Generative Models of Images

CS180: Intro to Comp. Vision and Comp. Photo
Alexei Efros, UC Berkeley, Fall 2025

Slides from Steve Seitz,
Bill Freeman, Andrew Owens, etc.

The Space of All Images

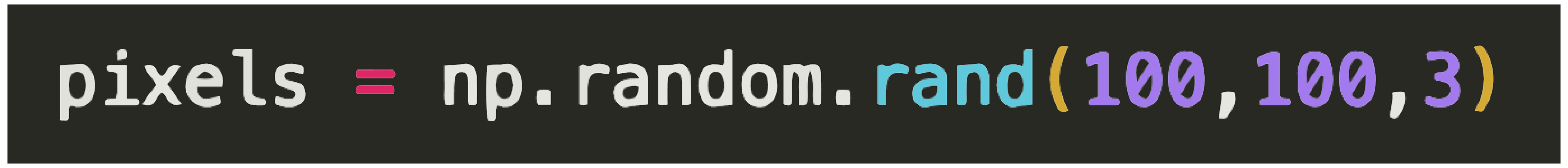
- Lets consider the space of all 100x100 images

- Now lets randomly sample that space…

- Conclusion: Most images are noise

Question:

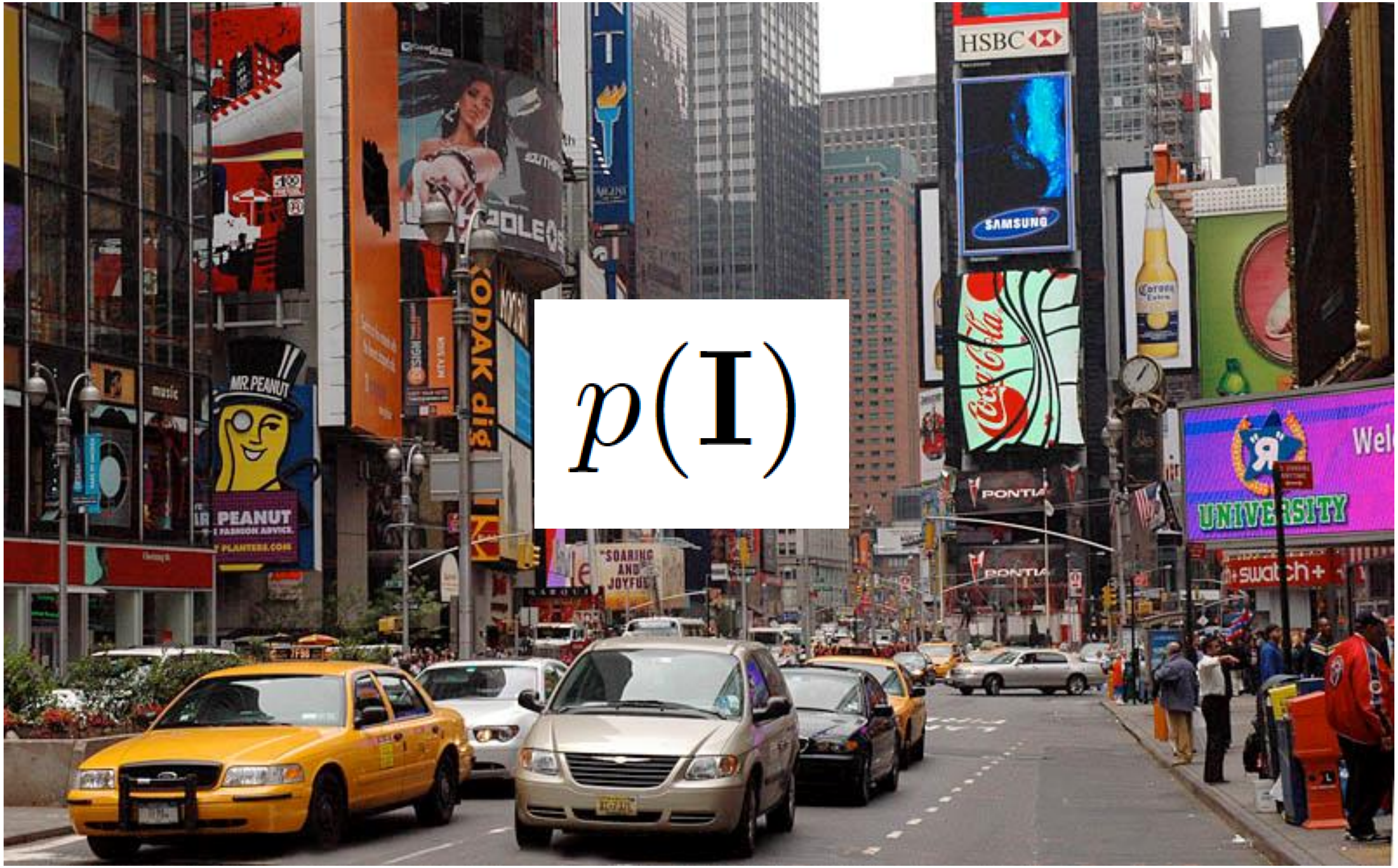
What do we expect a random uniform sample of all images to look like?



```
pixels = np.random.rand(100,100,3)
```

