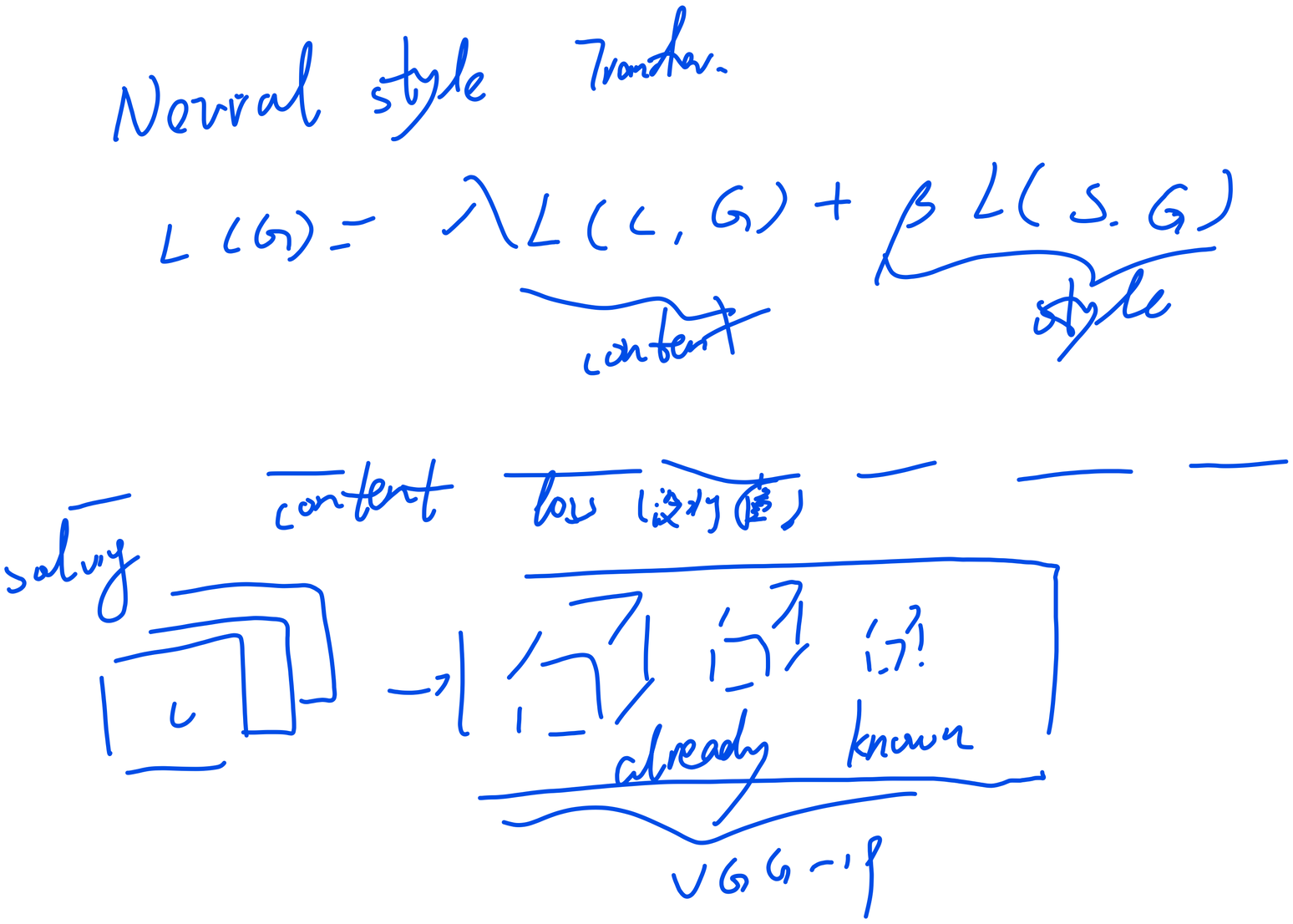
COMP0169: Machine Learning for Visual Computing

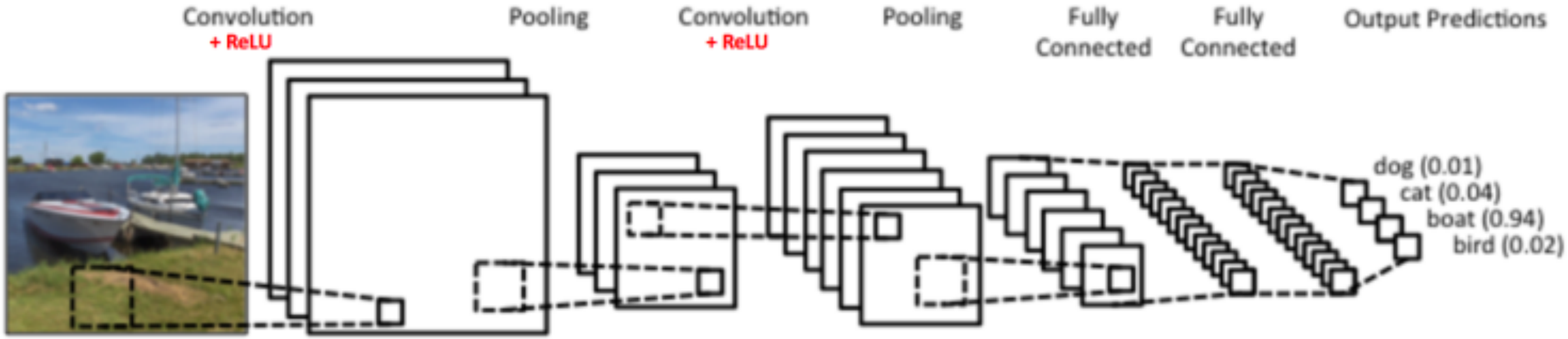
Style vs Content

UCL

Lectures will be Recorded



CNN Recap


Convolution
+ ReLU
Pooling
Convolution
+ ReLU
Pooling
Fully
Connected
Fully
Connected
Output Predictions
dog (0.01)
cat (0.04)
boat (0.94)
bird (0.02)

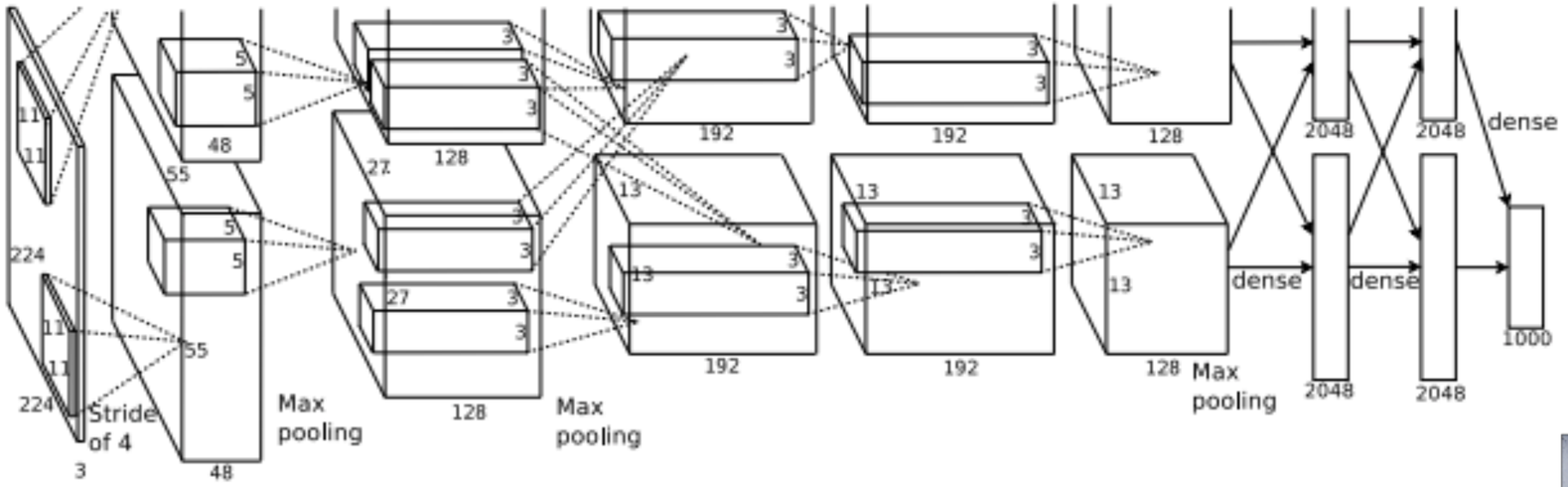
AlexNet

ImageNet Classification with Deep Convolutional Neural Networks

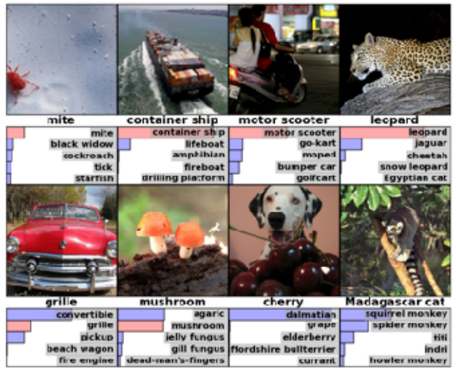
Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

