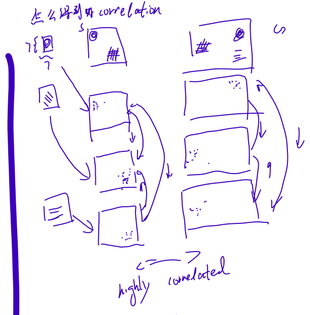
COMP0169: Machine Learning for Visual Computing

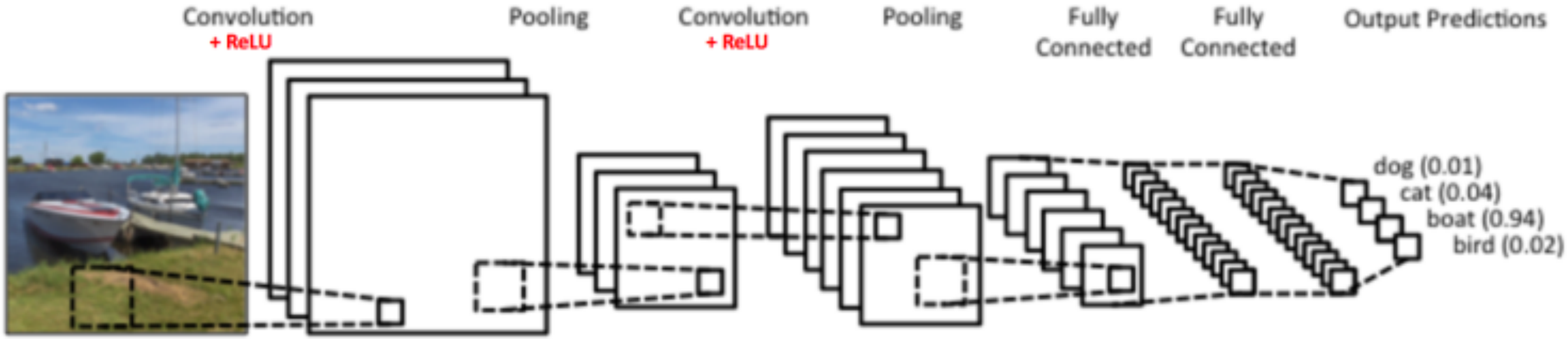
Style vs Content

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)

Neural Style Transfer

Content (C)

Generation (G)

Style (S)

$L_c + L_s \rightarrow G$

$\min$

Gram Matrix

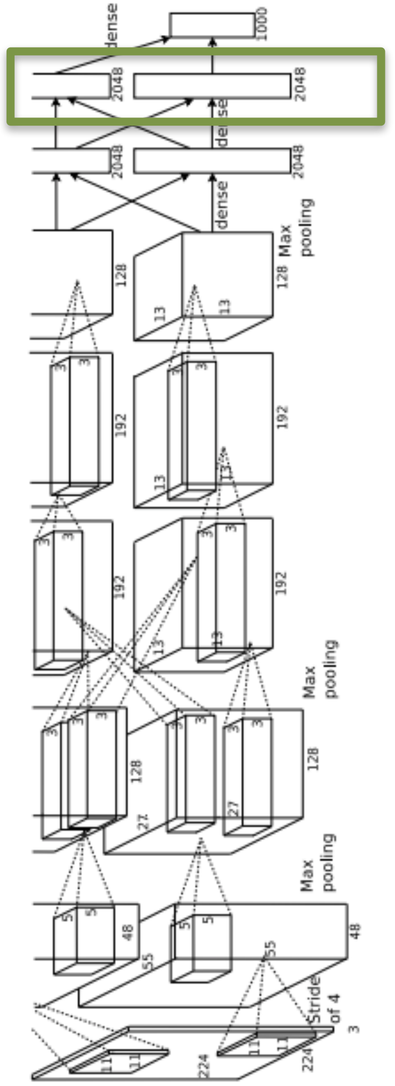
Last Layer

Feature size 2048+2048=4096.

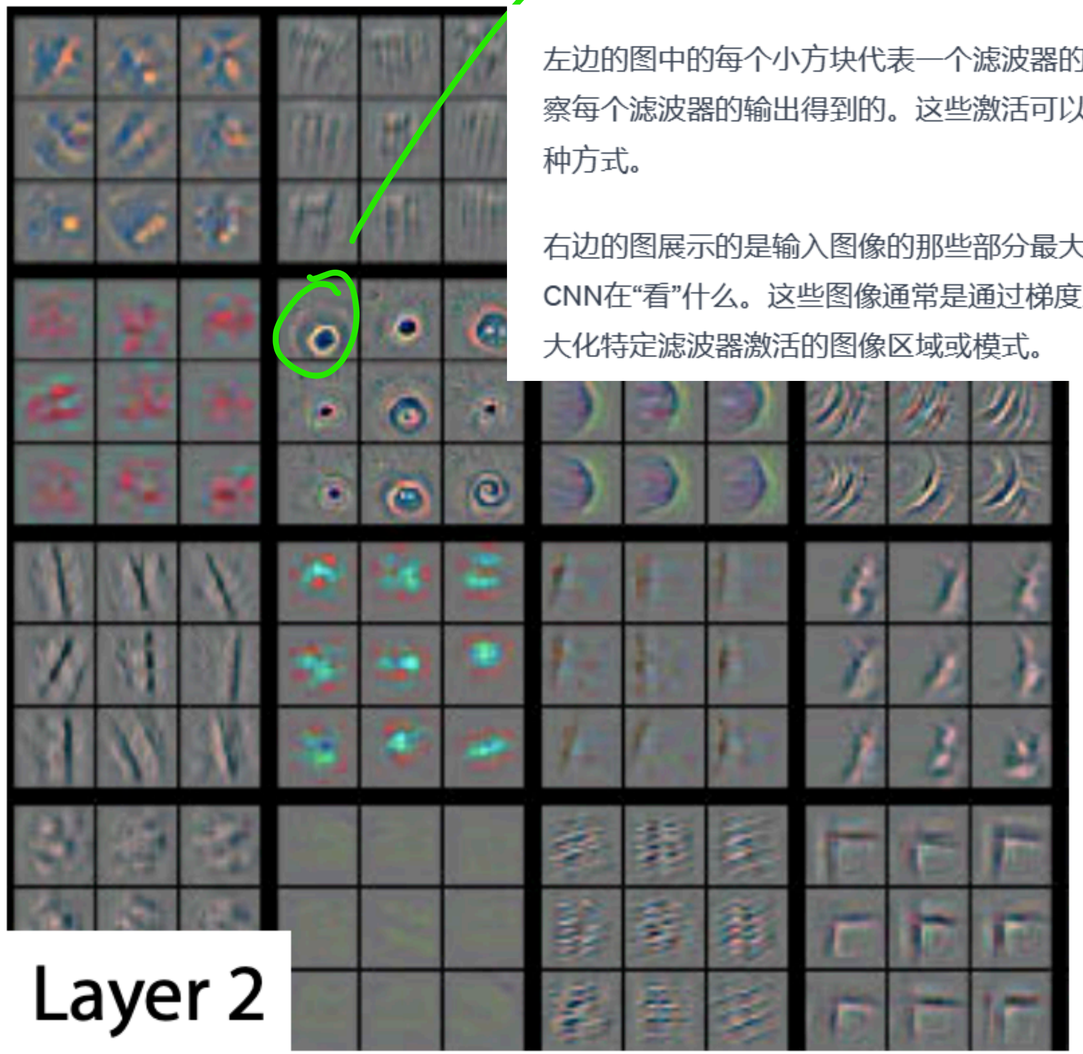
Pretrain network. Run ‘feature extractor’ on many images.
FC7 features.

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

池化层
(为过网条件)

