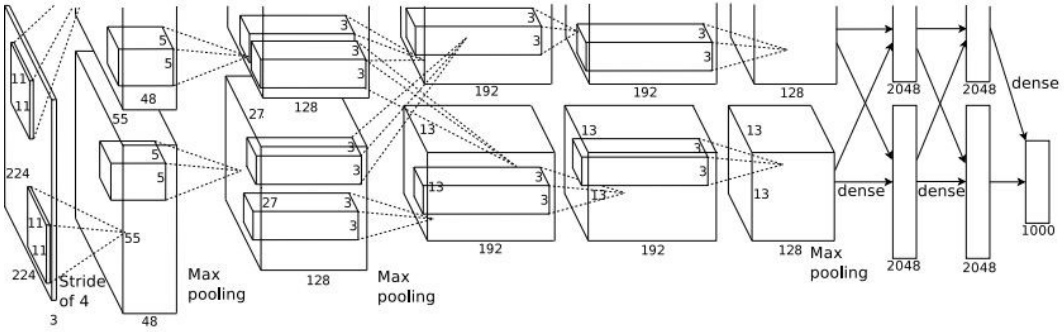
CSE 152: Computer Vision

Hao Su

Lecture 11: Visualizing Networks, Fun Applications

Credit: Stanford CS231n, L13

What's going on inside ConvNets?



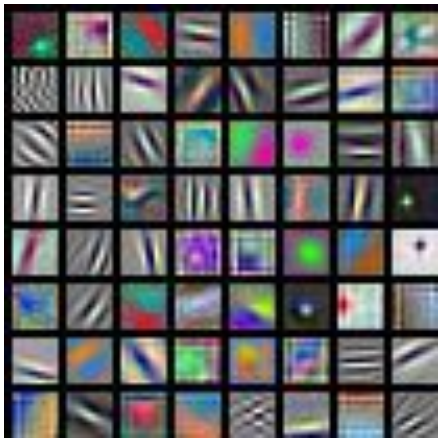
Input Image:
3 x 224 x 224

Class Scores:
1000 numbers

What are the intermediate features looking for?

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

First Layer: Visualize Filters

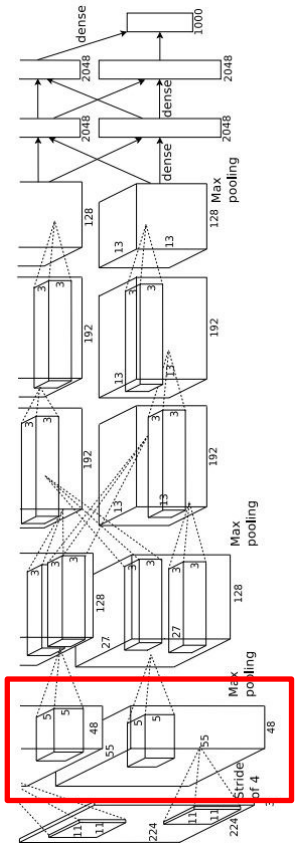


AlexNet:
64 x 3 x 11 x 11

ResNet-18: 64 x 3 x 7 x 7

ResNet-101: 64 x 3 x 7 x 7

DenseNet-121: 64 x 3 x 7 x 7



Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv 2014

He et al, “Deep Residual Learning for Image Recognition”, CVPR 2016

Huang et al, “Densely Connected Convolutional Networks”, CVPR 2017

Visualize the filters/kernels (raw weights)

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

(these are taken from ConvNetJS CIFAR-10 demo)

Weights:

layer 1 weights

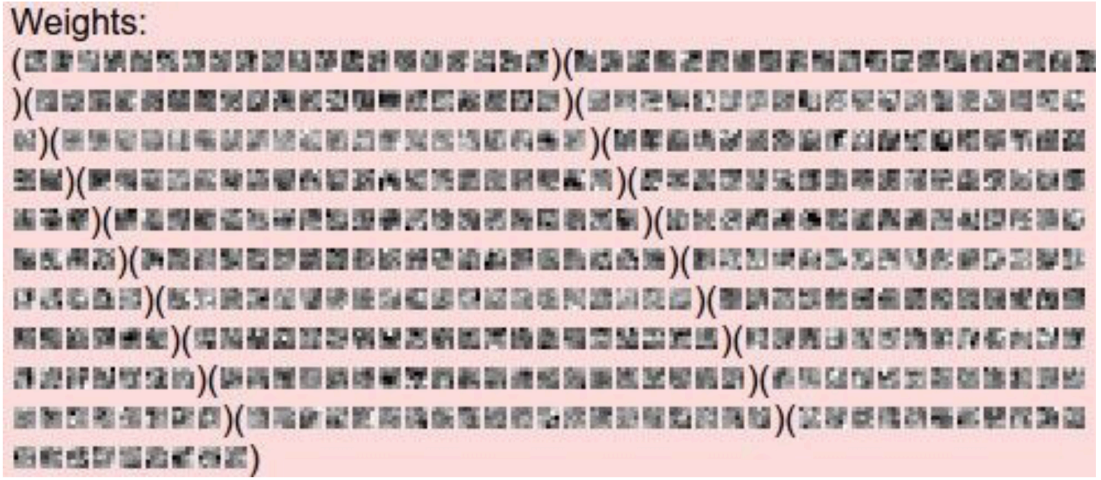
16 x 3 x 7 x 7

Weights:

layer 2 weights

20 x 16 x 7 x 7

Weights:

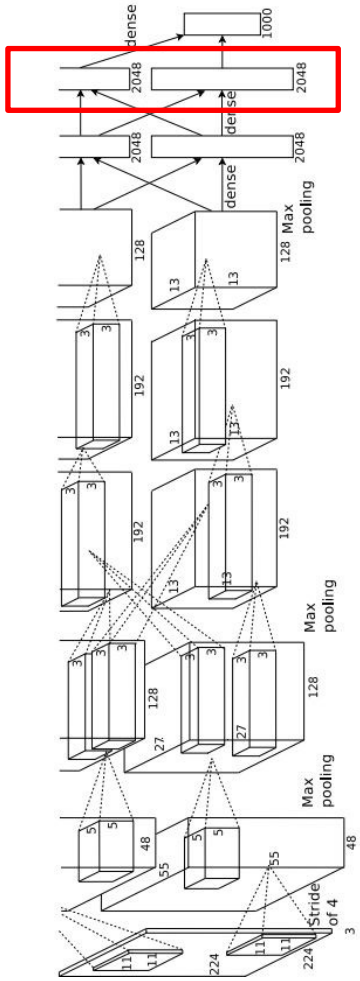


layer 3 weights

20 x 20 x 7 x 7

Last Layer

FC7 layer

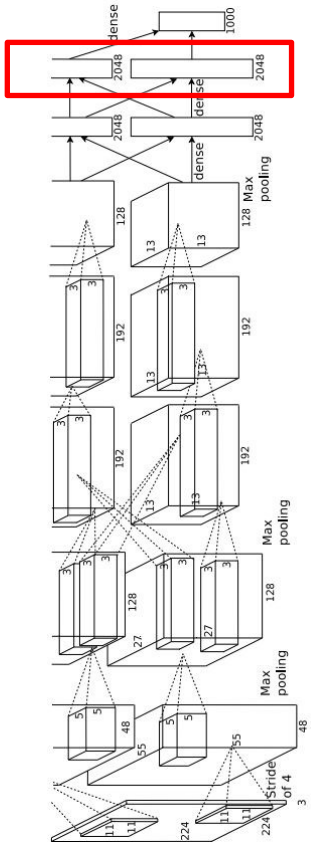


4096-dimensional feature vector for an image (layer immediately before the classifier)

