

COMP0169: Machine Learning for Visual Computing

Style vs Content

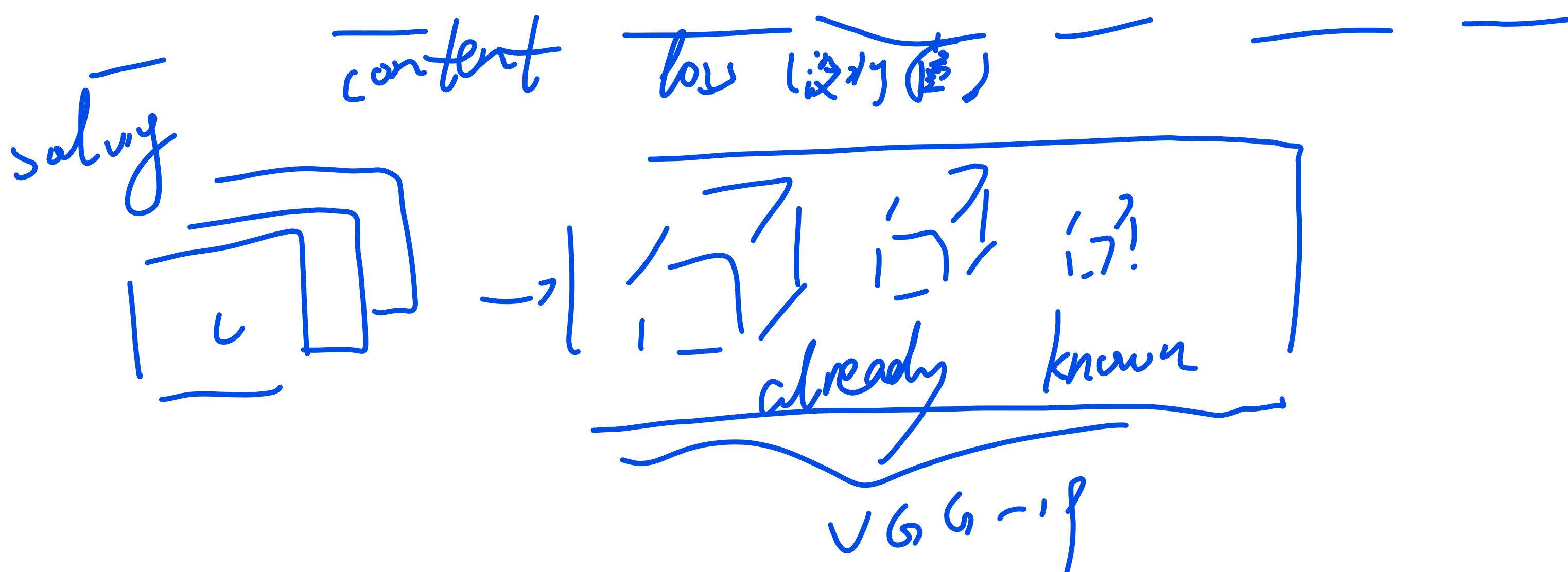


Lectures will be Recorded

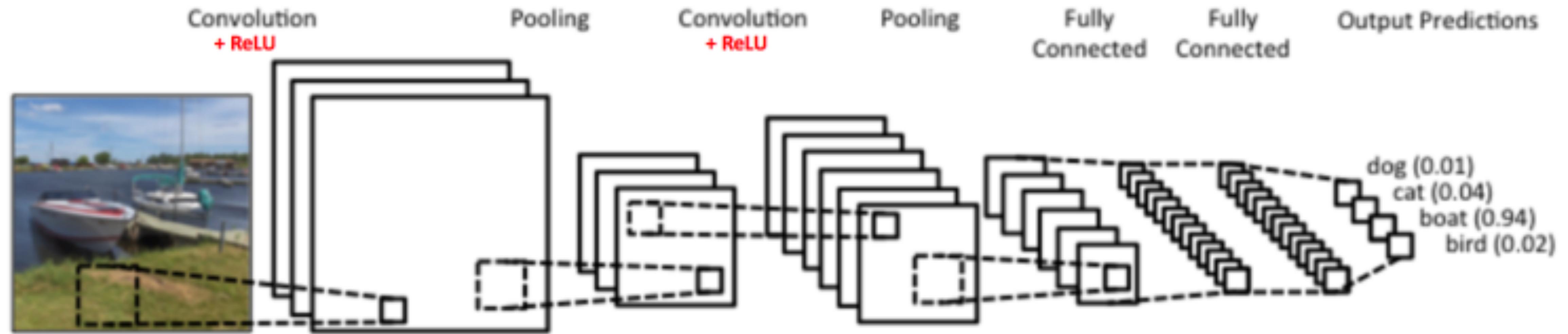
Normal style *Transfor.*

$$L(G) = \lambda L(C, G) + \beta L(S, G)$$

content *style*



CNN Recap



AlexNet

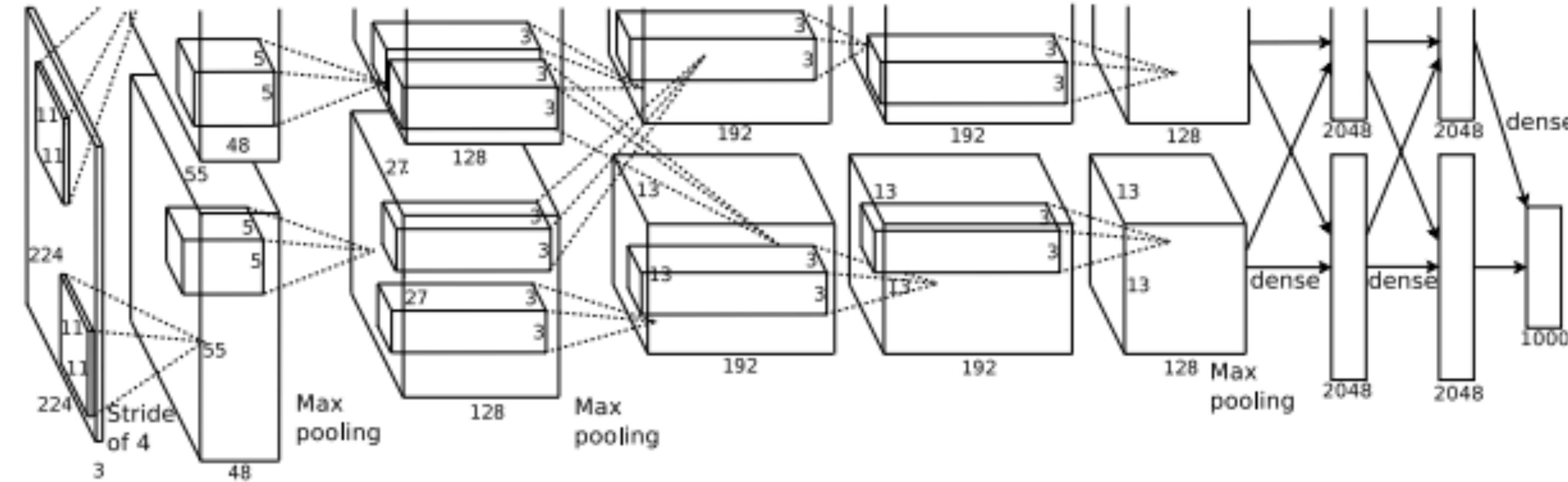


ImageNet Classification with Deep Convolutional Neural Networks

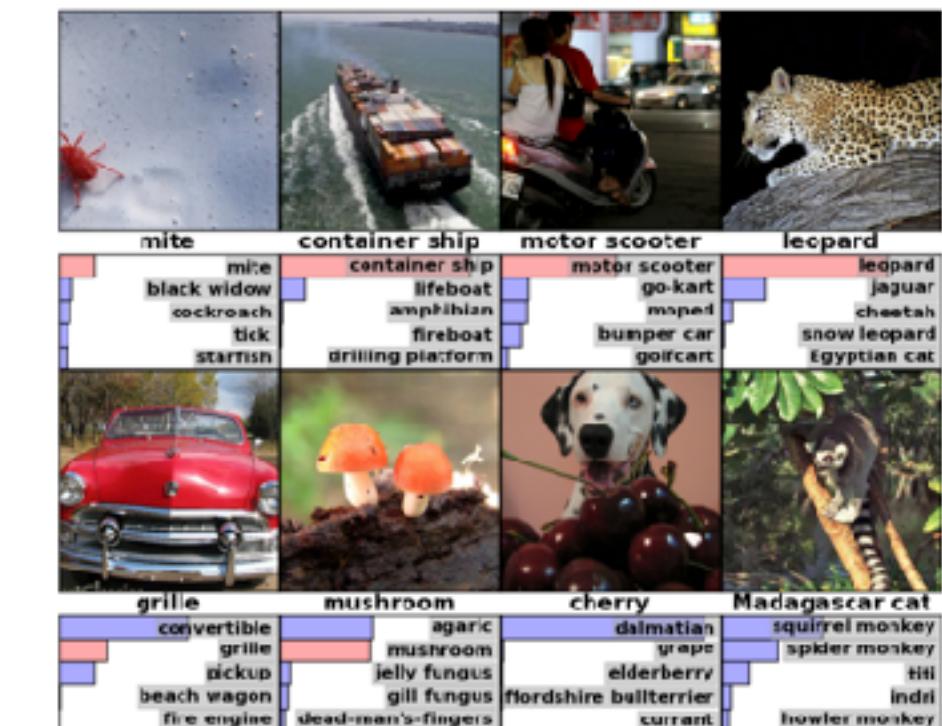
Alex Krizhevsky
University of Toronto
kriz@cs.toronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.toronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.toronto.ca