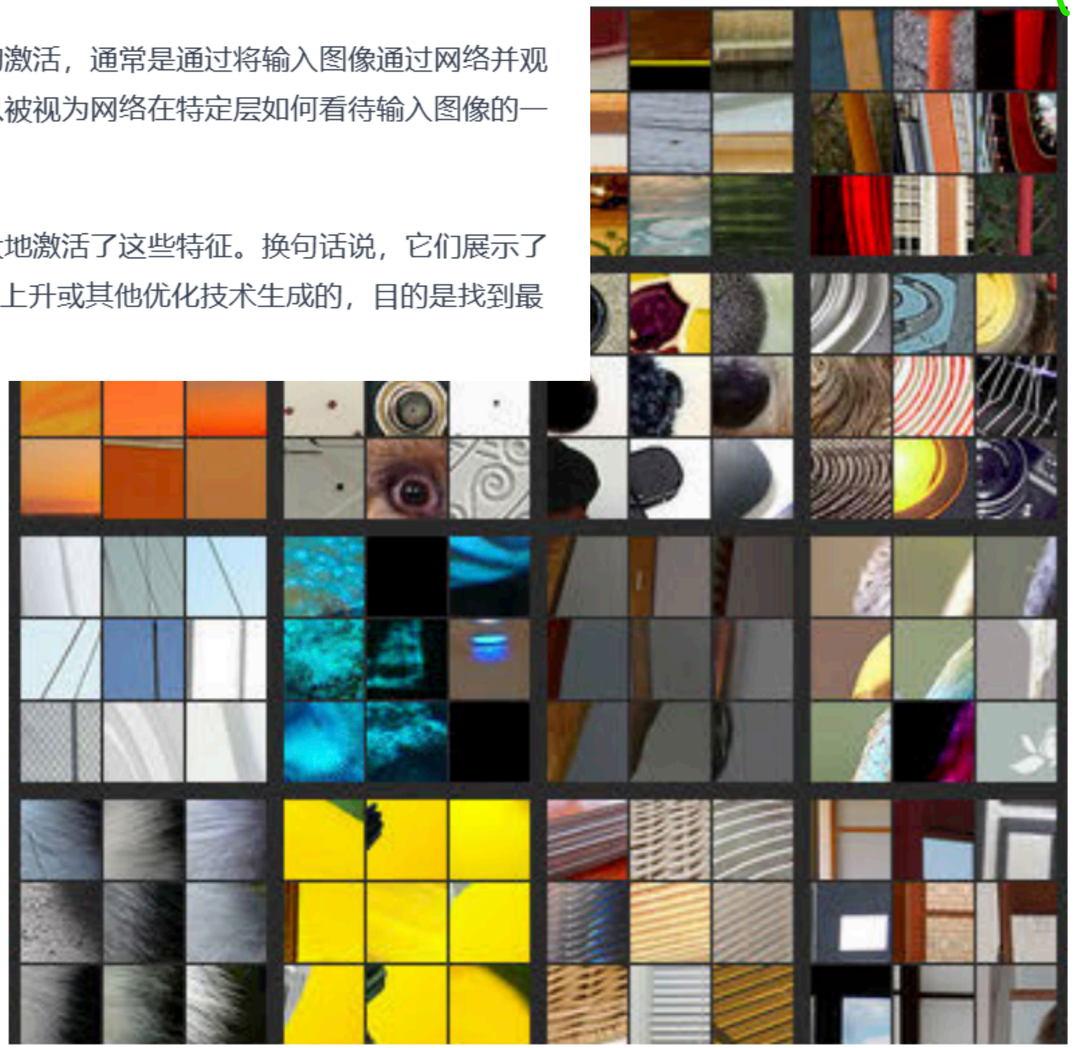


Feature Visualisation

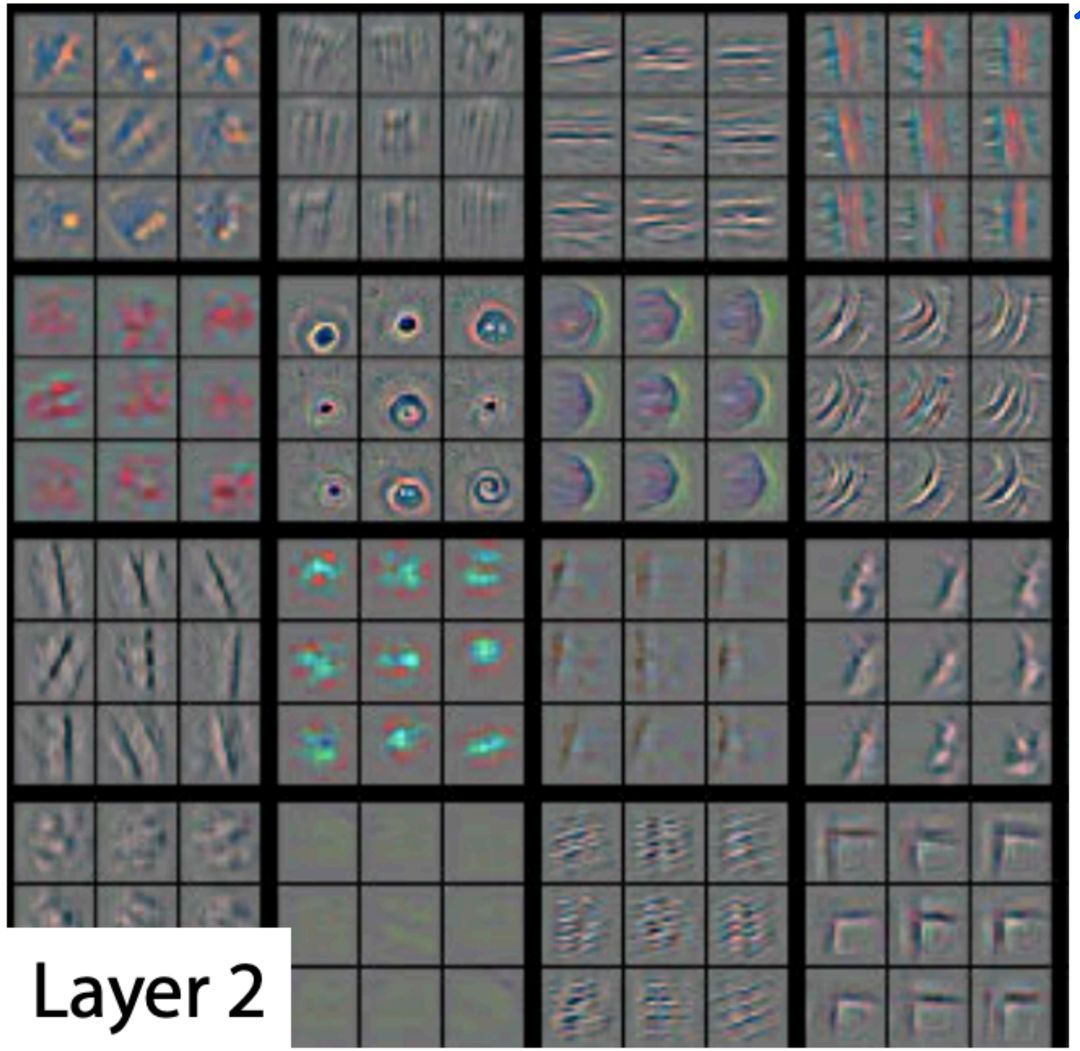


左边的图中的每个小方块代表一个滤波器的激活，通常是通过将输入图像通过网络并观察每个滤波器的输出得到的。这些激活可以被视为网络在特定层如何看待输入图像的一种方式。

右边的图展示的是输入图像的那些部分最大地激活了这些特征。换句话说，它们展示了CNN在“看”什么。这些图像通常是通过梯度上升或其他优化技术生成的，目的是找到最大化特定滤波器激活的图像区域或模式。



Feature Visualisation



Feature Inversion

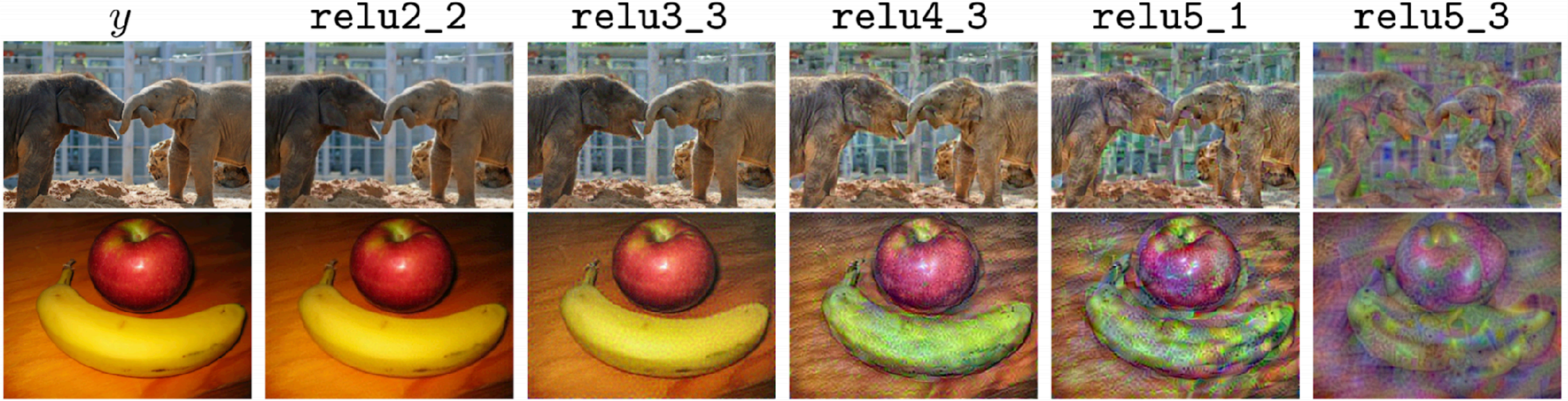
$$x^\star = \arg\min_I \{ l(\Phi(I), \Phi_0) + \lambda \mathcal{R}(I) \}$$

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

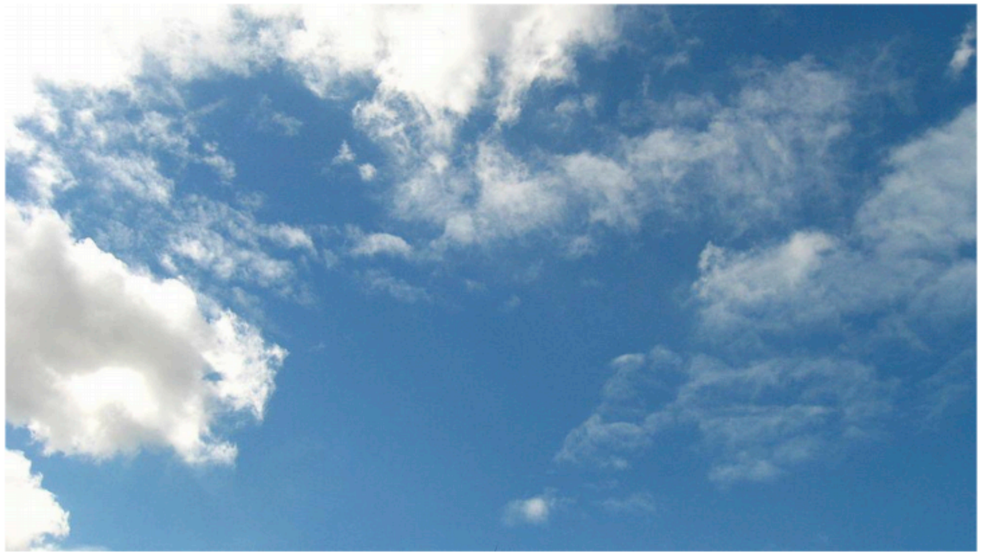
$$\mathcal{R}_\beta(I) = \sum_{i,j} \left(I(i, j+1) - I(i,j) \right)^2 + \left(I(i+1, j) - I(i,j) \right)^2 \Big)^{\frac{\beta}{2}}$$