Run the network on many images, collect the feature vectors

Last Layer: Nearest Neighbors

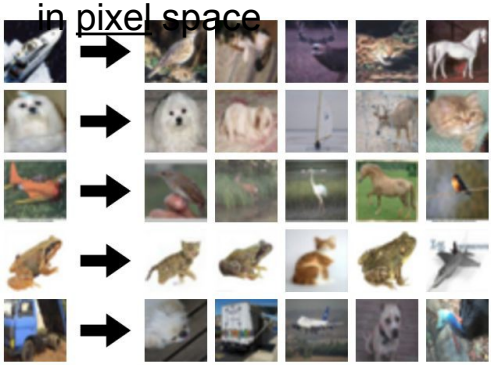
4096-dim vector



Test image

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

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



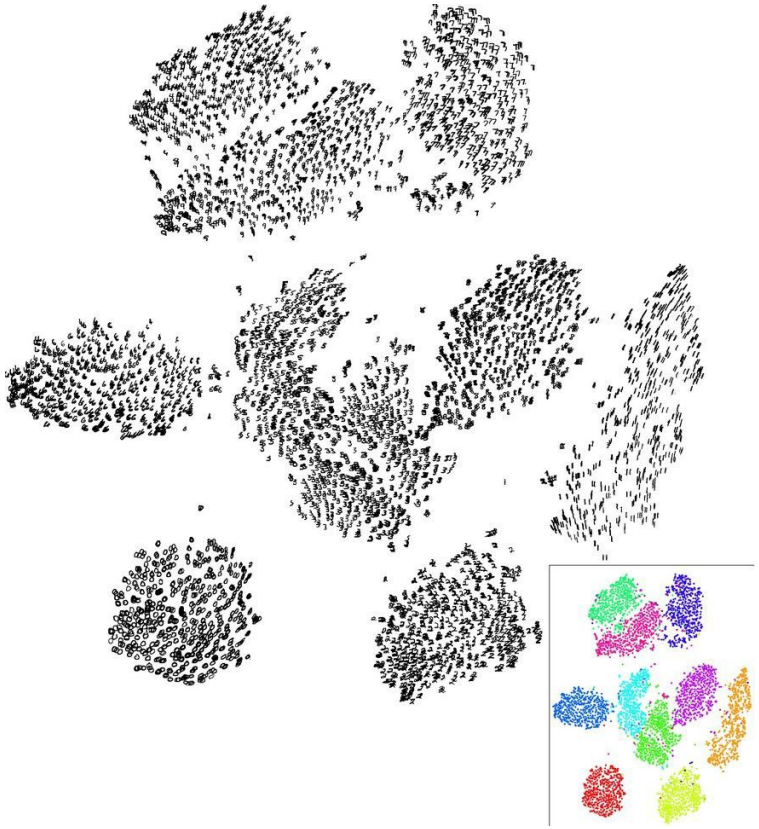
Krizhevsky et al, “ImageNet Classification with Deep Convolutional Neural Networks”, NIPS 2012. Figures reproduced with permission.

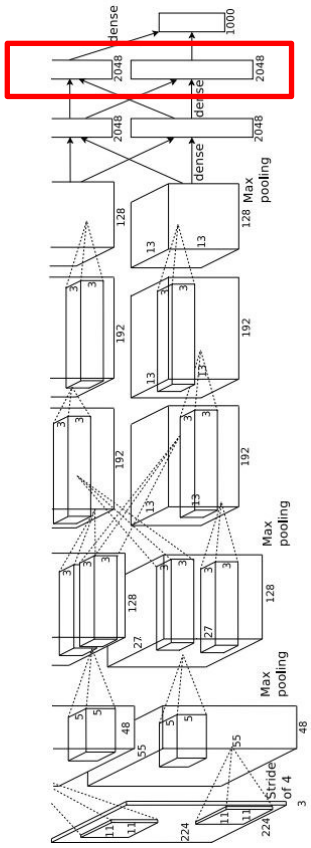
Last Layer: Dimensionality Reduction

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**

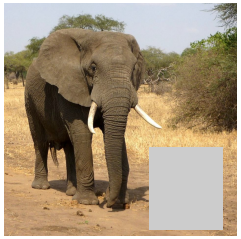


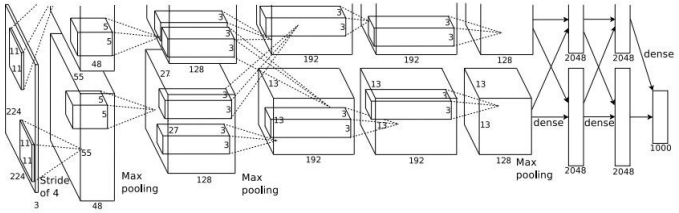


Van der Maaten and Hinton, “Visualizing Data using t-SNE”, JMLR 2008

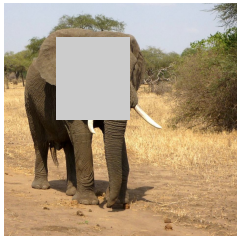
Which pixels matter: Saliency vs Occlusion

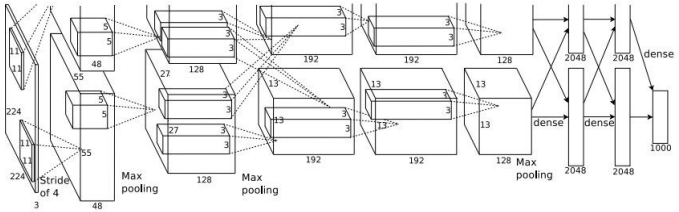
Mask part of the image before feeding to CNN, check how much predicted probabilities change





P(elephant) = 0.95



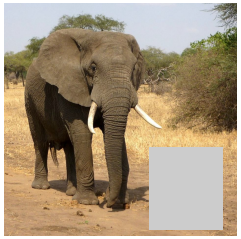


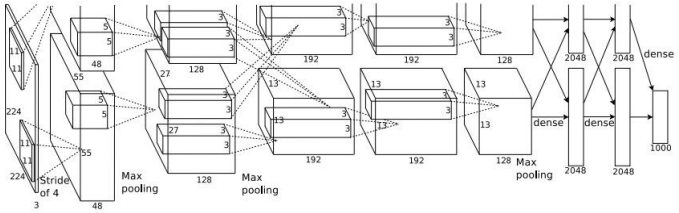
P(elephant) = 0.75

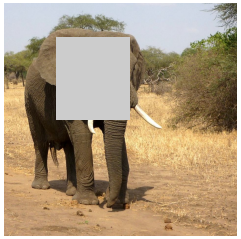
Zeiler and Fergus, “Visualizing and Understanding Convolutional Networks”, ECCV 2014

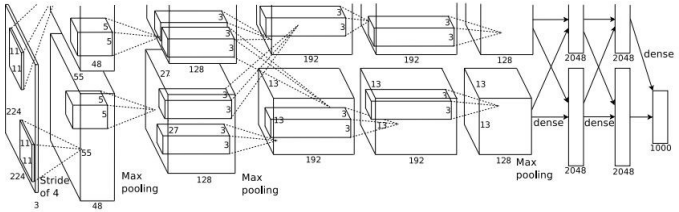
Which pixels matter: Saliency vs Occlusion