Class scores
for 1000
Classes

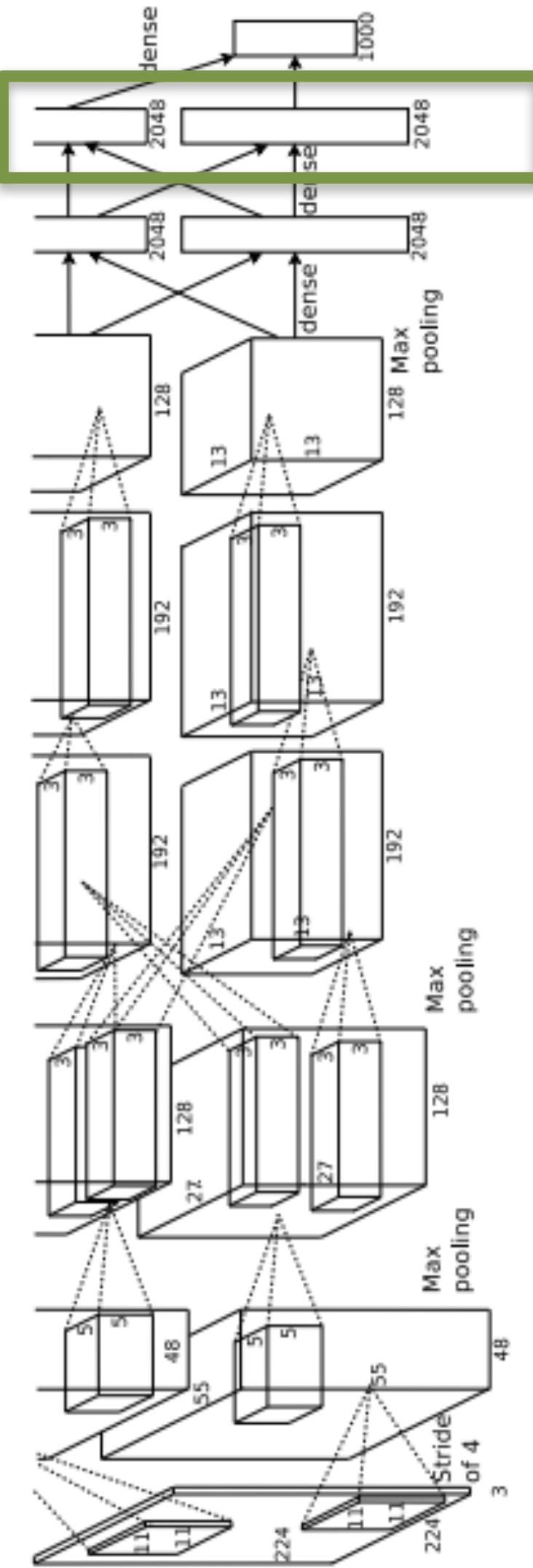


Last Layer

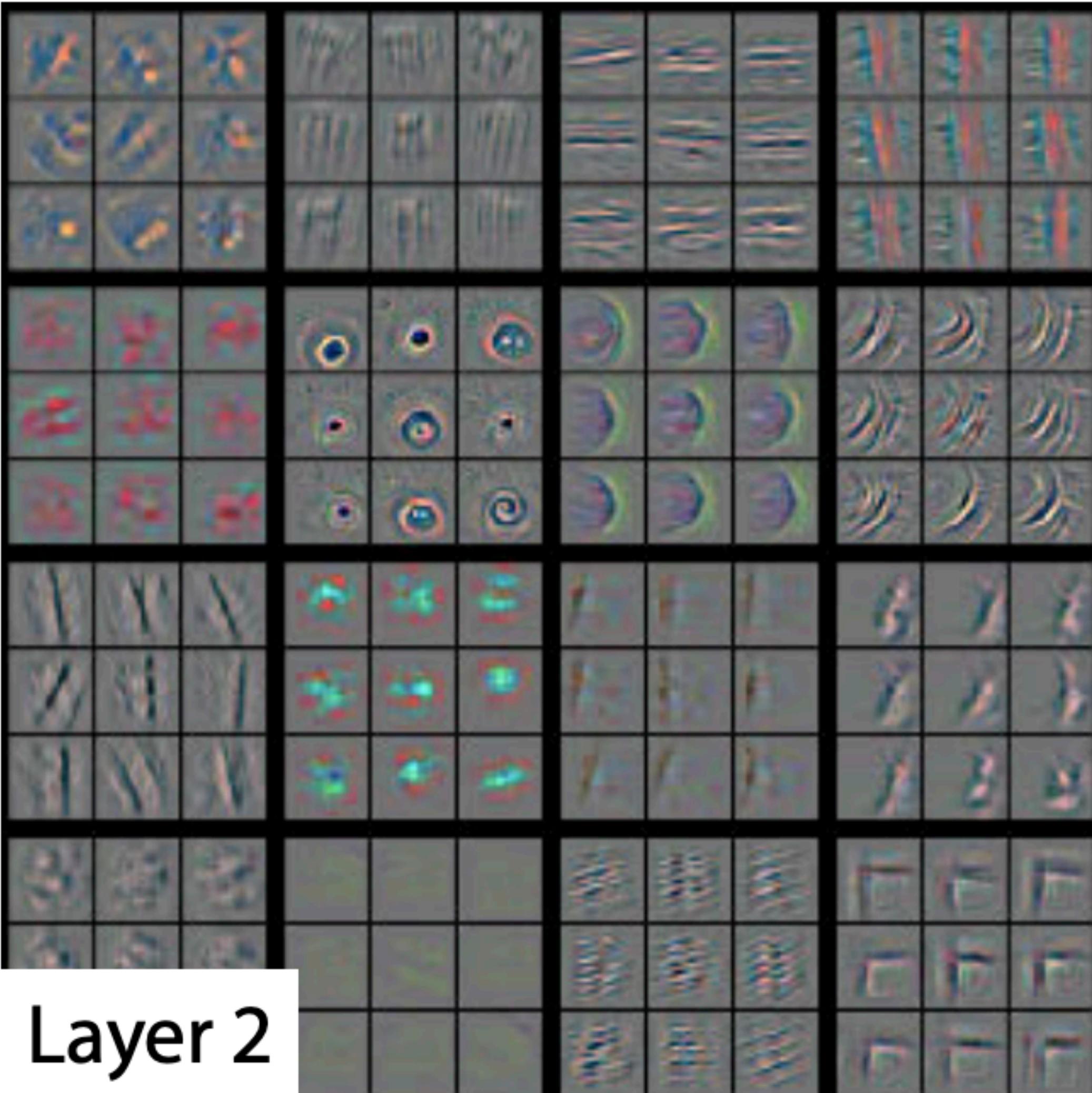
$$\mathbb{R}^{224 \times 224 \times 3} \rightarrow \mathbb{R}^{4096}$$

Feature size $2048+2048=4096$.

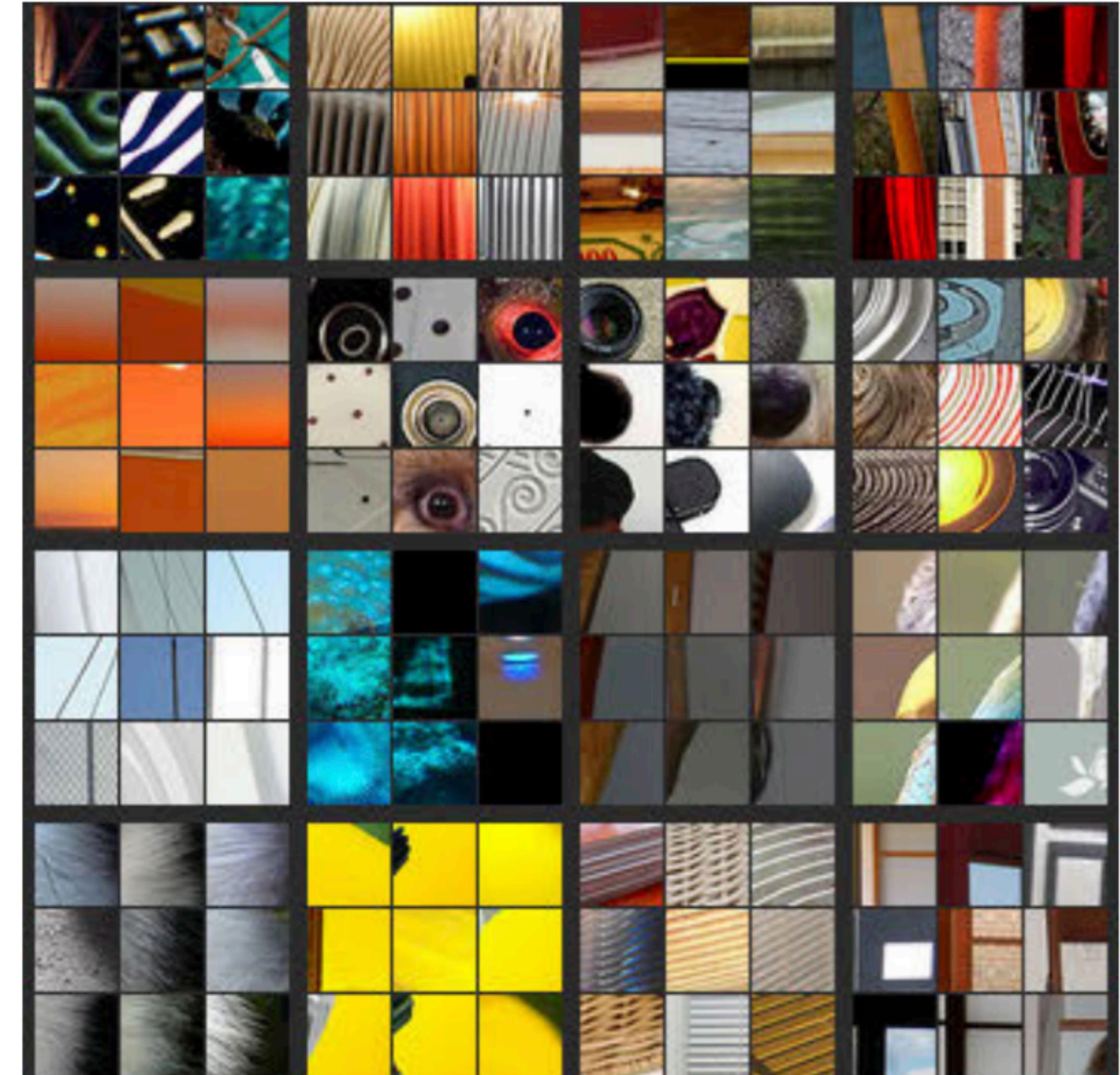
Pretrain network. Run ‘feature extractor’ on many images.
FC7 features.



Feature Visualisation



Layer 2



Style vs Content

Visualizing CNN Features: Gradient Ascent

- **Guided Backprop:**
Find parts of image that maximally responds to a neuron

- **Gradient Ascent:**
Find a synthetic image that maximally activates a neuron

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

Gradient Ascent

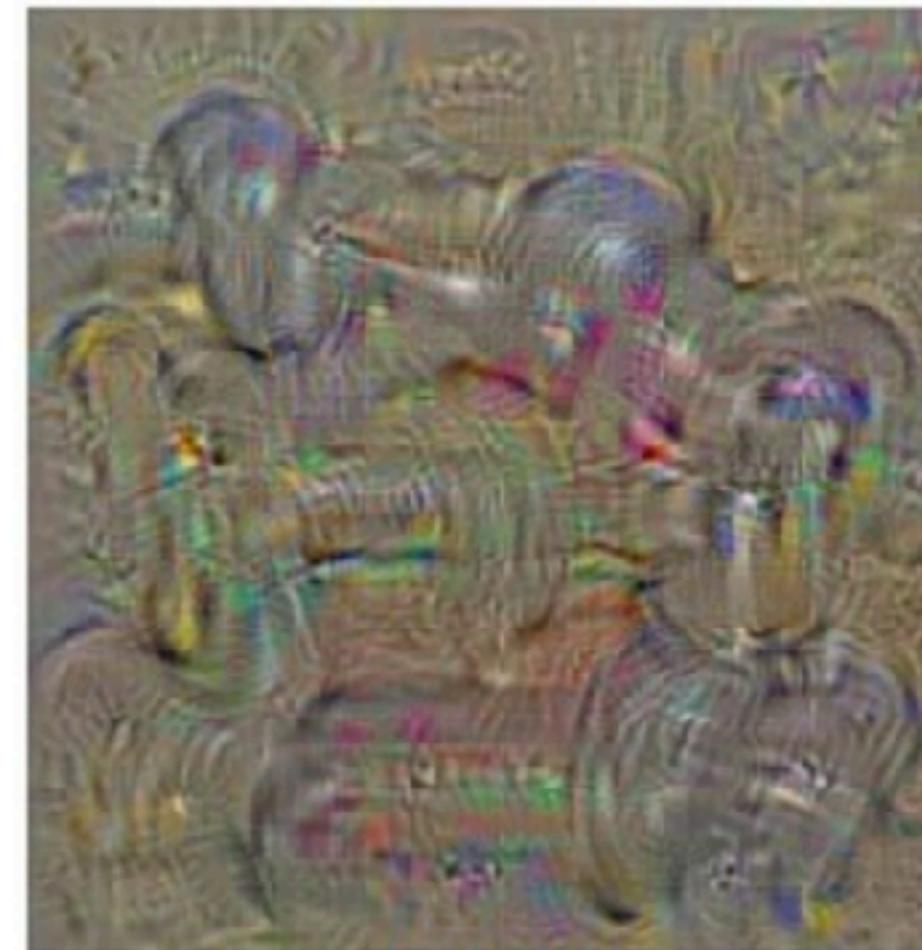
$$I^* = \arg \max_I \{S_c(I) - \lambda \|I\|^2\}$$

- Initialize image $I = 0$
- Forward image to compute scores $S_c(I)$
- Backprop to get gradients of neuron values wrt image pixels
- Update image as $I \leftarrow I + \frac{\partial S_c(I)}{\partial I}$

Gradient Ascent

Other regularizers:

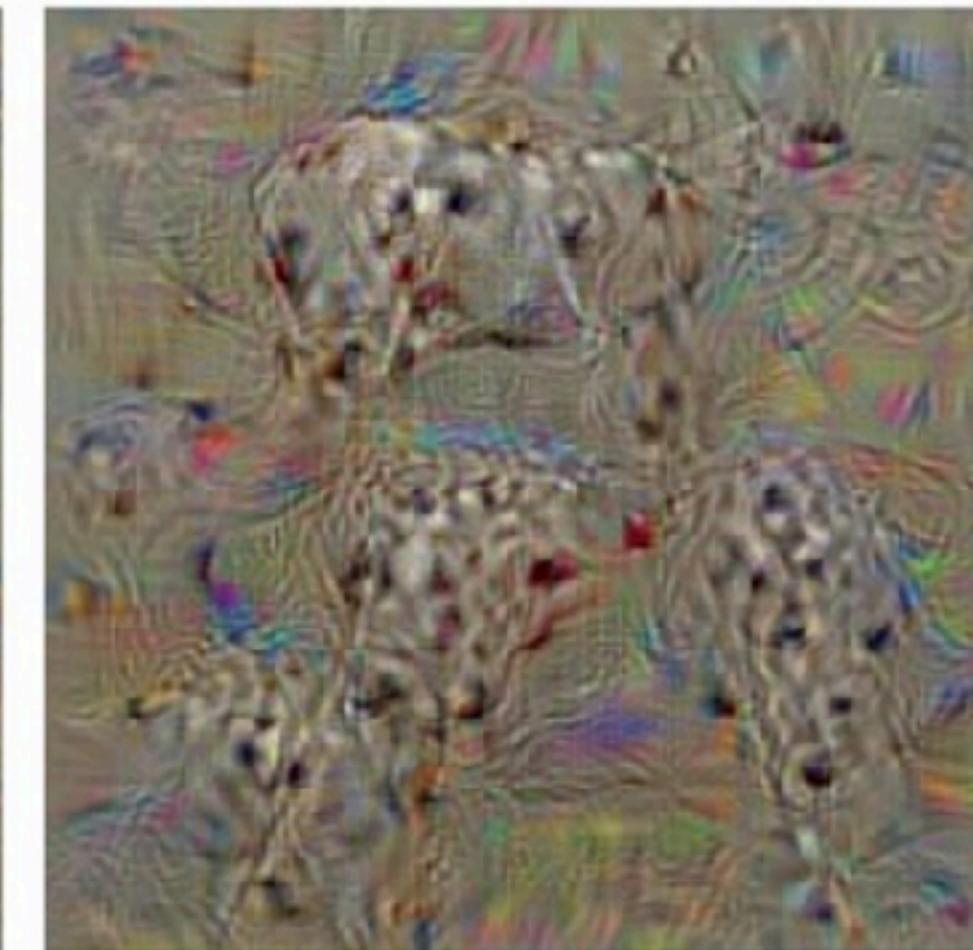
- 1.Gaussian blur
- 2.Clip pixels with values ~ 0
- 3.Clip pixels with gradients ~ 0