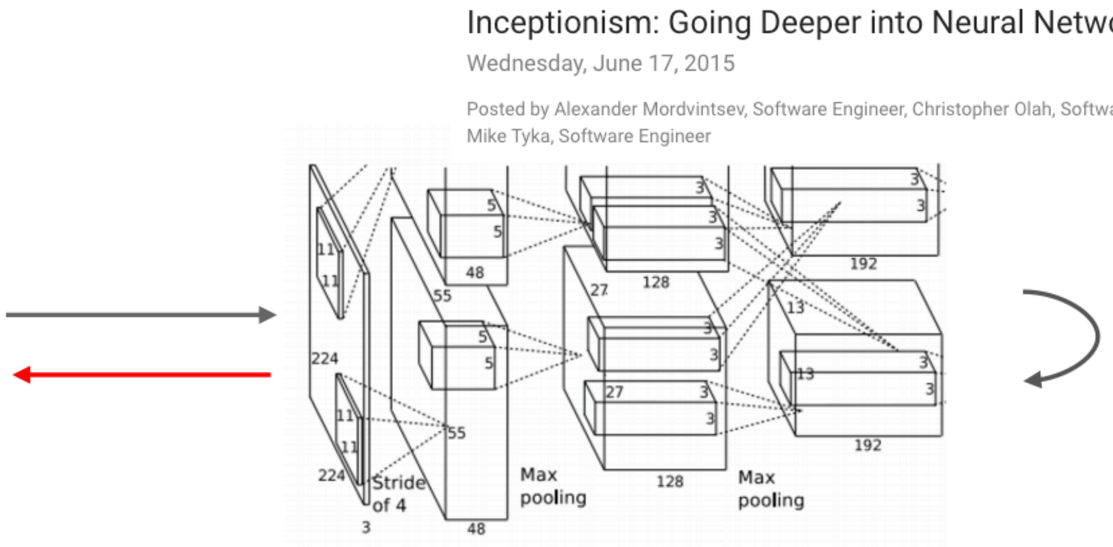
↑ Style (Z方向 position)

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

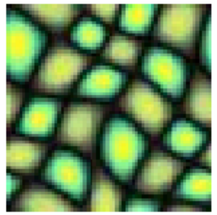
DeepDream

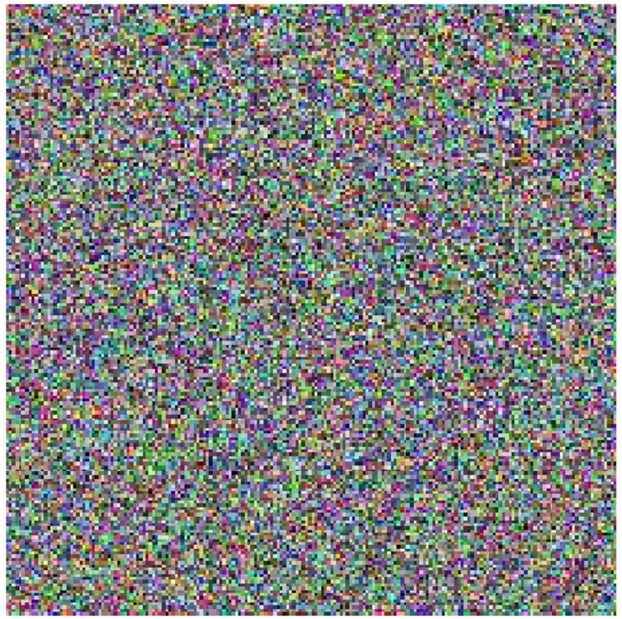
Texture Synthesis

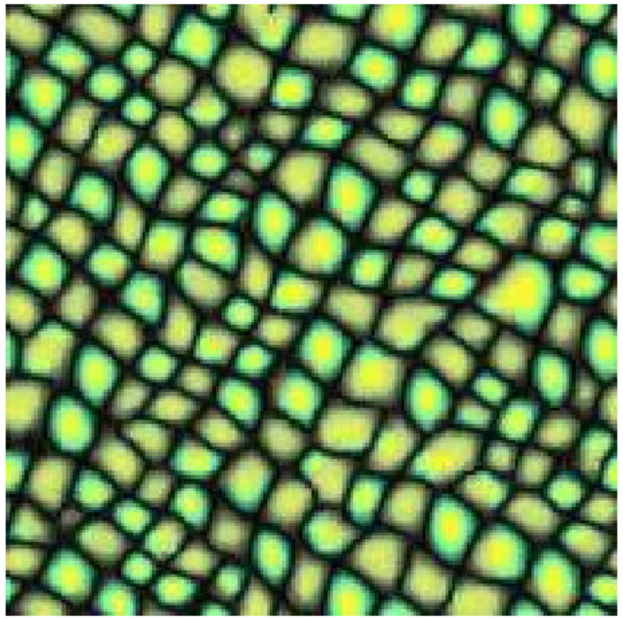
Fast Texture Synthesis using Tree-structured Vector Quantization

Li-Yi Wei Marc Levoy

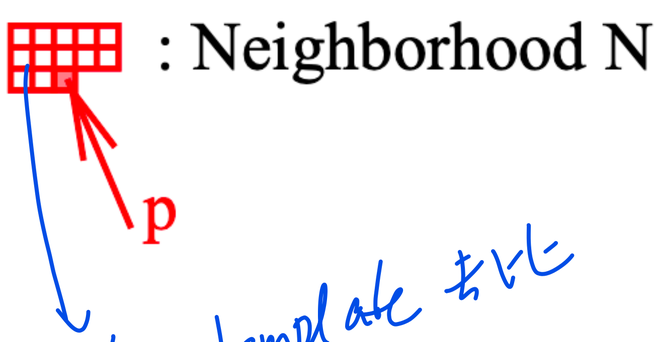
Stanford University *

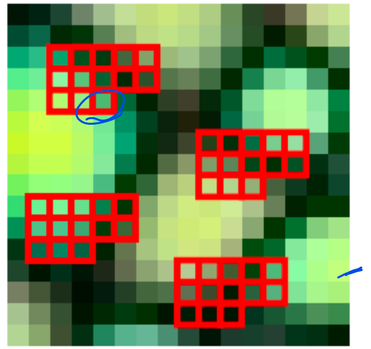






Texture Synthesis Stages





(a)

Matching Features vs Statistics

$$\sum \| X_i^g - X_i^c \|^2$$

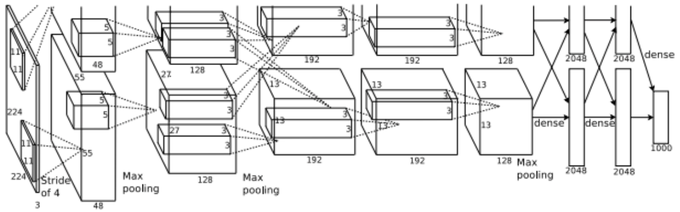
$$\sum X_i X_j^T$$

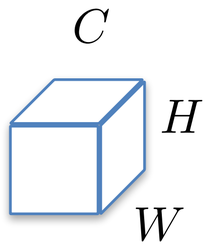
$$cxc$$

$$\sum X_j^g X_j^T$$

$$cxc$$

Neural Texture Synthesis: Gram Matrix





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

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

Neural Texture Synthesis

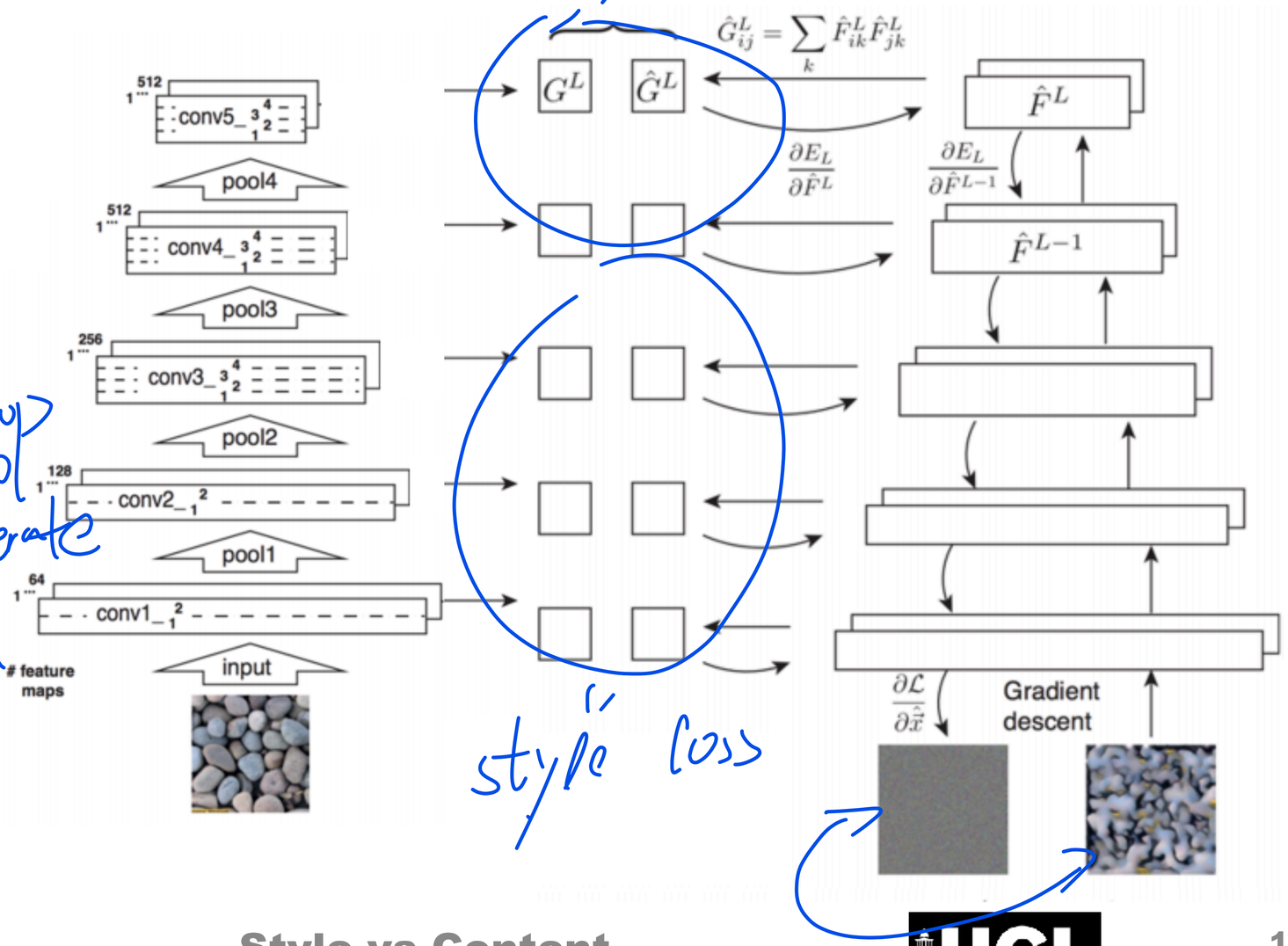
$$E_l = \frac{1}{N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - \hat{G}_{ij}^l \right)$$