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

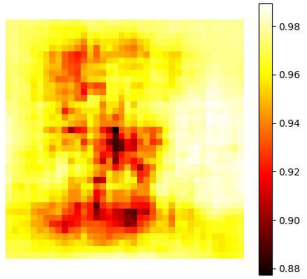




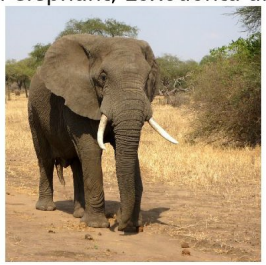


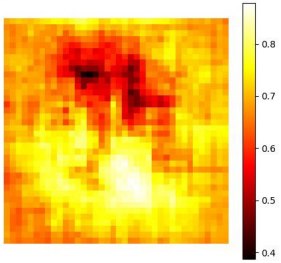


schooner

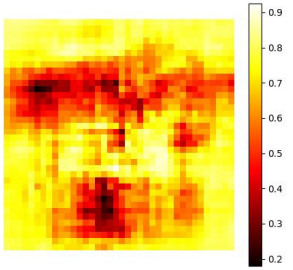


African elephant, Loxodonta africana





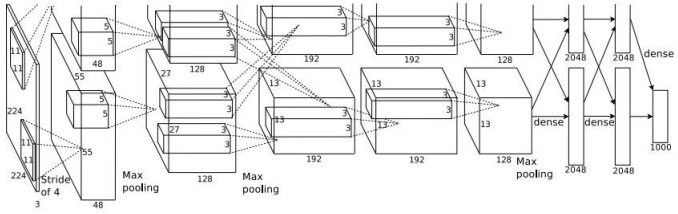
go-kart



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

Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities

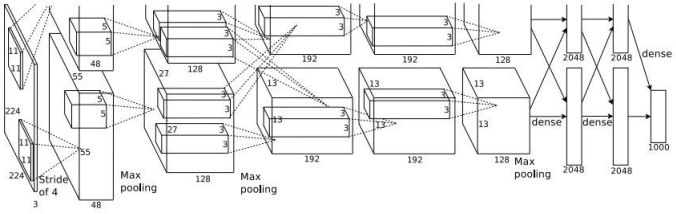


Dog

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

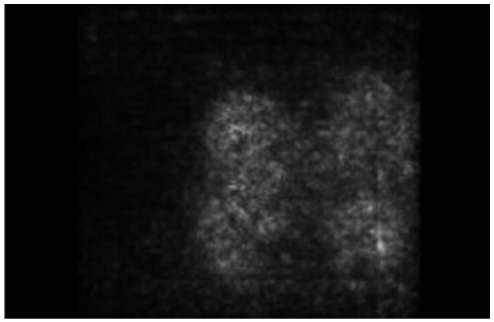
Which pixels matter: Saliency via Backprop

Forward pass: Compute probabilities



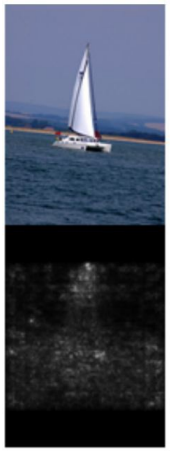
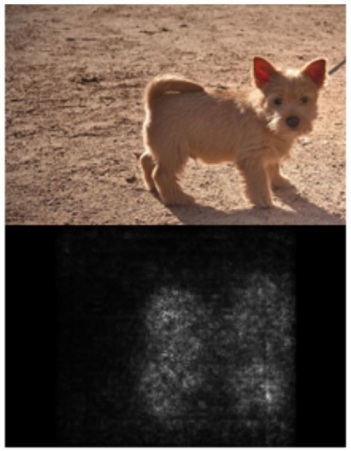
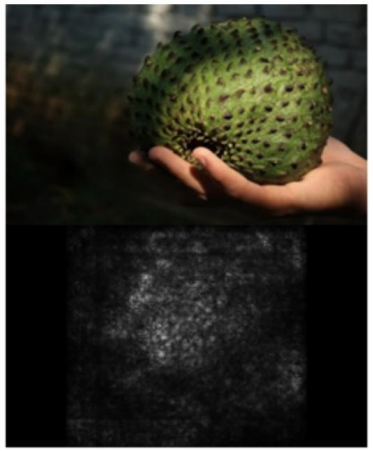
Dog

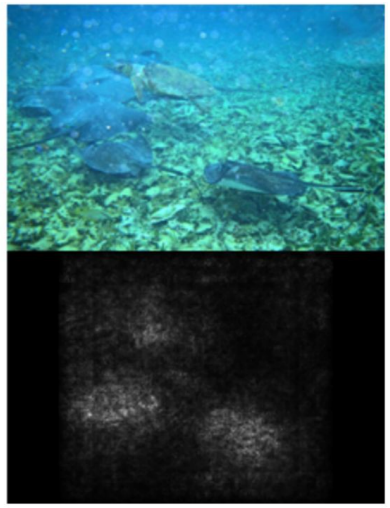
Compute gradient of (unnormalized) class score with respect to image pixels, take absolute value and max over RGB channels



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

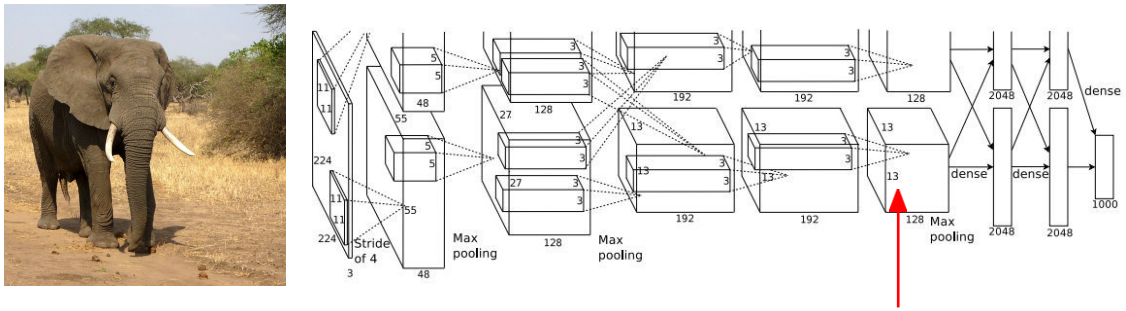
Saliency Maps



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

Intermediate Features via (guided) backprop



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

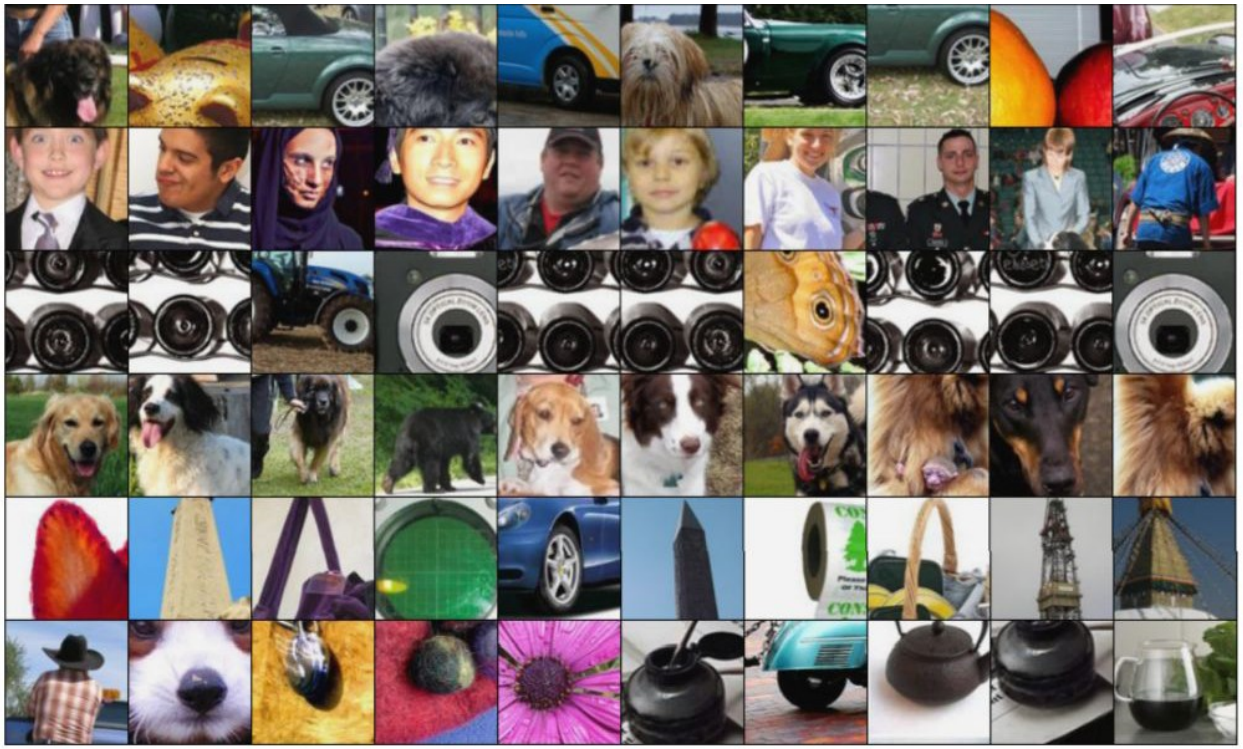
Compute gradient of neuron value with respect to image pixels

Intermediate features via (guided) backprop

Maximally activating patches
(Each row is a different neuron)