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

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



Class scores for 1000 Classes

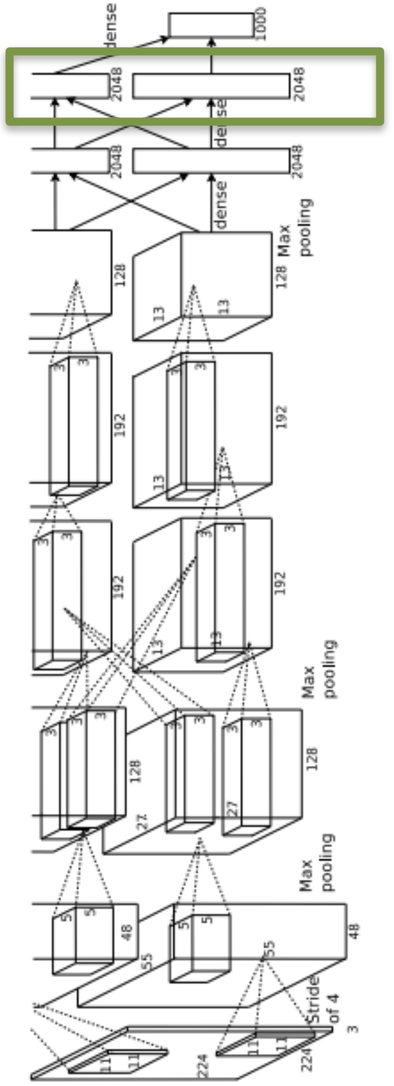


Last Layer

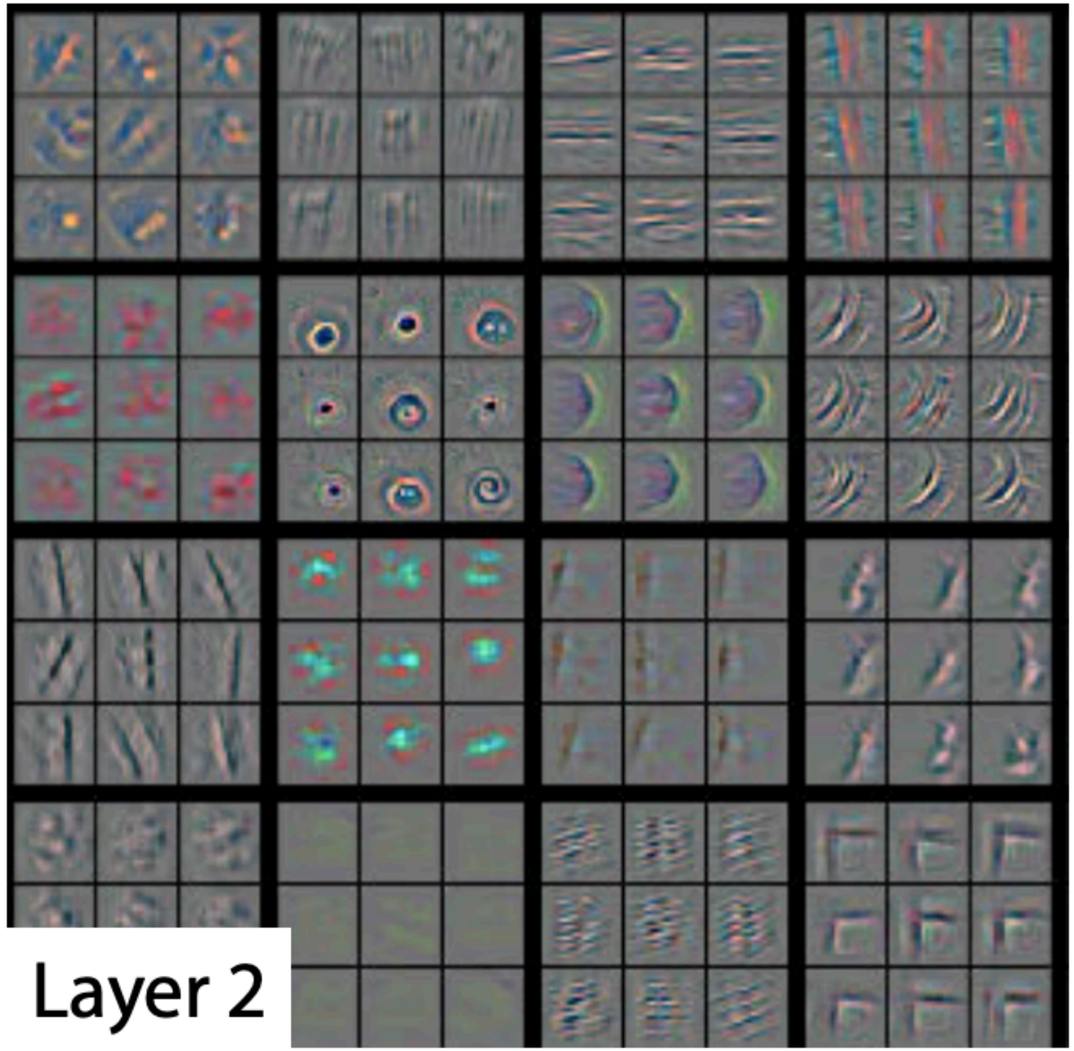
Feature size 2048+2048=4096.