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

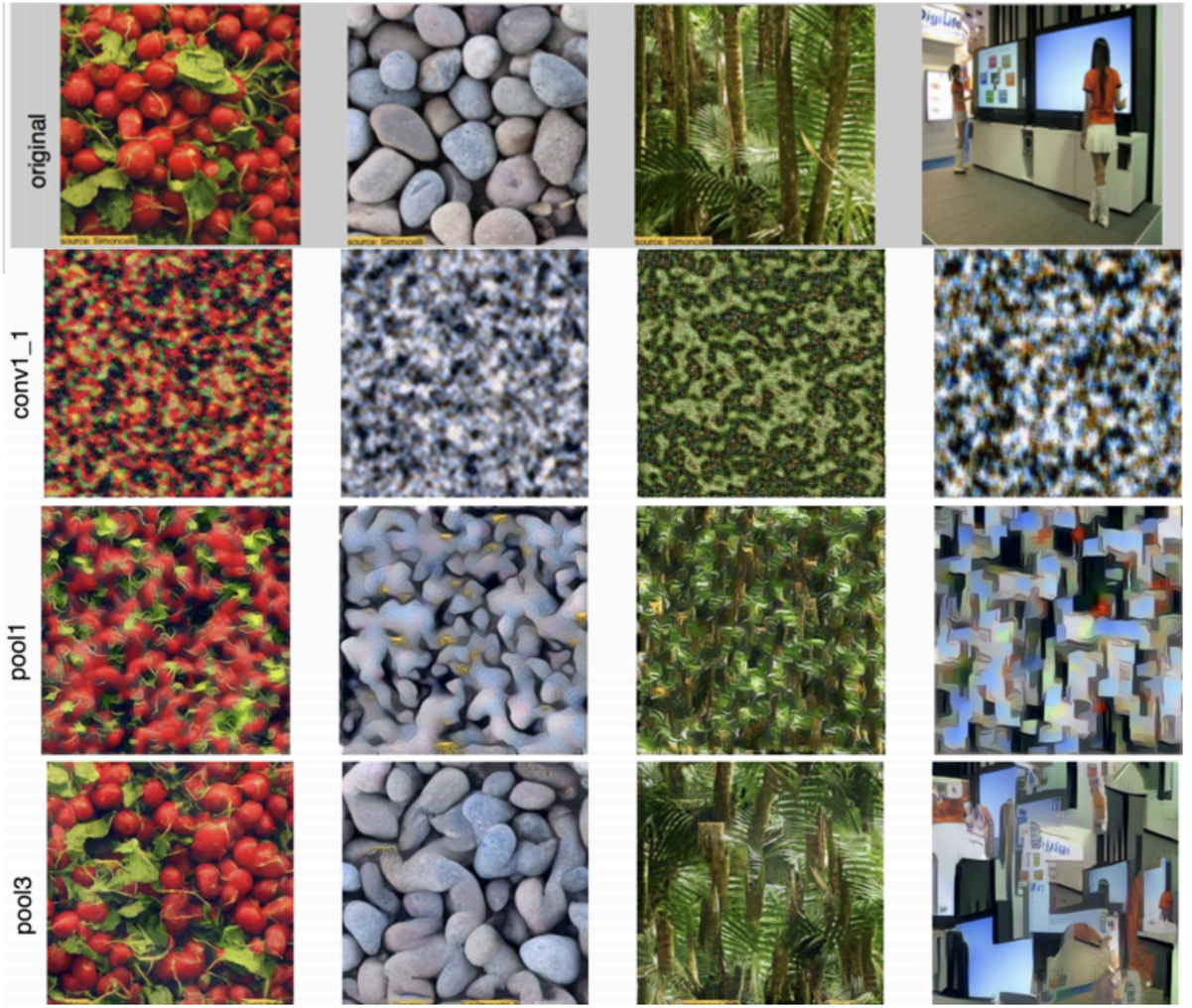
1. **Compute VGG features.**
2. **Given image I, compute features at different levels.**
3. **Compute Gram matrices at different levels.**

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

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

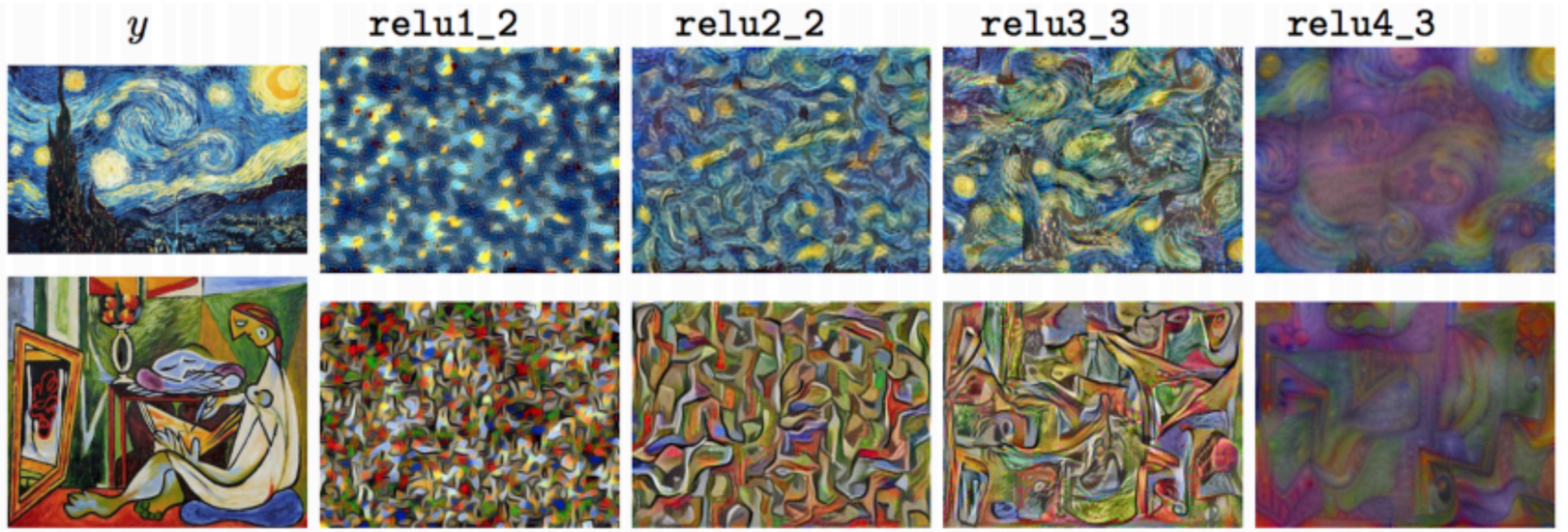


Neural Texture Synthesis



More Texture Synthesis

Texture synthesis (Gram reconstruction)