dumbbell



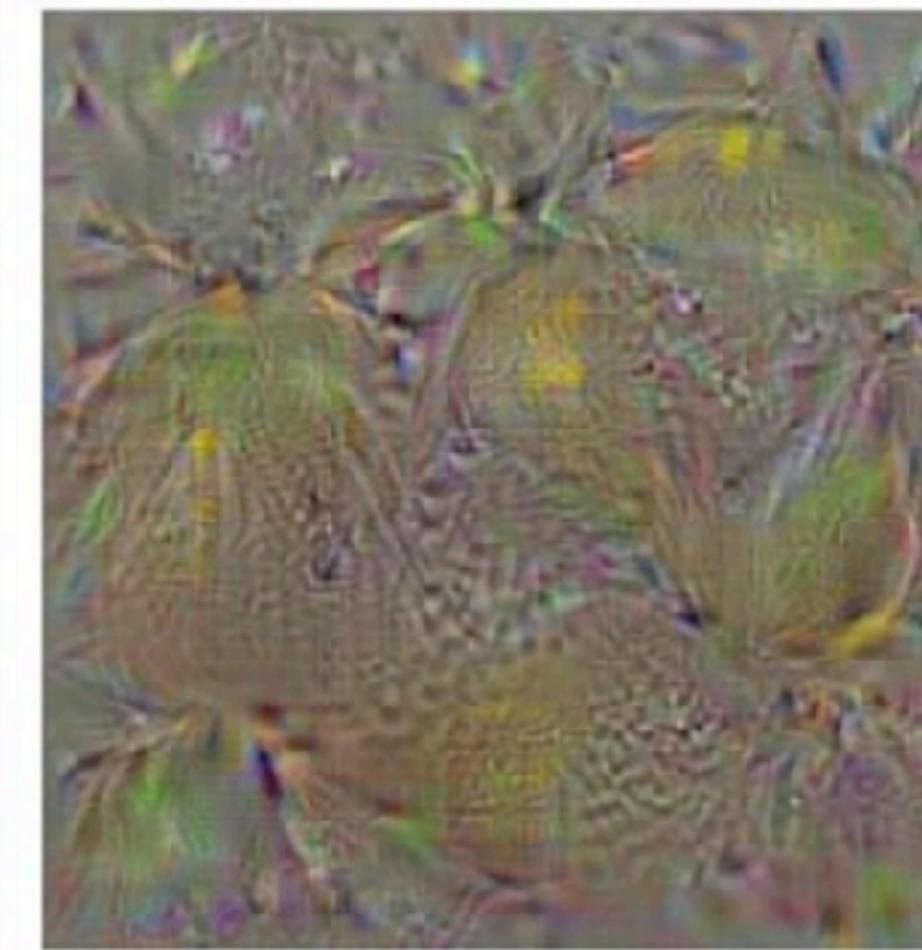
cup



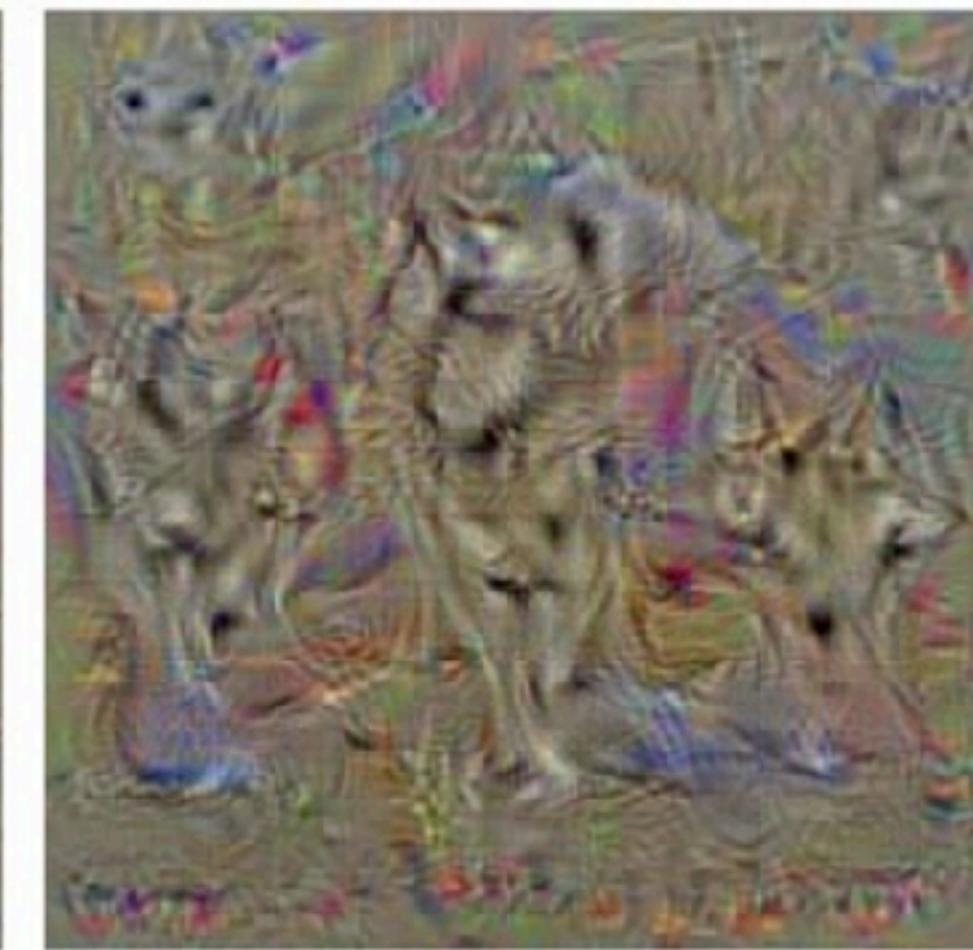
dalmatian



bell pepper

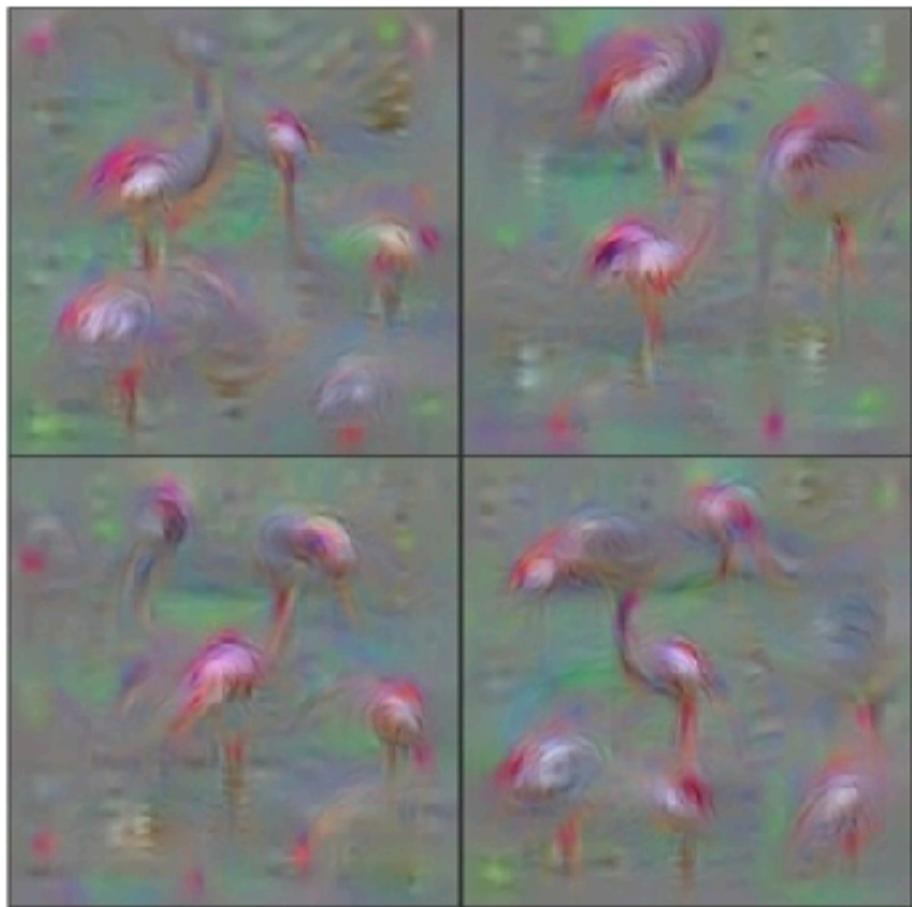


lemon

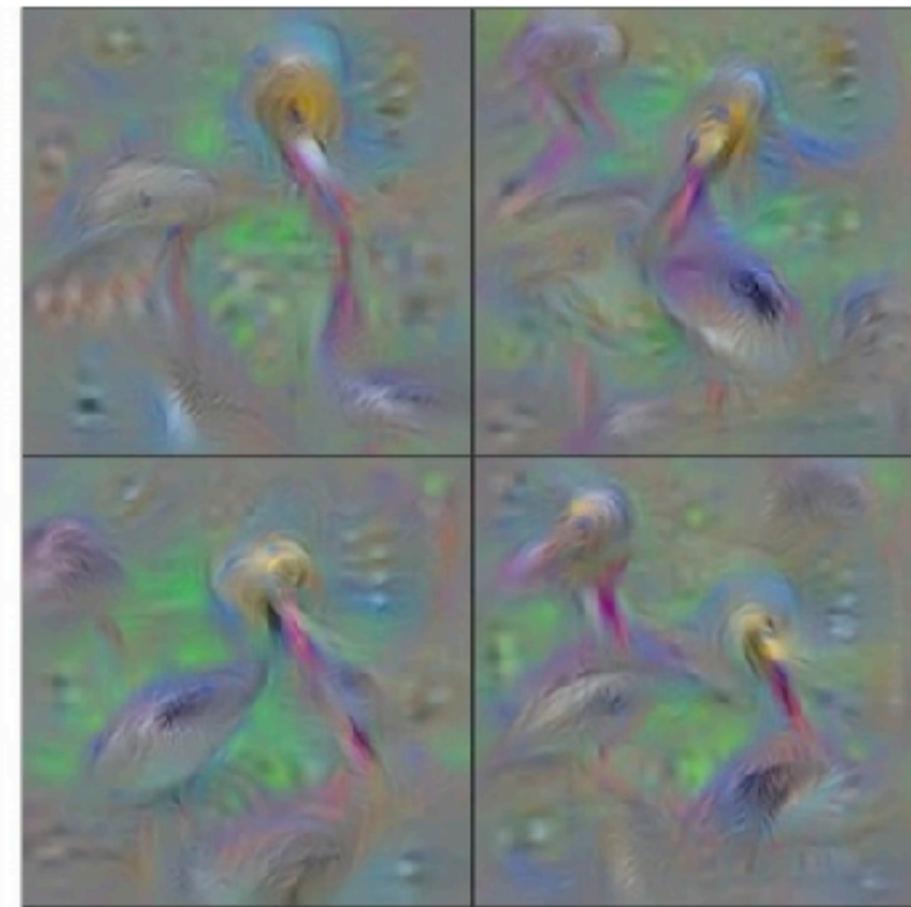


husky

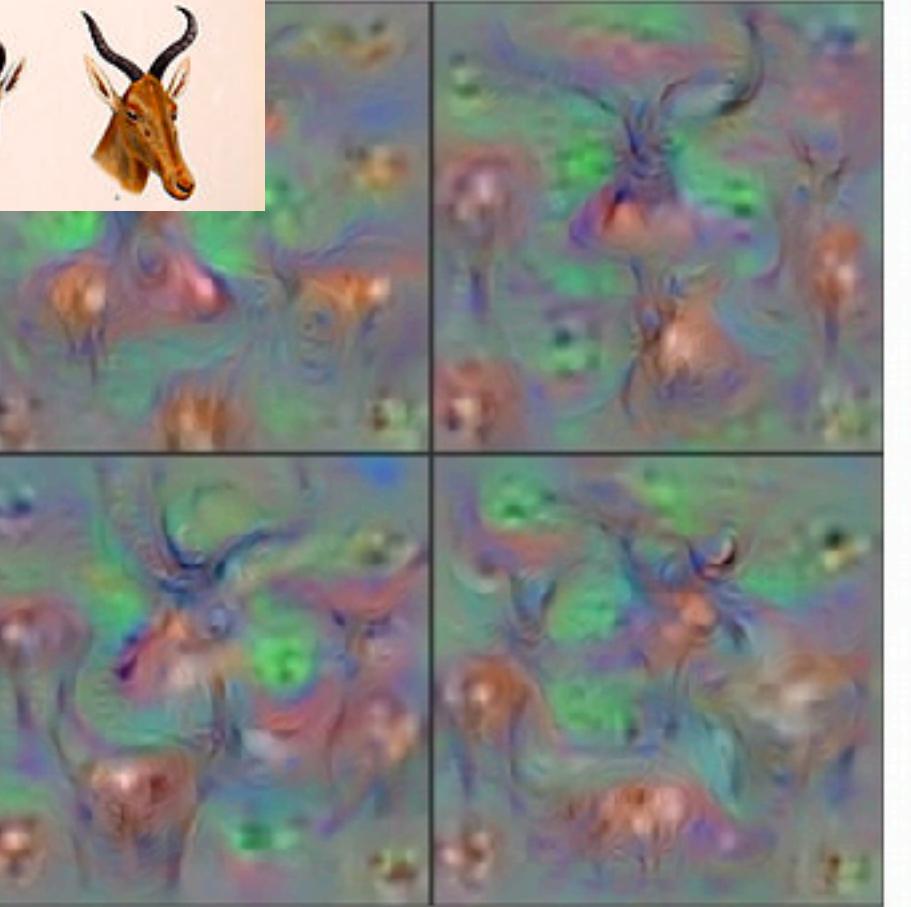
GA Results: More



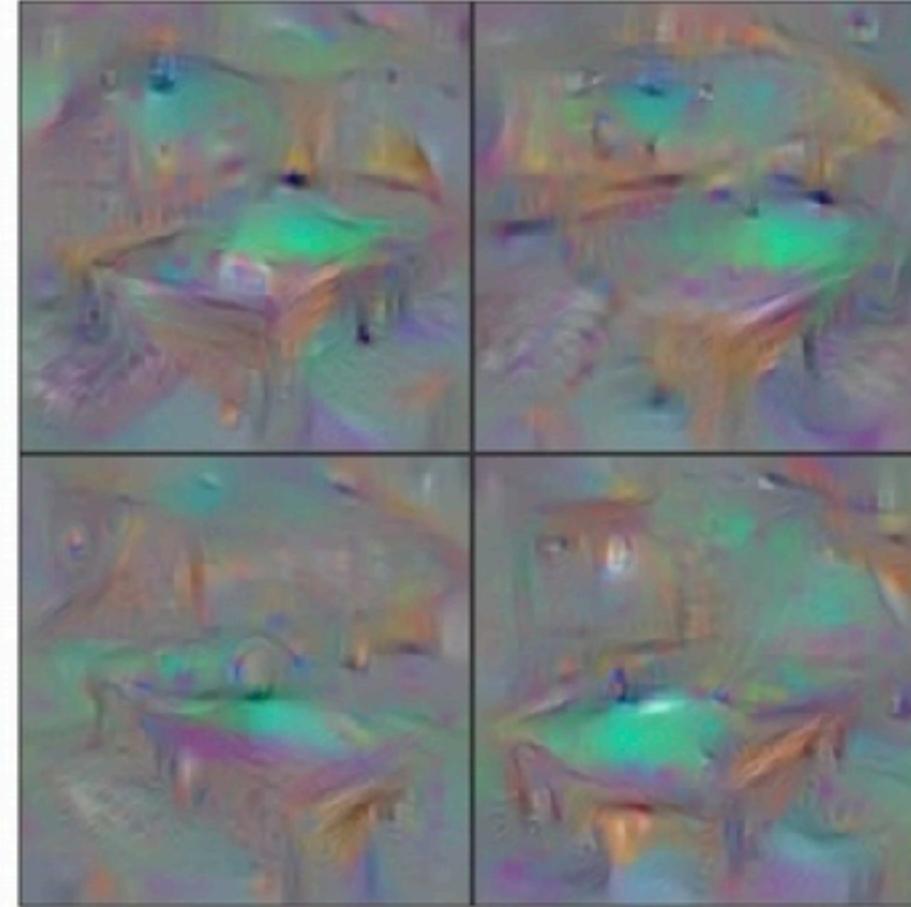
Flamingo



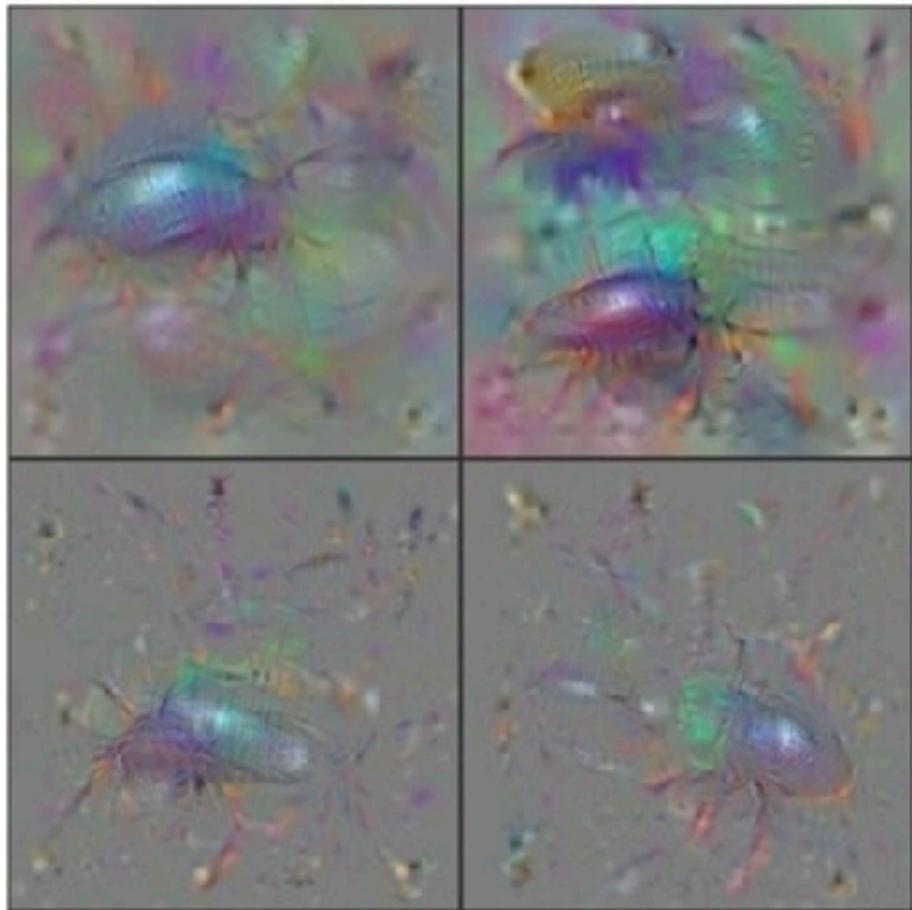
Pelican



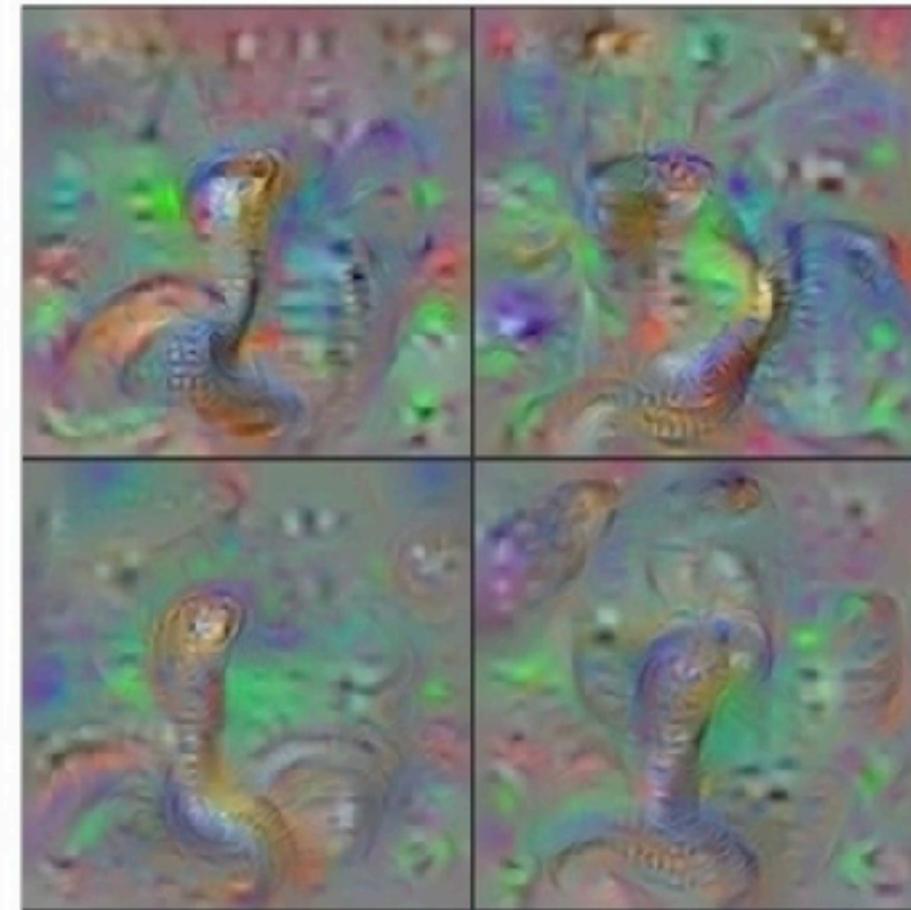
Hartebeest



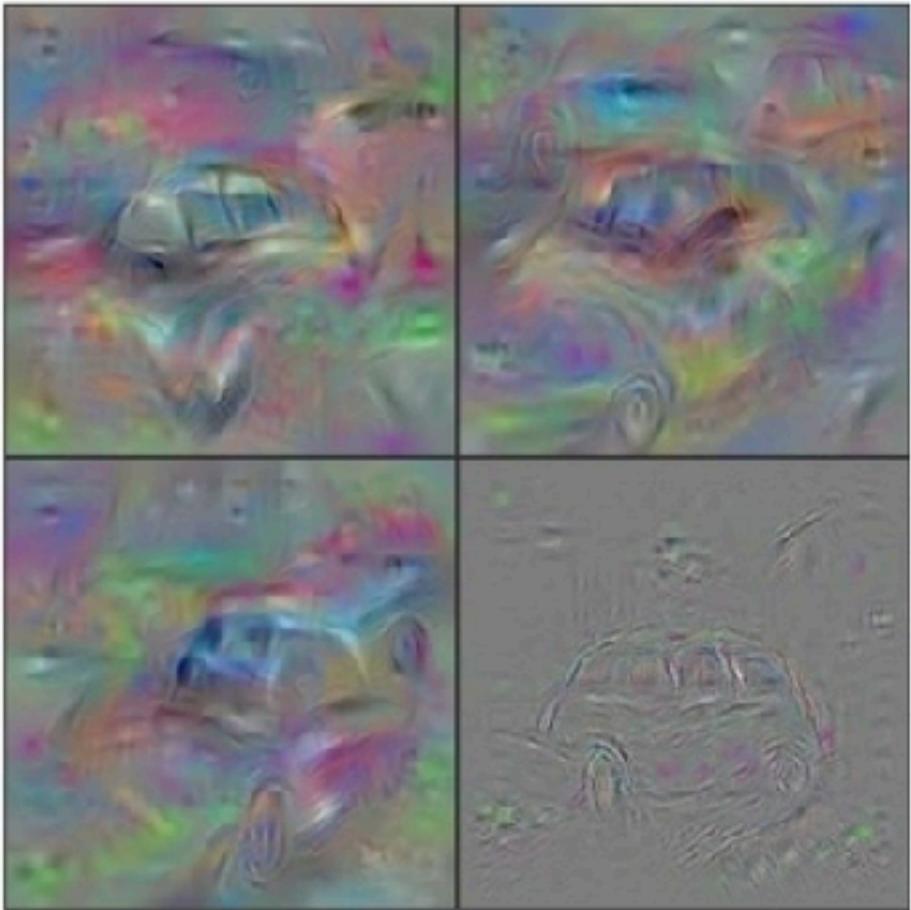
Billiard Table



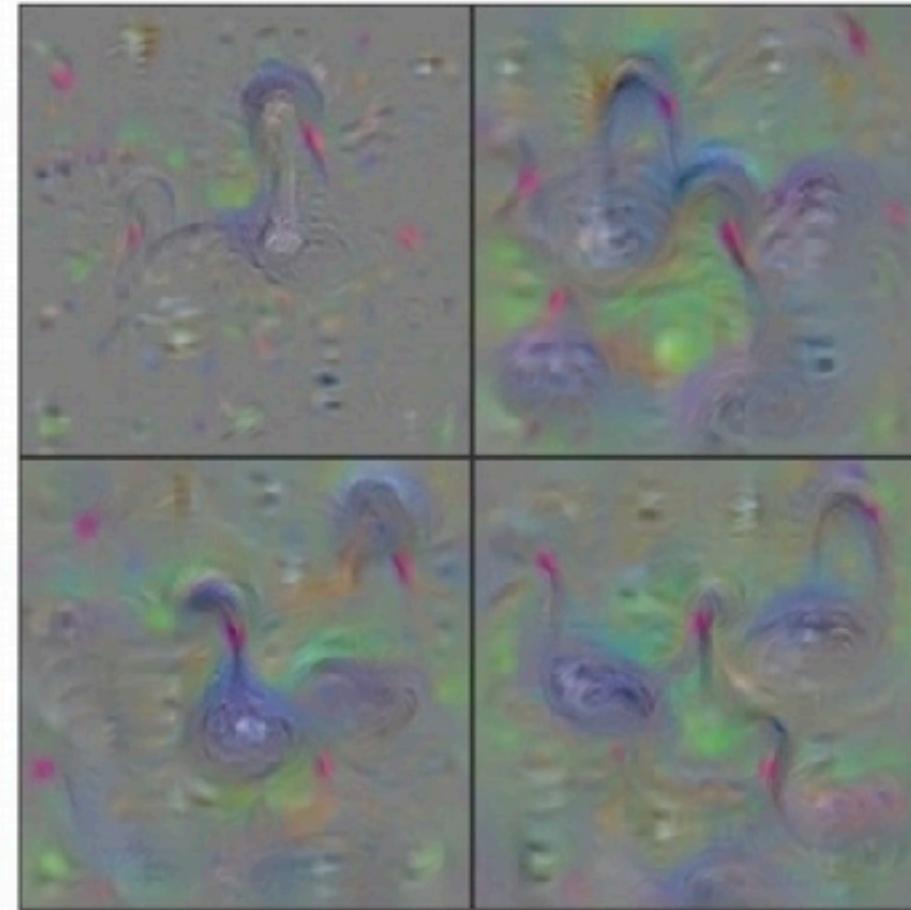
Ground Beetle