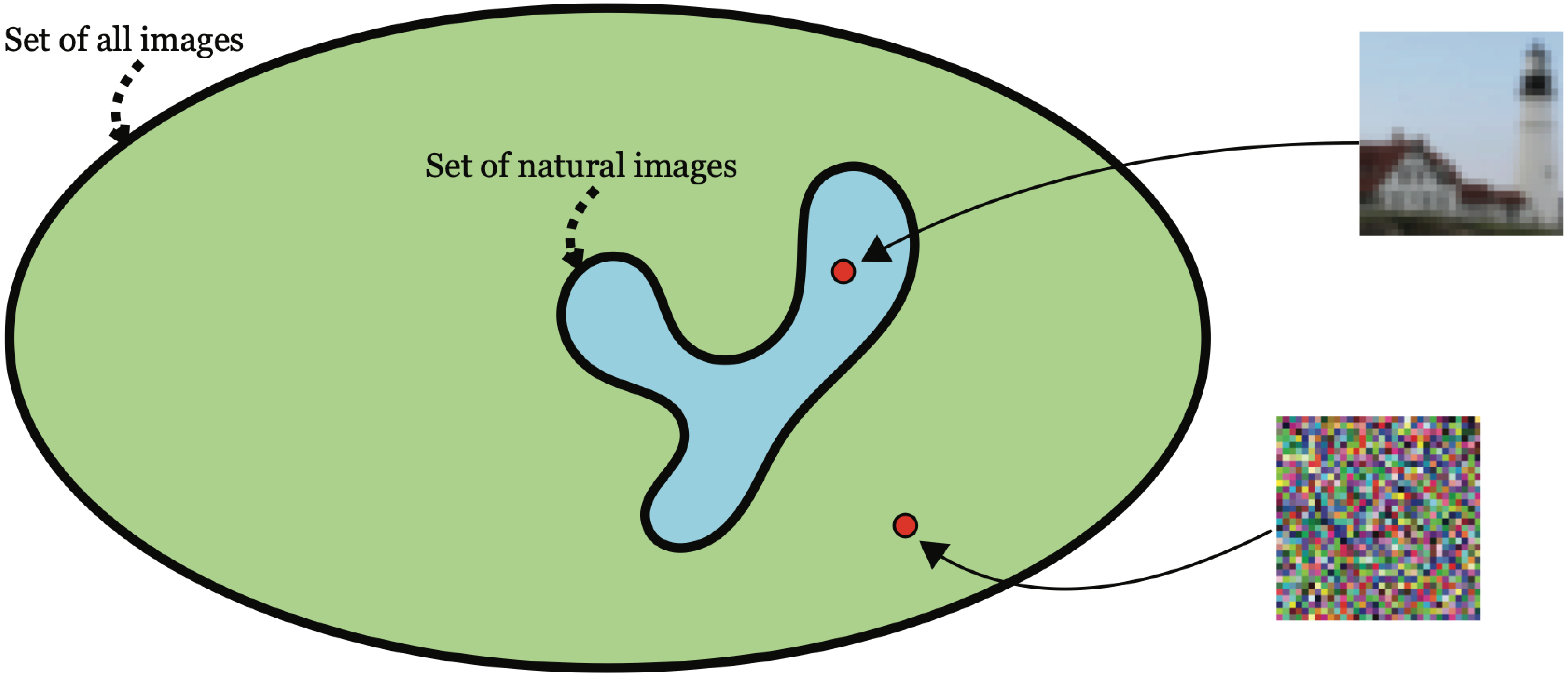
Slide source: Steve Seitz

Statistical modeling of images



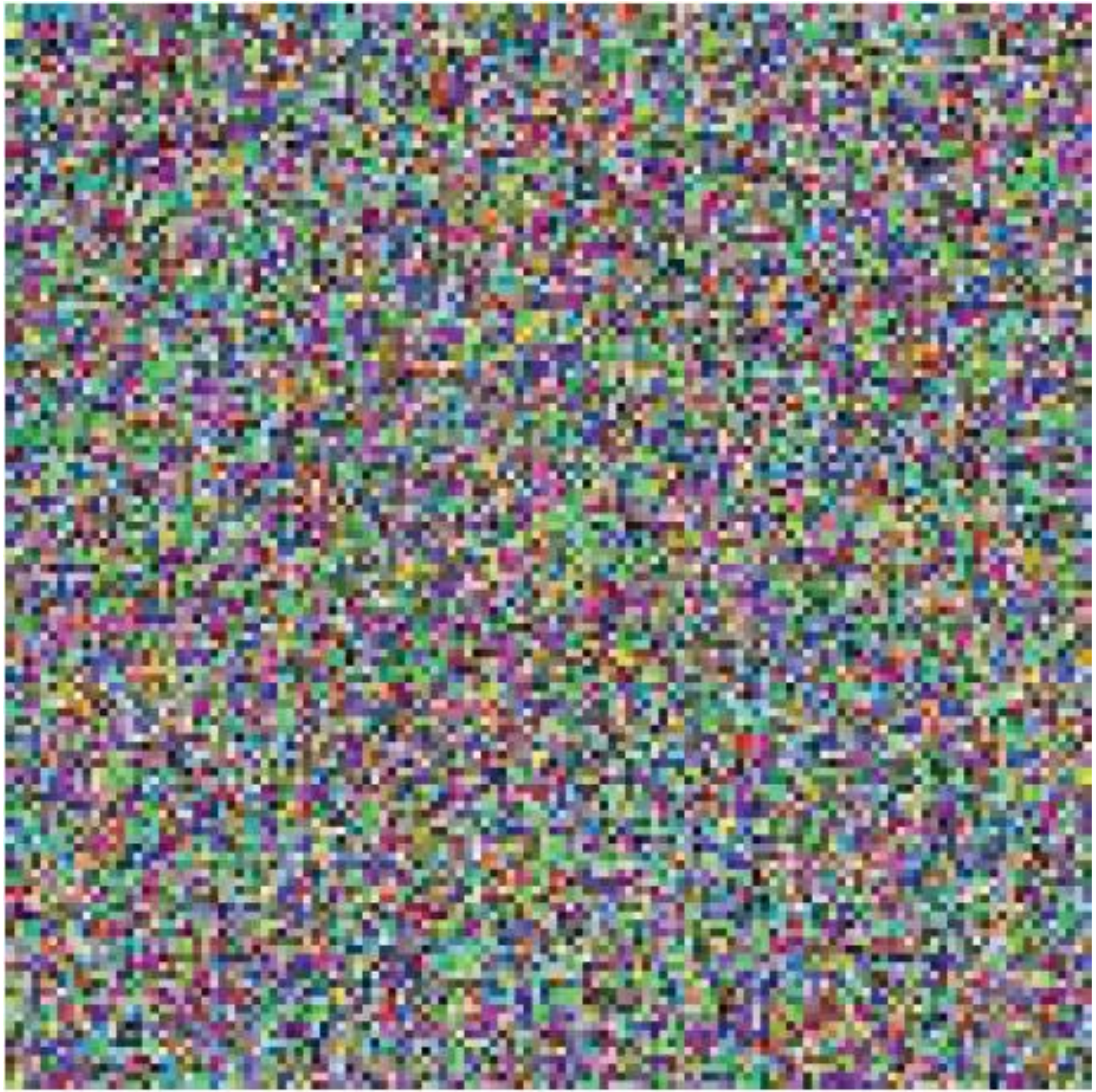
$$p(\mathbf{I})$$

Statistical modeling of images



Visual Worlds