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

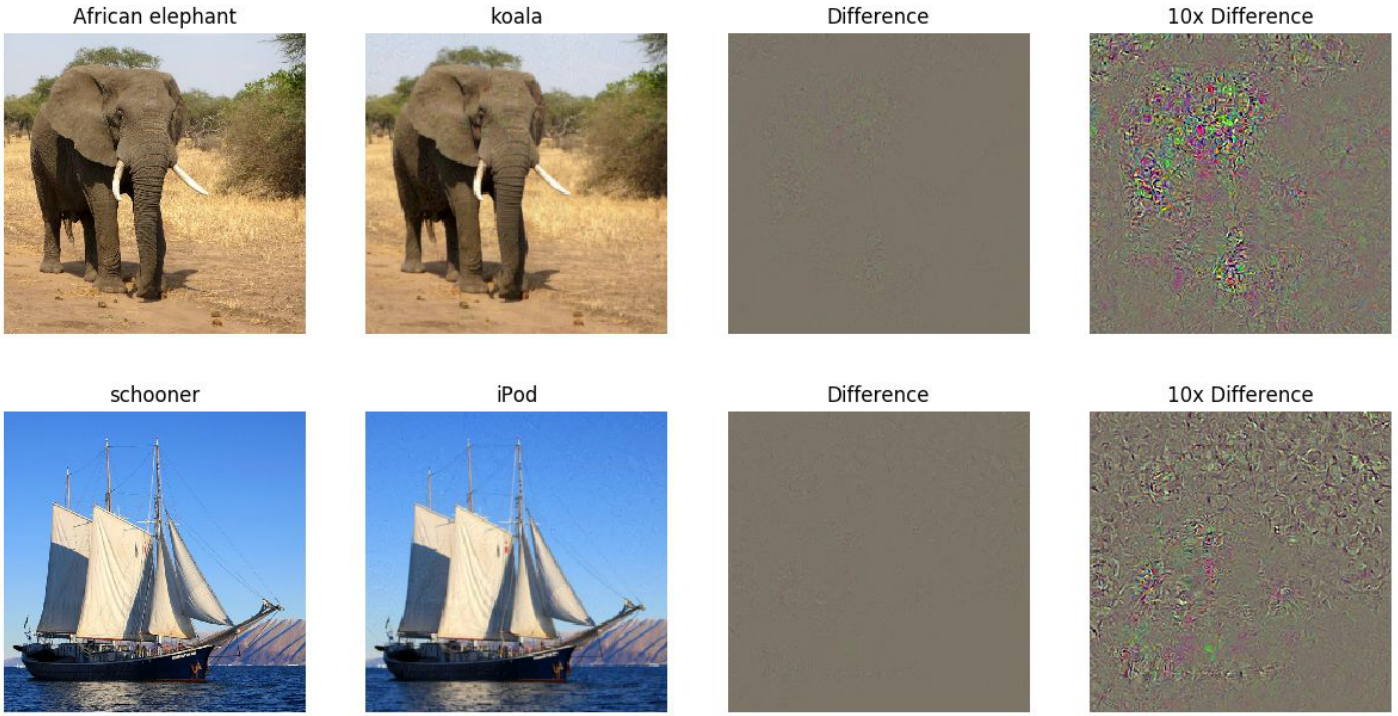
Intermediate features via (guided) backprop



Maximally activating patches
(Each row is a different neuron)

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

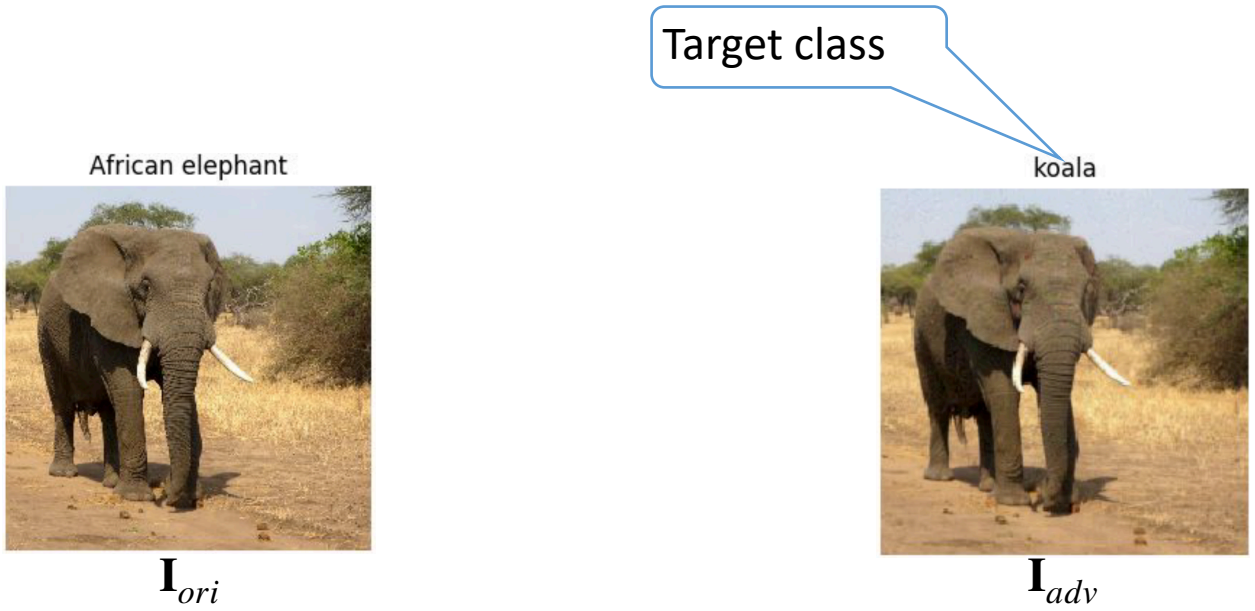
Fooling Images / Adversarial Examples



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

Optimization Formulation



$Score_c(\mathbf{I};\theta)$: the confidence score of an image belonging to class c, using a network of parameters θ

Attack: Modify the image $\mathbf{I}$ to increase $Score_{target\ class}(\mathbf{I};\theta)$

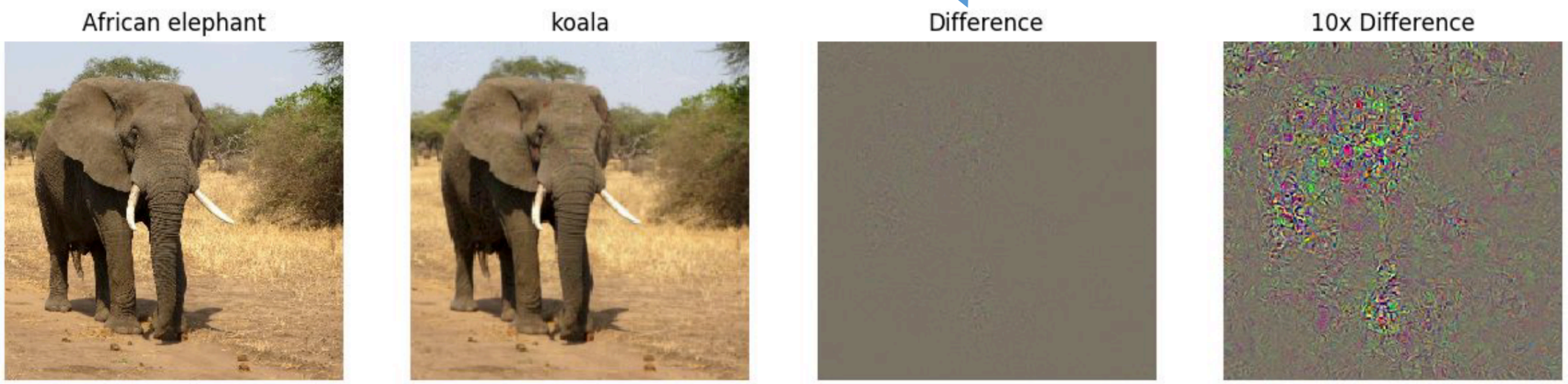
$$\underset{\mathbf{I}_{adv}}{\text{maximize}} \quad Score_{target\ class}(\mathbf{I}_{adv})$$

$$\text{subject to} \quad \|\mathbf{I}_{adv} - \mathbf{I}_{ori}\| \leq \epsilon$$