Neural Style Transfer

Content Image

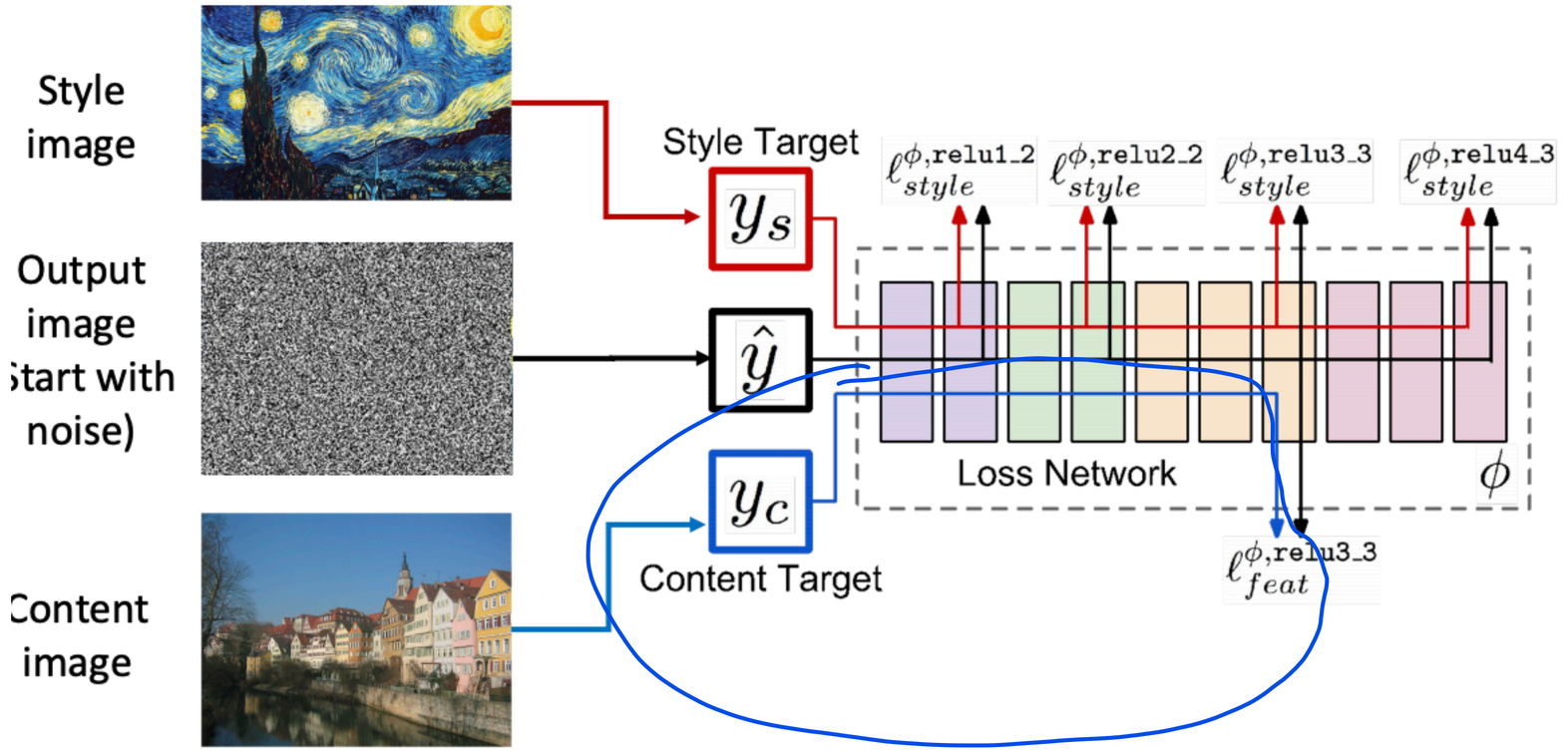
+

Style Image

=

Output Image

Neural Texture Synthesis



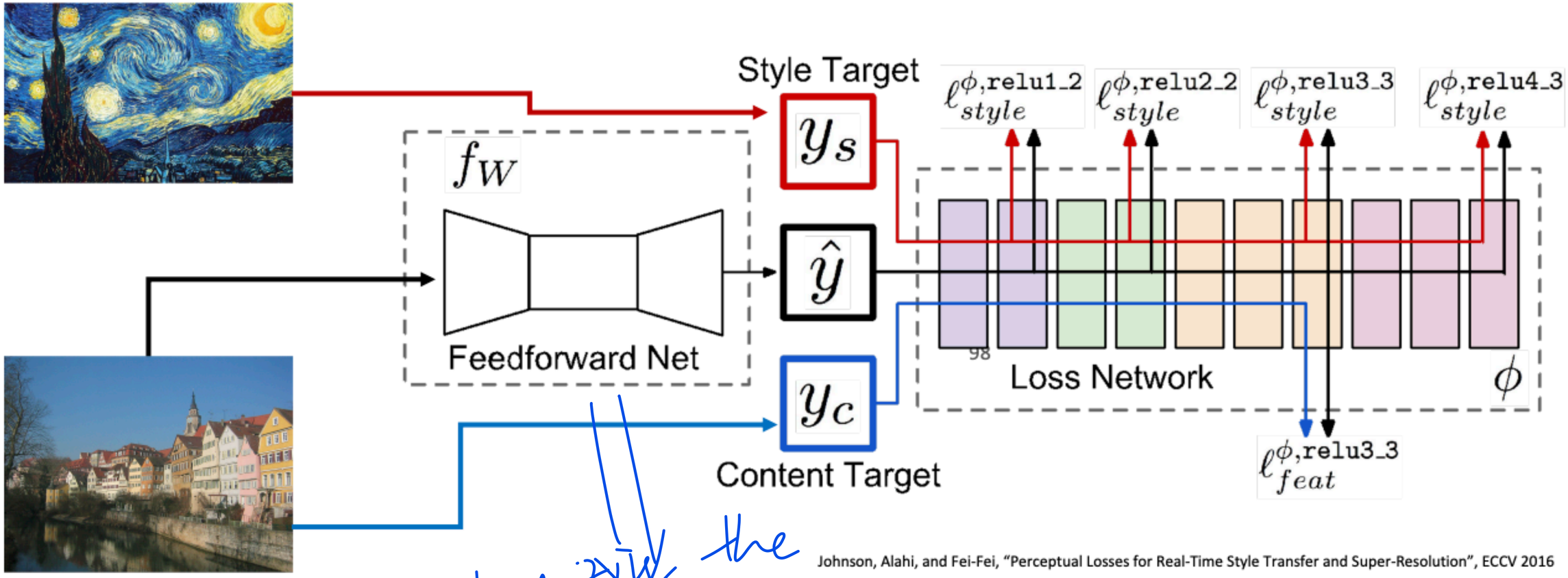
Gatys, Ecker, and Bethge, "Image style transfer using convolutional neural networks", CVPR 2016

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

Neural Style Transfer

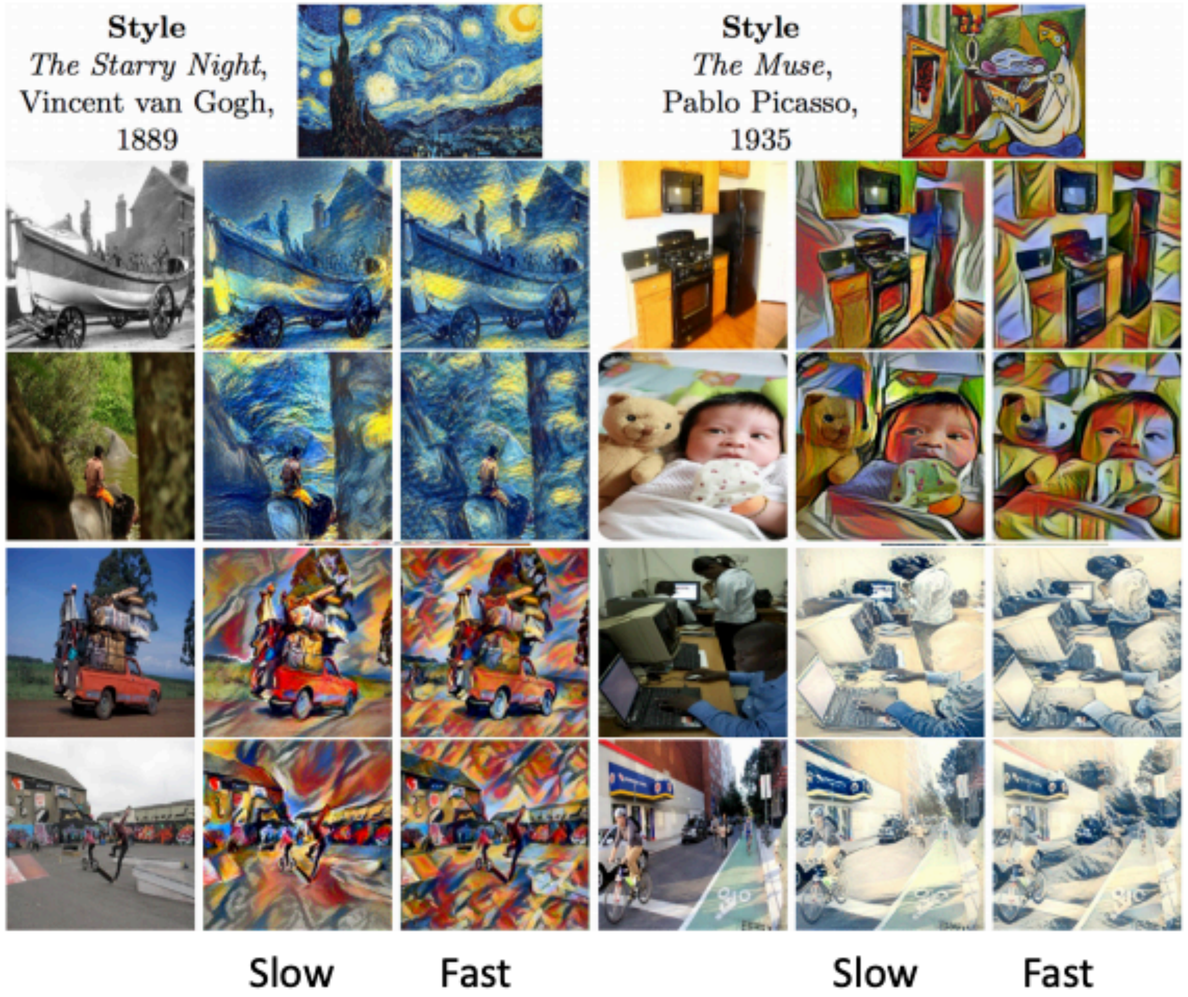
Pytorch implementation: https://github.com/jcjohnson/neural-style