Visual Worlds



Visual Worlds

Visual Worlds

Visual Worlds

Visual Worlds

Visual Worlds

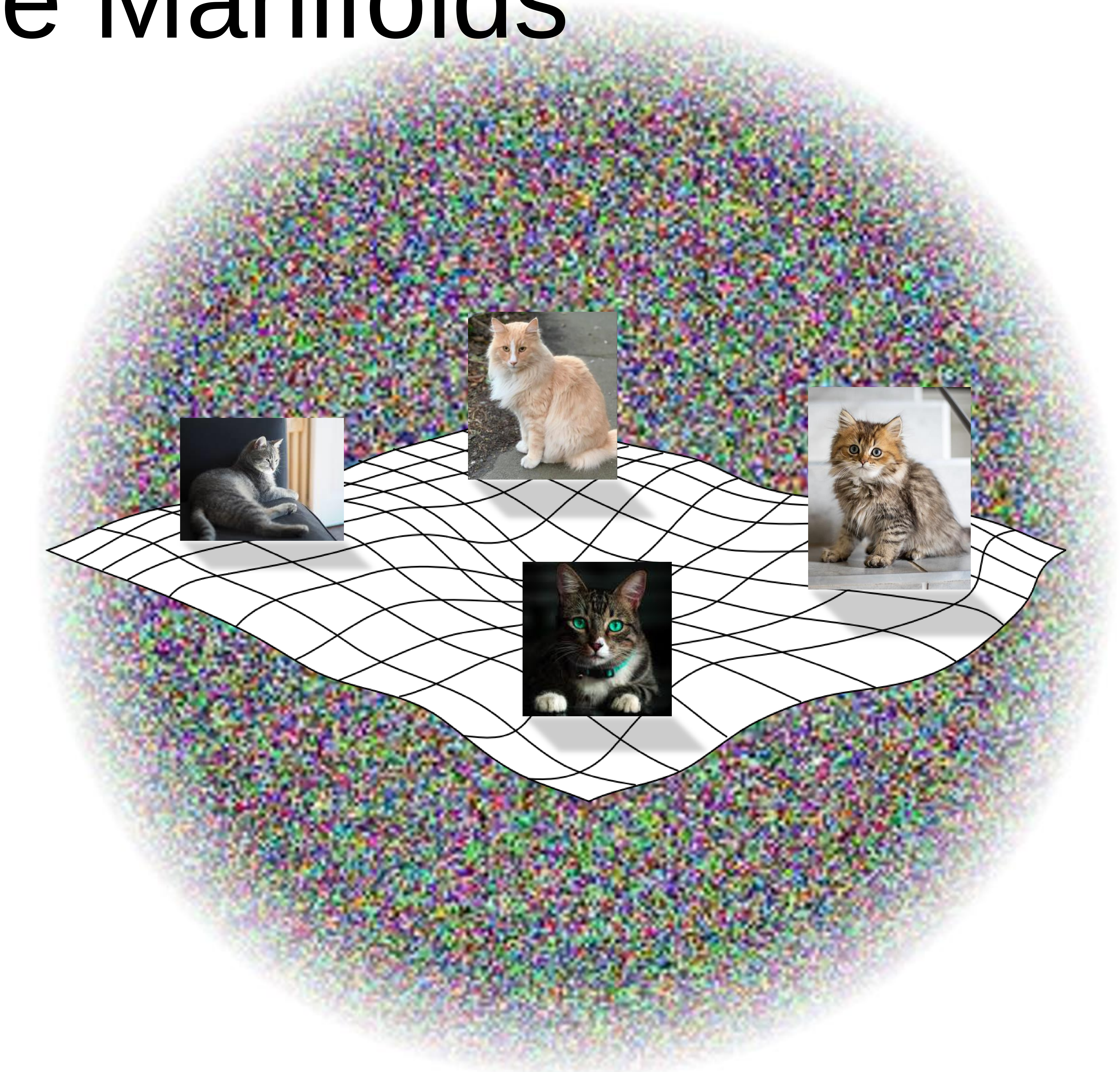
Visual Worlds

Visual Worlds

Natural Image Manifolds

- Most images are “noise”