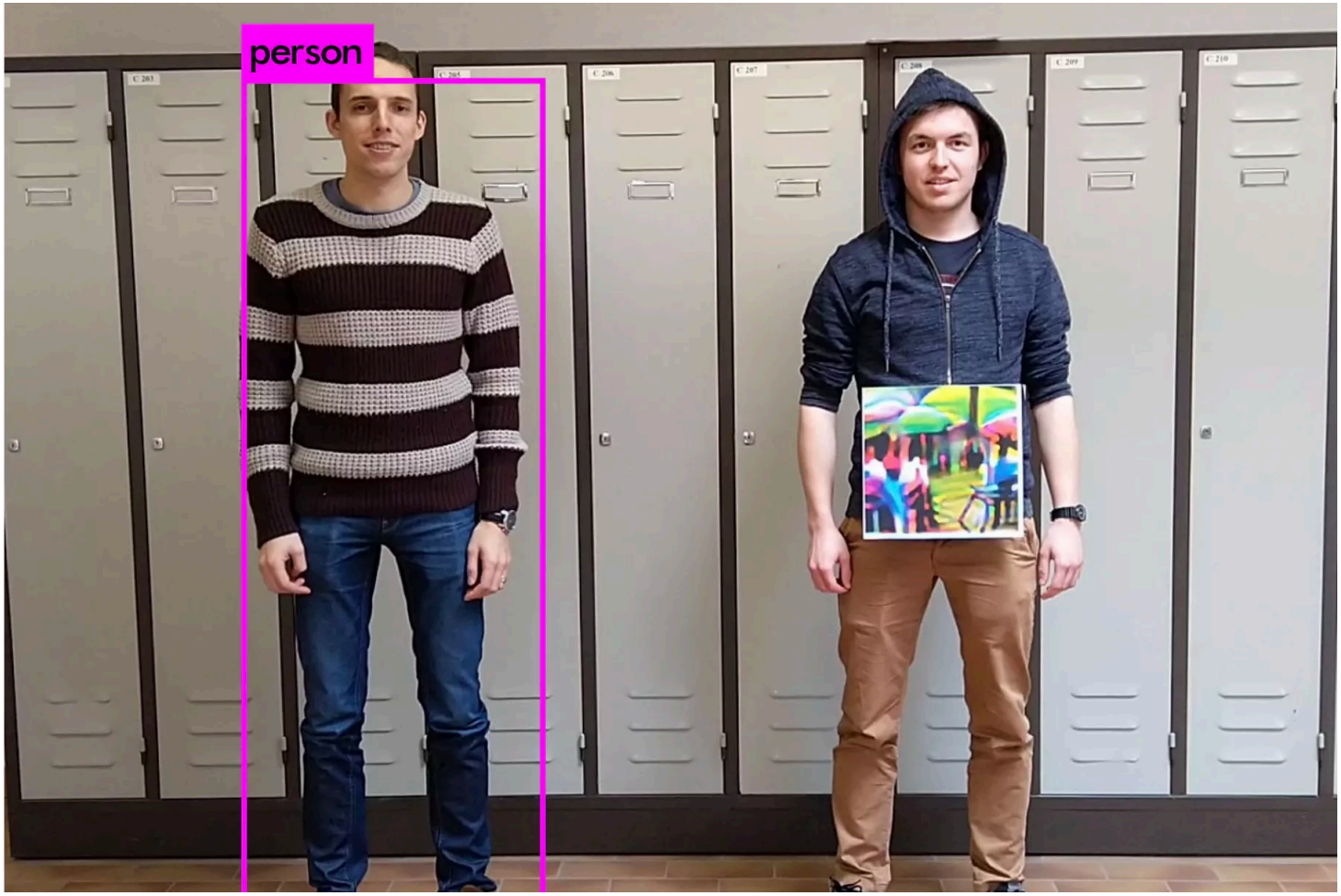
Gradient-based Attack

Fast Gradient Sign Method:

$$\mathbf{I}^{adv} = \mathbf{I} + \epsilon \text{sign}(\nabla_{\mathbf{I}} Score_{target\ class}(\mathbf{I}))$$



Patch-based Attack (Spatially Localized)



Dangerous!

(a) (b) (c) (d) (e)

Neural Style Transfer

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

Neural Style Transfer

Content Image

This image is licensed under CC-BY 3.0

+

Style Image

Starry Night by Van Gogh is in the public domain

=

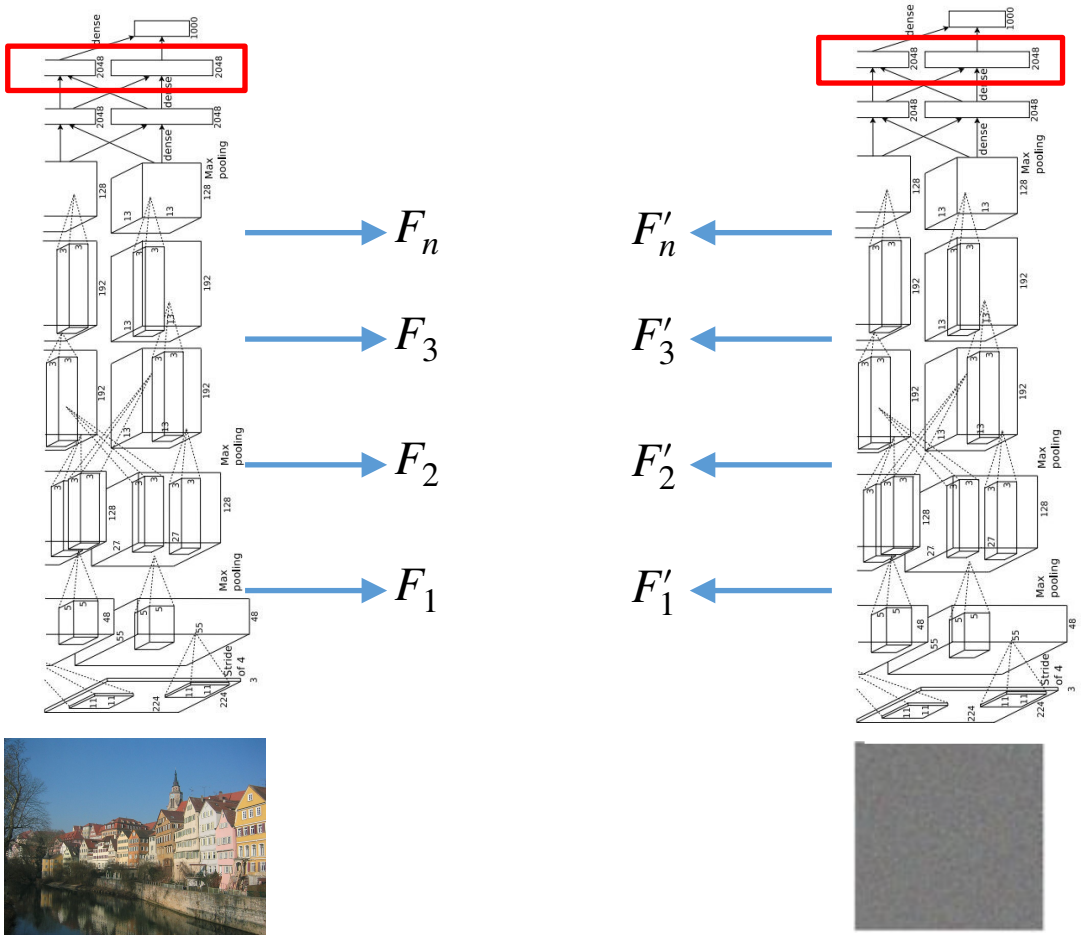
Style Transfer!