Style Transfer via Network



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

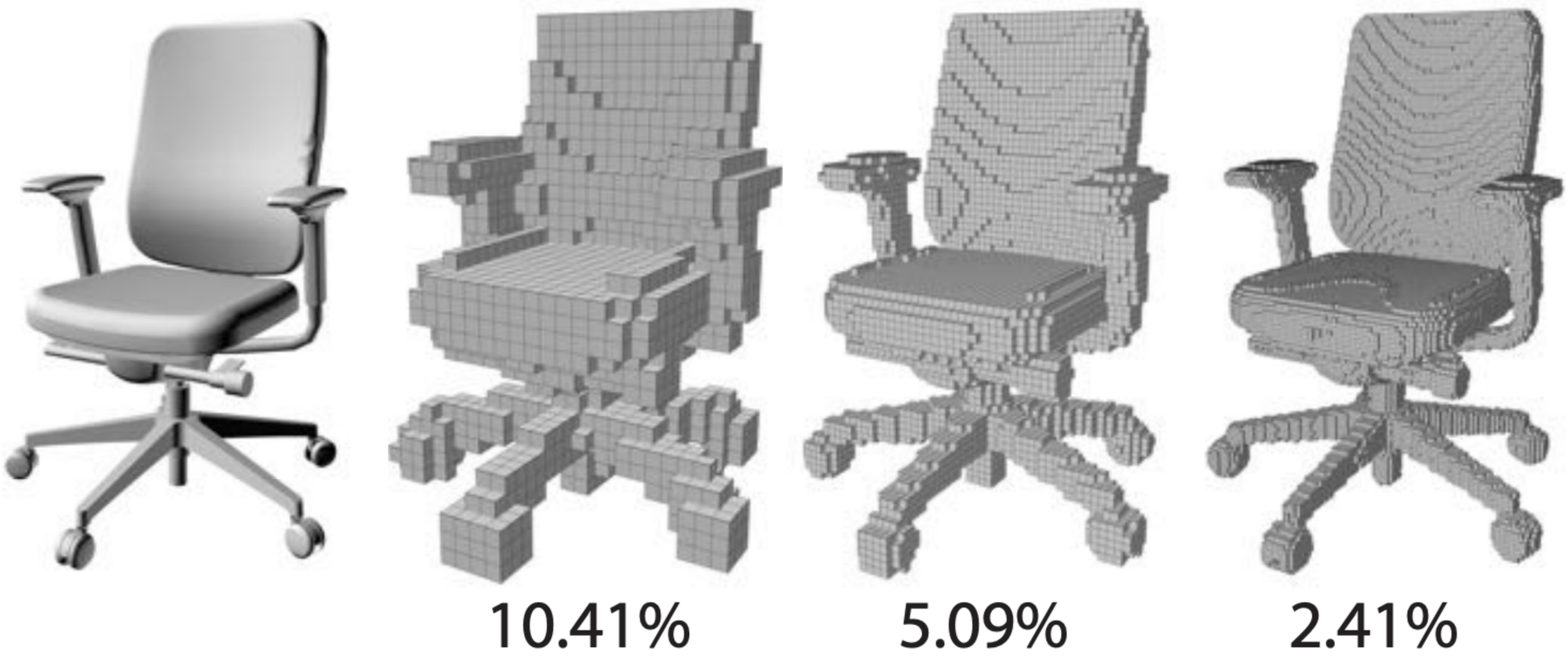
More Results



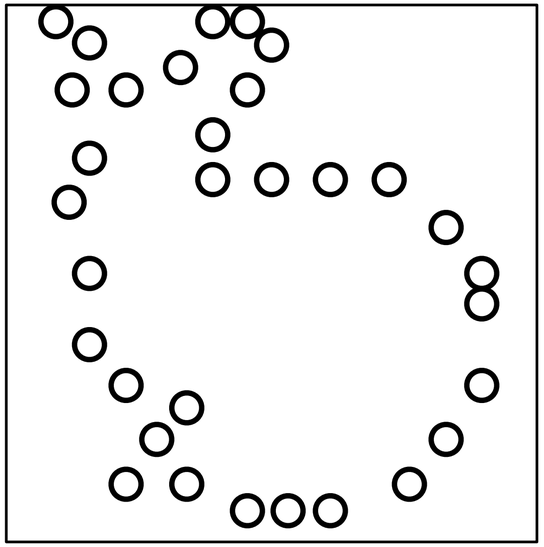
Overfit to an Image

What's Different in 3D?

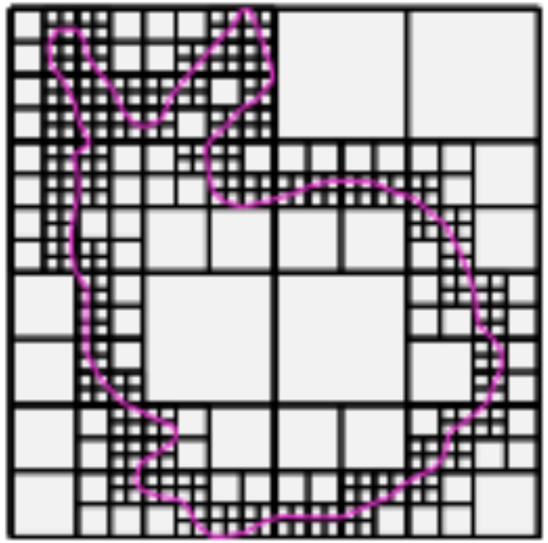
- **Number of Voxels grows as $O(n^3)$ versus occupied s**



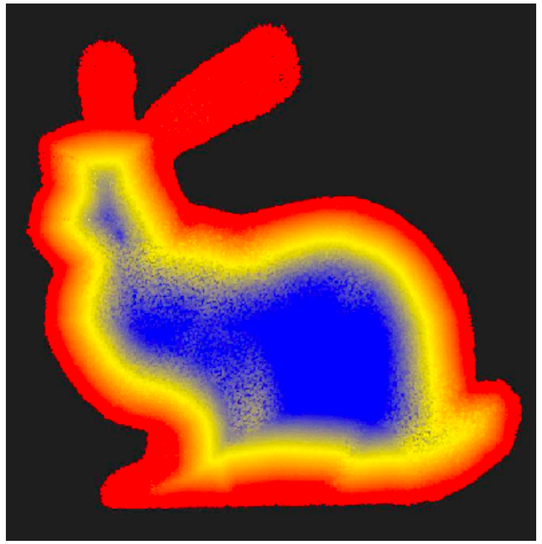
Data Representation: Many Possibilities!