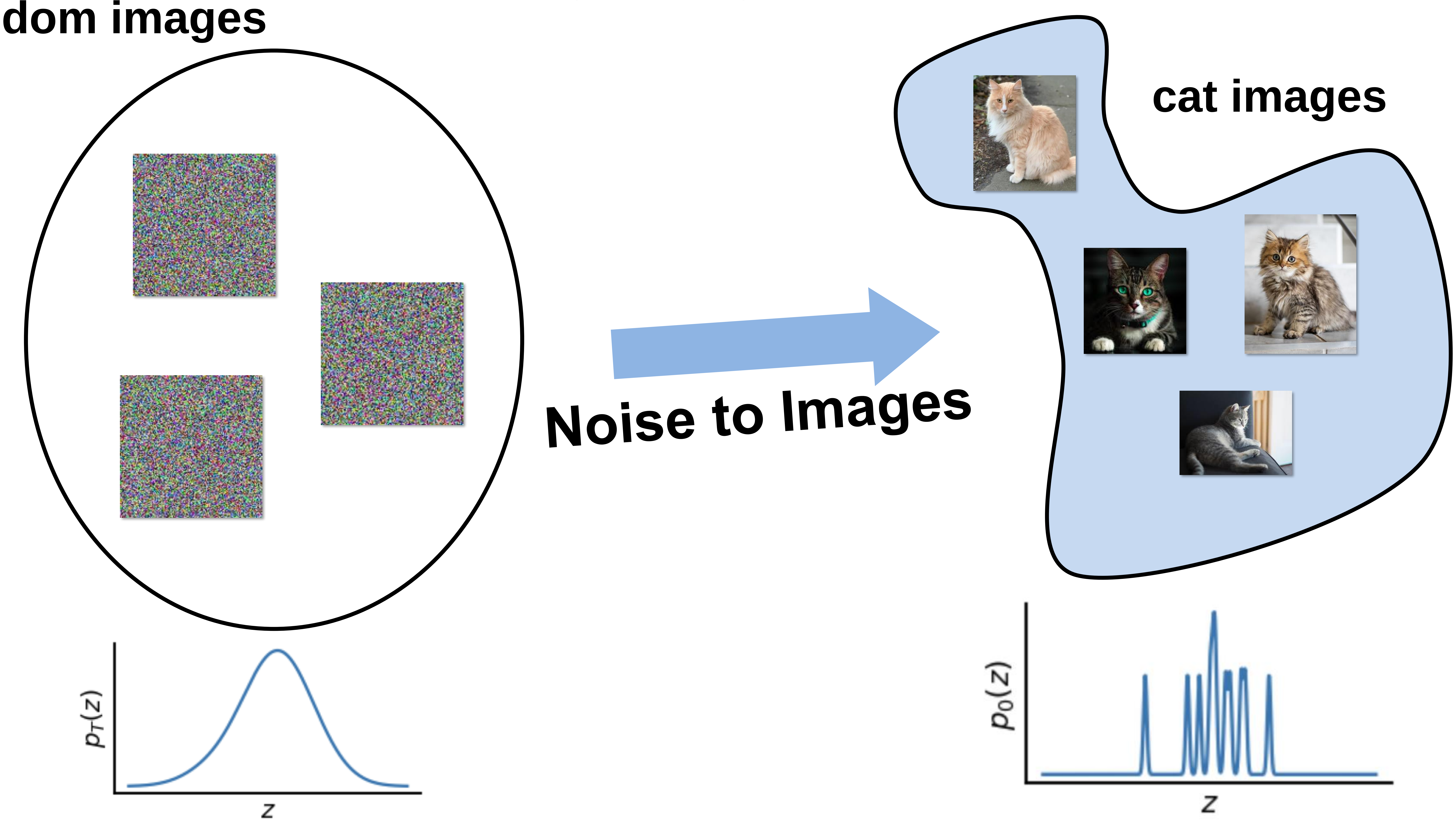
- “Meaningful” images tend to form some manifold within the space of all images