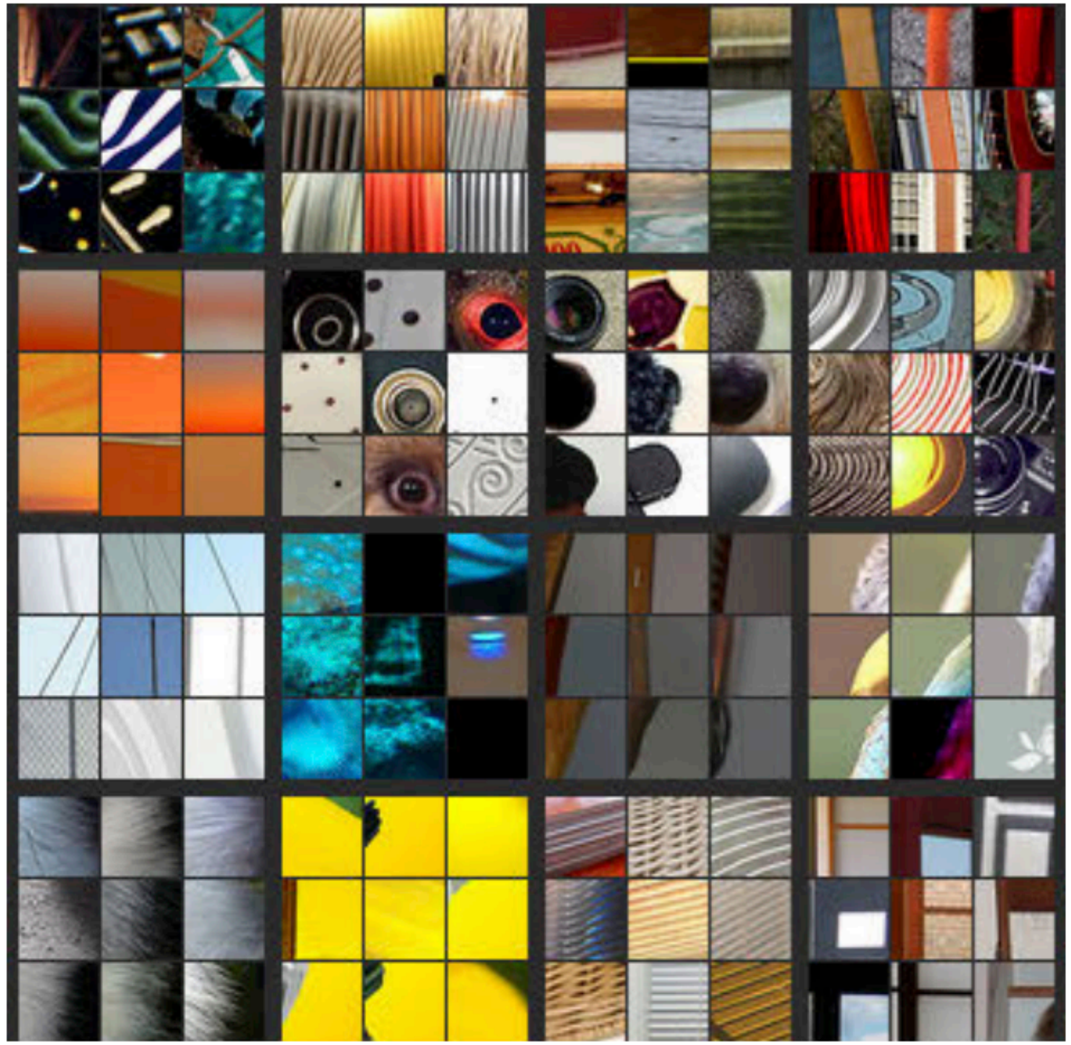
Pretrain network. Run ‘feature extractor’ on many images.
FC7 features.

$$\mathbb{R}^{224\times224\times3} \rightarrow \mathbb{R}^{4096}$$



Feature Visualisation





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

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

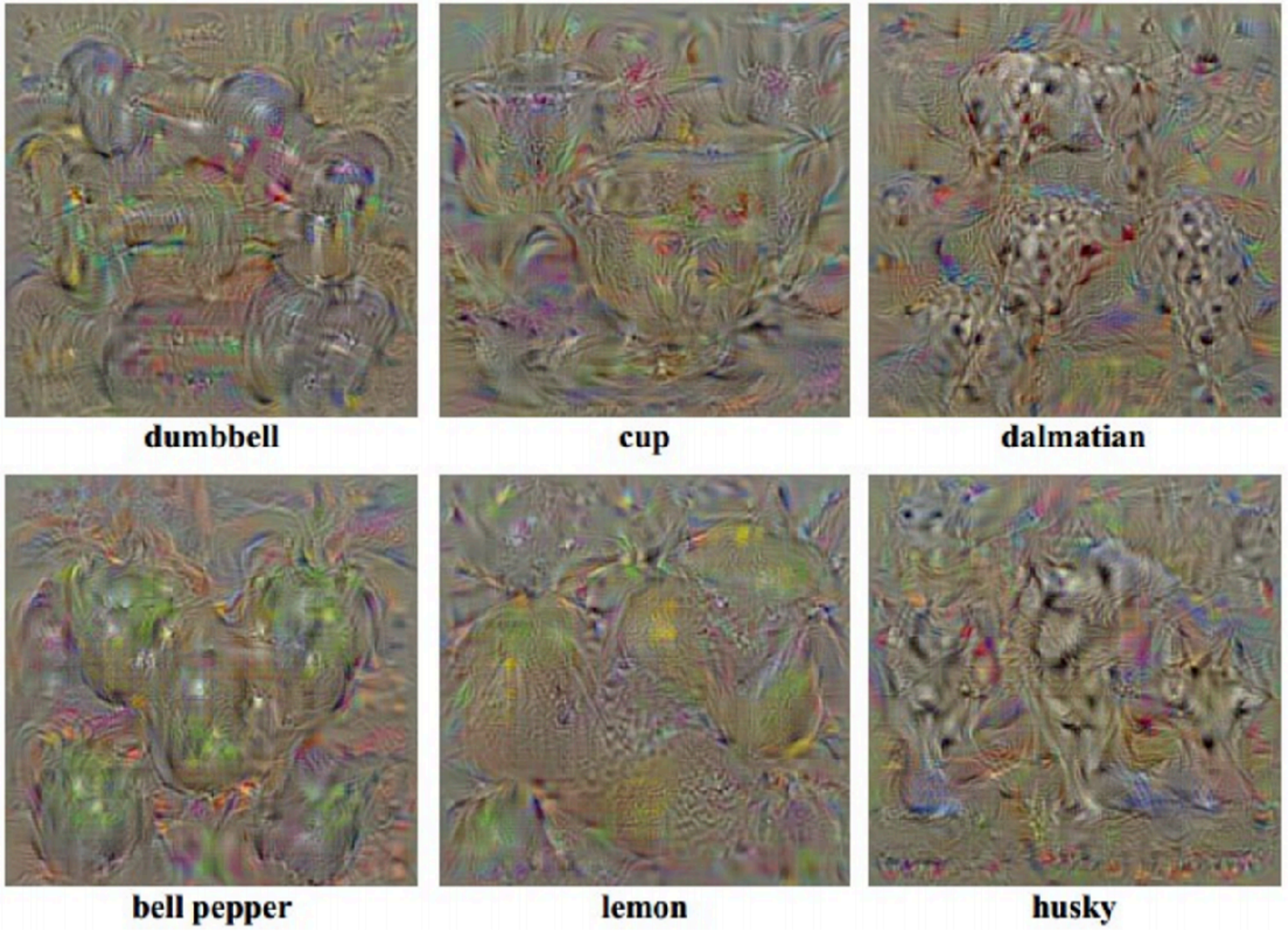
Gradient Ascent $I^\star = \arg\max_I \{S_c(I) - \lambda \|I\|^2\}$

- **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** $I \leftarrow I + \frac{\partial S_c(I)}{\partial I}$

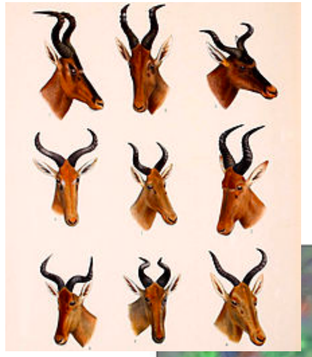
Gradient Ascent

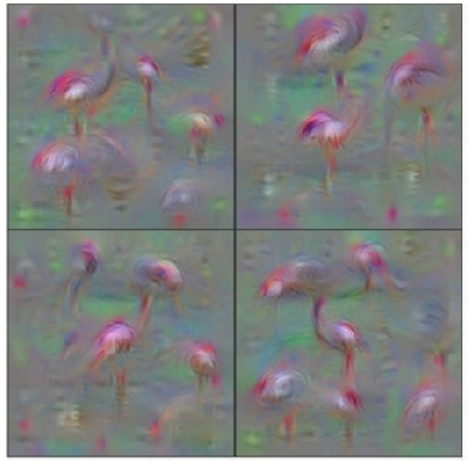
Other regularizers:

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

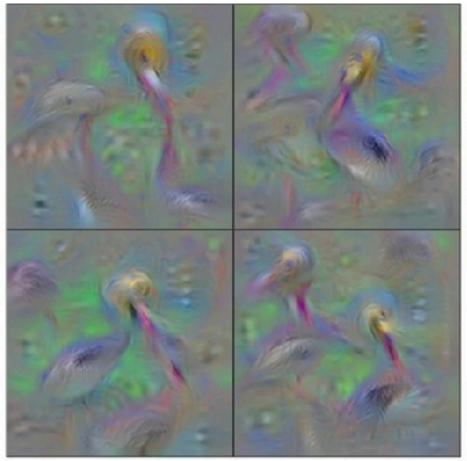


GA Results: More

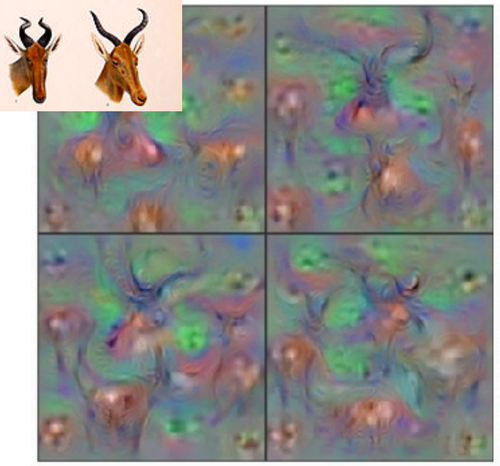




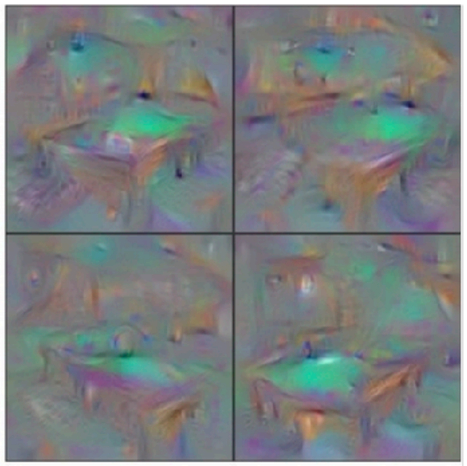
Flamingo



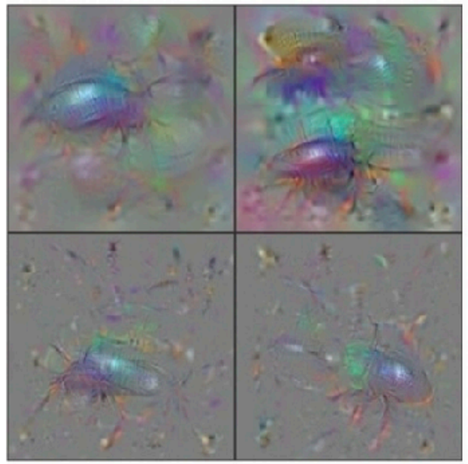
Pelican



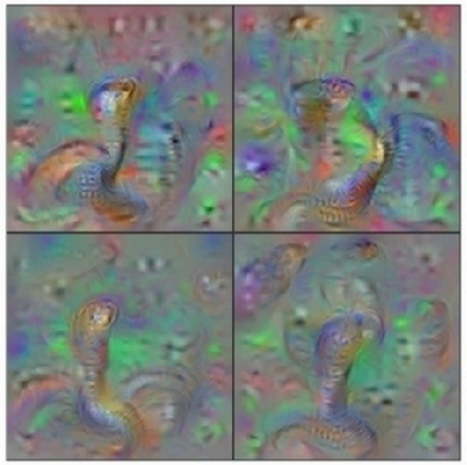
Hartebeest



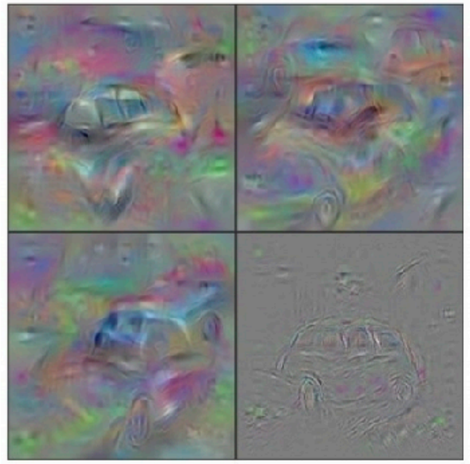
Billiard Table



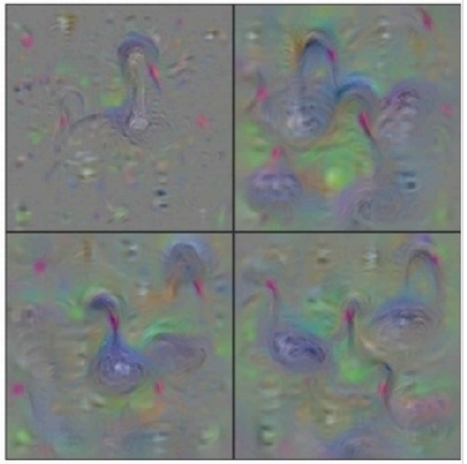
Ground Beetle



Indian Cobra

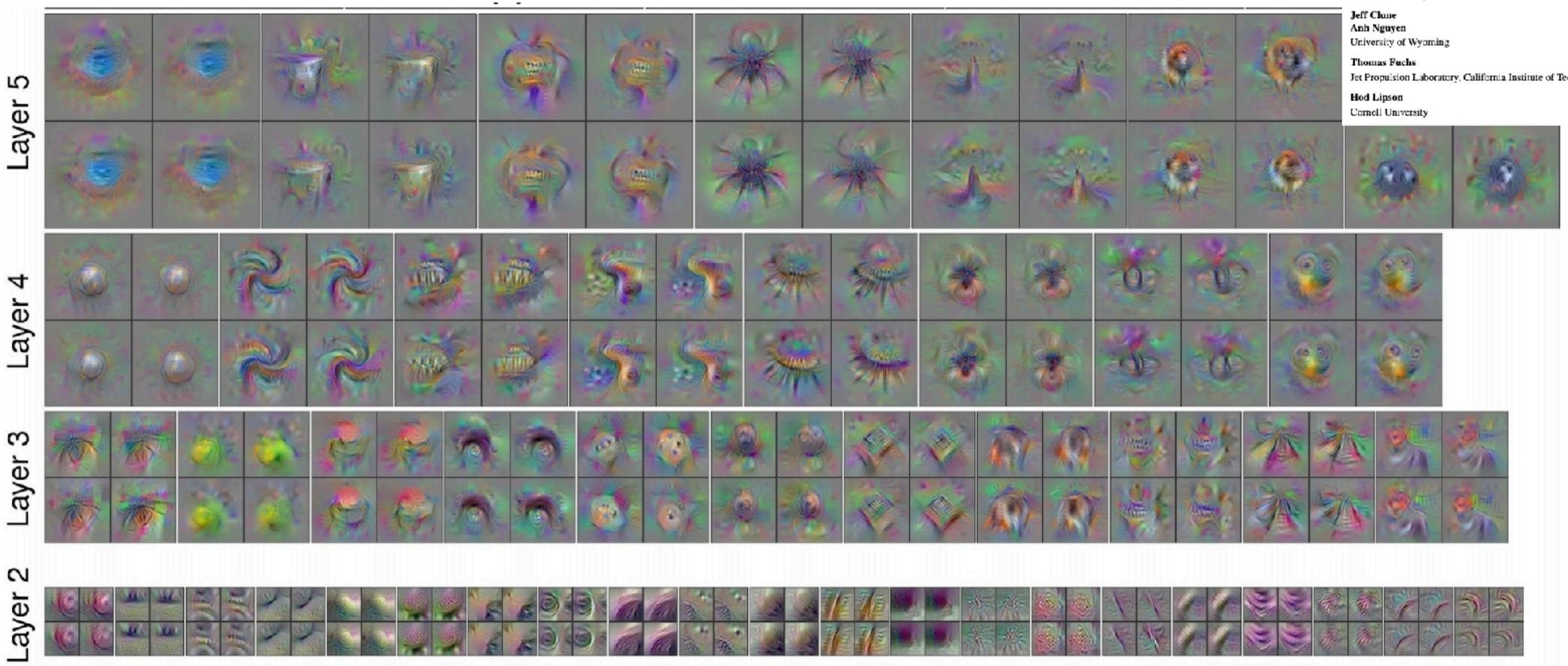


Station Wagon



Black Swan

Visualize Intermediate Features



Understanding Neural Networks Through Deep Visualization

Jason Yosinski YOSINSKI@CS.CORNELL.EDU