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

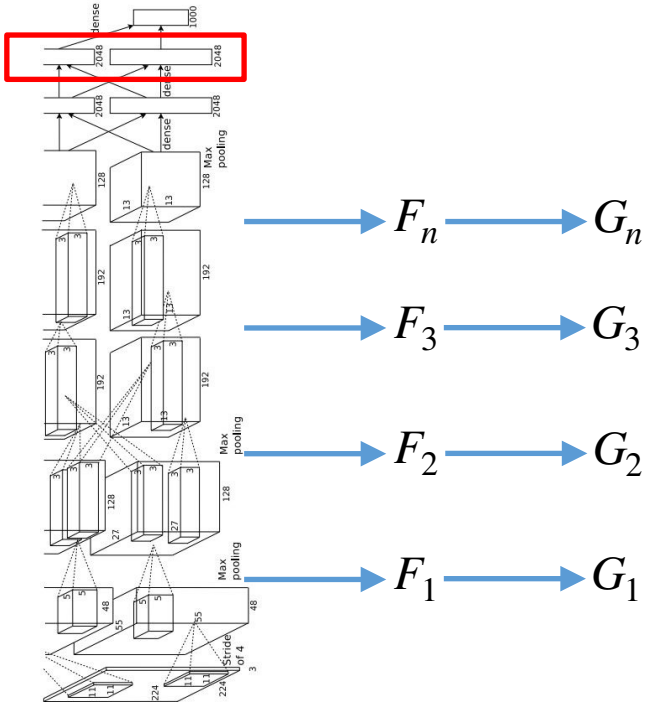
Gatys, Ecker, and Bethge, “Texture Synthesis Using Convolutional Neural Networks”, NIPS 2015

Content Discrepancy Loss

$$\mathscr{L}_{content}(\mathbf{I}) = \|\mathbf{F}(\mathbf{I}_{ref}) - \mathbf{F}(\mathbf{I})\|^2$$



Style Discrepancy Loss

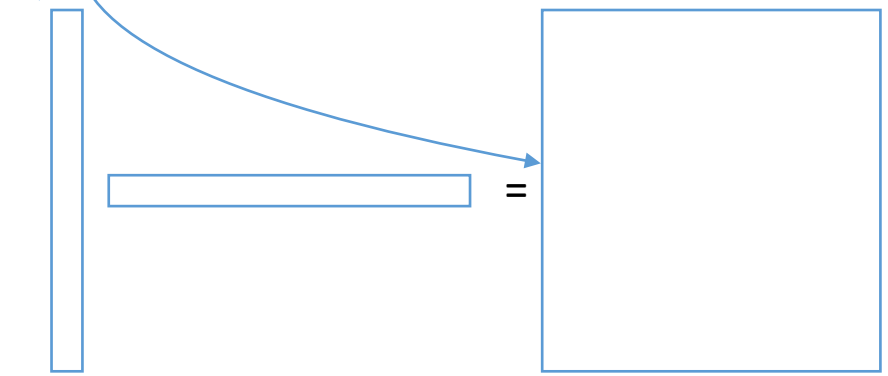


F: the centered feature tensor at some layer (as in PCA)

G: covariance matrix, a.k.a Gram matrix (as in PCA)

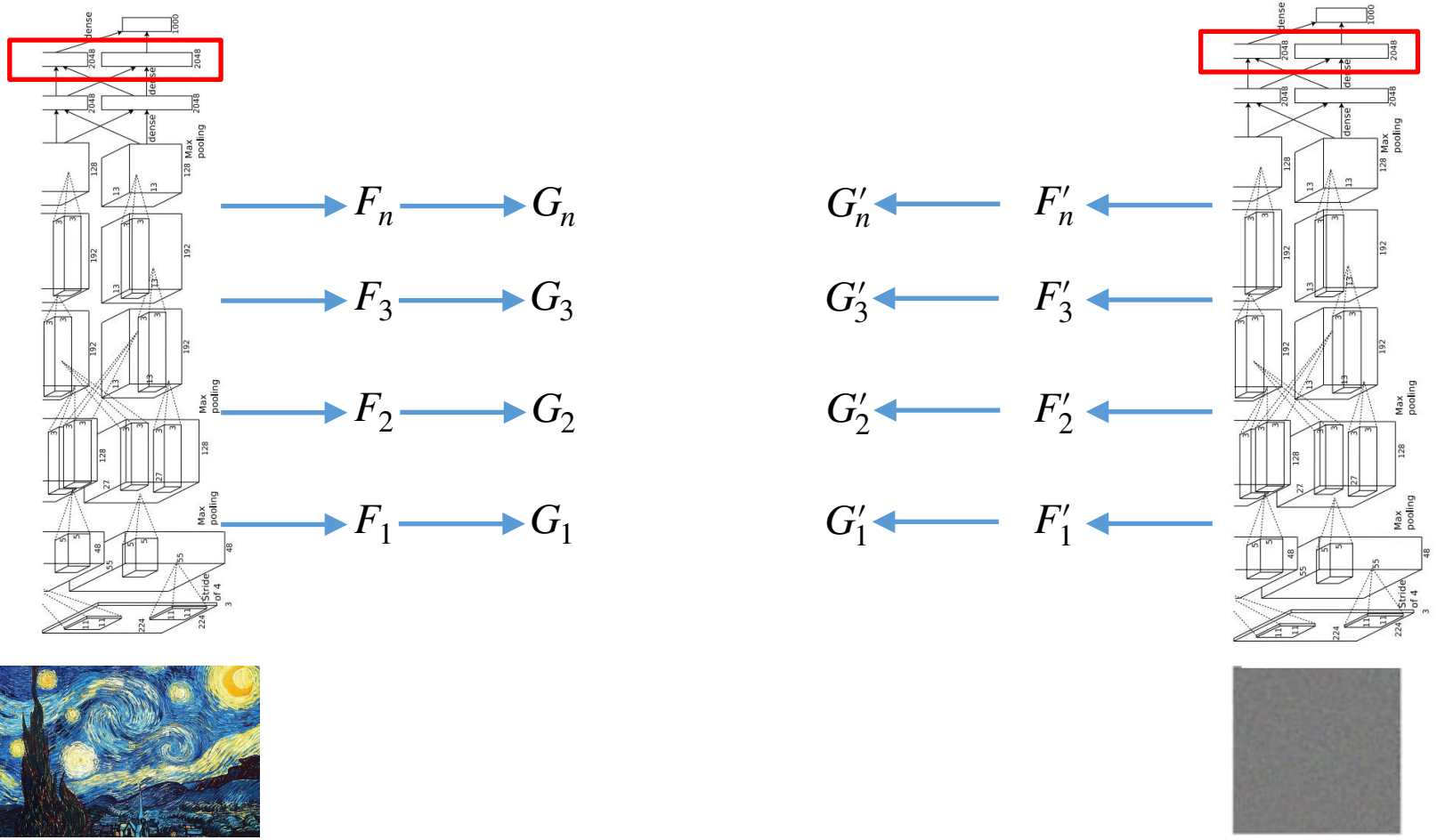
F=F.flatten().reshape(-1, 1)

G=F @ F.T

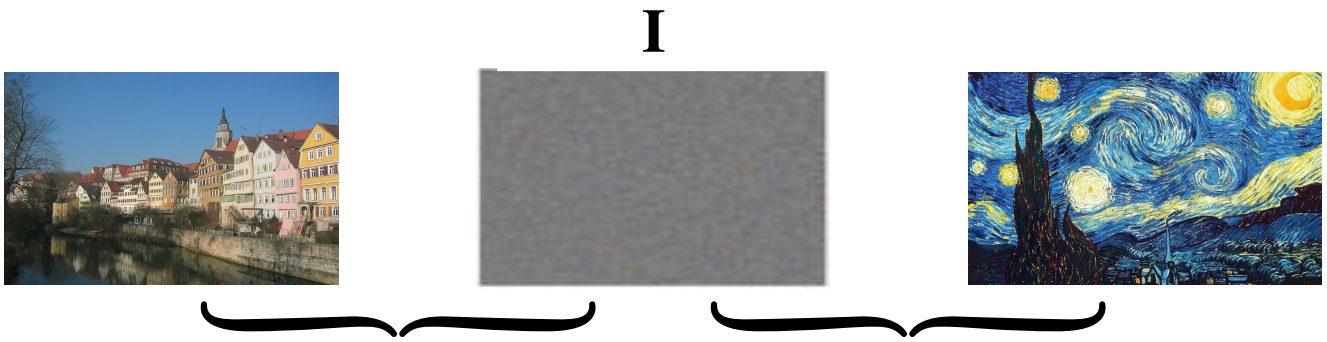


Style Discrepancy Loss

$$\mathscr{L}_{style}(\mathbf{I}) = \|\mathbf{G}(\mathbf{I}_{ref}) - \mathbf{G}(\mathbf{I})\|^2$$