The Space of All Images

Slide source: Steve Seitz

Generating Images from Noise



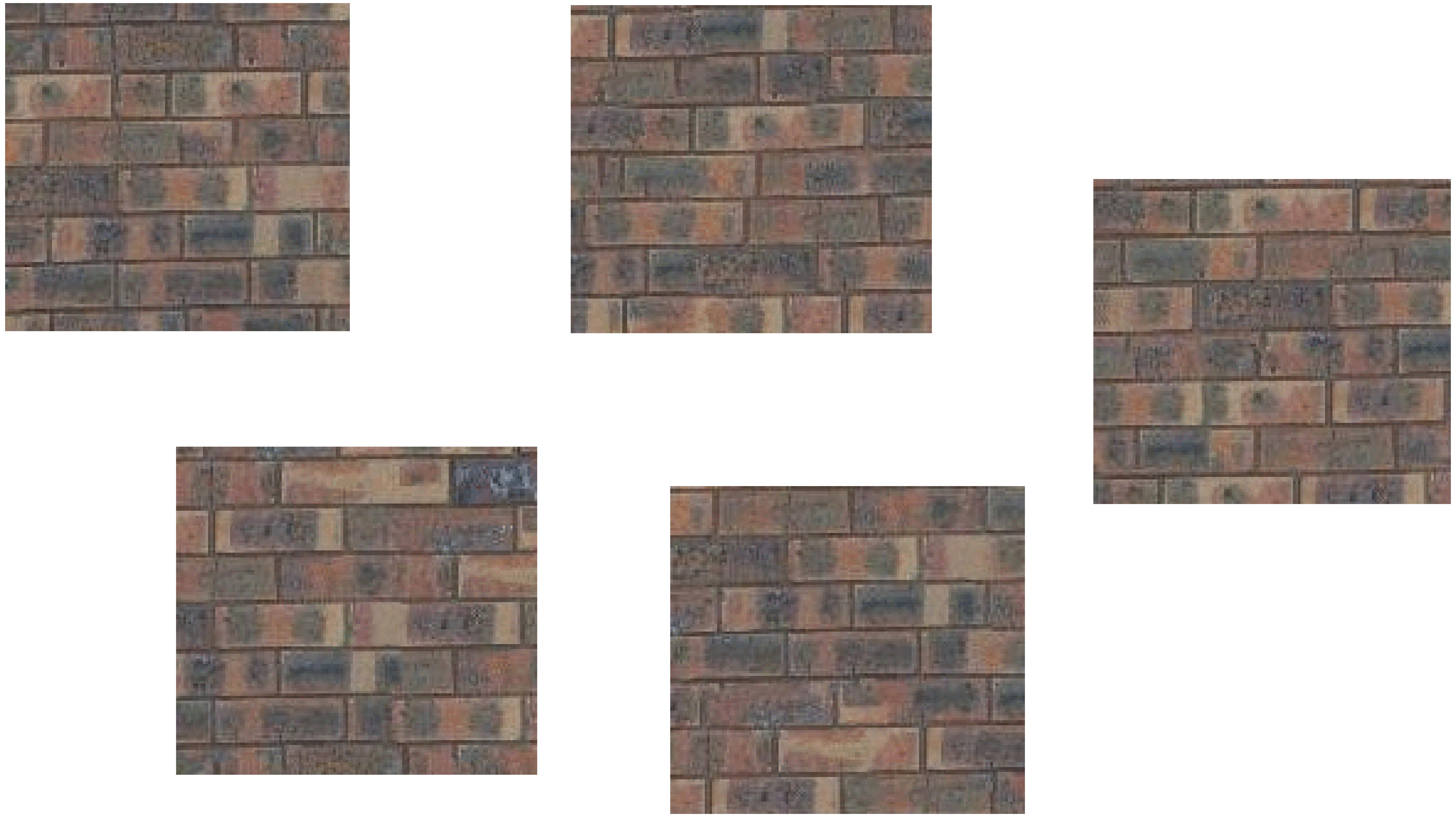
Random images

Noise to Images

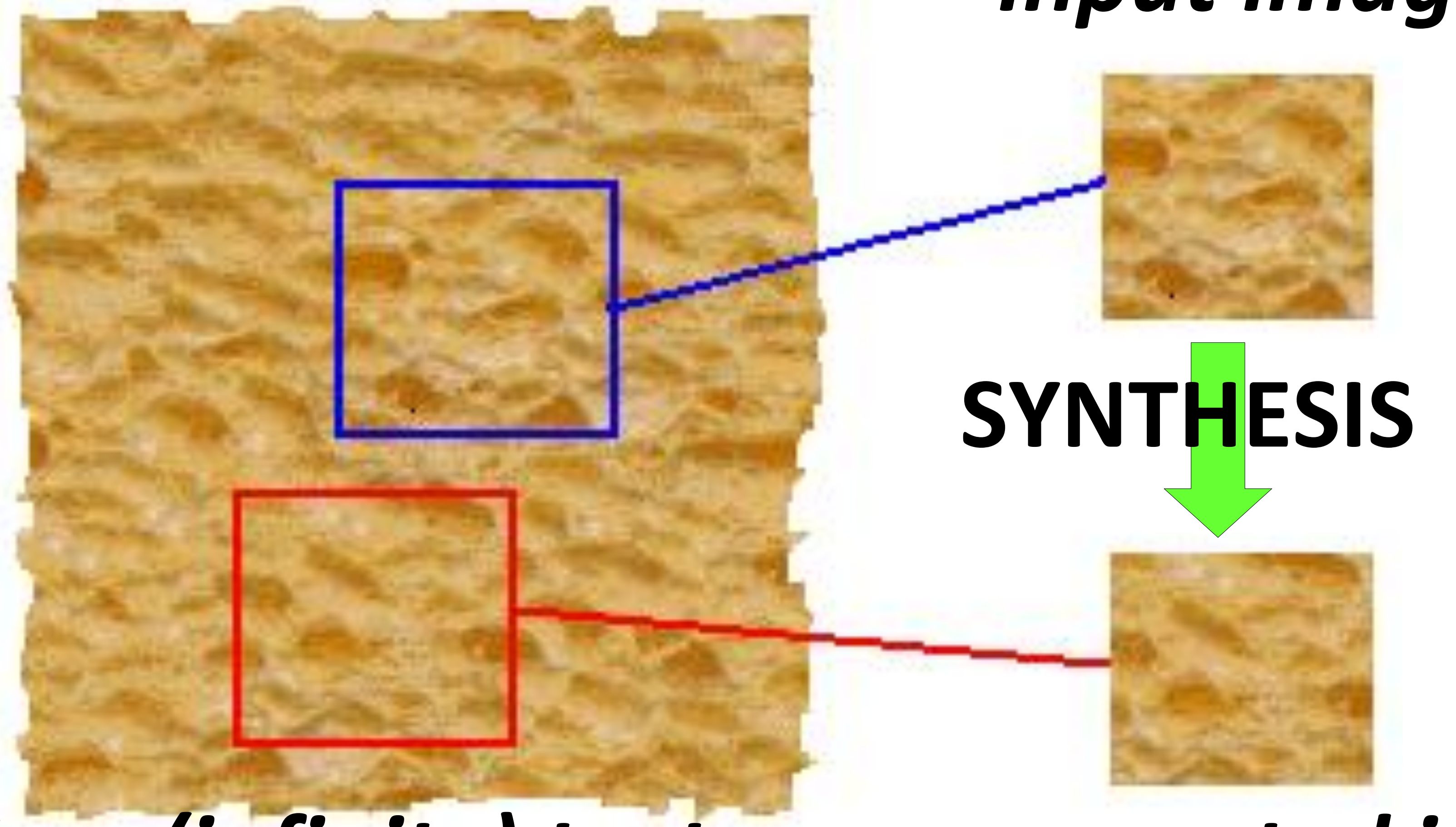
cat images

Slide source: Steve Seitz

Let’s start with manifold of textures



Parametric Texture Synthesis



input image

SYNTHESIS

True (infinite) texture

generated image

Goal: parametric **generative model** of the “infinite texture”

Texture-motivated Parametric Generative Image Models!

