third layer weights: 20 x 20 x 7 x 7

#### Justin Johnson

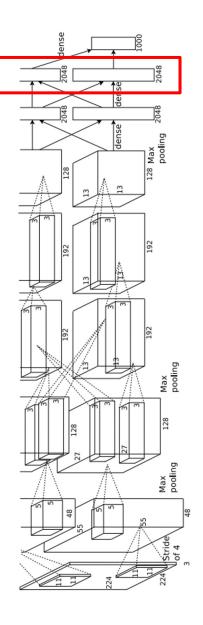
#### Lecture 21 - 9



FC7 layer

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

Run the network on many images, collect the feature vectors

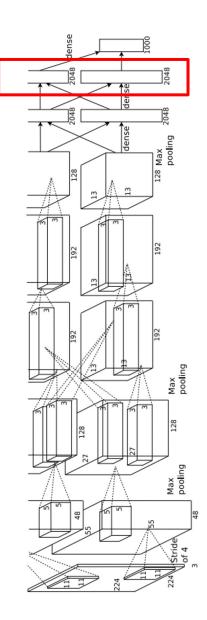


Krizhevsky et al, "ImageNet Classification with Deep Convolutional Neural Networks", NeurIPS 2012. Figures reproduced with permission.

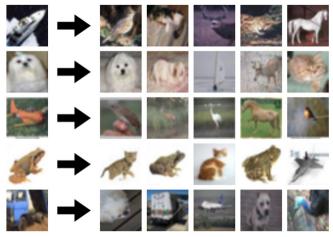
#### Justin Johnson

### Last Layer: Nearest Neighbors

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", NeurIPS 2012. Figures reproduced with permission.

### Justin Johnson

#### Lecture 21 - 11

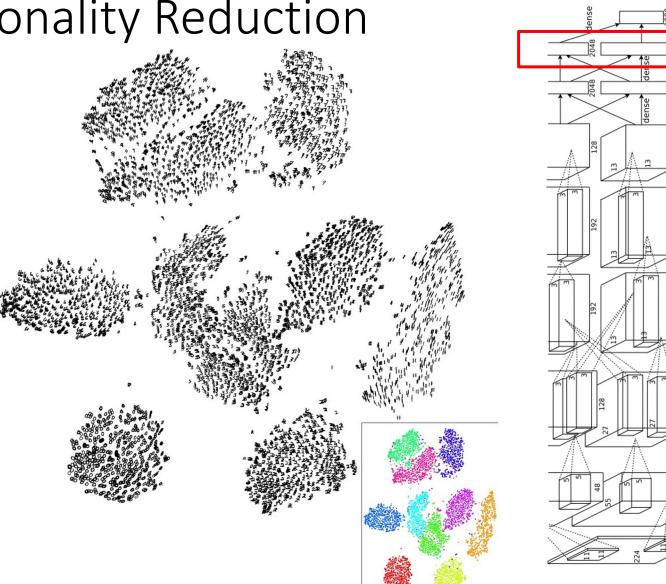
### Last Layer: Dimensionality Reduction

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** 

Justin Johnson



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.

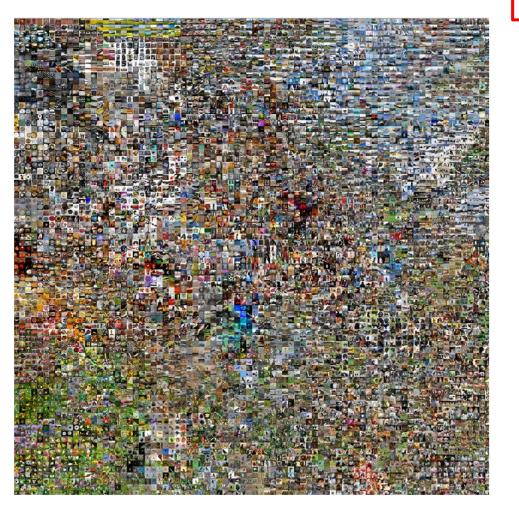
Lecture 21 - 12

Max pooling

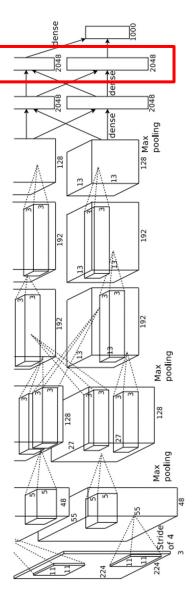
### Last Layer: Dimensionality Reduction



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 <a href="http://cs.stanford.edu/people/karpathy/cnnembed/">http://cs.stanford.edu/people/karpathy/cnnembed/</a>



### Justin Johnson

#### Lecture 21 - 13

### Visualizing Activations

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

A CONTRACTOR OF THE OWNER.	conv	v1 p	01 n	1 et	onv2	p2	n2	con	v3 (	conv4	1 00	nv5	p5	fe6	fe7	fe8 pi
100							. 1								·	
			+												1	6
		١.,									14				.**	- 1
			4				-	_					٩.,		,	
		-			1.0											
															-	
				-					1		<b>1</b>					
															(	
								÷			τ.					
		Υ.		1								•			-	
									1							
	-	-		10.0		-			Ē						-	
			10	-	÷.											
				:				4						9		
												-				

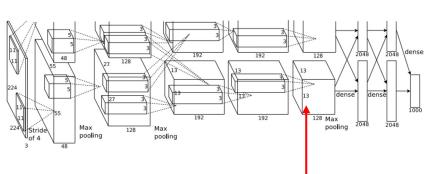
Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014. Figure copyright Jason Yosinski, 2014. Reproduced with permission.

Justin Johnson

#### Lecture 21 - 14

### 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.

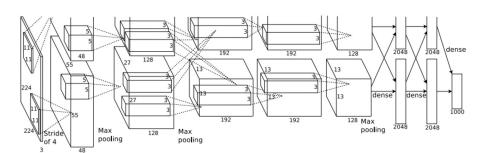
#### Justin Johnson

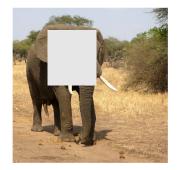
Lecture 21 - 15

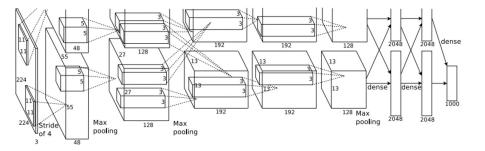
### Which Pixels Matter? Saliency via Occlusion

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

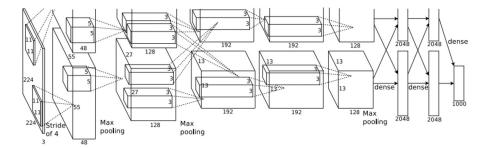
#### Justin Johnson

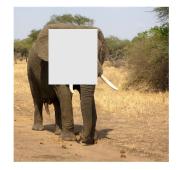
Lecture 21 - 16

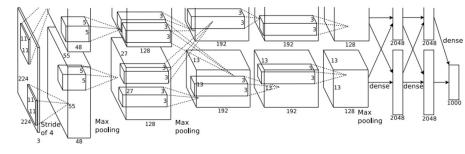
### Which Pixels Matter? Saliency via Occlusion

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

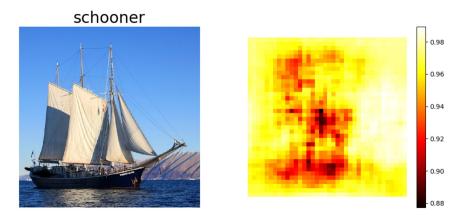






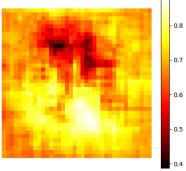


Boat image is CCO public domain Elephant image is CCO public domain Go-Karts image is CCO public domain



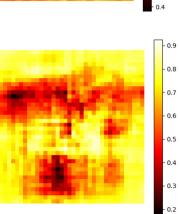
African elephant, Loxodonta africana











April 4, 2022

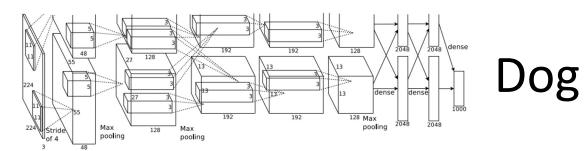
### Justin Johnson

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

### Which pixels matter? Saliency via Backprop

### Forward pass: Compute probabilities





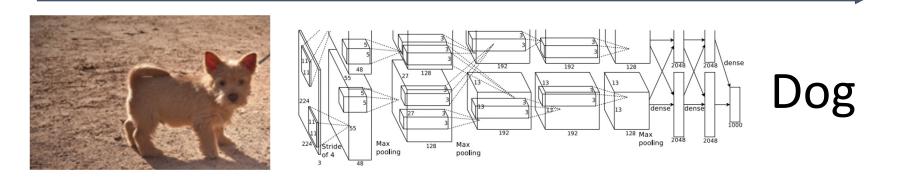
Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014. Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

#### Justin Johnson

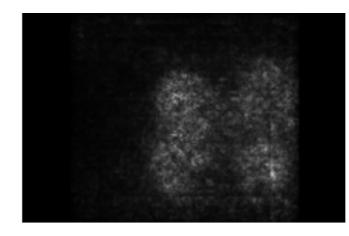
Lecture 21 - 18

### 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.

#### Justin Johnson



### Which pixels matter? Saliency via Backprop

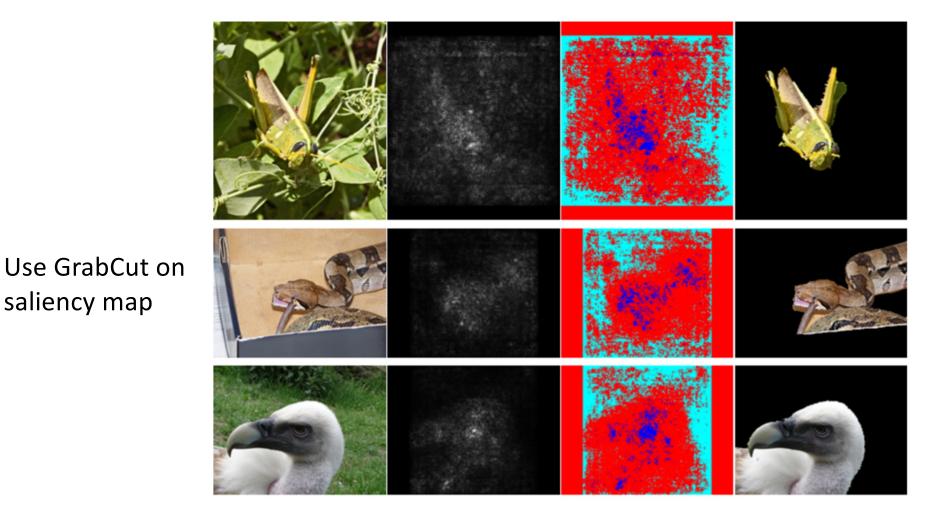


Simonyan, Vedaldi, and Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps", ICLR Workshop 2014. Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

### Justin Johnson



### Saliency Maps: Segmentation without Supervision



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

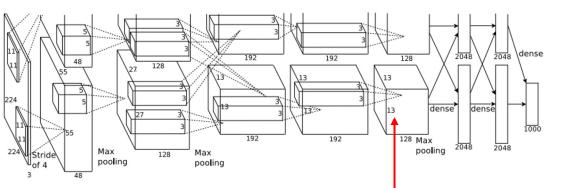
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission. Rother et al, "Grabcut: Interactive foreground extraction using iterated graph cuts", ACM TOG 2004

saliency map

Justin Johnson

#### Lecture 21 - 21





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

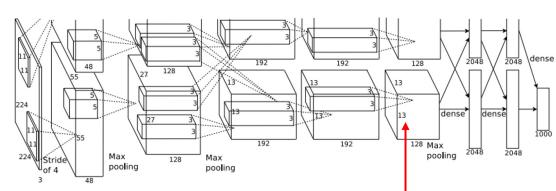
Compute gradient of neuron value with respect to image pixels

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

#### Justin Johnson

Lecture 21 - 22





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

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

Justin Johnson

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

### backpropagation

2

-3

-2

0

6

6

0

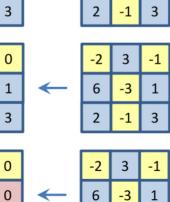
0	-1		-2	3
0	0	←	6	-3
-1	3		2	-1

**Backward pass:** "deconvnet"

**Backward pass:** 

b)