Indian Cobra



Station Wagon



Black Swan

Visualize Intermediate Features

Understanding Neural Networks Through Deep Visualization

Jason Yosinski
Cornell University

YOSINSKI@CS.CORNELL.EDU

Jeff Clune
Anh Nguyen
University of Wyoming

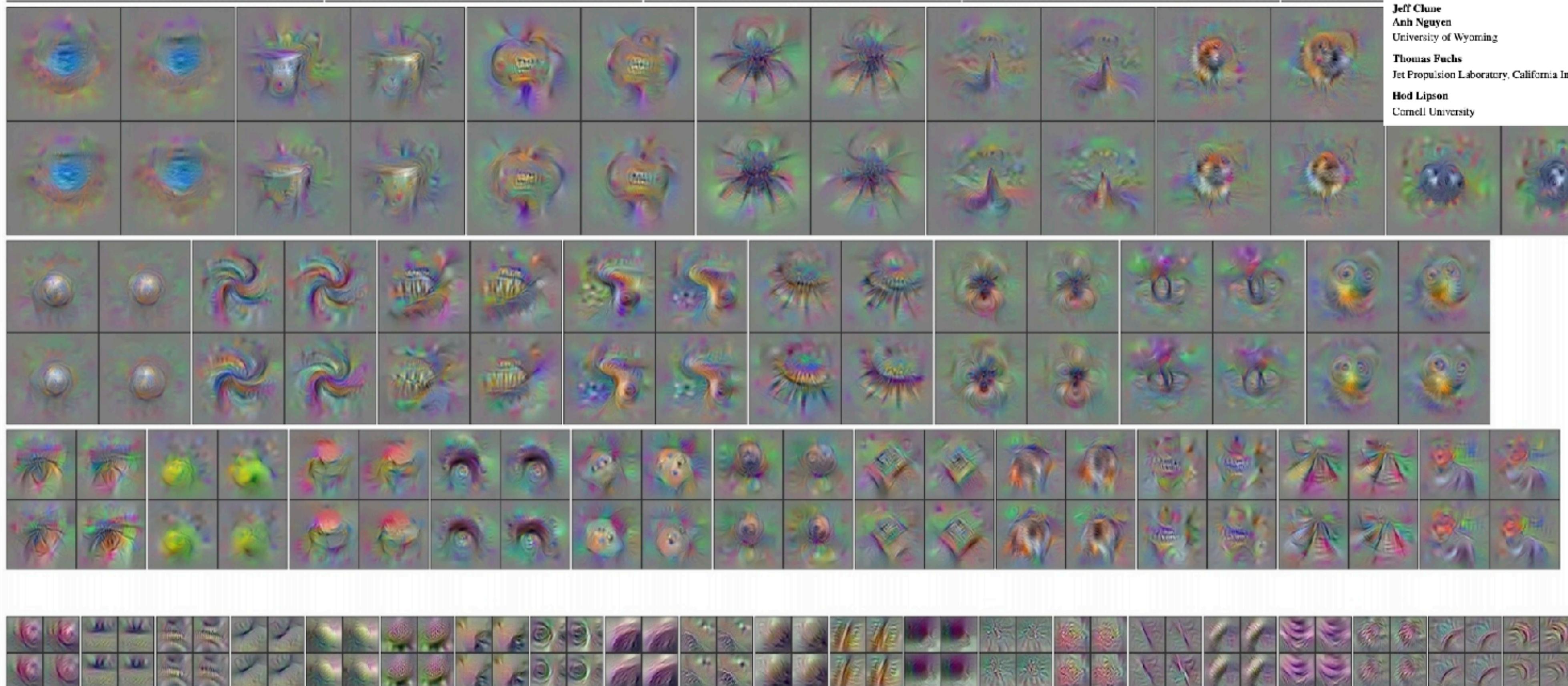
JEFFCLUNE@UWYO.EDU
ANGUYEN8@UWYO.EDU

Thomas Fuchs
Jet Propulsion Laboratory, California Institute of Technology

FUCHS@JPL.NASA.GOV
HOD.LIPSON@CORNELL.EDU

Hod Lipson
Cornell University

Layer 5
Layer 4
Layer 3
Layer 2



More results: Stronger Regularizers