Cornell University

Jeff Clune JEFFCLUNE@UWYO.EDU
Anh Nguyen ANGUYEN8@UWYO.EDU
University of Wyoming

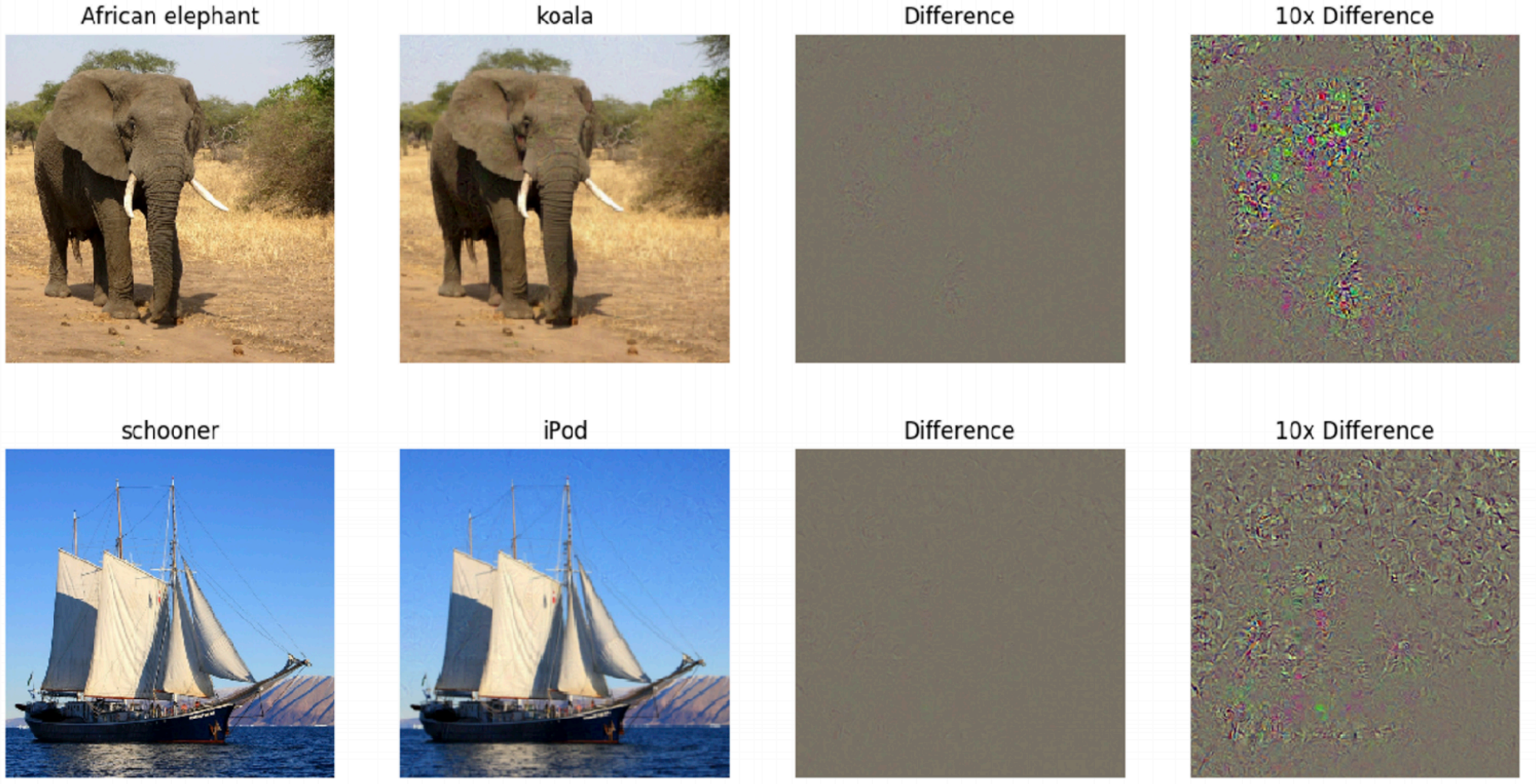
Thomas Fuchs FUCHS@CALTECH.EDU
Jet Propulsion Laboratory, California Institute of Technology

Hod Lipson HOD.LIPSON@CORNELL.EDU
Cornell University

More results: Stronger Regularizers

Relation to Adversarial Training

Adversarial Training: Examples



Feature Inversion

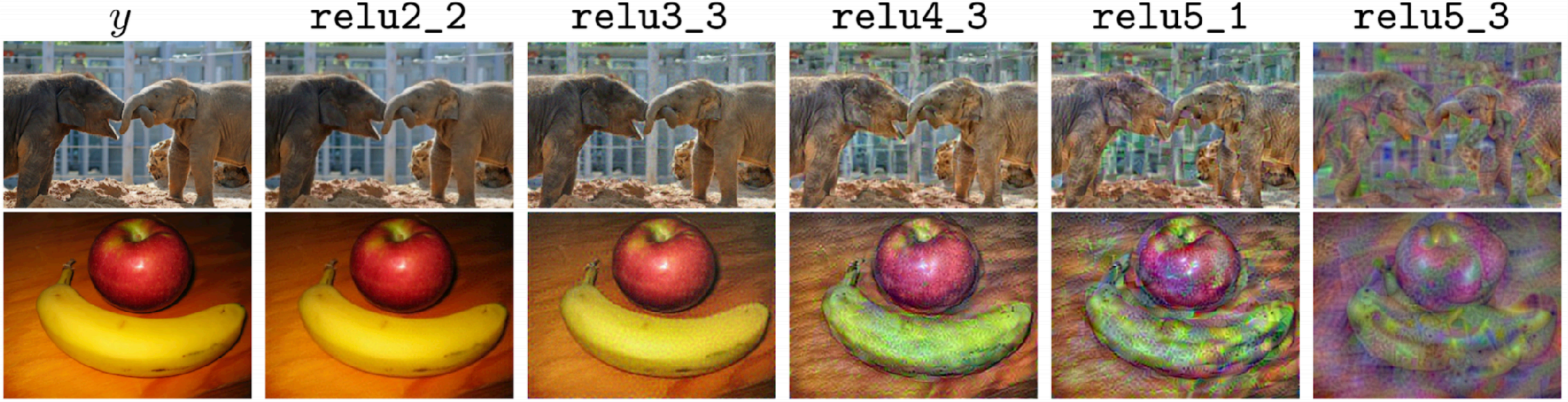
$$x^\star = \arg\min_I \{ l(\Phi(I), \Phi_0) + \lambda \mathcal{R}(I) \}$$

go through the network and find the layer

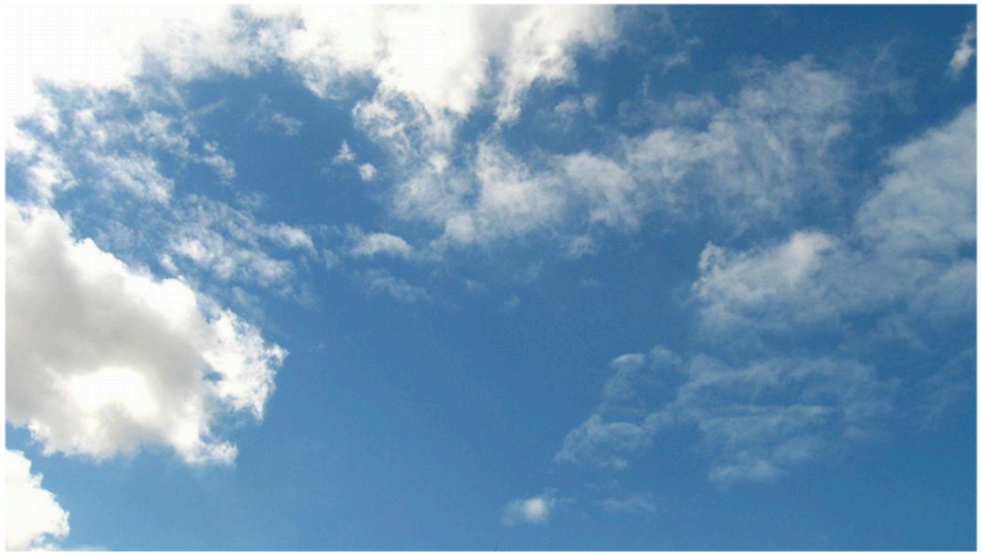
$$l(\Phi(I), \Phi_0) = \|\Phi(I) - \Phi_0\|^2$$