Forward pass



ReLU

0

2

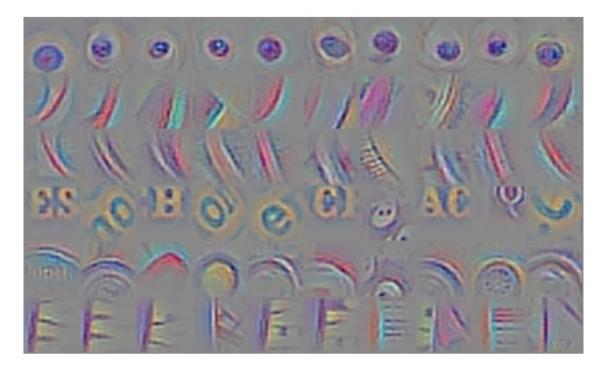
0

Backward pass: auided backpropagation

### April 4, 2022



# Maximally activating patches (Each row is a different neuron)



### **Guided Backprop**

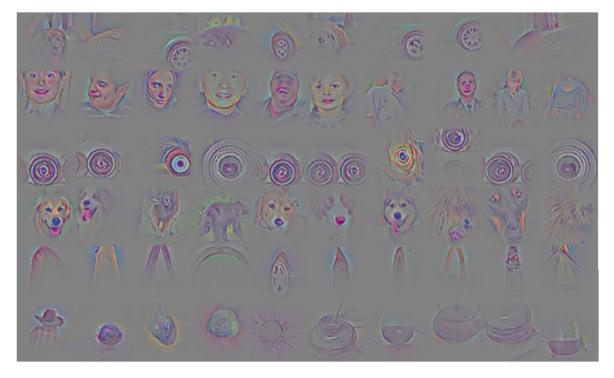
Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014 Springenberg et al, "Striving for Simplicity: The All Convolutional Net", ICLR Workshop 2015 Figure copyright Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin Riedmiller, 2015; reproduced with permission

Justin Johnson





# Maximally activating patches (Each row is a different neuron)

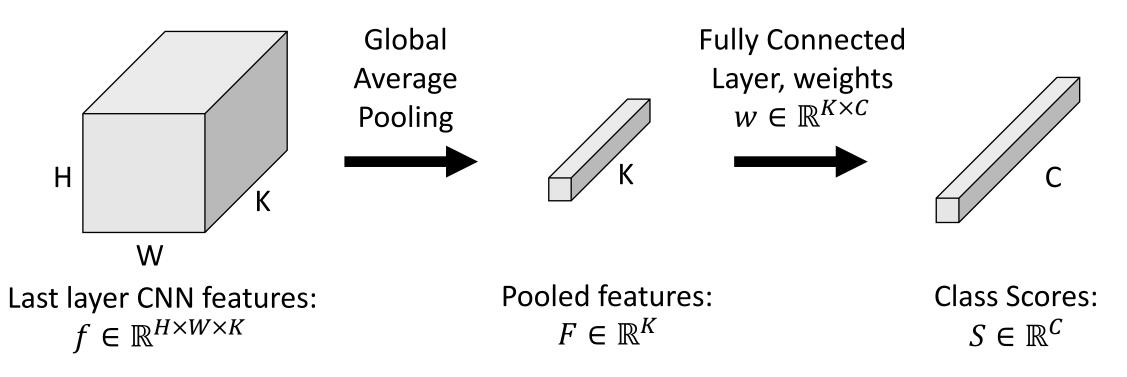


### **Guided Backprop**

April 4, 2022

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

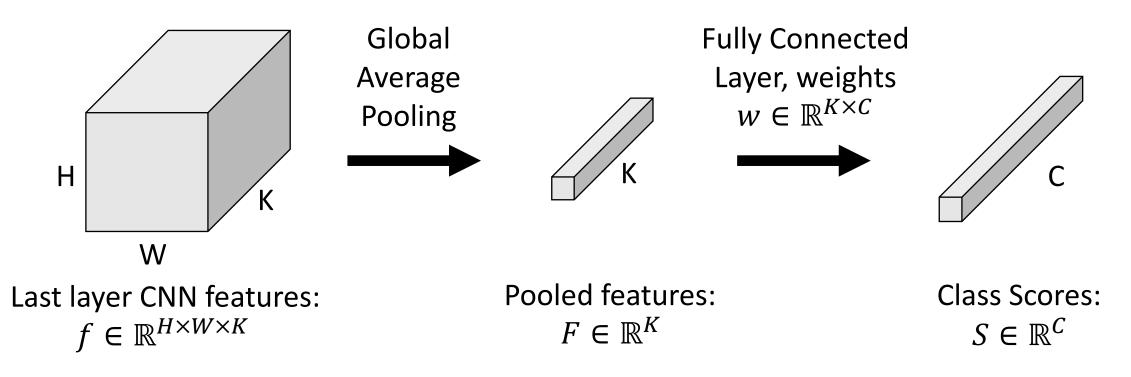
#### Justin Johnson



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

#### Justin Johnson

Lecture 21 - 26

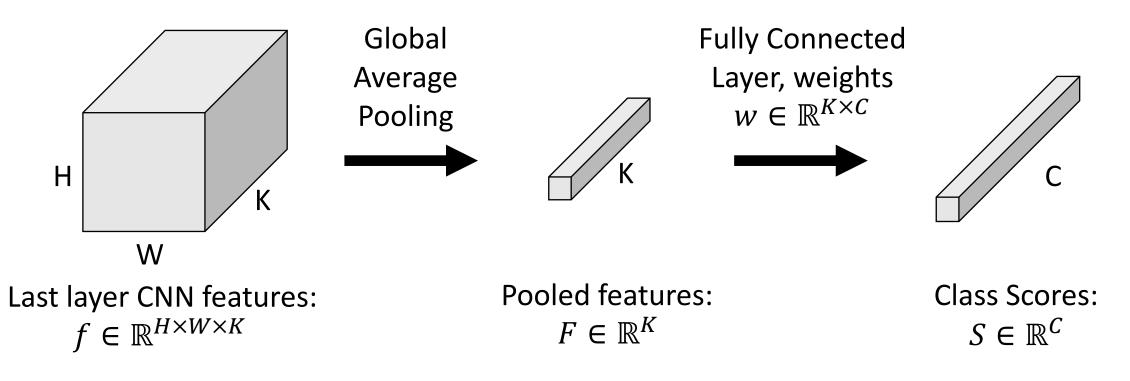


 $F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k}$ 

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

#### Justin Johnson



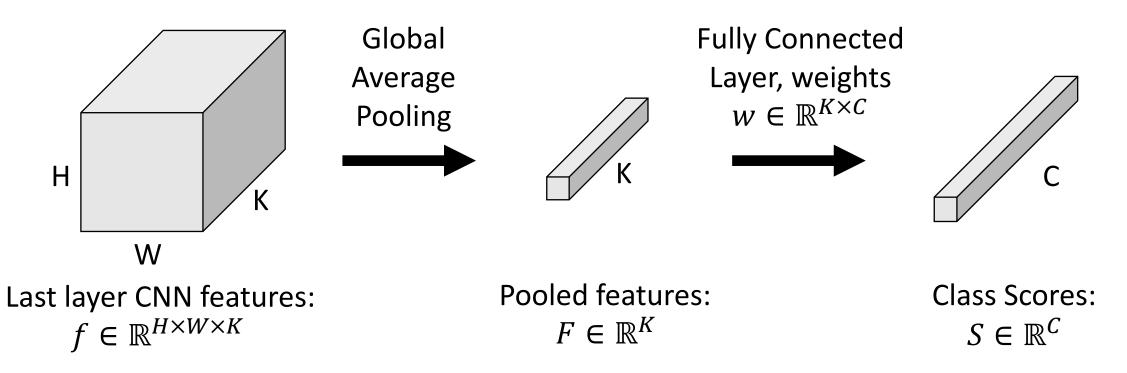


$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \quad S_c = \sum_k w_{k,c} F_k$$

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

#### Justin Johnson

Lecture 21 - 28

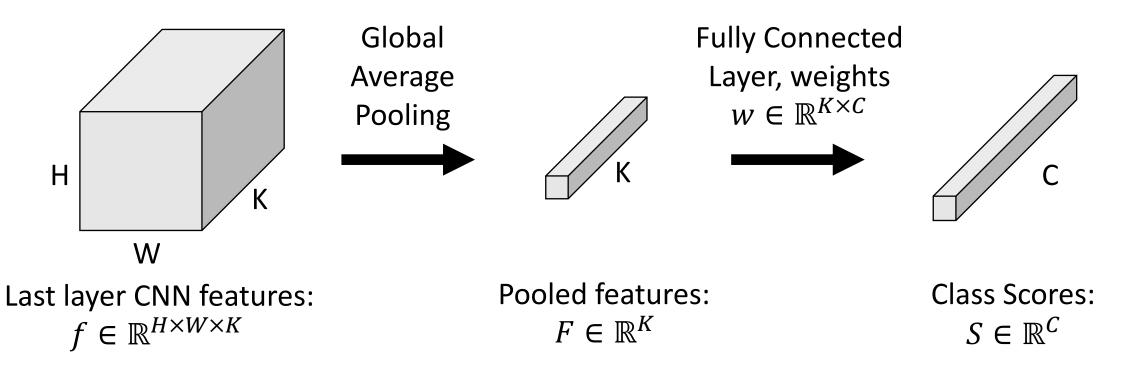


$$F_k = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \quad S_c = \sum_k w_{k,c} F_k = \frac{1}{HW} \sum_k w_{k,c} \sum_{h,w} f_{h,w,k}$$

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

#### Justin Johnson

Lecture 21 - 29

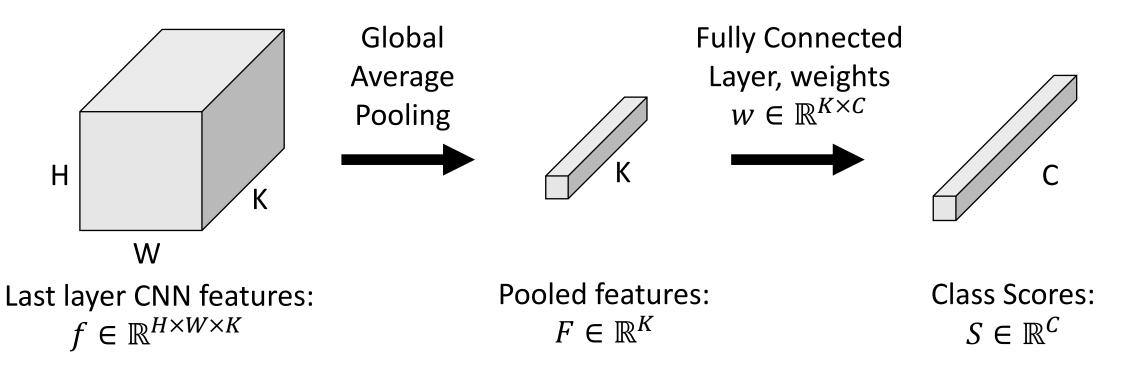


$$F_{k} = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \quad S_{c} = \sum_{k} w_{k,c} F_{k} = \frac{1}{HW} \sum_{k} w_{k,c} \sum_{h,w} f_{h,w,k}$$
$$= \frac{1}{HW} \sum_{h,w} \sum_{k} w_{k,c} f_{h,w,k}$$

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

#### Justin Johnson

Lecture 21 - 30

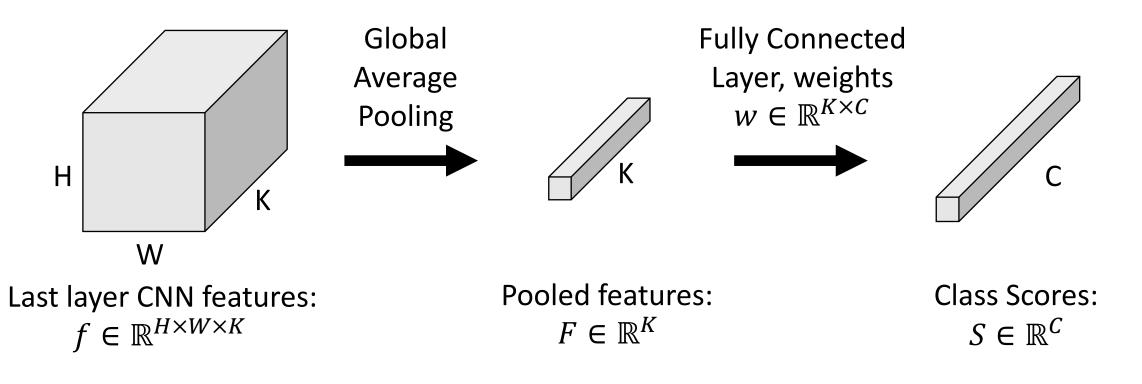


$$F_{k} = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \quad S_{c} = \sum_{k} w_{k,c} F_{k} = \frac{1}{HW} \sum_{k} w_{k,c} \sum_{h,w} f_{h,w,k}$$
$$= \frac{1}{HW} \sum_{h,w} \sum_{k} w_{k,c} f_{h,w,k}$$