Relation to Adversarial Training

议題
Adversarial

Adversarial Training: Examples

African elephant



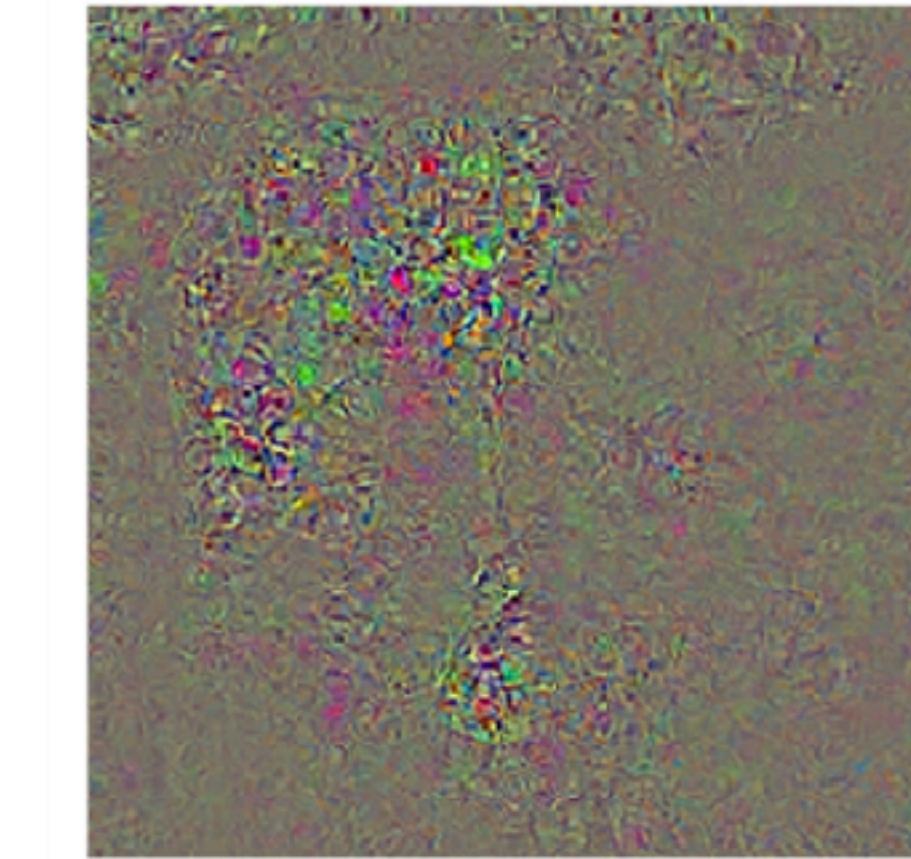
koala



Difference



10x Difference



schooner



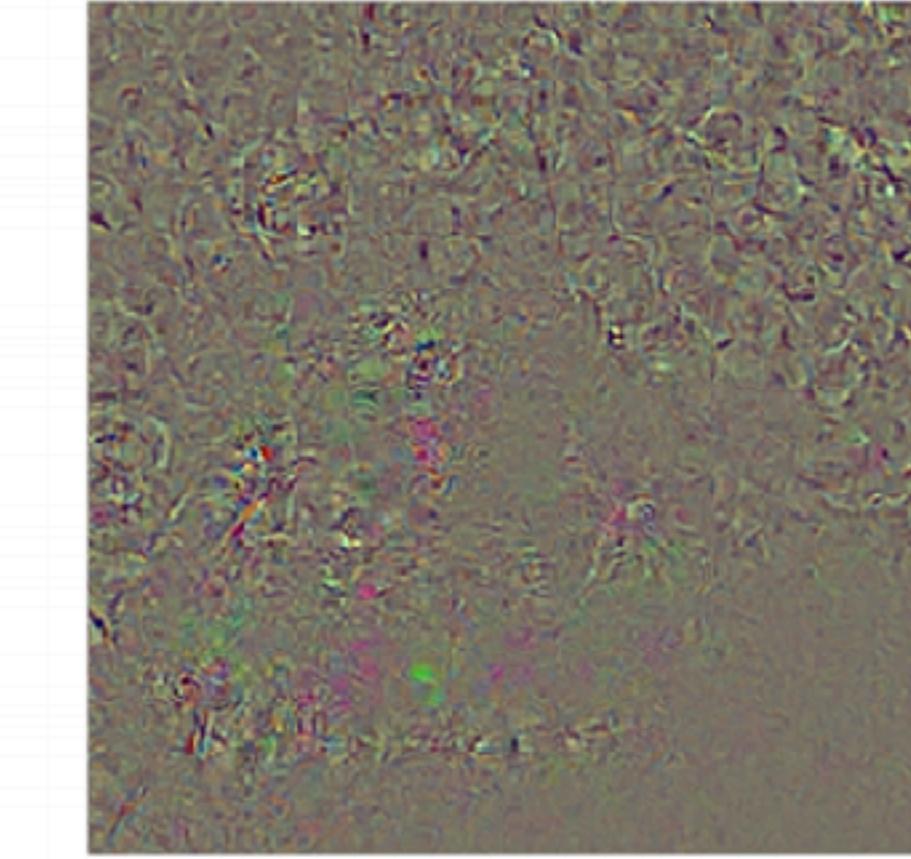
iPod



Difference



10x Difference



Feature Inversion

$$x^* = \arg \min_I \{l(\Phi(I), \Phi_0) + \lambda \mathcal{R}(I)\}$$

go through the network and find the layer

$$l(\Phi(I), \Phi_0) = \|\Phi(I) - \Phi_0\|^2$$

$$\mathcal{R}_\beta(I) = \sum_{i,j} (I(i, j+1) - I(i, j))^2 + (I(i+1, j) - I(i, j))^2)^{\frac{\beta}{2}}$$