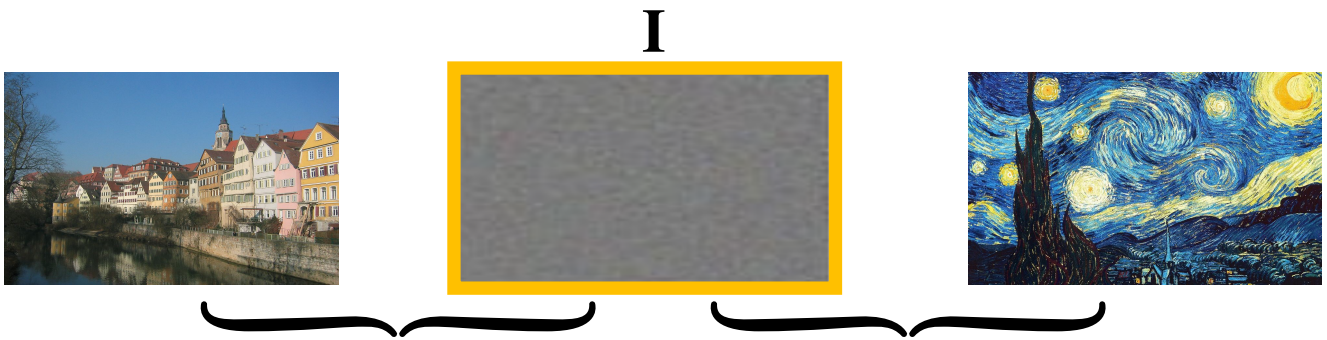


Total Loss



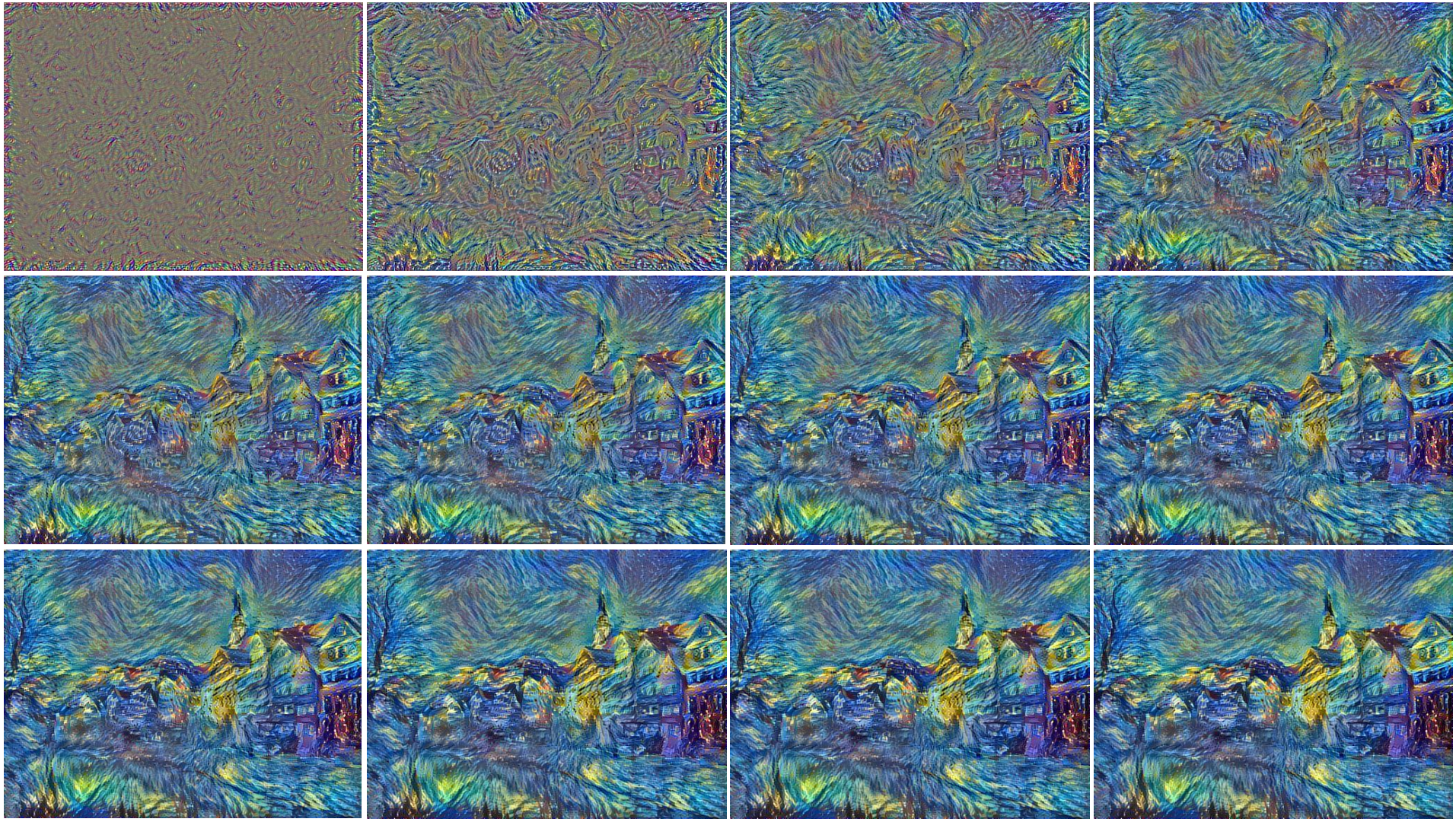
$$\mathcal{L}_{total}(\mathbf{I}) = \alpha\,\mathcal{L}_{content}(\mathbf{I}) + \beta\,\mathcal{L}_{style}(\mathbf{I})$$

Total Loss



$$\mathcal{L}_{total}(\mathbf{I}) = \alpha\,\mathcal{L}_{content}(\mathbf{I}) + \beta\,\mathcal{L}_{style}(\mathbf{I})$$