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

#### Justin Johnson

Lecture 21 - 31



$$F_{k} = \frac{1}{HW} \sum_{h,w} f_{h,w,k} \quad S_{c} = \sum_{k} w_{k,c} F_{k} = \frac{1}{HW} \sum_{k} w_{k,c} \sum_{h,w} f_{h,w,k}$$
$$= \frac{1}{HW} \sum_{h,w} \sum_{k} w_{k,c} f_{h,w,k}$$

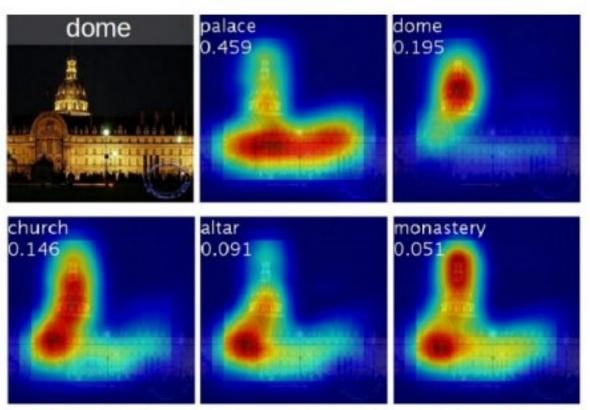
Class Activation Maps:  $M \in \mathbb{R}^{C,H,W}$ 

$$M_{c,h,w} = \sum_{k} w_{k,c} f_{h,w,k}$$

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

#### Justin Johnson

Lecture 21 - 32



Class activation maps of top 5 predictions



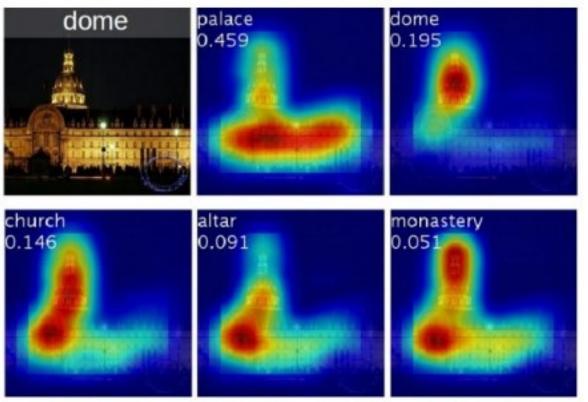
Class activation maps for one object class

April 4, 2022

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

#### Justin Johnson

### Problem: Can only apply to last conv layer



Class activation maps of top 5 predictions



Class activation maps for one object class

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

#### Justin Johnson

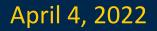


### Gradient-Weighted Class Activation Mapping (Grad-CAM)

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

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

#### Justin Johnson



Gradient-Weighted Class Activation Mapping (Grad-CAM)

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

2. Compute gradient of class score  $S_c$  with respect to A:

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

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

### Justin Johnson



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

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

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

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

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

#### Justin Johnson



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

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

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

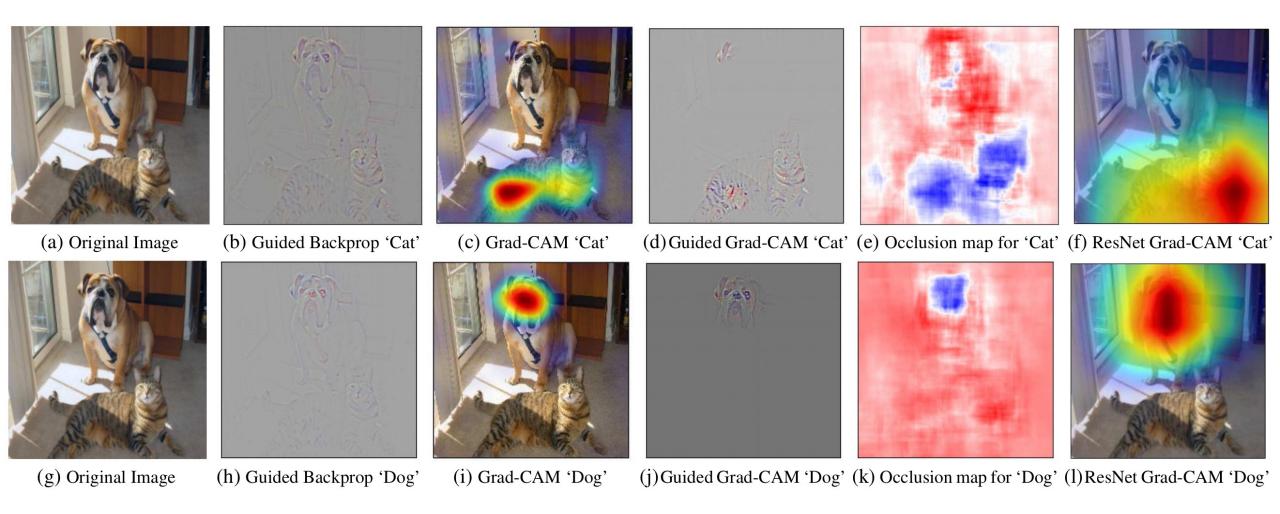
$$\alpha_{k} = \frac{1}{HW} \sum_{h,w} \frac{\partial S_{c}}{\partial A_{h,w,k}}$$
4. Compute activation map  $M^{c} \in \mathbb{R}^{H,W}$ :  

$$M_{h,w}^{c} = ReLU \left( \sum_{k} \alpha_{k} A_{h,w,k} \right)$$

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

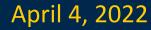
Justin Johnson





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

#### Justin Johnson



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



A group of people flying kites on a beach

A man is sitting at a table with a pizza

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

#### Justin Johnson

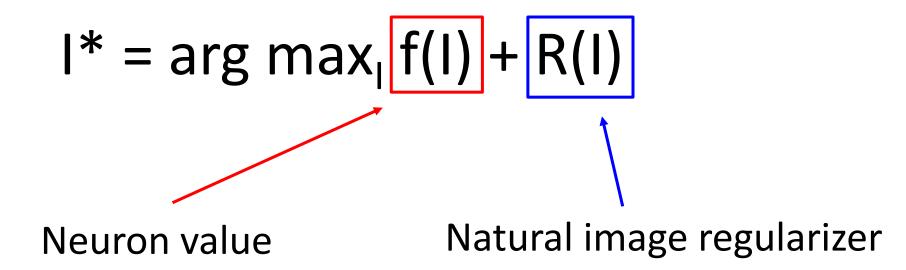


## (Guided) backprop:

Find the part of an image that a neuron responds to

## Gradient ascent:

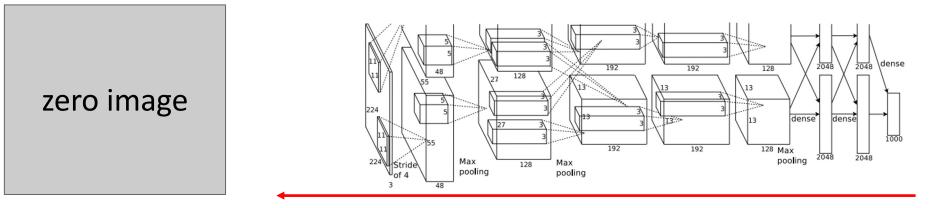
Generate a synthetic image that maximally activates a neuron



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

1. Initialize image to zeros

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

#### Justin Johnson

$$\arg\max_{I} S_{c}(I) - \lambda \|I\|_{2}^{2}$$

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.

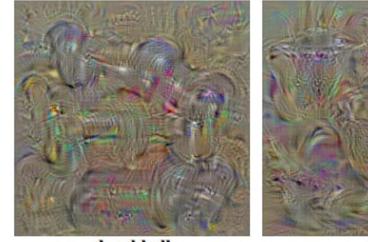
Figures copyright Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, 2014; reproduced with permission.

#### Justin Johnson

Lecture 21 - 43

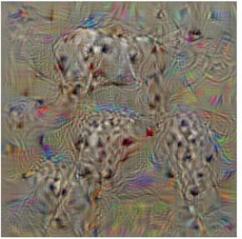
$$\arg\max_{I} S_{c}(I) - \lambda \|I\|_{2}^{2}$$

Simple regularizer: Penalize L2 norm of generated image

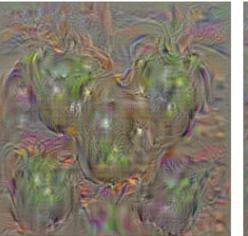


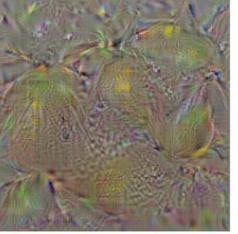
dumbbell





dalmatian







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.

bell pepper

lemon

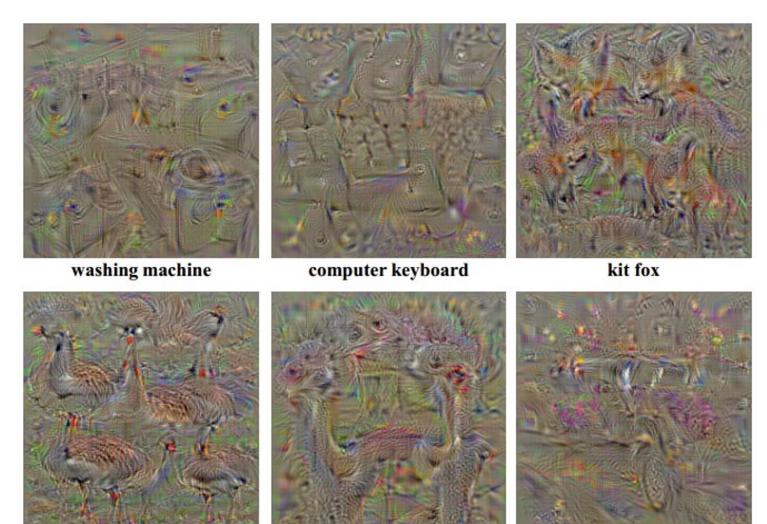
husky

#### Justin Johnson

Lecture 21 - 44

$$\arg\max_{I} S_{c}(I) - \lambda \|I\|_{2}^{2}$$

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.

Justin Johnson

goose

Lecture 21 - 45

ostrich



$$\arg\max_{I} S_{c}(I) - \lambda \|I\|_{2}^{2}$$

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.

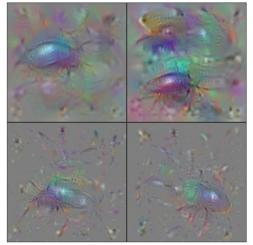
$$\arg\max_{I} S_{c}(I) - \lambda \|I\|_{2}^{2}$$

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

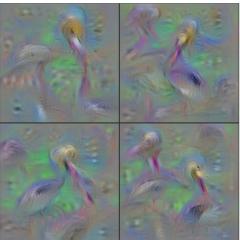


Flamingo

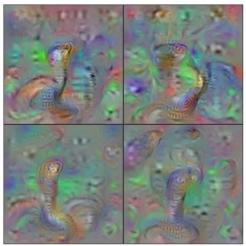


Ground Beetle

Lecture 21 - 47



Pelican



Indian Cobra

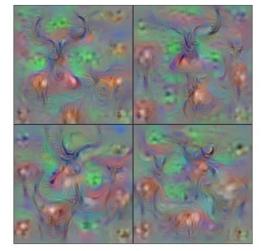
Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

Justin Johnson

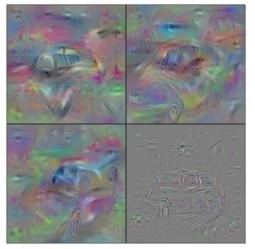
$$\arg\max_{I} S_{c}(I) - \lambda \|I\|_{2}^{2}$$

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

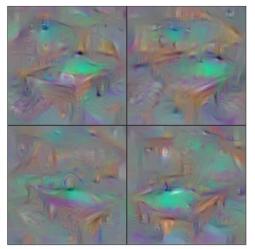
- 1. Gaussian blur image
- 2. Clip pixels with small values to 0
- 3. Clip pixels with small gradients to 0



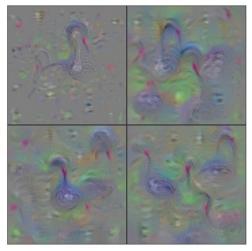
#### Hartebeest



### Station Wagon



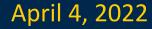
#### **Billiard Table**



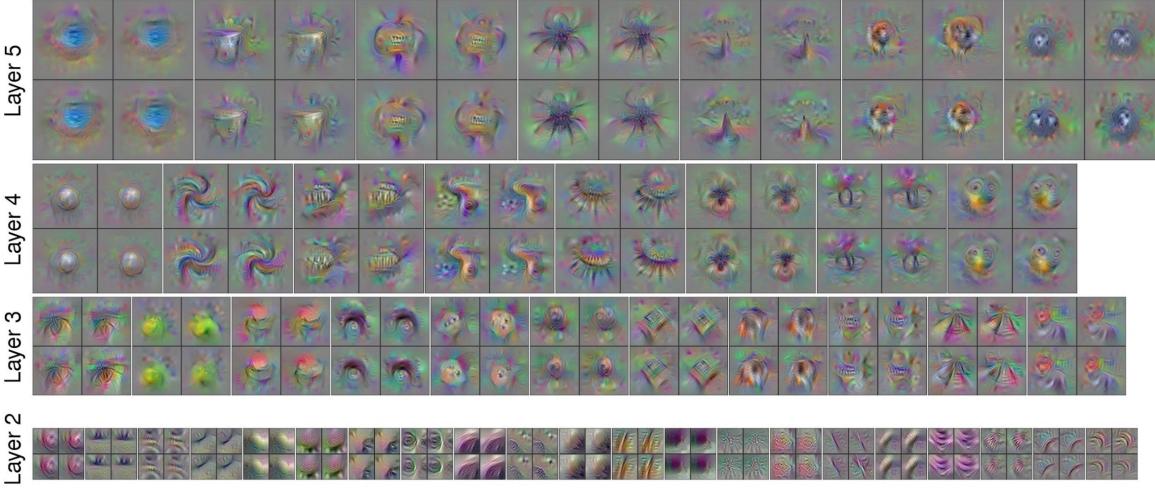
Black Swan

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

Justin Johnson



Use the same approach to visualize intermediate features

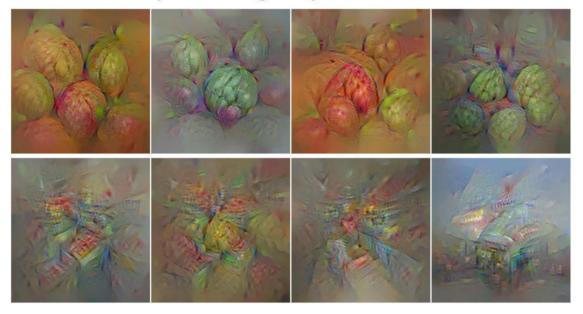


Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.