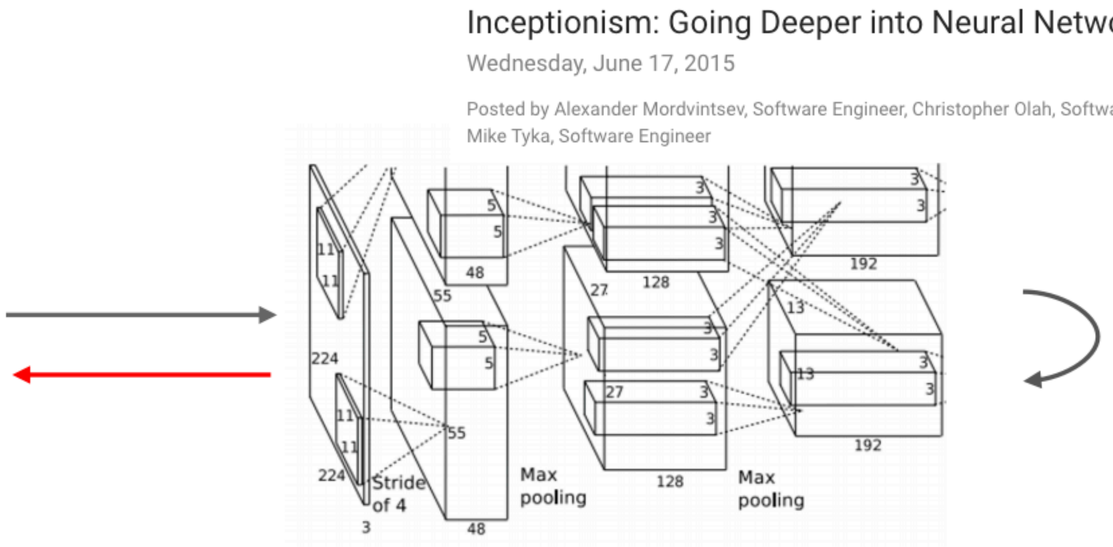
$$\mathcal{R}_\beta(I) = \sum_{i,j} \left(I(i, j+1) - I(i,j) \right)^2 + \left(I(i+1, j) - I(i,j) \right)^2 \Big)^{\frac{\beta}{2}}$$

Feature Inversion



Deep Dream: Amplify Features





- **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**

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

DeepDream

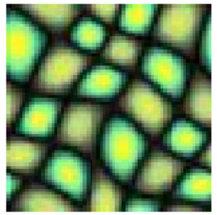
DeepDream

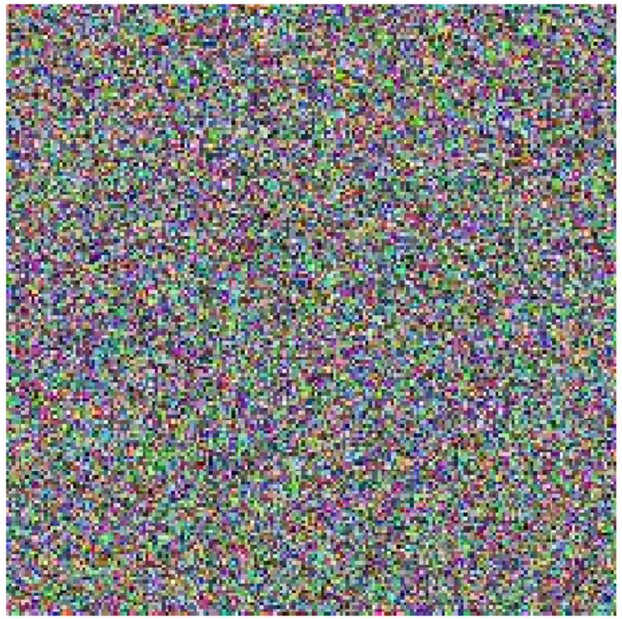
Texture Synthesis

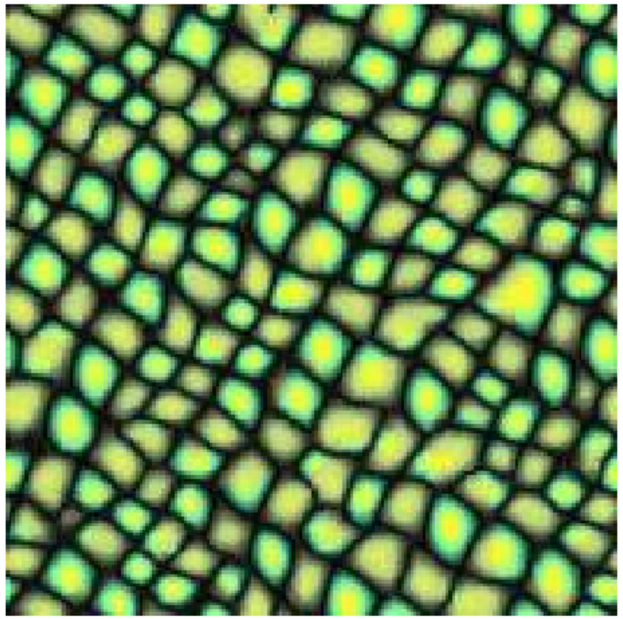
Fast Texture Synthesis using Tree-structured Vector Quantization

Li-Yi Wei Marc Levoy

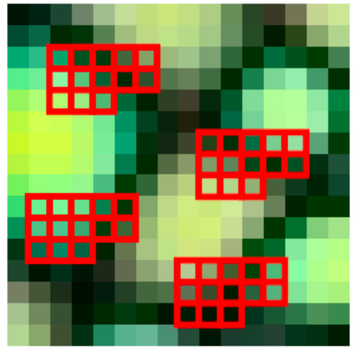
Stanford University *





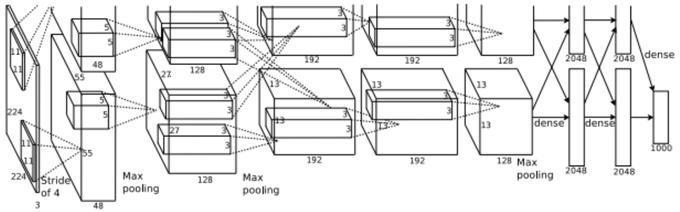


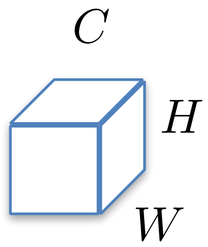
Texture Synthesis Stages



(a)