Feature Inversion

y



relu2_2



relu3_3



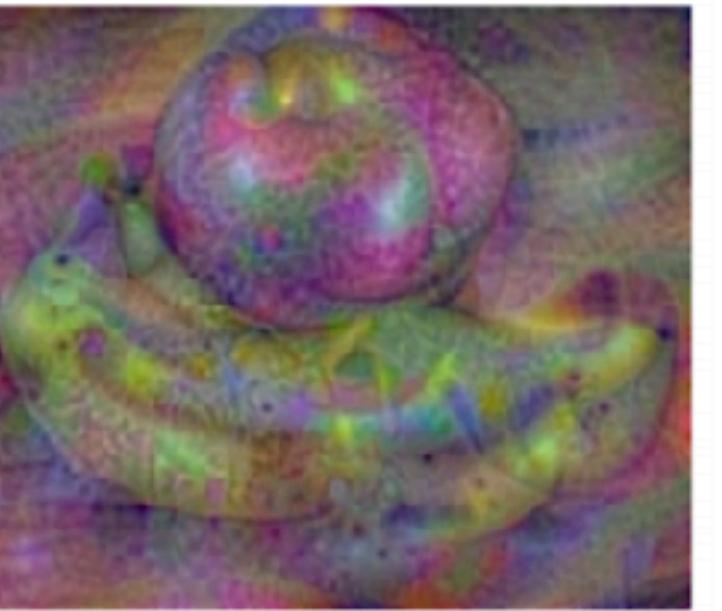
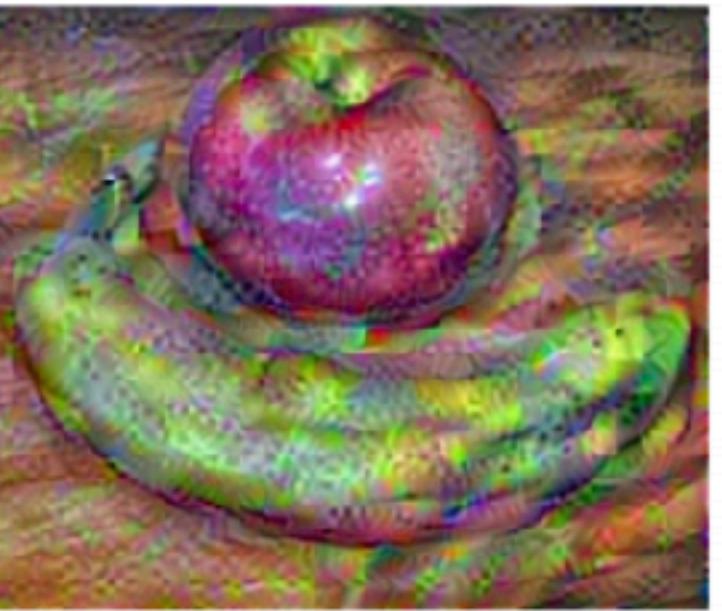
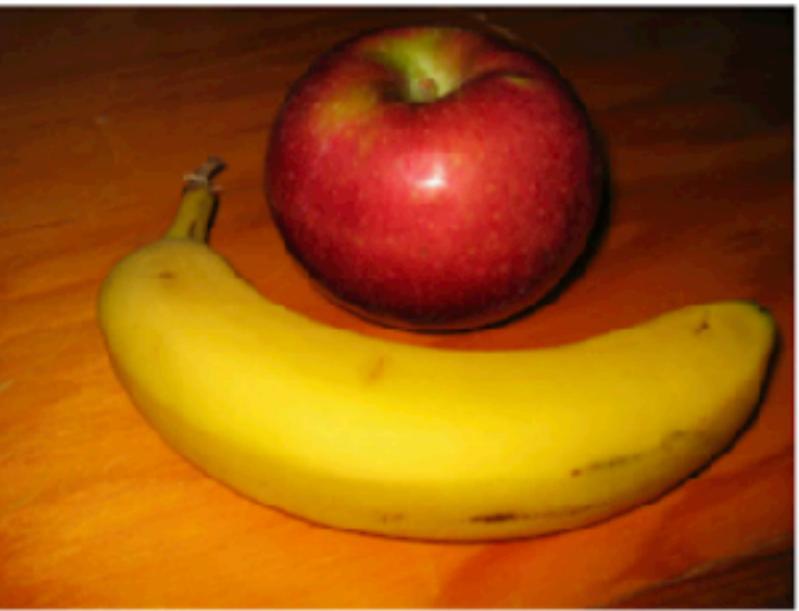
relu4_3



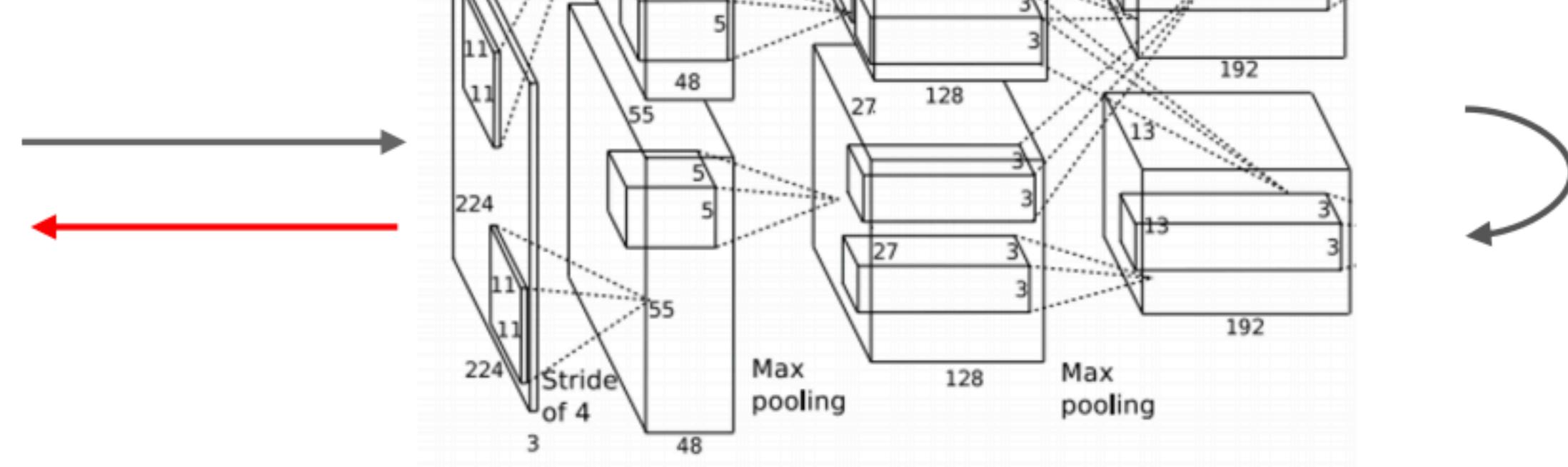
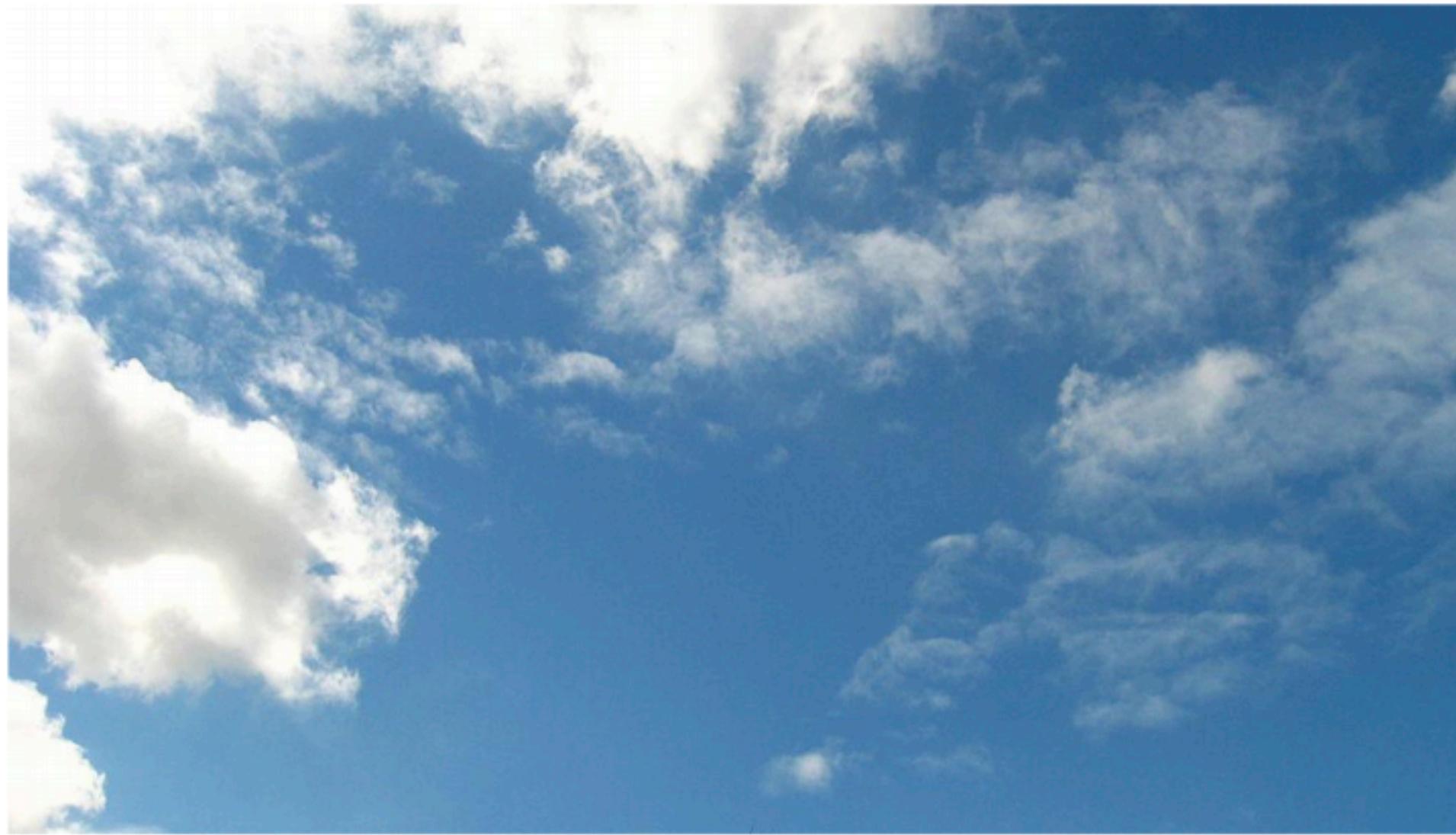
relu5_1



relu5_3



Deep Dream: Amplify Features



Inceptionism: Going Deeper into Neural Networks

Wednesday, June 17, 2015

Posted by Alexander Mordvintsev, Software Engineer, Christopher Olah, Software Engineering Intern and Mike Tyka, Software Engineer

- Compute activations at chosen layer via forward pass
- Set gradient of chosen layer equal to its activation
- Compute gradient on image via backprop
- Update image