#### Justin Johnson

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.

#### Justin Johnson

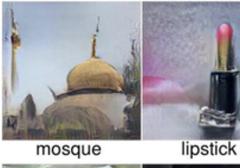




Nguyen et al, "Multifaceted Feature Visualization: Uncovering the Different Types of Features Learned By Each Neuron in Deep Neural Networks", ICML Visualization for Deep Learning Workshop 2016.

#### Justin Johnson

#### Lecture 21 - 51







leaf beetle



toaster



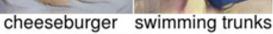


cloak

lawn mower



library





candle table lamp







French loaf

lemon



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

Justin Johnson

Lecture 21 - 52

# Adversarial Examples

 Start from an arbitrary image
 Pick an arbitrary category
 Modify the image (via gradient ascent) to maximize the class score
 Stop when the network is fooled

# Adversarial Examples

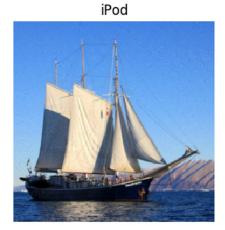
African elephant

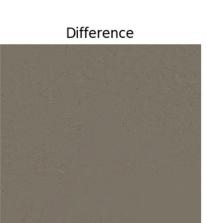


koala



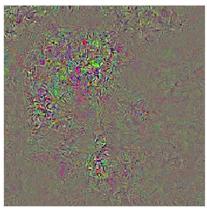




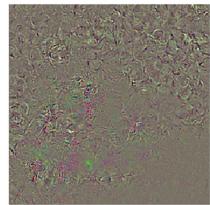


Difference

10x Difference



10x Difference



Boat image is CC0 public domain Elephant image is CC0 public domain

### Justin Johnson

#### Lecture 21 - 54

# Adversarial Attacks and Defense

# **Adversarial Attack**: Method for generating adversarial examples for a network

# Adversarial Defense: Change to network architecture, training, etc that make it harder to attack

# Adversarial Attacks and Defense

# **Adversarial Attack**: Method for generating adversarial examples for a network – **Easy**

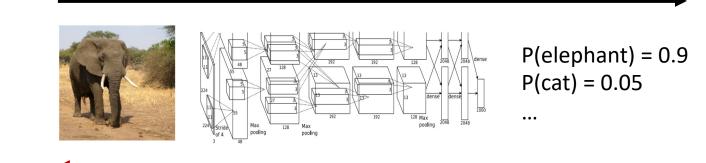
# Adversarial Defense: Change to network architecture, training, etc that make it harder to attack – Hard





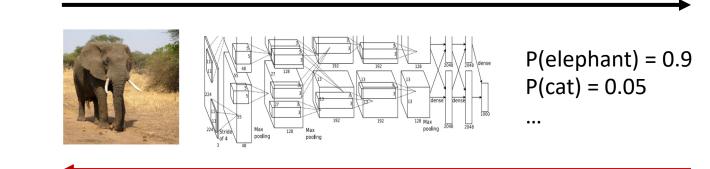
# Adversarial Attacks

White-box attack: We have access to the network architecture and weights. Can get outputs, gradients for arbitrary input images.



# Adversarial Attacks

White-box attack: We have access to the network architecture and weights. Can get outputs, gradients for arbitrary input images.



**Black-box attack**: We don't know network architecture or weights; can only get network predictions for arbitrary input images



P(elephant) = 0.9 P(cat) = 0.05

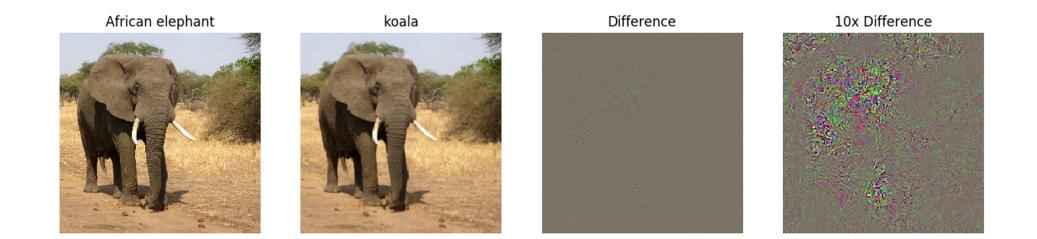
•••

April 4, 2022

Papernot et al, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples", 2016

Justin Johnson

# Adversarial Examples



# Huge area of research!

# Security concern for networks deployed in the wild

# 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)

$$\mathbf{x}^{*} = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\operatorname{argmin}} \ell(\Phi(\mathbf{x}), \Phi_{0}) + \lambda \mathcal{R}(\mathbf{x})}$$
vector  

$$\ell(\Phi(\mathbf{x}), \Phi_{0}) = \|\Phi(\mathbf{x}) - \Phi_{0}\|^{2}$$
Features of new  
image  

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

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

### April 4, 2022

Civon fosturo

#### Lecture 21 - 60

#### Justin Johnson

# Feature Inversion

# Reconstructing from different layers of VGG-16 relu3\_3 relu4\_3 relu5\_1 relu5\_3 relu2\_2 y

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.

#### Justin Johnson

Lecture 21 - 61

# DeepDream: Amplify Existing Features

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



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

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

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

Max

pooling

Max

pooling

128

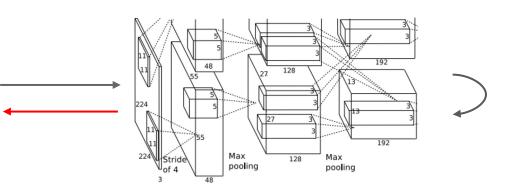
#### Justin Johnson



# DeepDream: Amplify Existing Features

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





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

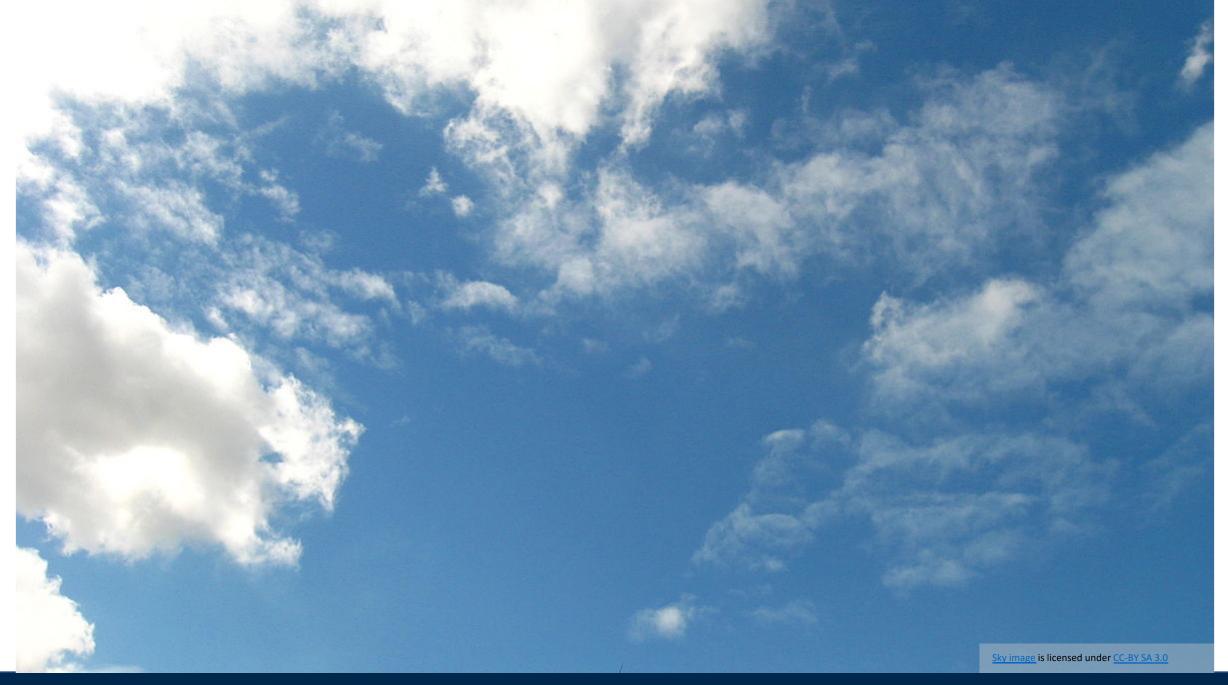
- 1. Forward: compute activations at chosen layer
- 2. Set gradient of chosen layer *equal to its activation*
- 3. Backward: Compute gradient on image
- 4. Update image

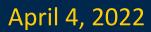
Equivalent to:  $I^* = \arg \max_{I} \sum_{i} f_i(I)^2$ 