Neural Texture Synthesis: Gram Matrix





$$G$$

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

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

Neural Texture Synthesis

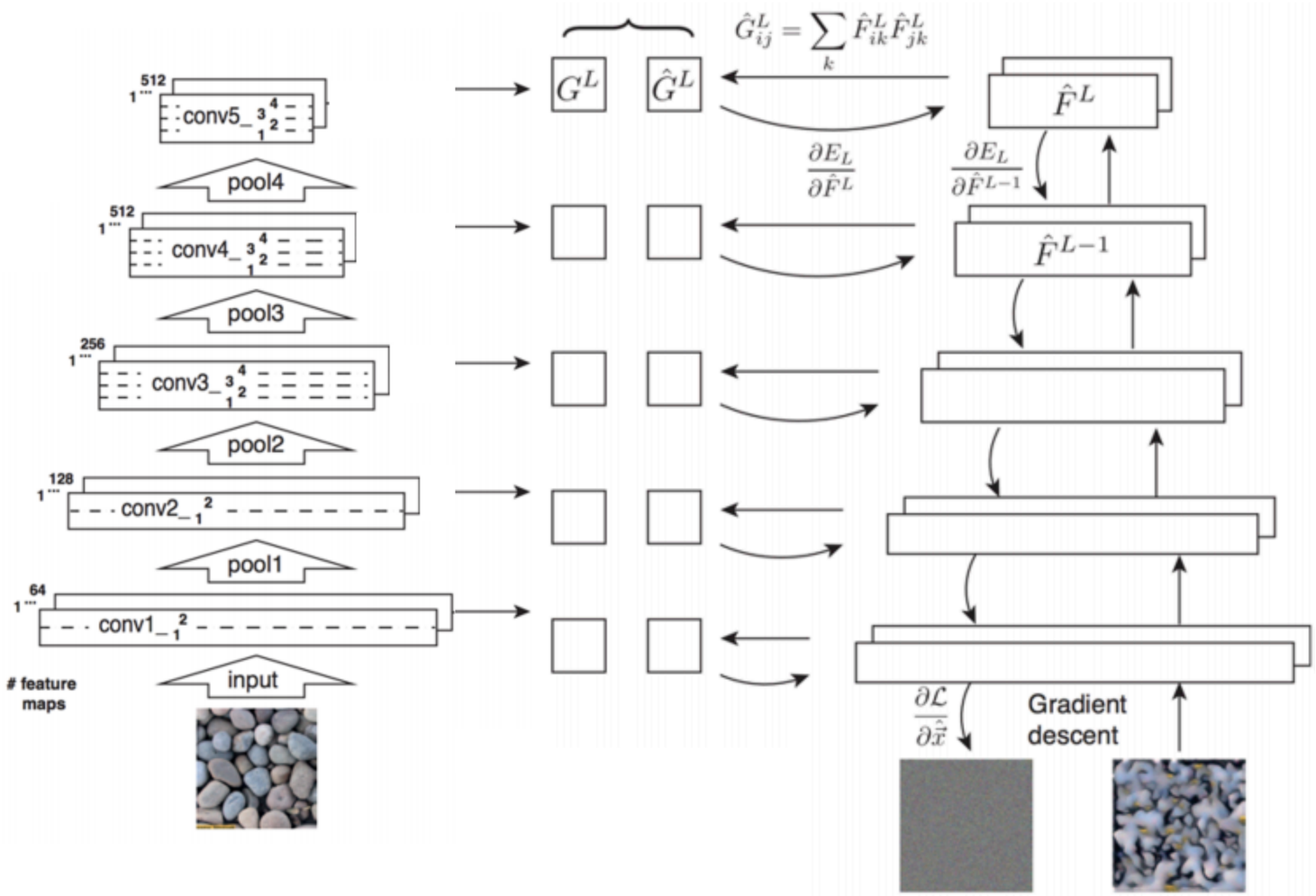
$$E_l = \frac{1}{N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - \hat{G}_{ij}^l \right)$$

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

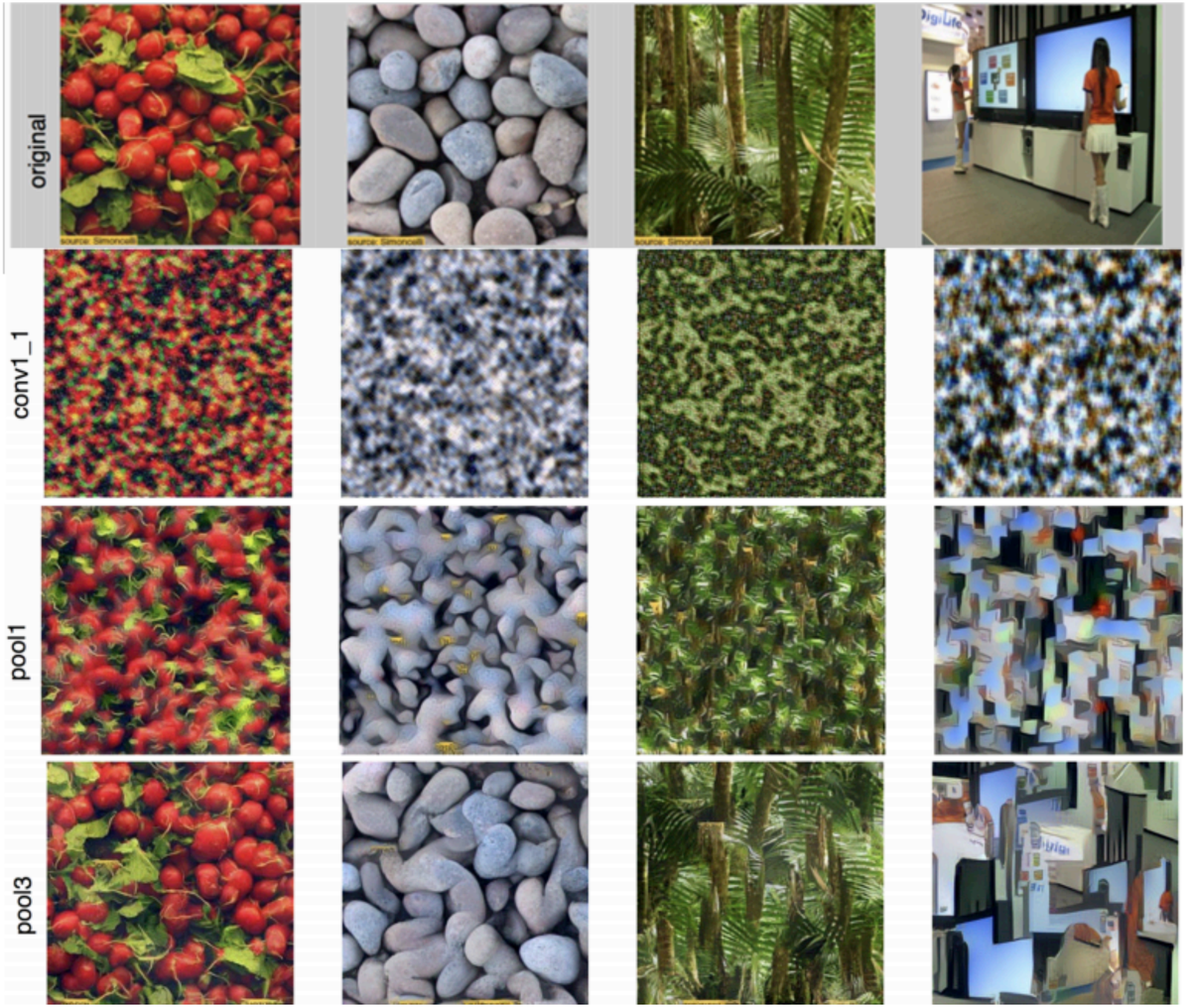
1. **Compute VGG features.**
2. **Given image I, compute features at different levels.**
3. **Compute Gram matrices at different levels.**

$$G_{ij}^l := \sum_k F_{ik}^l F_{jk}^l$$

4. **Initialize with random (noise) matrix.**
5. **Compute features at each level.**
6. **Compute loss, backprop wrt image pixel. Goto step 5.**