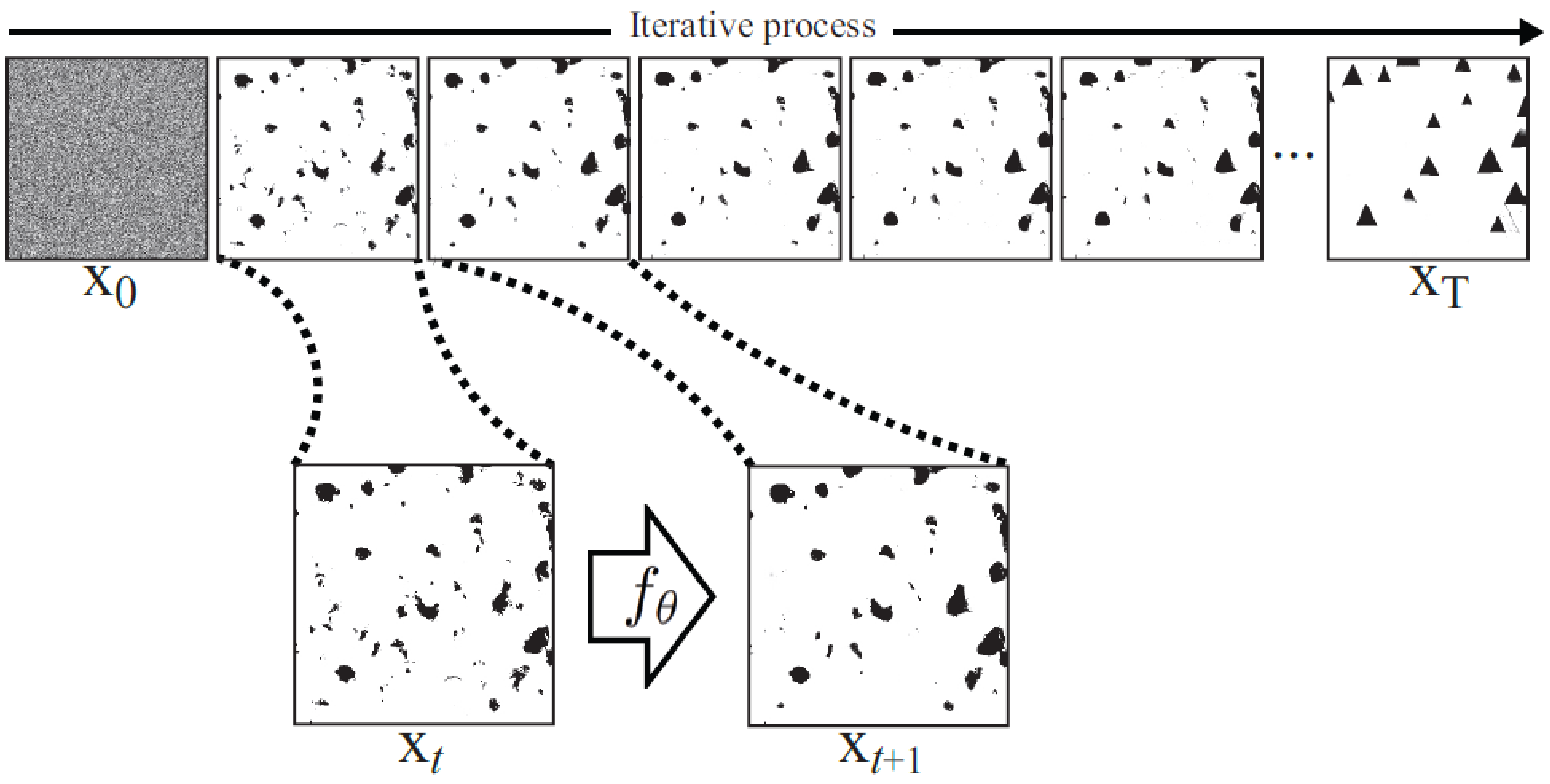