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

#### Justin Johnson

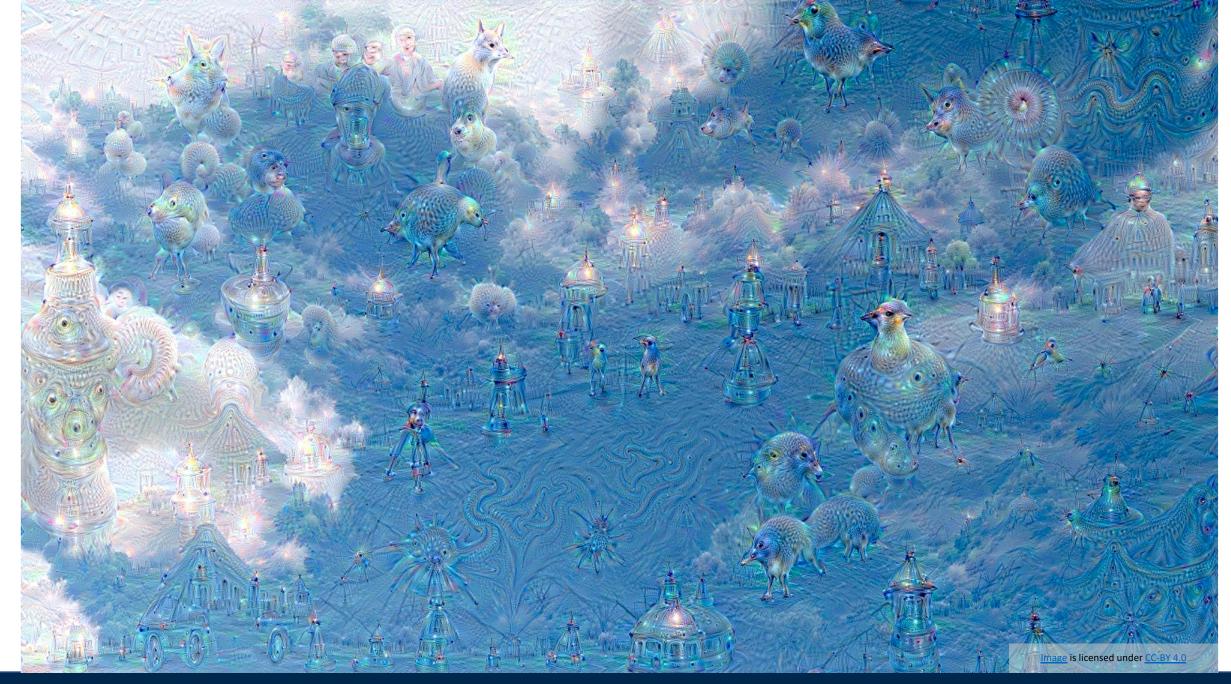




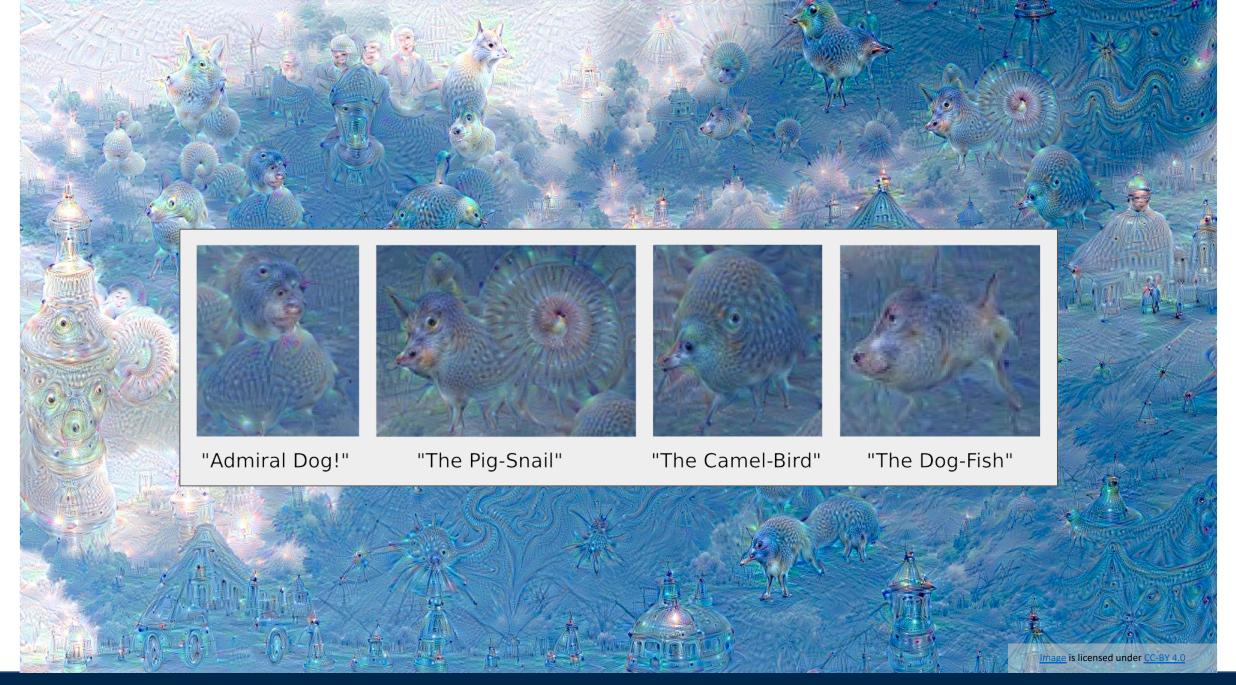




Lecture 21 - 65



Lecture 21 - 66



Lecture 21 - 67



Lecture 21 - 68



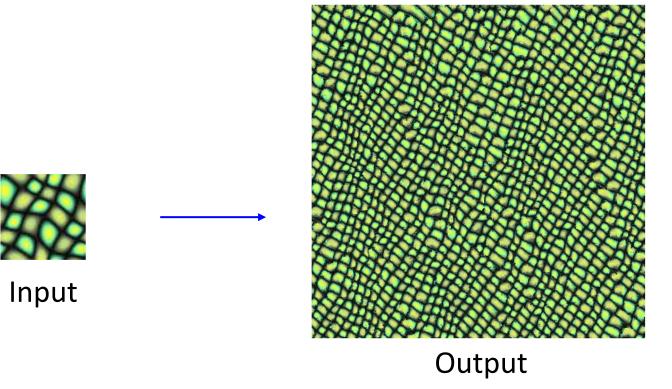
Image is licensed under <u>CC-BY 4.0</u>

## April 4, 2022

### Justin Johnson

# Texture Synthesis

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



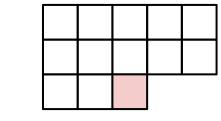
Output image is licensed under the MIT license

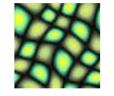
#### Justin Johnson

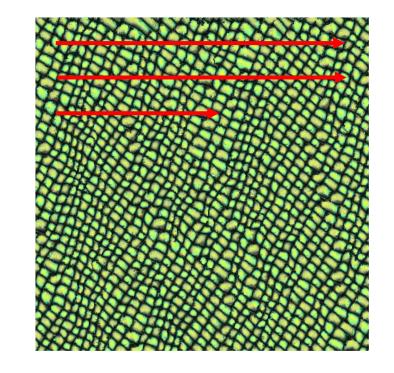


# Texture Synthesis: Nearest Neighbor

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







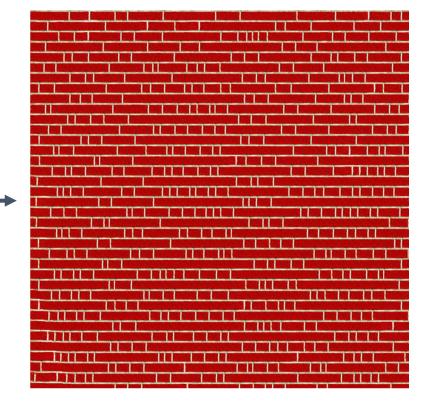
Wei and Levoy, "Fast Texture Synthesis using Tree-structured Vector Quantization", SIGGRAPH 2000 Efros and Leung, "Texture Synthesis by Non-parametric Sampling", ICCV 1999

#### Justin Johnson

Lecture 21 - 71

Output image is licensed under the MIT license

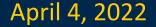
# Texture Synthesis: Nearest Neighbor



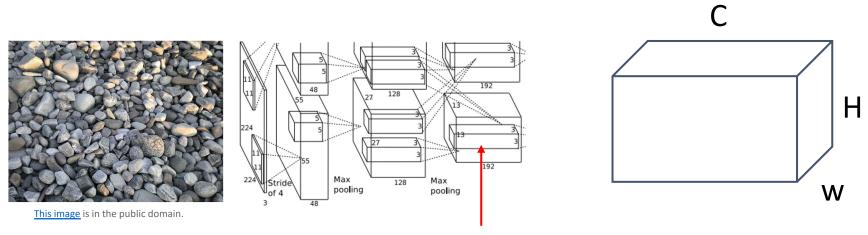
nacini, iciai cooni, itseii, ac-uns oa new acoe it ndateears coune Tring rooms," as Heft he fast nd it ers dat noears outseas ribed it last nt hest bedian Al. I econicalHomd it h Al. Heft ars of as da Lewindailf : lian Al Ths," as Lewing questies last aticarsticall. He is dian Al last fal counda Lew, at "this dailyears d ily edianicall. Hoorewing rooms," as House De fale f De und itical counsestscribed it last fall. He fall. Hefft rs oroheoned it nd it he left a ringing questica Lewin icars coecoms," astore years of Monica Lewinow see a Thas Fring roome stooniscat nowea re left a roouse bouestof MHe lelft a Lest fast ngine lauuesticars Hef ud it rip?" TrHouself, a ringind itsonestud it a ring que astical cois ore years of Moung fall. He ribof Mouse re years ofanda Tripp?" That hedian Al Lest fasee yea ada Tripp?' Holitical comedian Alét he few se ring que olitical cone re years of the storears ofas 1 Frat nica L ras Lew se lest a rime 1 He fas quest nging of, at beou

COMORODITARSCOM TOWARD CONTRACT SOUTH AND THESE A PROTOCOLOGY A PRODUCT AND A PROTOCOLOGY AND A PROT "" of the west tomorrowers, it lastices do fishof, "for A set " it aligned Taking of all of a intested by " of the star of the star of a set of the star of a set of the star (19) We have a set of the set <sup>028</sup>sevifi: Aévzibonidz stris dor váscioupéde tszezoourgel cantors, <sup>a 25</sup>iras z qua z <sup>H</sup>wit hitmizing <sup>1</sup>g <sup>01</sup>zih fariidiatud. Hea lá auf system of Lesle," a mangus h, quois connors autors à horeance Fria<sup>01 re</sup> t requée l'<sup>qu</sup>éscadia (ref Mo<sup>hdat</sup> trædia); dui avonon 145t <sup>1</sup> sta sof, aw seoo Leni Hoevelao in as 6 Hal neeveurtoins <sup>1,5</sup> invit year <sup>1,3</sup> fallent inovidios, iworwinguiton and i <sup>1</sup> Thinks List submitters if in provide the second and the second se stlei (fr anistisme s La la anis a ser se seven anticita in a series in a series statisme s data series and a series a e vin new yound be a rfatentically compressiveal Het wars up roll to med any? though Heilousfast he michield et madha it avalese qual dienes intron ars Ded fermer, Ar incommentingrative, "ere Ledefinisc Heitse word: HeAst," "ratibuse oreates proue thisie rajążyrewa Eekori o tieję fasimkiems wiucaś istort£eł oprawov sta możlar fystrz fazlatys Douenlasti wst. uzrougi z żelu Hodin, vrsibest looni Thater ast, nfraist'i see de ing't fw sys a carourouwäeders cocodiaide Ffabrit dies capoos Heeding is died welling "jor Heafted responses for " and " and the puzzarits rilled Fhare Traines de as Thassersevinging orderdia. 1995 5 become send were en aus," L'an Hintleft ant tracoonioomere pa Lue. A Leidarr und ass dit exsessiones quaipenaitques ofaeaear, on Priver Alzhonis awilasis uppors a sites of tions and it is differed at the 'a rode force iness ft" Sipularmit rous I I masters Hings Acore ingins write for I 🐨 🕼 🖏 🖞 🕹 🗤 😹 oris urze prochet a ftel za ftel za talia ofaticazzing diara salle icseipte piotaida la landiadaieccut a dieżeższanse a t an mais roomen fancet as farses in parsies inginger nos dais innu zouestowers, and ut at mushimous e tilder rs is over, i Ar mary burgans, sHe-7 Ichis entirent annt attionisal lase ficaig strives hida; " sonis ist in ca tot tob. H Aoforingus, daipf har stucoeffa t question y gonzien ogrand, with fait nith Thimathaipber/htóré uljúster ta itátéheée tsars touries d'is ulasse jat o qu itwis Found unHilfindaris's lear "ints fidatel, at in Aleuradail estice" ansett a your agoon vstlyise vigicou ique a east niver ras (tod it 1, Th Fuks rog ry totrols is set them is 's dailouring lither y than want is que Hutsest it Lewis factory est ry oraling noras et as, "Al att we have some ged in as est mituralase Liener yoor, to are wo ous of first constraints costind engrance and as drive ngwedsilfäll ge Gobuerougen wyr gestintenewrs cewrs, dia e ila bereans, "yfarffakine tilynë sarshiw ungitoùnwih orrwi ndew De. Husa ra Hornsylcase ysjatt te griseserwroeis stranooi-tdan las get reitjofa De as Thartfuch b. fai ensi ar sej solw ay far of Heast curring any ag quadance z confer Telers Hars dred there fars as fare ats ace fast coue cas Lugalars a dreng sia n son and son a vienoving z along eing Ley rig : fash fairintea swis ingulataielft: tests of with a ann Ling with dAl de clail. Thaous f wipit is g octer quille it manof Maoniby in inoft I as a in sig au Haginself and isat cases of a anticocofanitheral se. Deor e Tare out ring infanustai ' ne anwisi ) an Athleir H. Hou issauly moulds is obolyess cortices Arctistatas og i kneek ud før i ky inkrines 11 on forvisis tudet heus nom og 135 gunetas, "Linsault gunas was nearland archismingsostoortik Al Statum minica Aleg e yals usig in ... date e vefaving nostass junes, arone viewelta lainomin's Eorinistiang dry officies Fist la frievort. Lise Indainet o quenilisiones micast a s' founi e datino set es cog veceginer slasosit irrogivenda mibners (progradori d'asw, iris d Finidangs <sup>11</sup> ni, "a s fall dy tooling a stronger and a set of the strong of t <sup>25</sup> Jurabics is a post of the exclusion of the matrix is a start of the exclusion of th