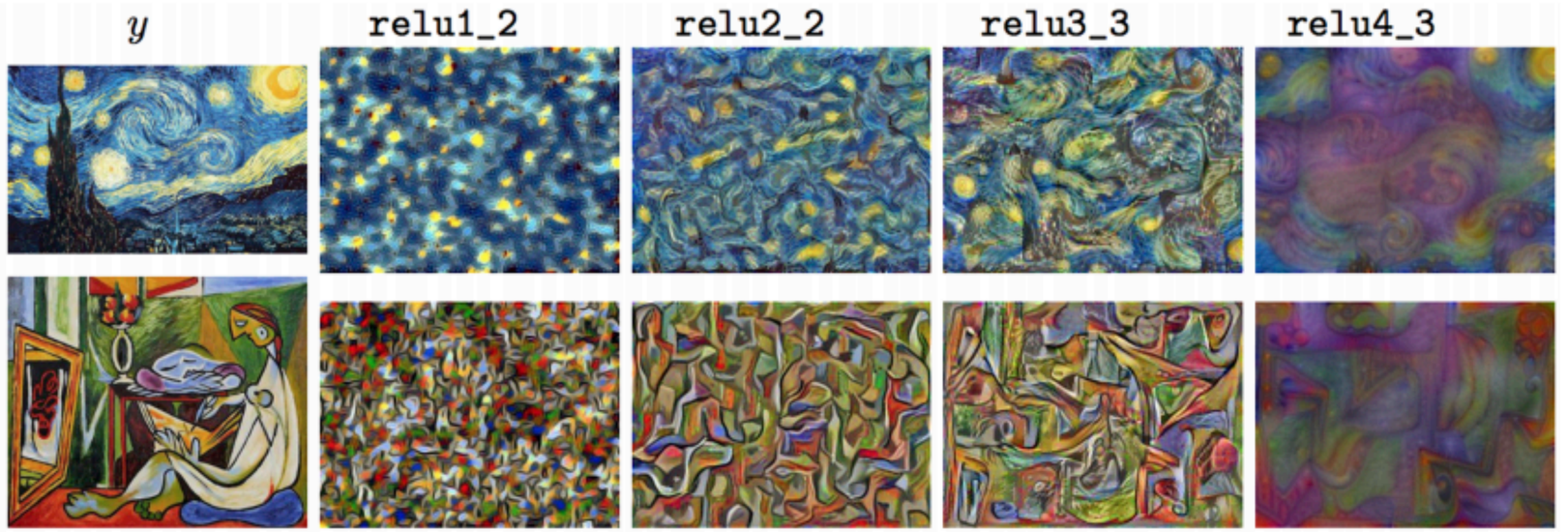


Neural Texture Synthesis



More Texture Synthesis

Texture synthesis (Gram reconstruction)



Neural Style Transfer

Content Image

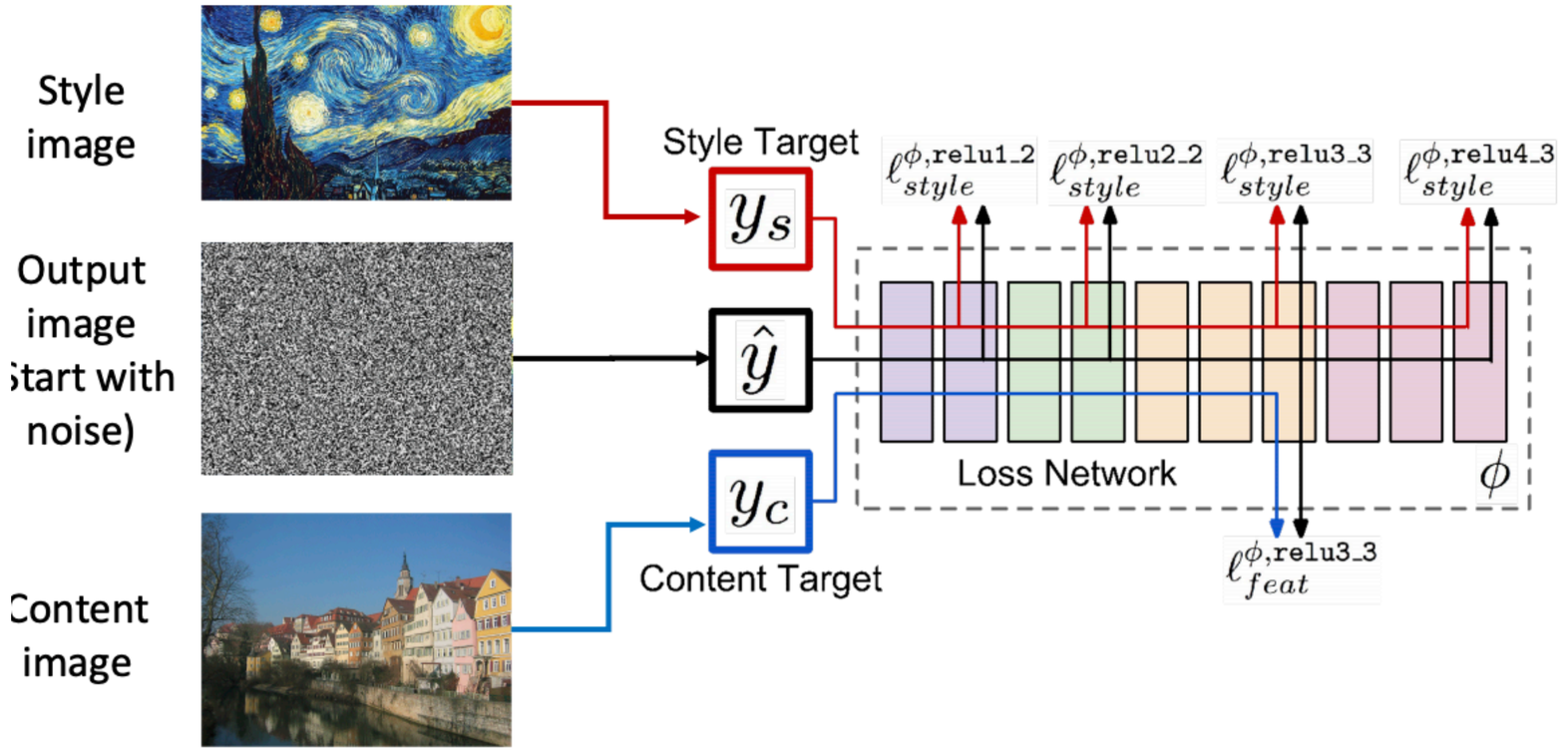
+

Style Image

=

Output Image

Neural Texture Synthesis



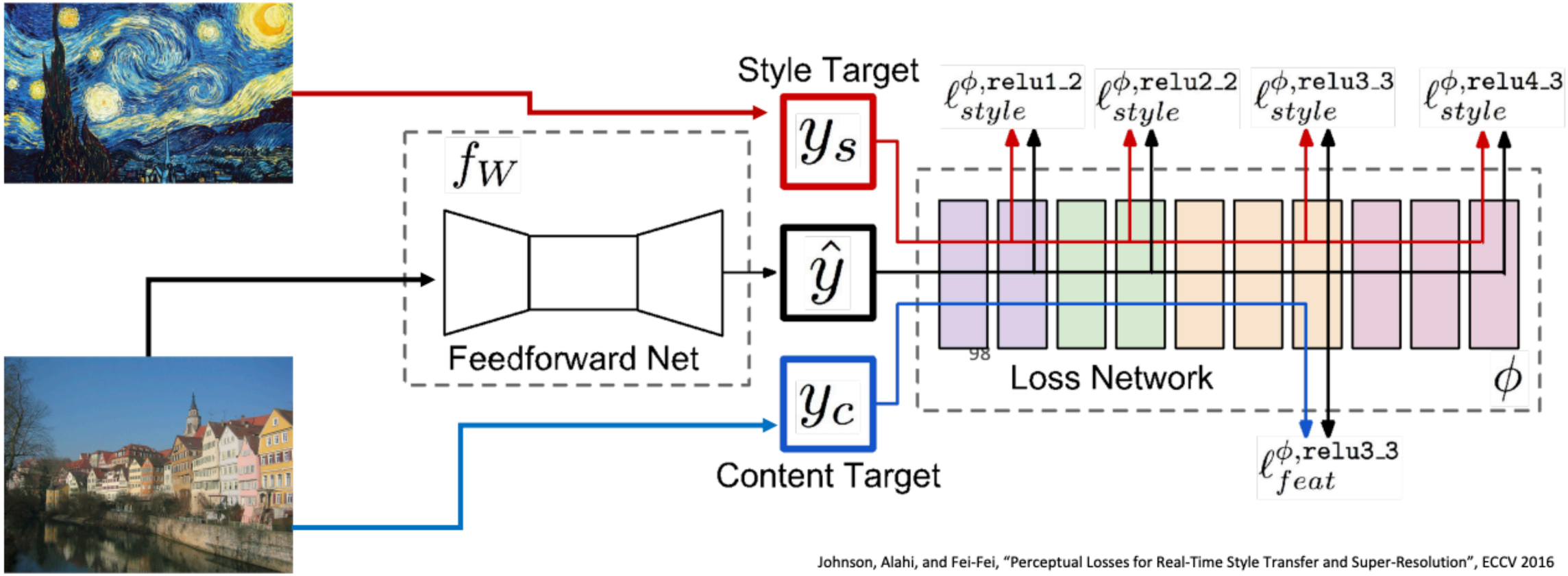
Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016

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



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

More Results