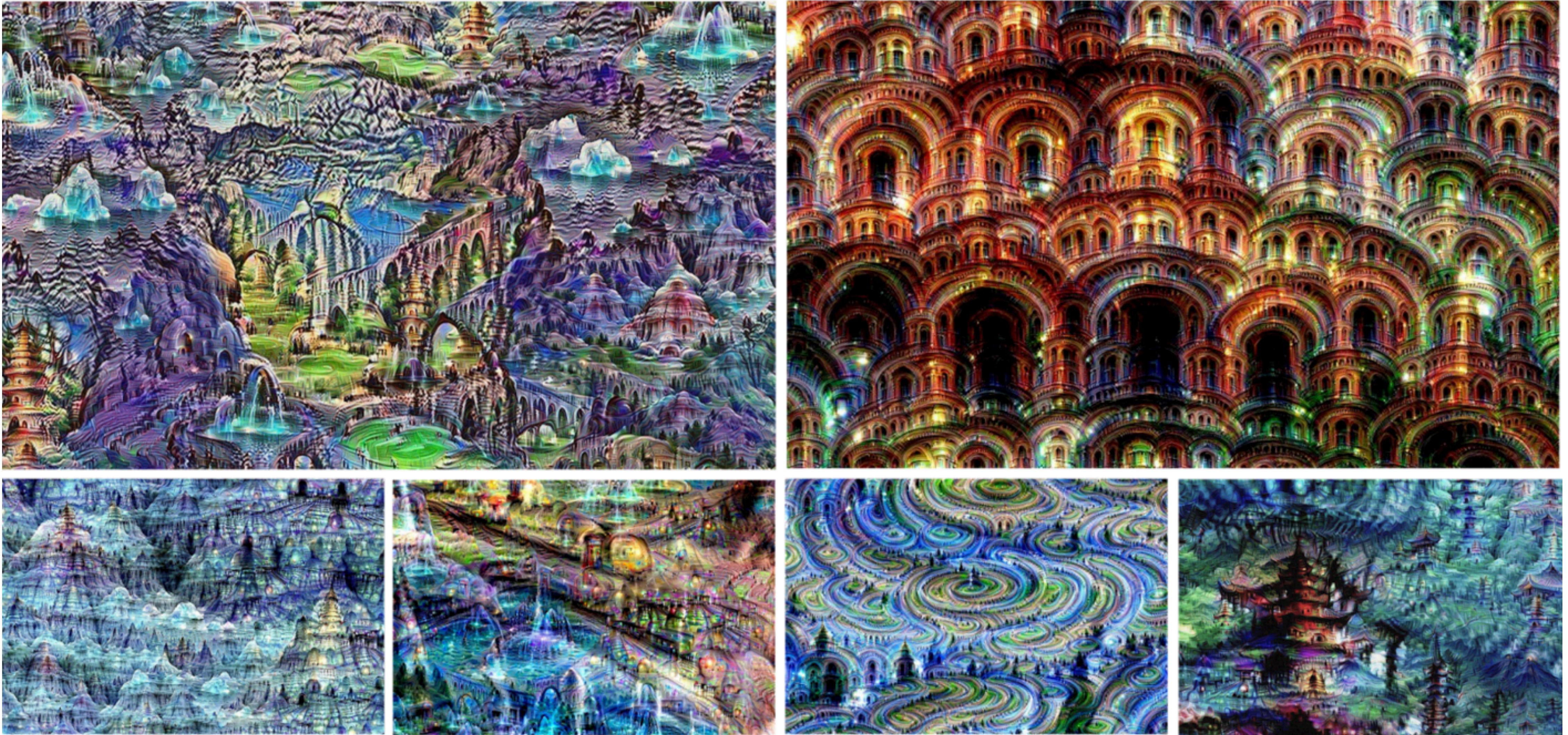
$$I^* = \arg \max_I \sum_i f_i(I)^2$$

DeepDream



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

DeepDream



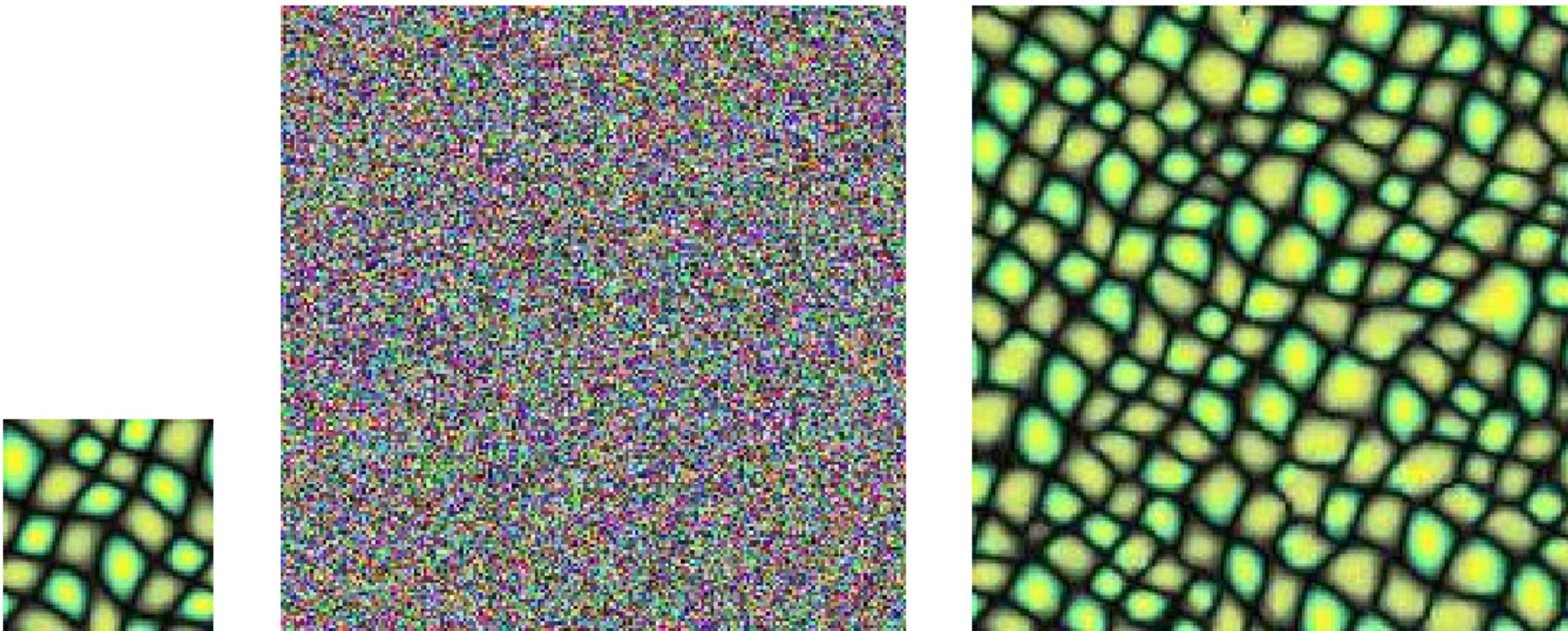
Texture Synthesis

Fast Texture Synthesis using Tree-structured Vector Quantization

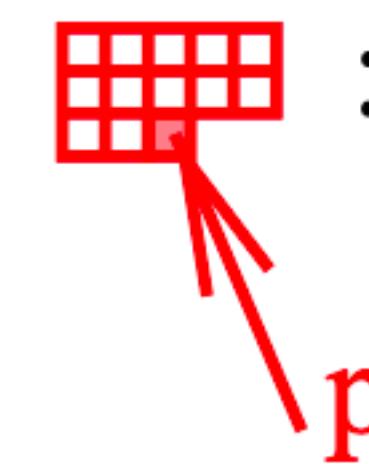
Li-Yi Wei

Marc Levoy

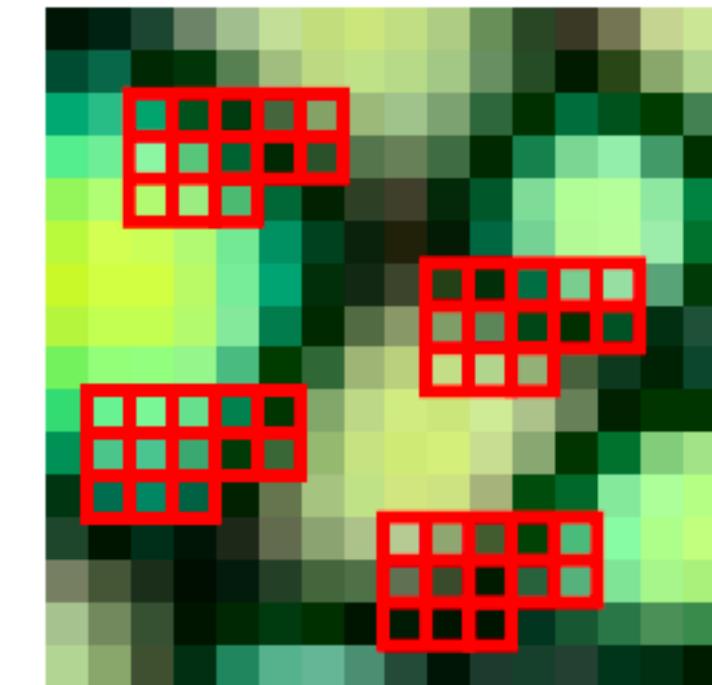
Stanford University *



Texture Synthesis Stages

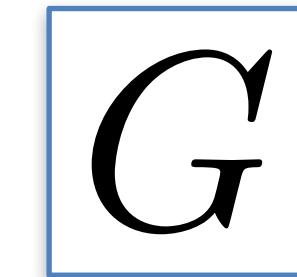
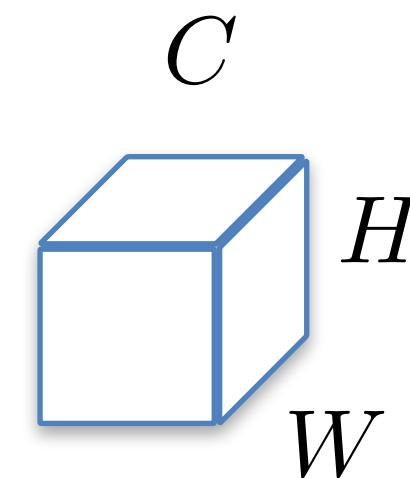
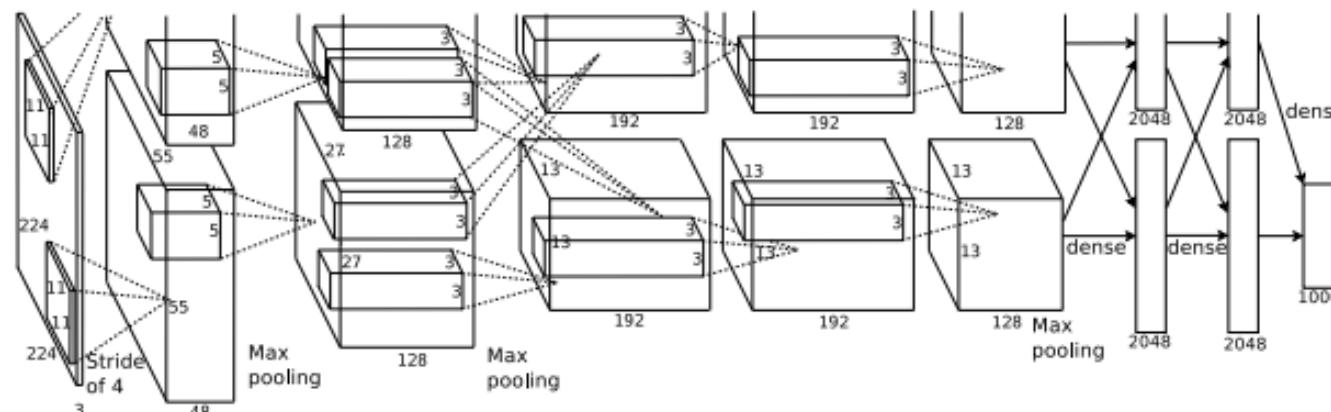


: Neighborhood N



(a)

Neural Texture Synthesis: Gram Matrix



Outer product of two C -dimensional vectors outputs C times C matrix (covariance matrix)

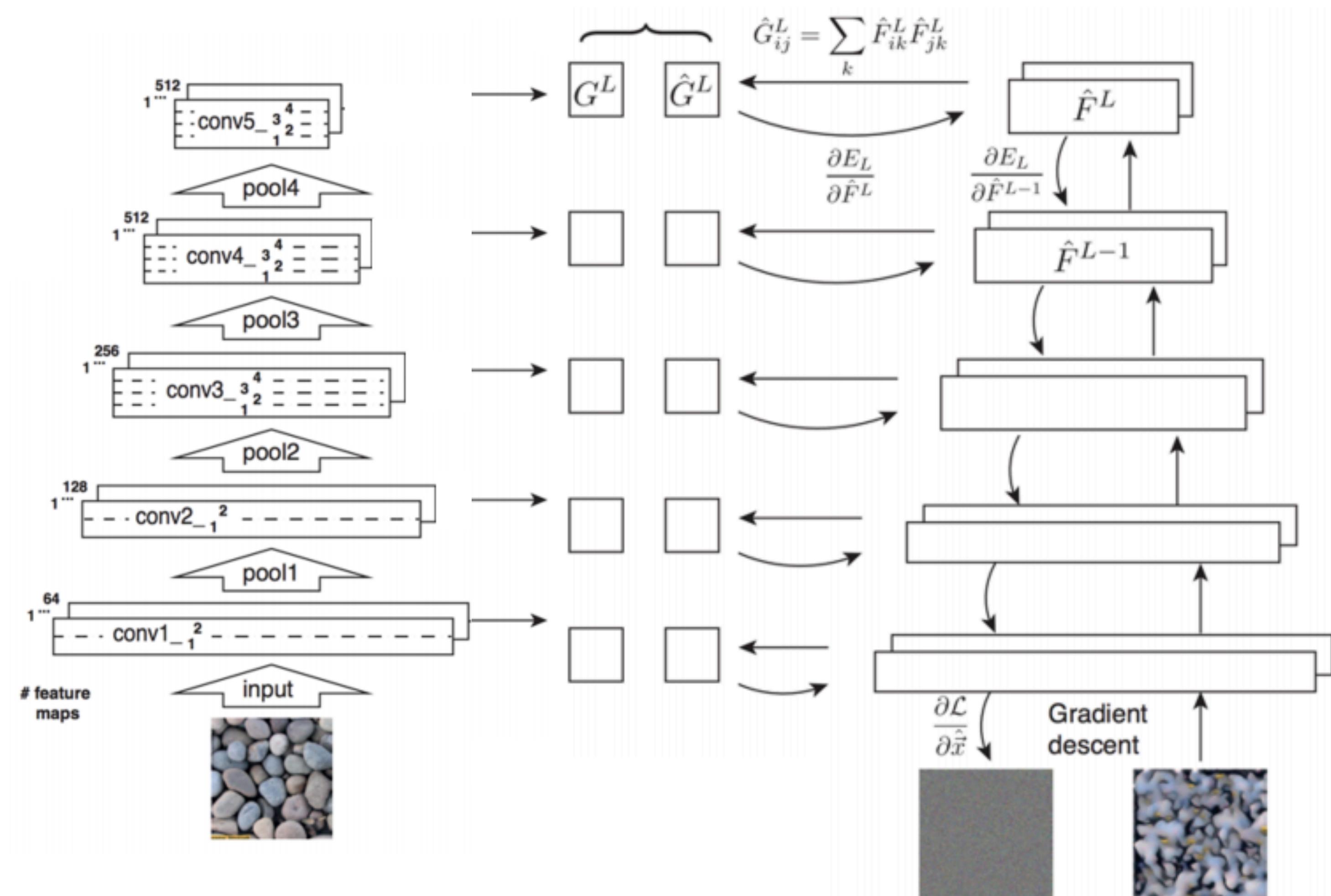
$$G := \sum_{i=1}^{HW} \mathbf{x}_i \mathbf{x}_i^T$$

$$E_l = \frac{1}{N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - \hat{G}_{ij}^l \right)$$