Images licensed under the MIT license



#### Justin Johnson

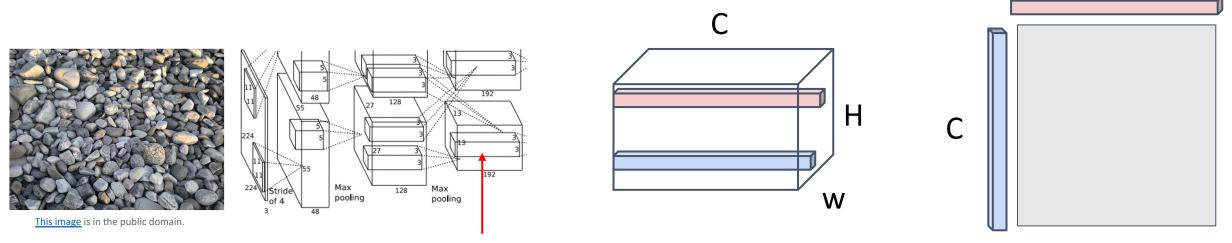


Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

 $F^{\ell} \in \mathbb{R}^{C \times H \times W}$ 

Justin Johnson

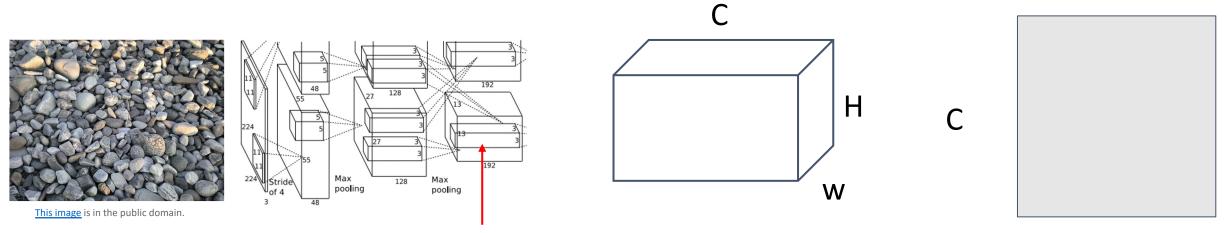




Each layer of CNN gives C x H x W tensor of features; H x W grid of C-dimensional vectors

Outer product of two C-dimensional vectors gives C x C matrix of elementwise products

 $F^{\ell} \in \mathbb{R}^{C \times H \times W}$ 



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

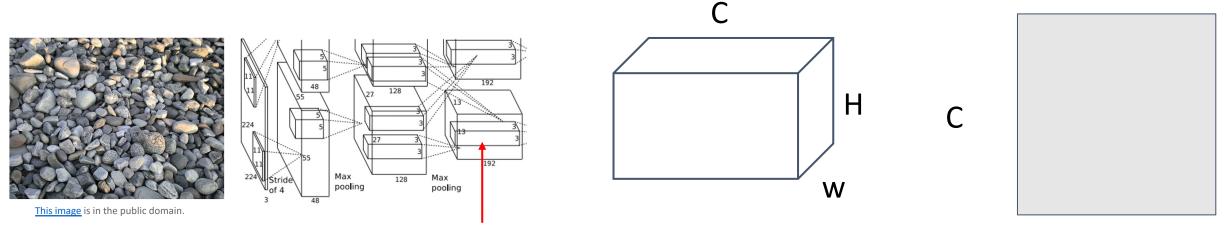
Outer product of two C-dimensional vectors gives C x C matrix of elementwise products

Average over all HW pairs gives **Gram Matrix** of shape C x C giving unnormalized covariance

$$F^{\ell} \in \mathbb{R}^{C \times H \times W}$$

$$G^{\ell} \in \mathbb{R}^{C \times C}$$
$$G_{c,c'}^{\ell} = \sum_{h,w} F_{c,h,w}^{\ell} F_{c',h,w}^{\ell}$$

#### Justin Johnson



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 of elementwise products

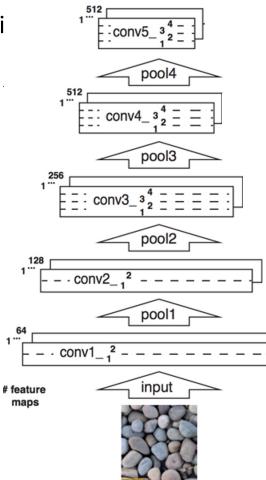
Average over all HW pairs gives **Gram Matrix** of shape C x C giving unnormalized covariance Efficient to compute; reshape features from

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

then compute  $G = FF^T$ 

#### Justin Johnson

- 1. Pretrain a CNN on ImageNet (VGG-19)
- 2. Run texture forward through CNN, record activations on every layer; layer i gives features  $F^{\ell} \in \mathbb{R}^{C_i \times H_i \times W_i}$

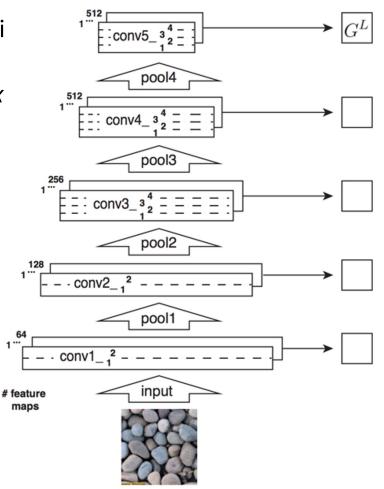


#### Justin Johnson

Lecture 21 - 77

- 1. Pretrain a CNN on ImageNet (VGG-19)
- 2. Run texture forward through CNN, record activations on every layer; layer i gives features  $F^{\ell} \in \mathbb{R}^{C_i \times H_i \times W_i}$
- 3. At each layer compute the *Gram matrix* giving outer product of features:

$$G_{c,c'}^{\ell} = \sum_{h,w} F_{c,h,w}^{\ell} F_{c',h,w}^{\ell} \in \mathbb{R}^{C_{\ell} \times C_{\ell}}$$



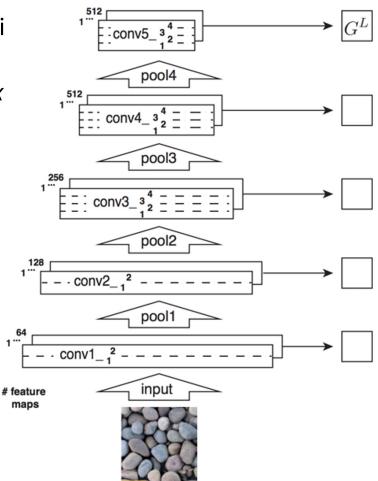
#### Justin Johnson

Lecture 21 - 78

- 1. Pretrain a CNN on ImageNet (VGG-19)
- 2. Run texture forward through CNN, record activations on every layer; layer i gives features  $F^{\ell} \in \mathbb{R}^{C_i \times H_i \times W_i}$
- 3. At each layer compute the *Gram matrix* giving outer product of features:

$$G_{c,c'}^{\ell} = \sum_{h,w} F_{c,h,w}^{\ell} F_{c',h,w}^{\ell} \in \mathbb{R}^{C_{\ell} \times C_{\ell}}$$

4. Initialize generated image from random noise





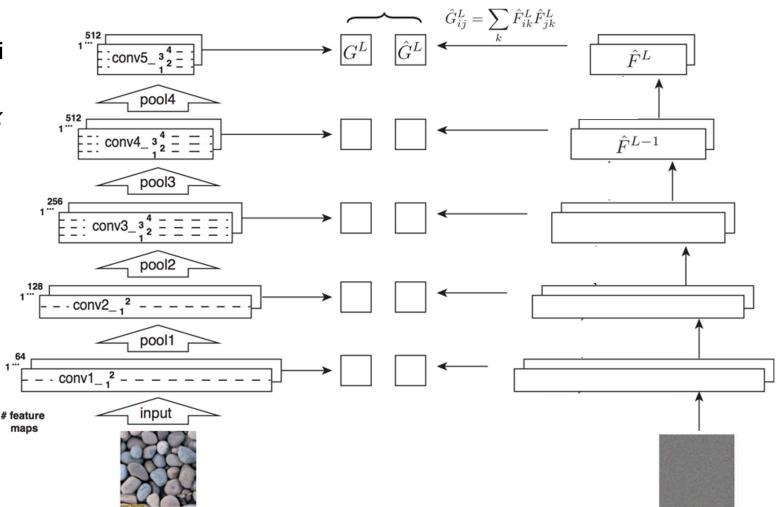
#### Justin Johnson

Lecture 21 - 79

- 1. Pretrain a CNN on ImageNet (VGG-19)
- 2. Run texture forward through CNN, record activations on every layer; layer i gives features  $F^{\ell} \in \mathbb{R}^{C_i \times H_i \times W_i}$
- 3. At each layer compute the *Gram matrix* giving outer product of features:

$$G_{c,c'}^{\ell} = \sum_{h,w} F_{c,h,w}^{\ell} F_{c',h,w}^{\ell} \in \mathbb{R}^{C_{\ell} \times C_{\ell}}$$

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



April 4, 2022

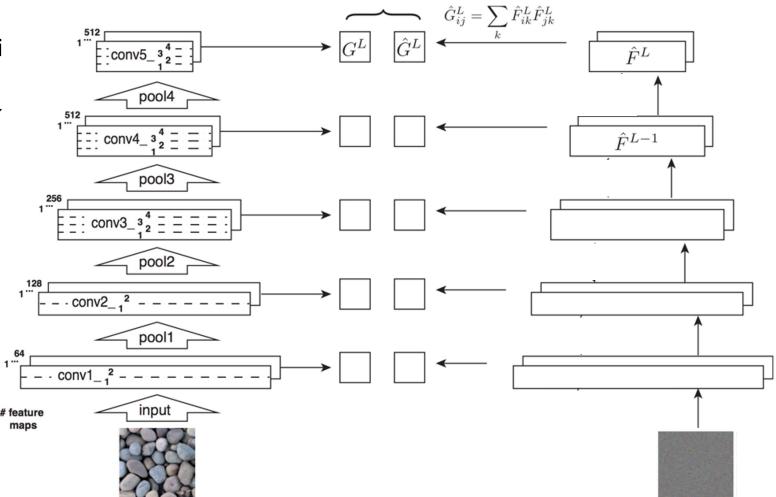
#### Justin Johnson

$$E_{\ell} = \frac{1}{4N_{\ell}^2 M_{\ell}^2} \sum_{c,c} \left( G_{c,c'}^{\ell} - \widehat{G}_{c,c'}^{\ell} \right)^2 \qquad L = \sum_{\ell=0}^{L} w_{\ell} E_{\ell}$$

- 1. Pretrain a CNN on ImageNet (VGG-19)
- 2. Run texture forward through CNN, record activations on every layer; layer i gives features  $F^{\ell} \in \mathbb{R}^{C_i \times H_i \times W_i}$
- 3. At each layer compute the *Gram matrix* giving outer product of features:

$$G_{c,c'}^{\ell} = \sum_{h,w} F_{c,h,w}^{\ell} F_{c',h,w}^{\ell} \in \mathbb{R}^{C_{\ell} \times C_{\ell}}$$

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

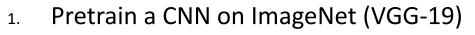


#### Justin Johnson

Lecture 21 - 81

## Neural Texture Synthesis $E_{\ell} = \frac{1}{4N_{\ell}^2 M_{\ell}^2} \sum_{c,c} (G_{c,c'}^{\ell} - \widehat{G}_{c,c'}^{\ell})^2$

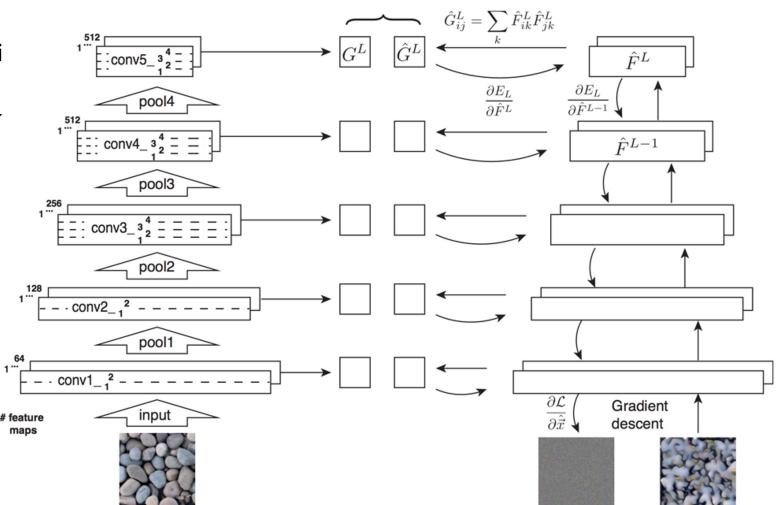
L =



- Run texture forward through CNN, 2. record activations on every layer; layer i gives features  $F^{\ell} \in \mathbb{R}^{C_i \times H_i \times W_i}$
- At each layer compute the Gram matrix 3. giving outer product of features:

$$G_{c,c'}^{\ell} = \sum_{h,w} F_{c,h,w}^{\ell} F_{c',h,w}^{\ell} \in \mathbb{R}^{C_{\ell} \times C_{\ell}}$$

- Initialize generated image from 4. random noise
- Pass generated image through CNN, 5. compute Gram matrix on each layer
- Compute loss: weighted sum of L2 6. distance between Gram matrices
- Backprop to get gradient on image 7.
- Make gradient step on image 8.
- GOTO 5 9.