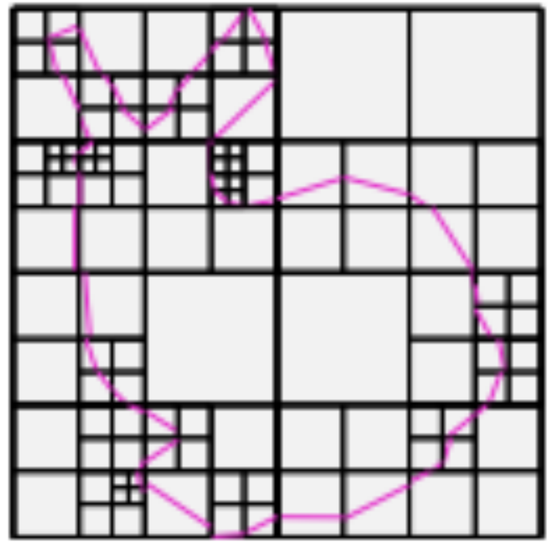
points



voxels

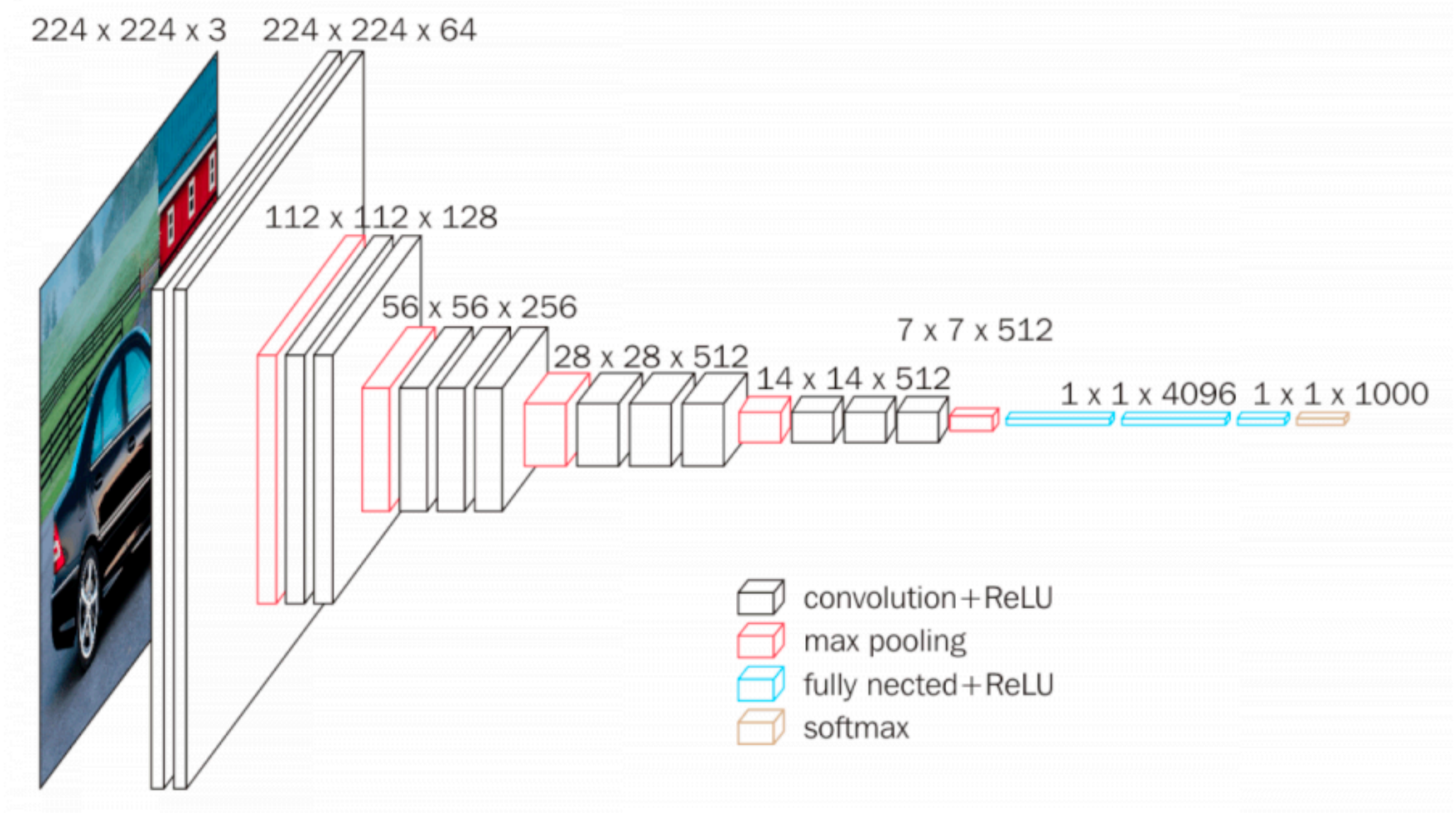


SDF



patches

VGG Network


224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax

Representation for 3D: Multi-view CNN

- **Image-based**
- **V**
- **I**
- **S**

regular image analysis networks

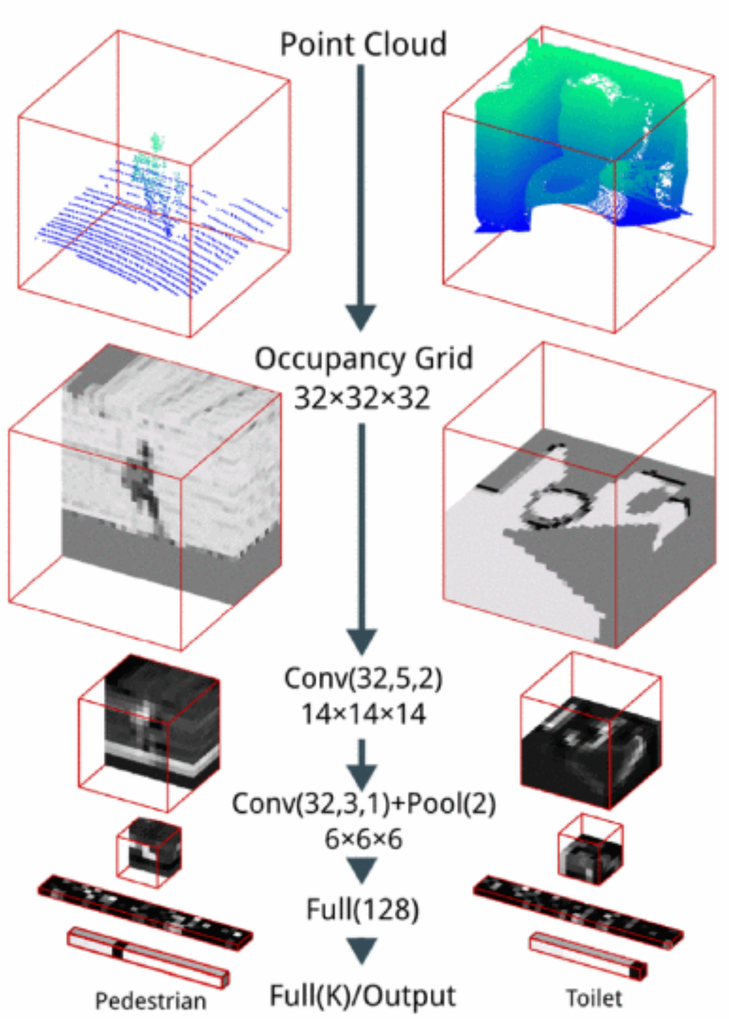
3D shape model
rendered with
different virtual cameras

[Kalogerakis et al. 2015]

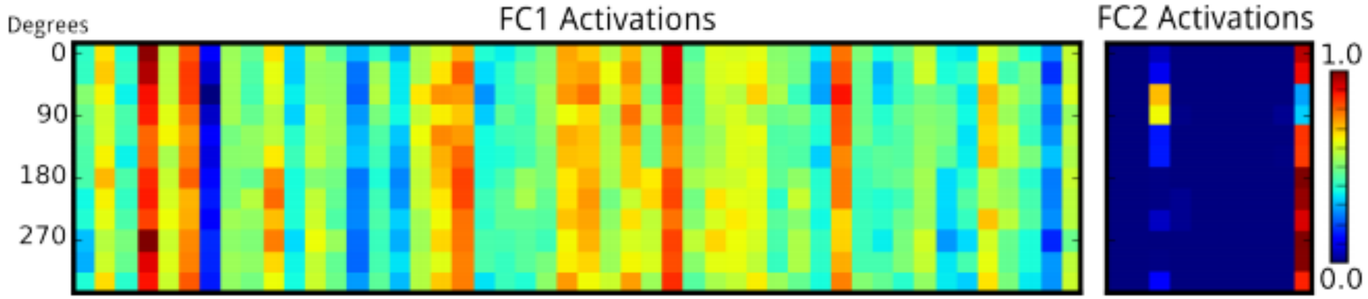
VoxNet

[Maturana et al. 2014]

- *) VOXNET: A 3D CONVOLUTIONAL NE
  FOR REAL-TIME OBJECT RECOGNITIO
  AL. 2015]
  - Binary occupancy, density grid, etc.