Minimize total loss



Neural Style Transfer

Example outputs

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

Neural Style Transfer

More weight to content loss

More weight to style loss

Neural Style Transfer

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

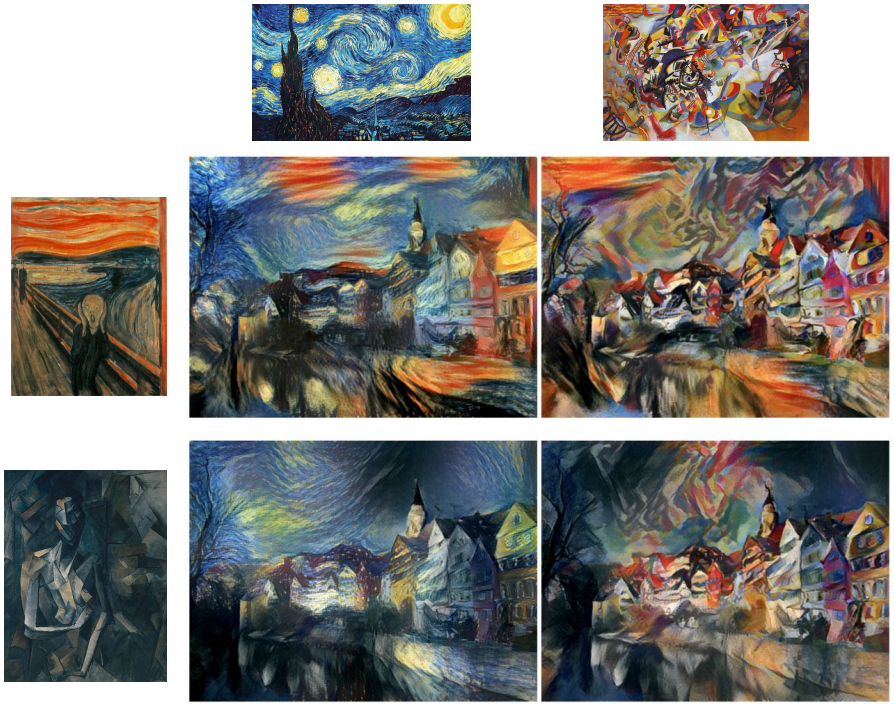
Larger style image

Smaller style image

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

Neural Style Transfer: Multiple Style Images

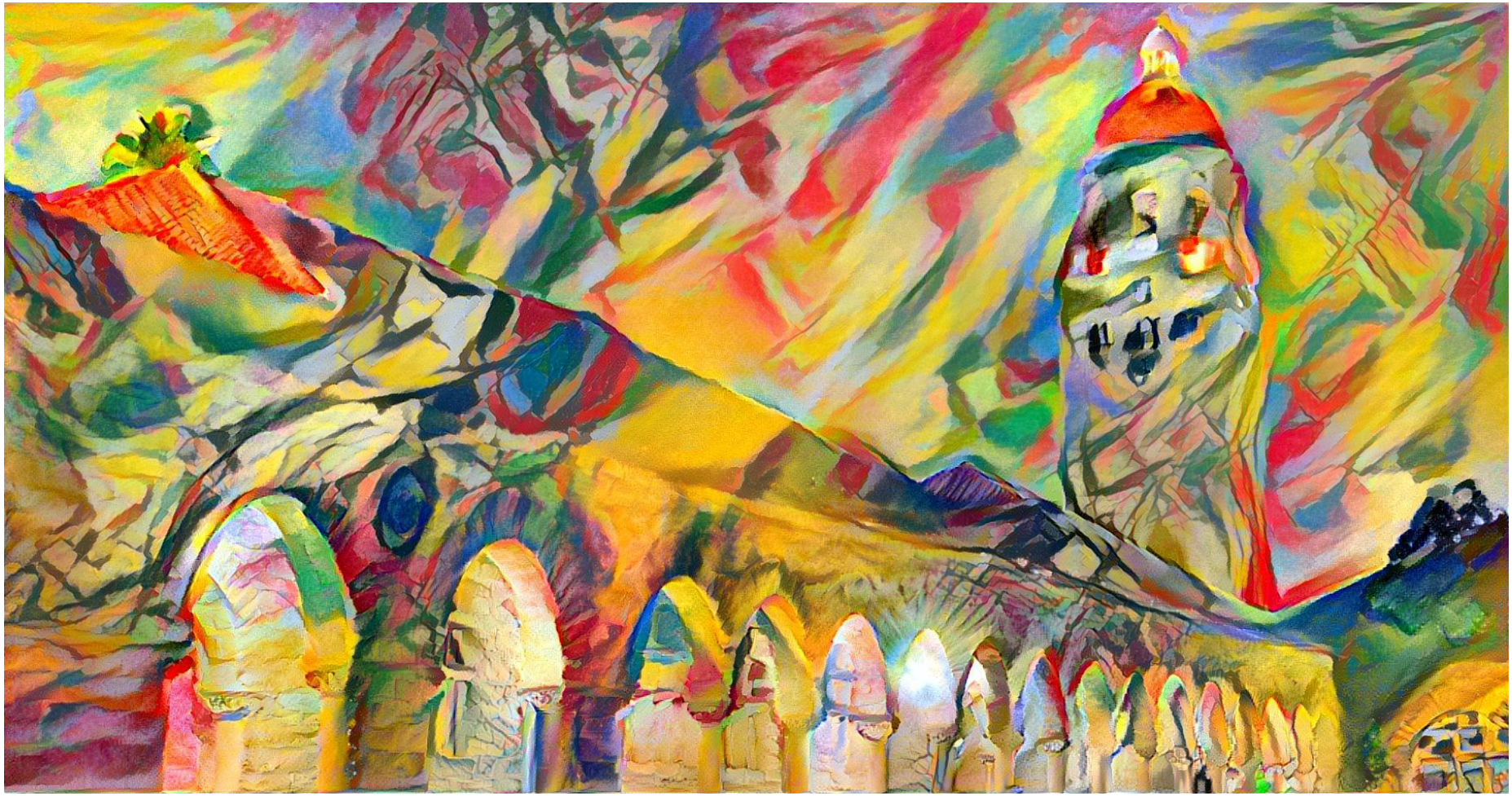
Mix style from multiple images by taking a weighted average of Gram matrices



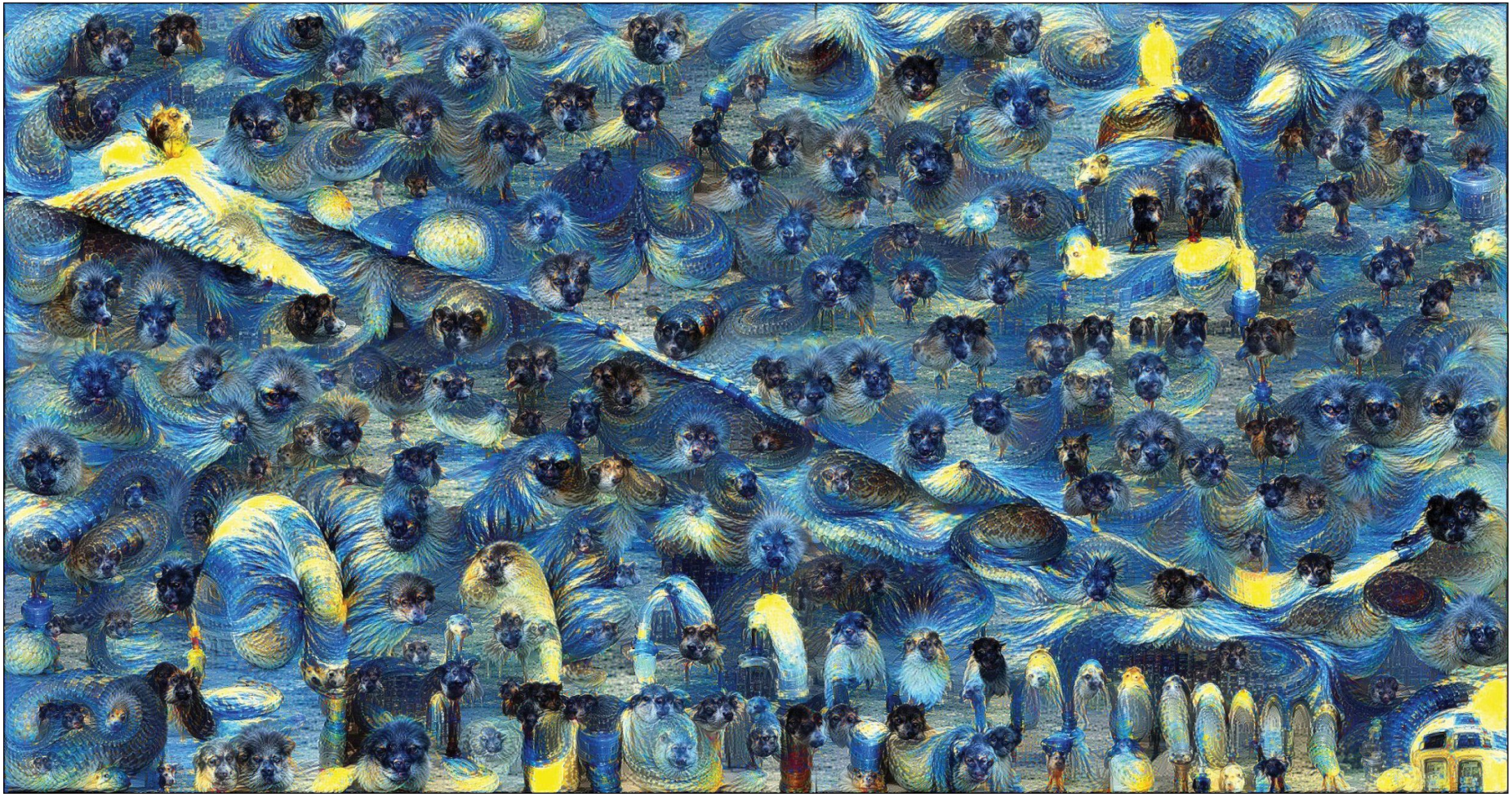
Gatys, Ecker, and Bethge, “Image style transfer using convolutional neural networks”, CVPR 2016 Figure copyright Justin Johnson, 2015.

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

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



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



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