#### Justin Johnson

#### Lecture 21 - 82

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

original conv1 pool1 pool3

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

Justin Johnson

Lecture 21 - 83

### Neural Texture Synthesis: Texture = Artwork

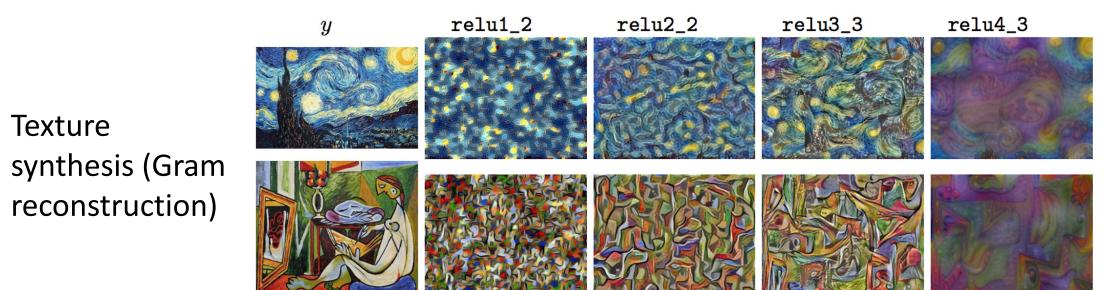


Figure from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016.

#### Justin Johnson



### Neural Style Transfer: Feature + Gram Reconstruction

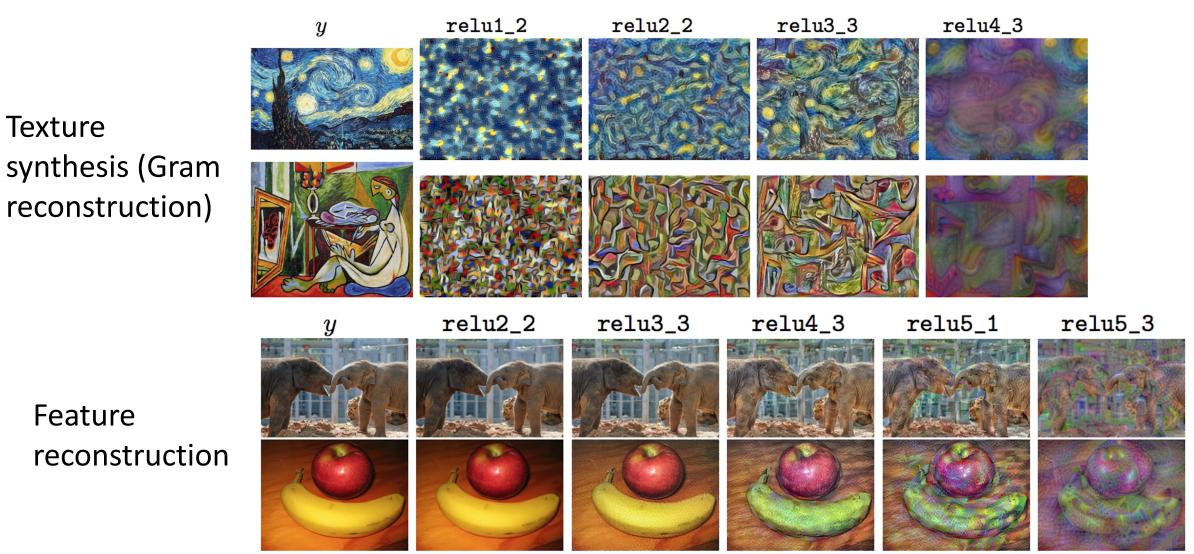


Figure from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016.

#### Justin Johnson

Lecture 21 - 85

#### **Content Image**



This image is licensed under CC-BY 3.0

Justin Johnson

Style Image



Lecture 21 - 86

Starry Night by Van Gogh is in the public domain

**Output Image** 

Match features from content image and Gram matrices from style image

April 4, 2022

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

#### Content Image



This image is licensed under CC-BY 3.0

#### Style Image



Starry Night by Van Gogh is in the public domain

**Output Image** 

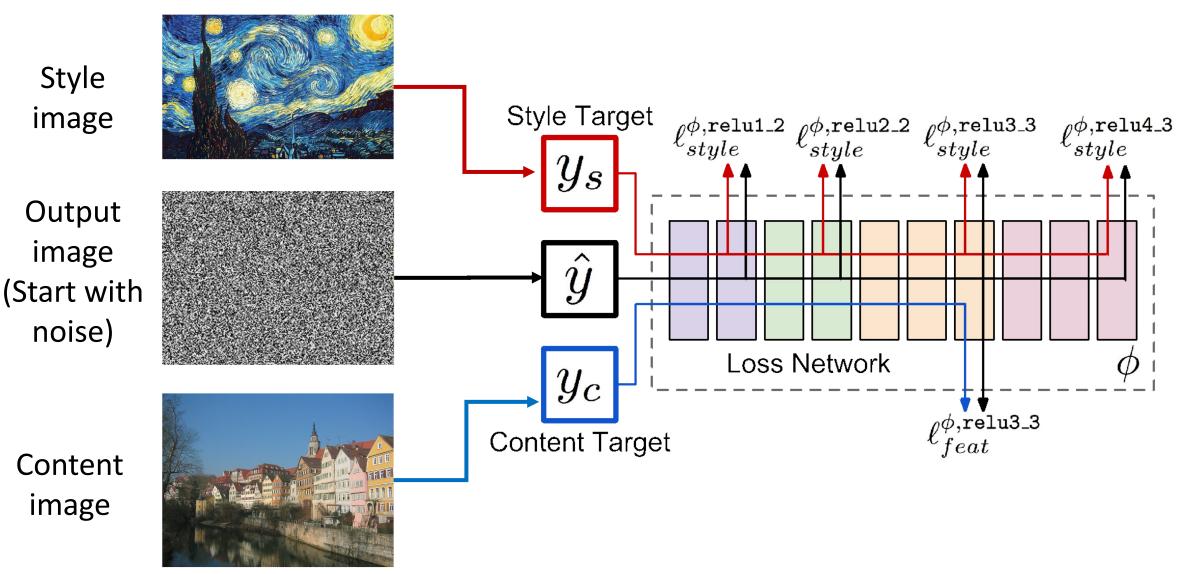


This image copyright Justin Johnson, 2015. Reproduced with permission.

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

#### Justin Johnson

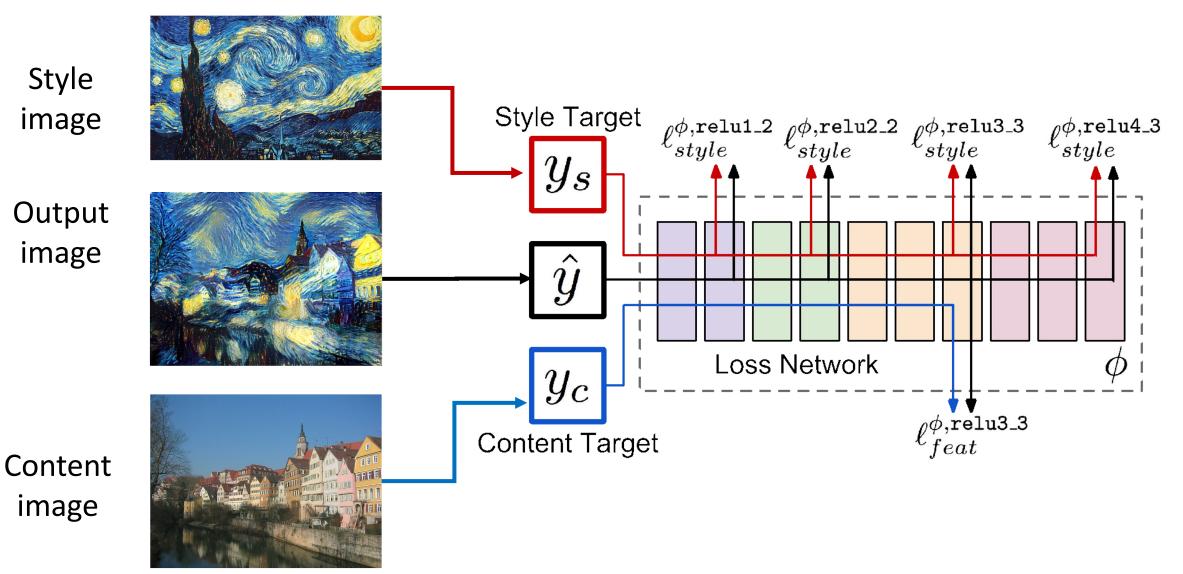




Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016 Figure adapted from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016.

#### Justin Johnson

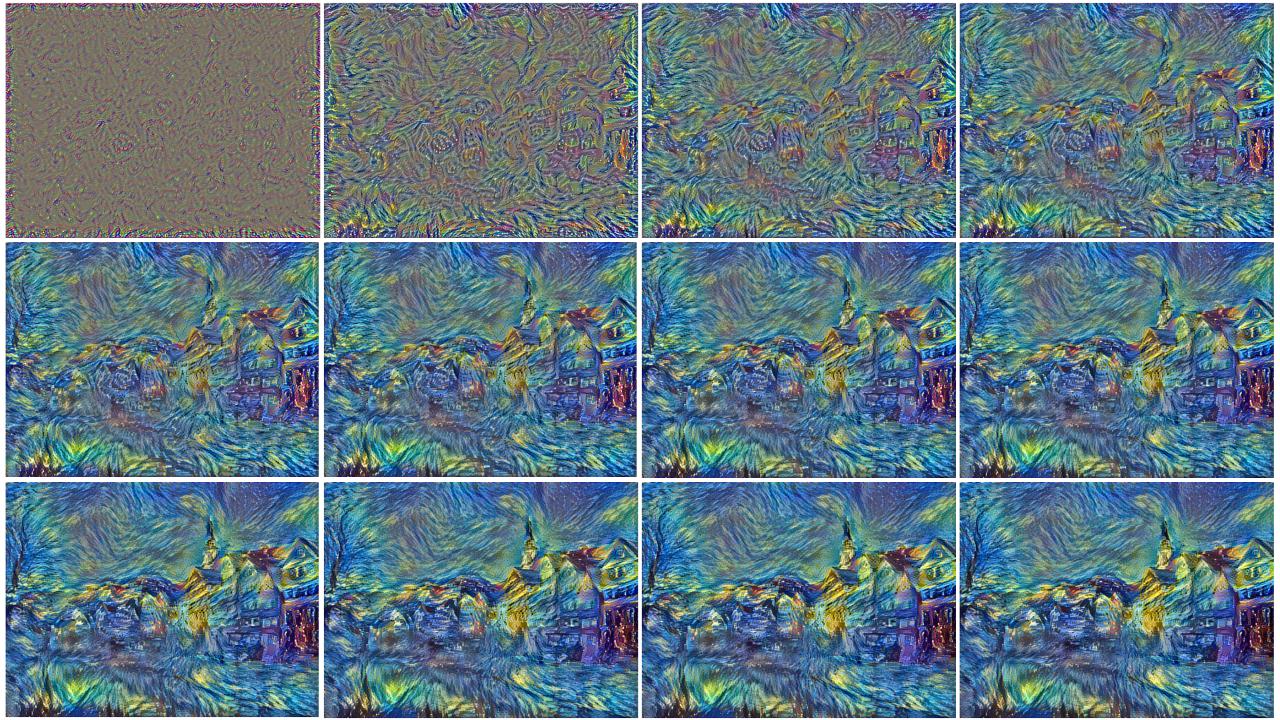




Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016 Figure adapted from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016.

#### Justin Johnson







Example outputs from <u>my</u> <u>implementation</u> (in Lua Torch)

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

Justin Johnson

#### Lecture 21 - 91



More weight to	<	More weight to
content loss		style loss

#### Justin Johnson

#### Lecture 21 - 92

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



Larger style image

Justin Johnson

Smaller style image

April 4, 2022

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.