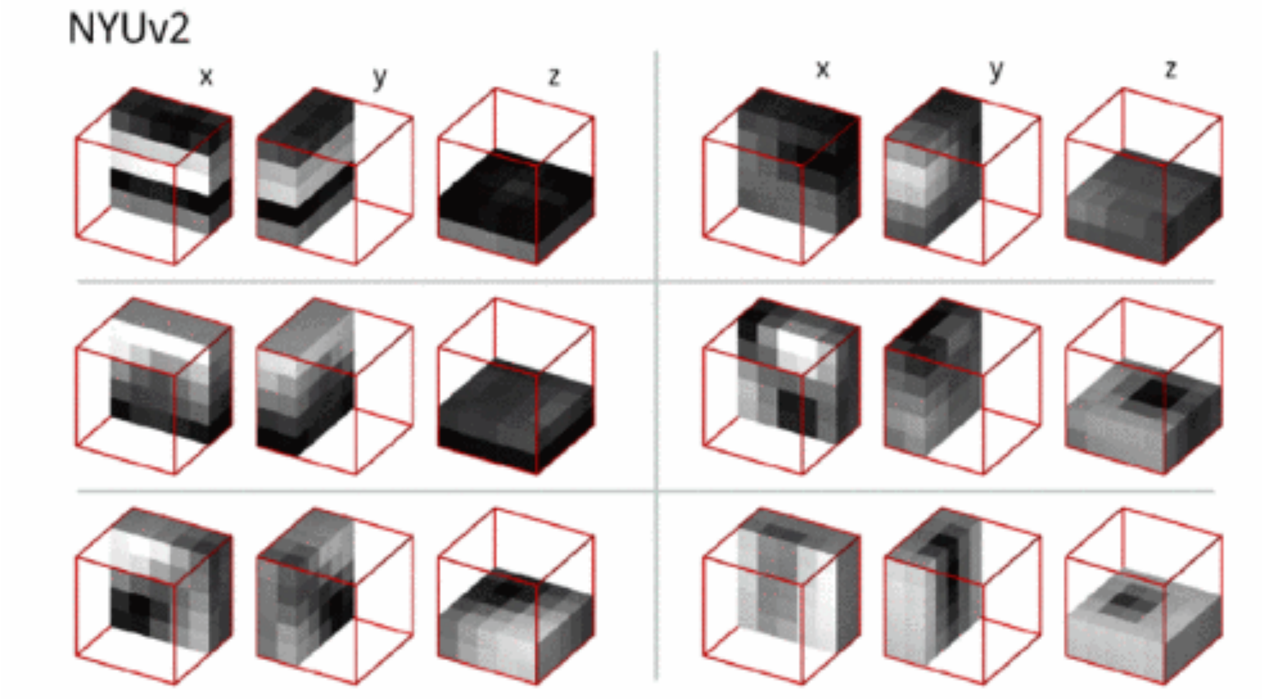


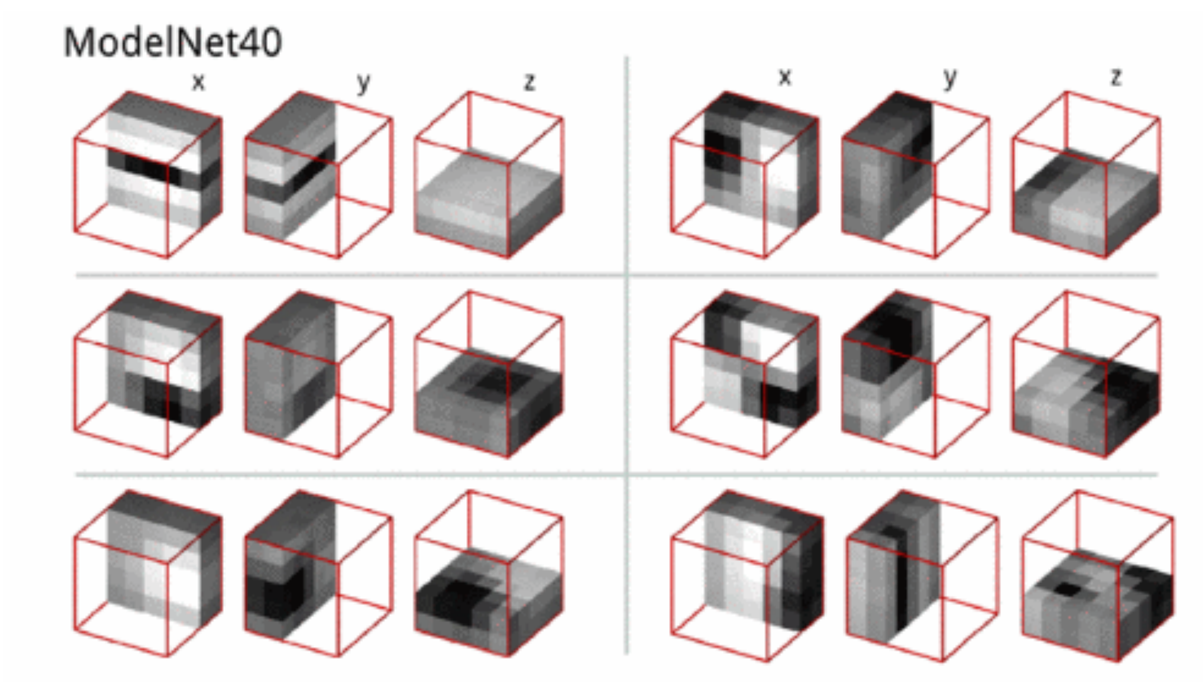
rotational invariance



Visualization of First Level Filters

VISUALISATION OF FIRST LAYER FILTERS

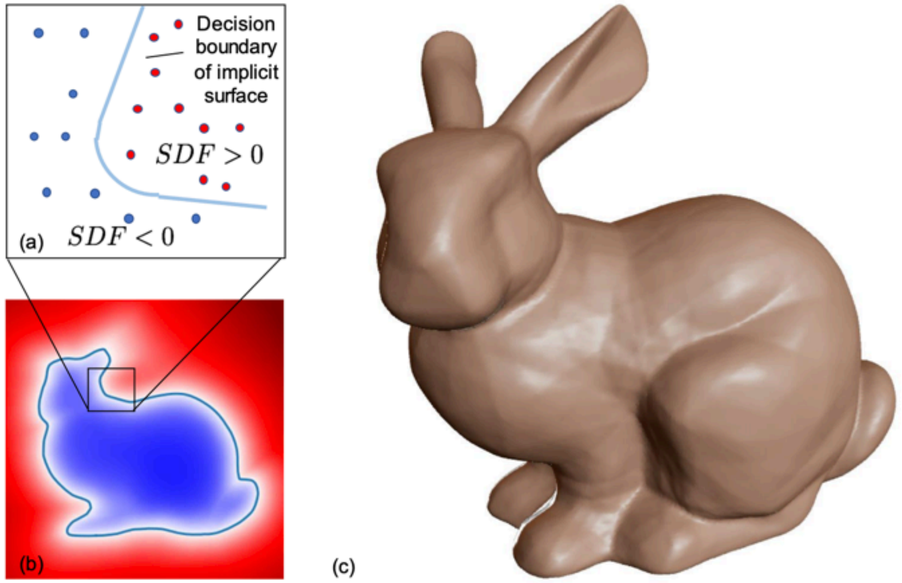




DeepSDF

- What are SDFs?

- MLP to capture SDF



DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation

Jeong Joon Park[1,4†] Peter Florence [2,4†] Julian Straub[3] Richard Newcombe[3] Steven Lovegrove[3]
[1]University of Washington [2]Massachusetts Institute of Technology [3]Facebook Reality Labs

Figure 1: DeepSDF represents signed distance functions (SDFs) of shapes via latent code-conditioned feed-forward decoder networks. Above images are raycast renderings of DeepSDF interpolating between two shapes in the learned shape latent space. Best viewed digitally.

Abstract

Computer graphics, 3D computer vision and robotics communities have produced multiple approaches to representing 3D geometry for rendering and reconstruction. These provide trade-offs across fidelity, efficiency and compression capabilities. In this work, we introduce DeepSDF, a learned continuous Signed Distance Function (SDF) representation of a class of shapes that enables high quality shape representation, interpolation and completion from partial and noisy 3D input data. DeepSDF, like its classical counterpart, represents a shape's surface by a continuous volumetric field: the magnitude of a point in the field represents the distance to the surface boundary and the sign indicates whether the region is inside (-) or outside (+) of the shape, hence our representation implicitly encodes a shape's boundary as the zero-level-set of the learned function while explicitly representing the classification of space as being part of the shapes interior or not. While classical SDF's both in analytical or discretized voxel form typically represent the surface of a single shape, DeepSDF can represent an entire class of shapes. Furthermore, we show state-of-the-art performance for learned 3D shape representation and completion while reducing the model size by an order of magnitude compared with previous work.

† Work performed during internship at Facebook Reality Labs.

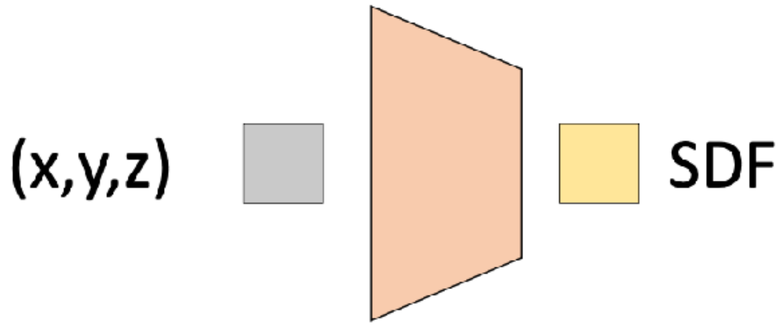
1. Introduction

Deep convolutional networks which are a mainstay of image-based approaches grow quickly in space and time complexity when directly generalized to the 3rd spatial dimension, and more classical and compact surface representations such as triangle or quad meshes pose problems in training since we may need to deal with an unknown number of vertices and arbitrary topology. These challenges have limited the quality, flexibility and fidelity of deep learning approaches when attempting to either input 3D data for processing or produce 3D inferences for object segmentation and reconstruction.

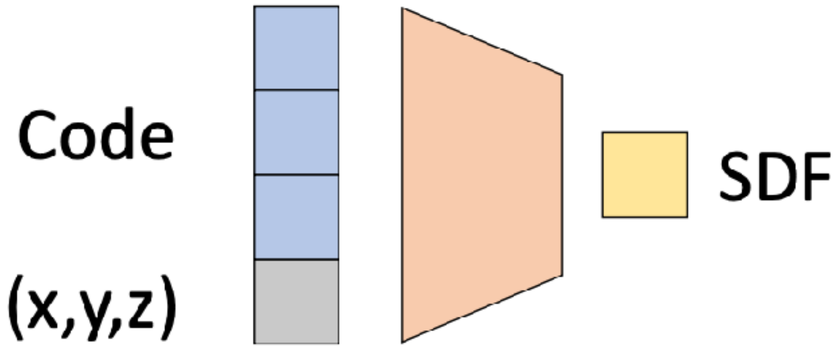
In this work, we present a novel representation and approach for generative 3D modeling that is efficient, expressive, and fully continuous. Our approach uses the concept of a SDF, but unlike common surface reconstruction techniques which discretize this SDF into a regular grid for evaluation and measurement denoising, we instead learn a generative model to produce such a continuous field.

The proposed continuous representation may be intuitively understood as a learned shape-conditioned classifier for which the decision boundary is the surface of the shape itself, as shown in Fig. 2. Our approach shares the generative aspect of other works seeking to map a latent space to a distribution of complex shapes in 3D [54], but critically differs in the central representation. While the notion of an

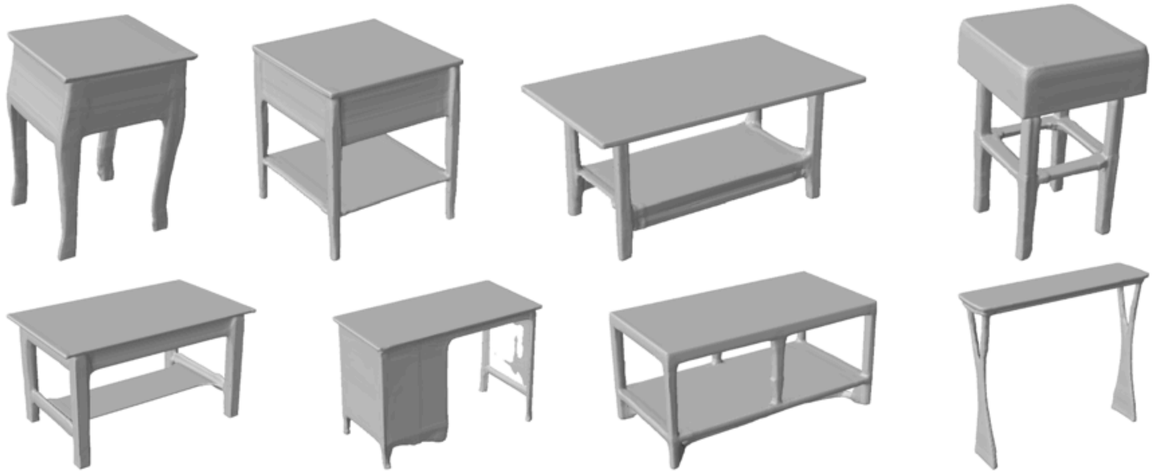
One vs Many Shape



$$f_\theta(\mathbf{x}_i) \approx SDF(\mathbf{x}_i)$$



$$g_\theta(c, \mathbf{x}_i) \approx SDF(\mathbf{x}_i)$$



Global vs Local Encoding