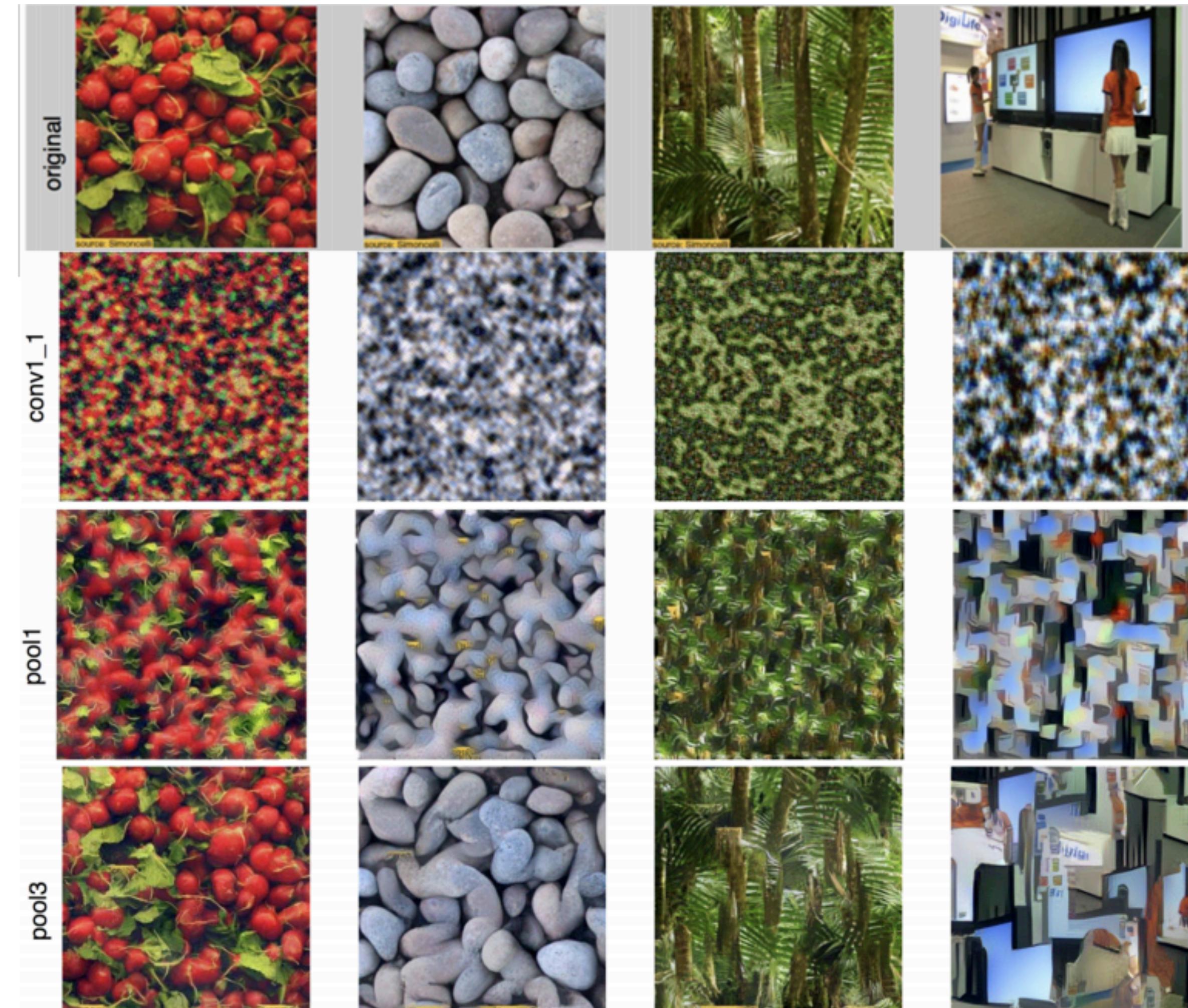
$$\mathcal{L}(I) := \sum_l \alpha_l E_l$$

Neural Texture Synthesis

- 1. Compute VGG features.**
 - 2. Given image I , compute features at different levels.**
 - 3. Compute Gram matrices at different levels.**
- $$G_{ij}^l := \sum_k F_{ik}^l F_{jk}^l$$
- 4. Initialize with random (noise) matrix.**
 - 5. Compute features at each level.**
 - 6. Compute loss, backprop wrt image pixel.
Goto step 5.**

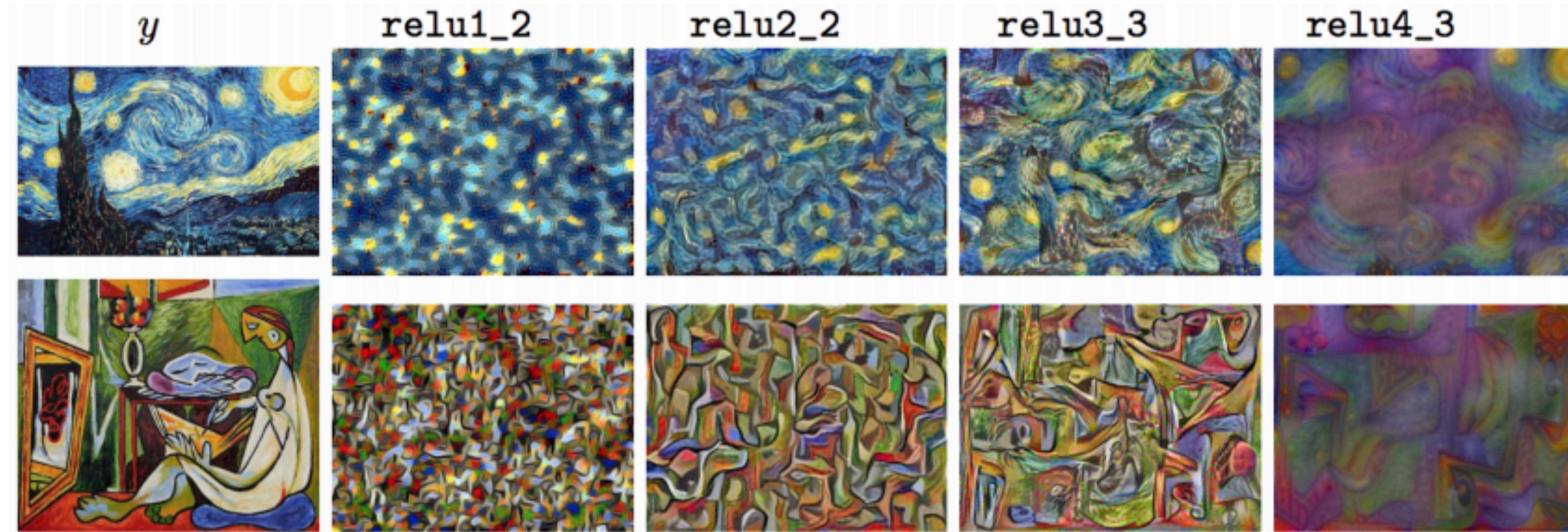


Neural Texture Synthesis



More Texture Synthesis

Texture
synthesis (Gram
reconstruction)



Neural Style Transfer

Content Image



+

Style Image

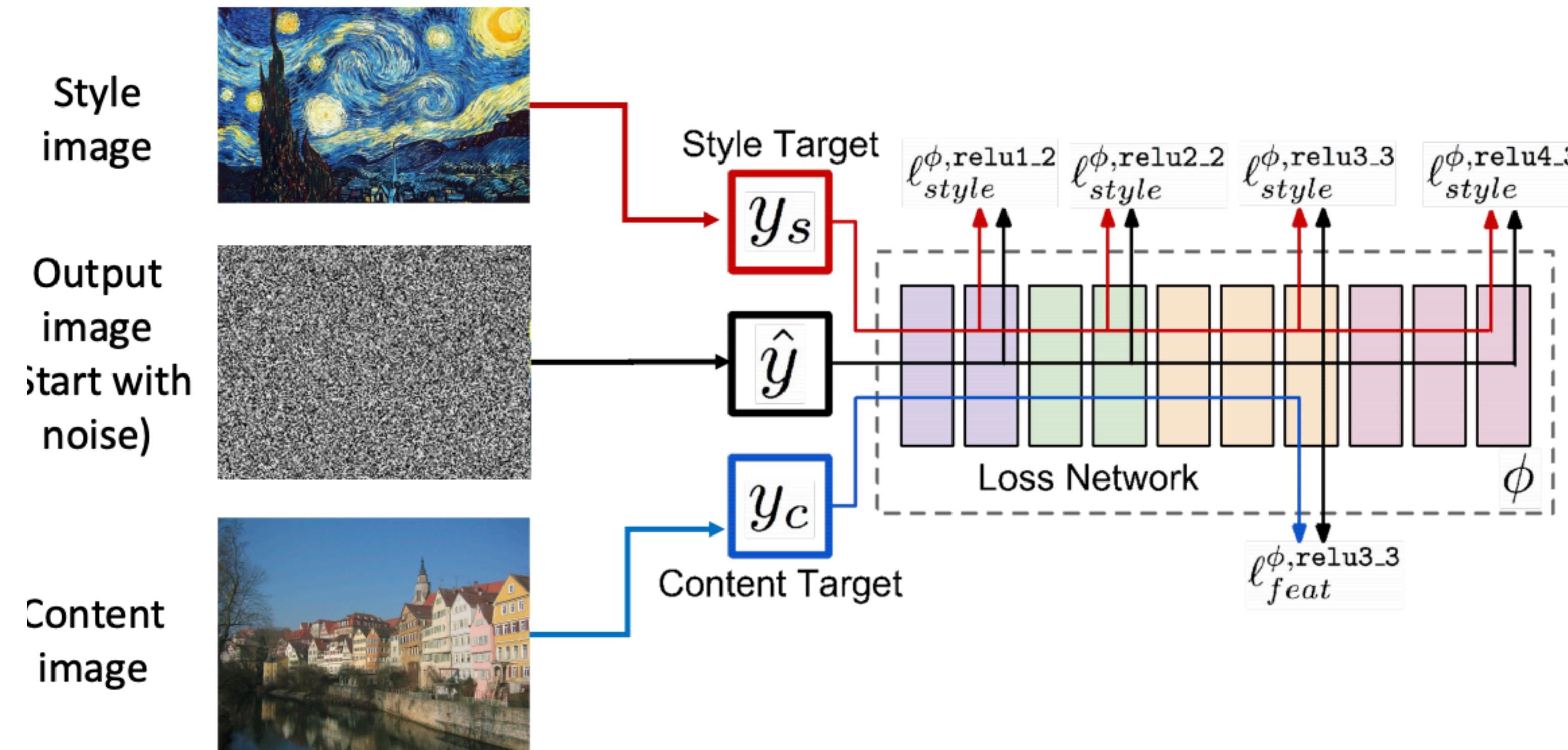


=

Output Image



Neural Texture Synthesis



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

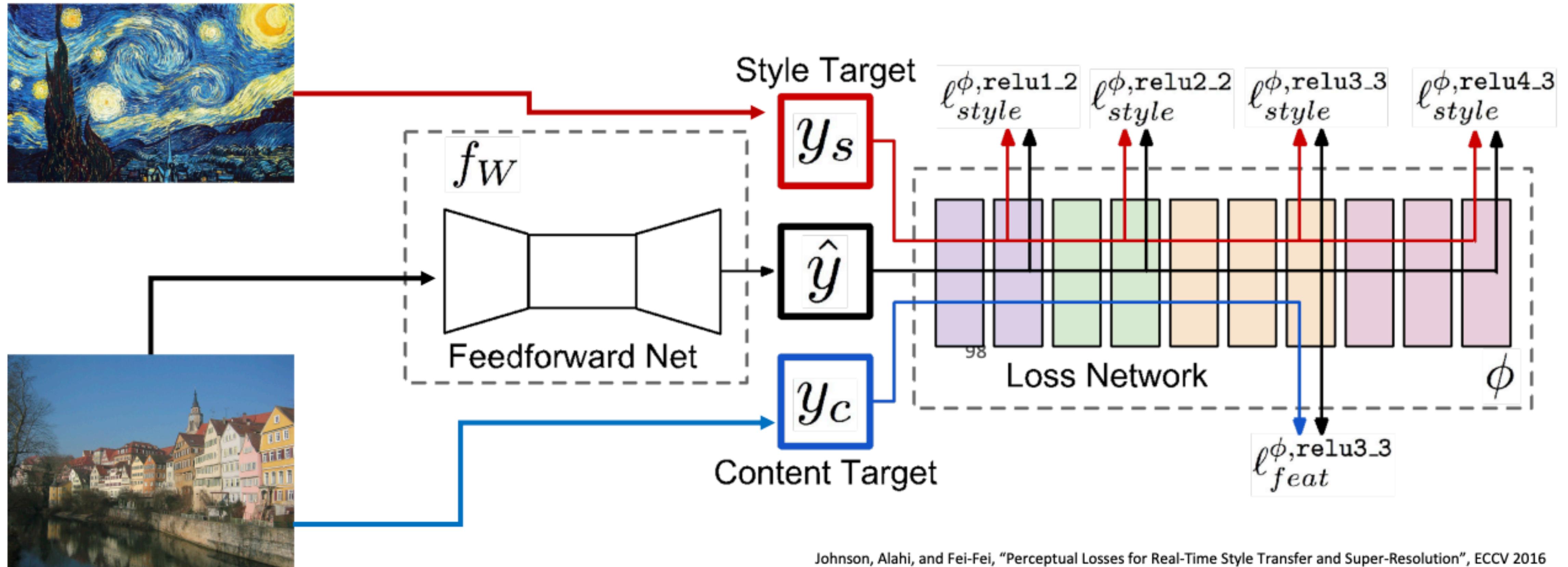
Figure adapted from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016.

Neural Style Transfer



Pytorch implementation: <https://github.com/jcjohnson/neural-style>

Style Transfer via Network



More Results