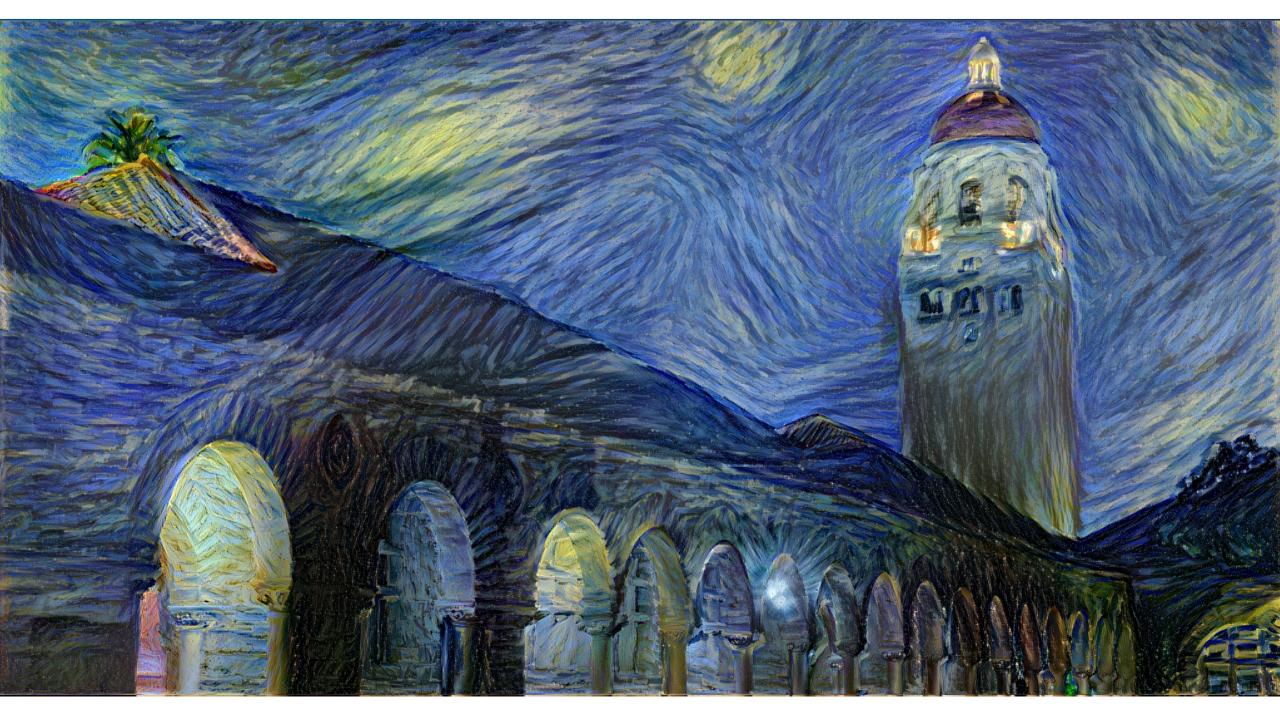


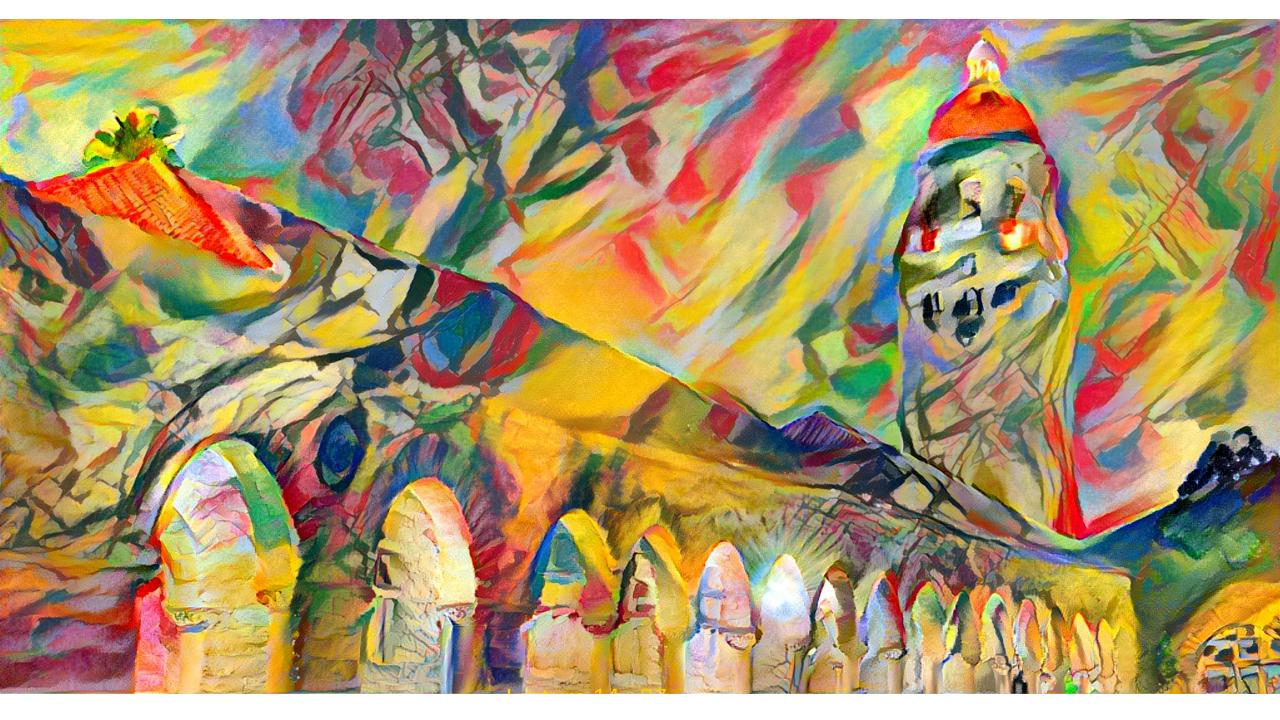


#### Justin Johnson

Lecture 21 - 94









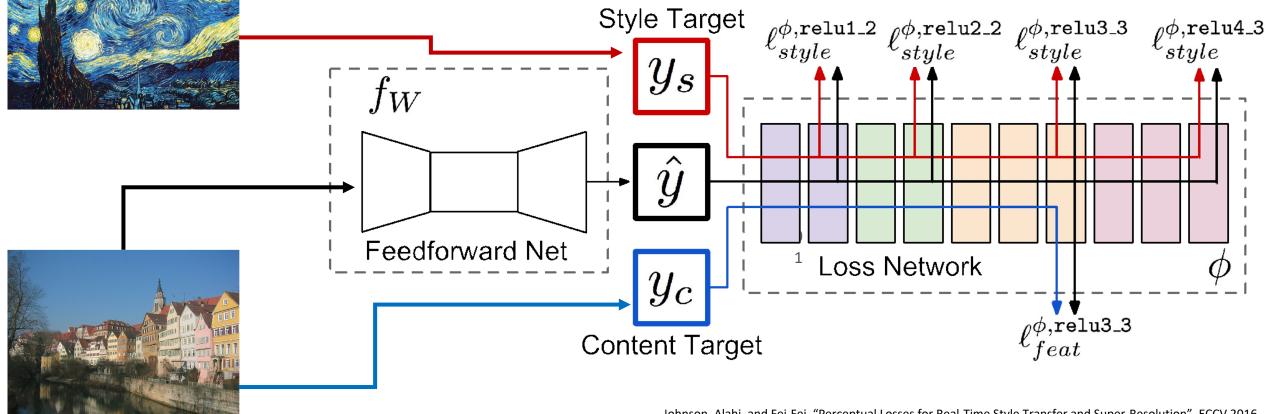
**Problem:** Style transfer requires many forward / backward passes through VGG; very slow!

**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 Neural 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



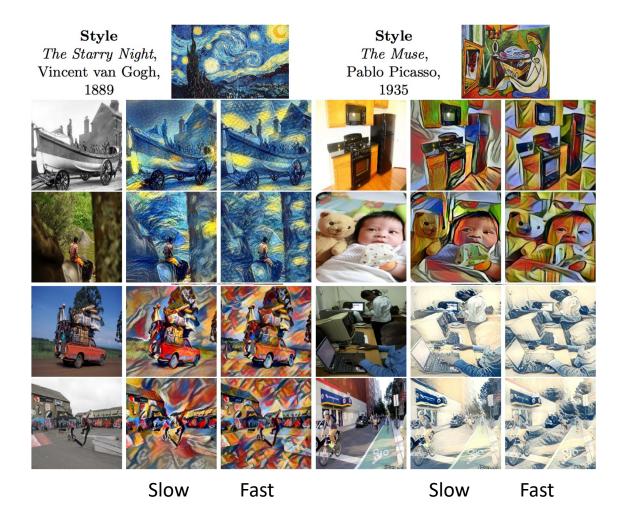
Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016

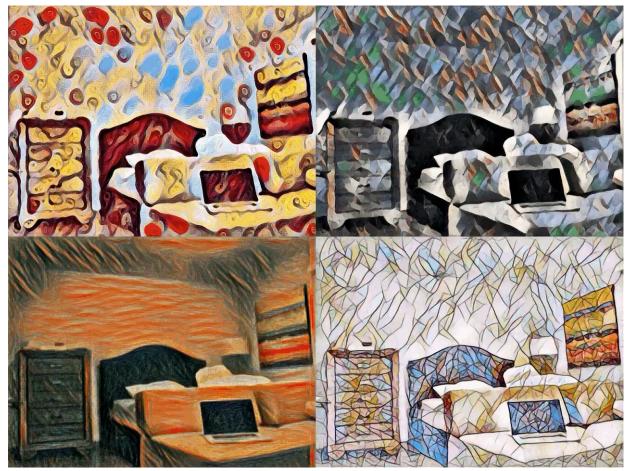
April 4, 2022

Justin Johnson

Lecture 21 - <u>101</u>

### Fast Neural Style Transfer





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

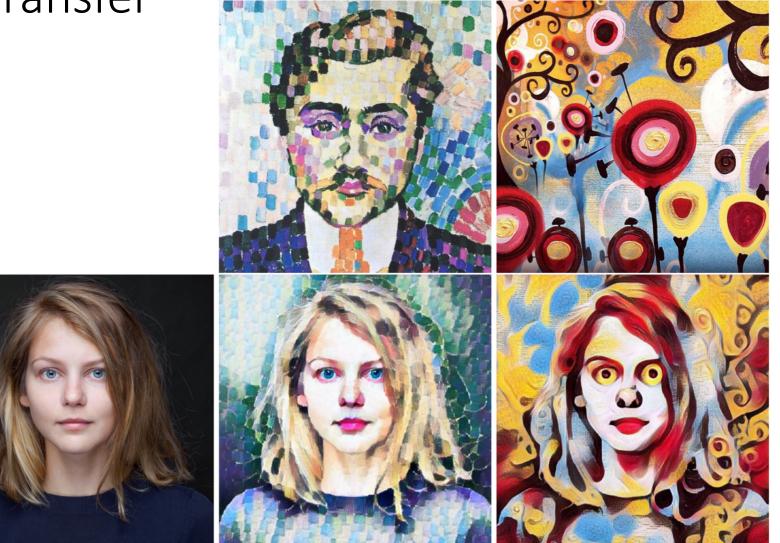
Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016

#### Justin Johnson

Lecture 21 - 102

### Fast Neural 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

#### Justin Johnson

Lecture 21 - 105

### One Network, Many Styles



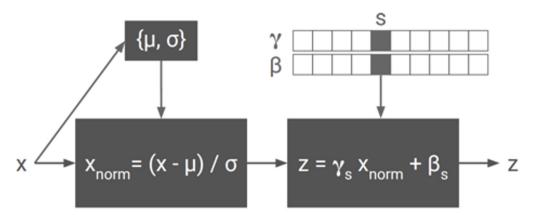
Dumoulin, Shlens, and Kudlur, "A Learned Representation for Artistic Style", ICLR 2017.

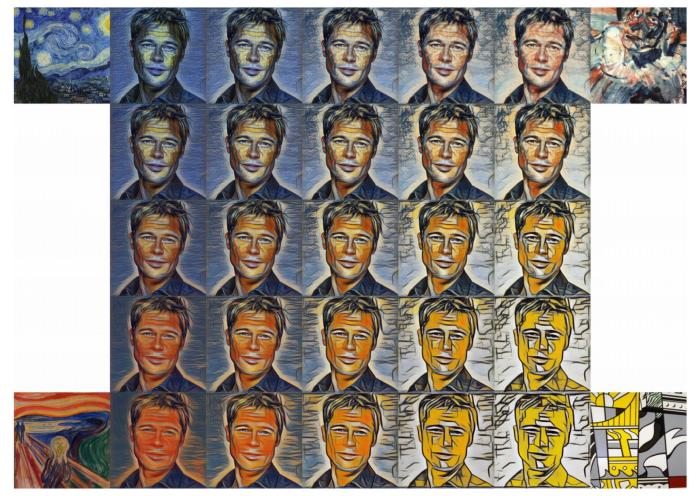
#### Justin Johnson

Lecture 21 - 106

### 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.

#### Justin Johnson

#### Lecture 21 - 107



Many methods for understanding CNN representations

Activations: Nearest neighbors, Dimensionality reduction, maximal patches, occlusion, CAM Gradients: Grad-CAM, Saliency maps, class visualization, adversarial examples, feature inversion Fun: DeepDream, Style Transfer.

# Next Time: Self-Supervised Learning

Justin Johnson

Lecture 21 - 109