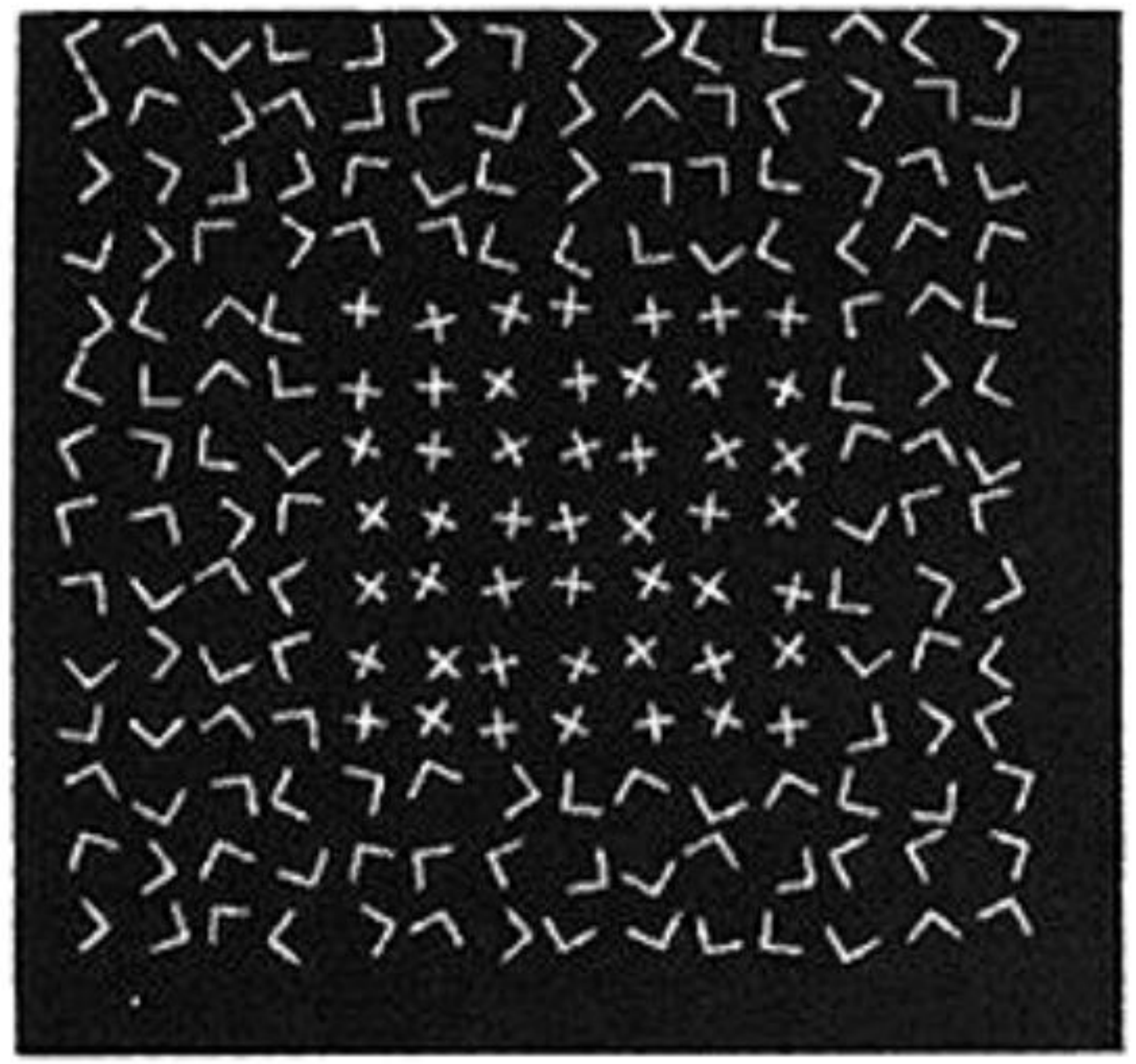
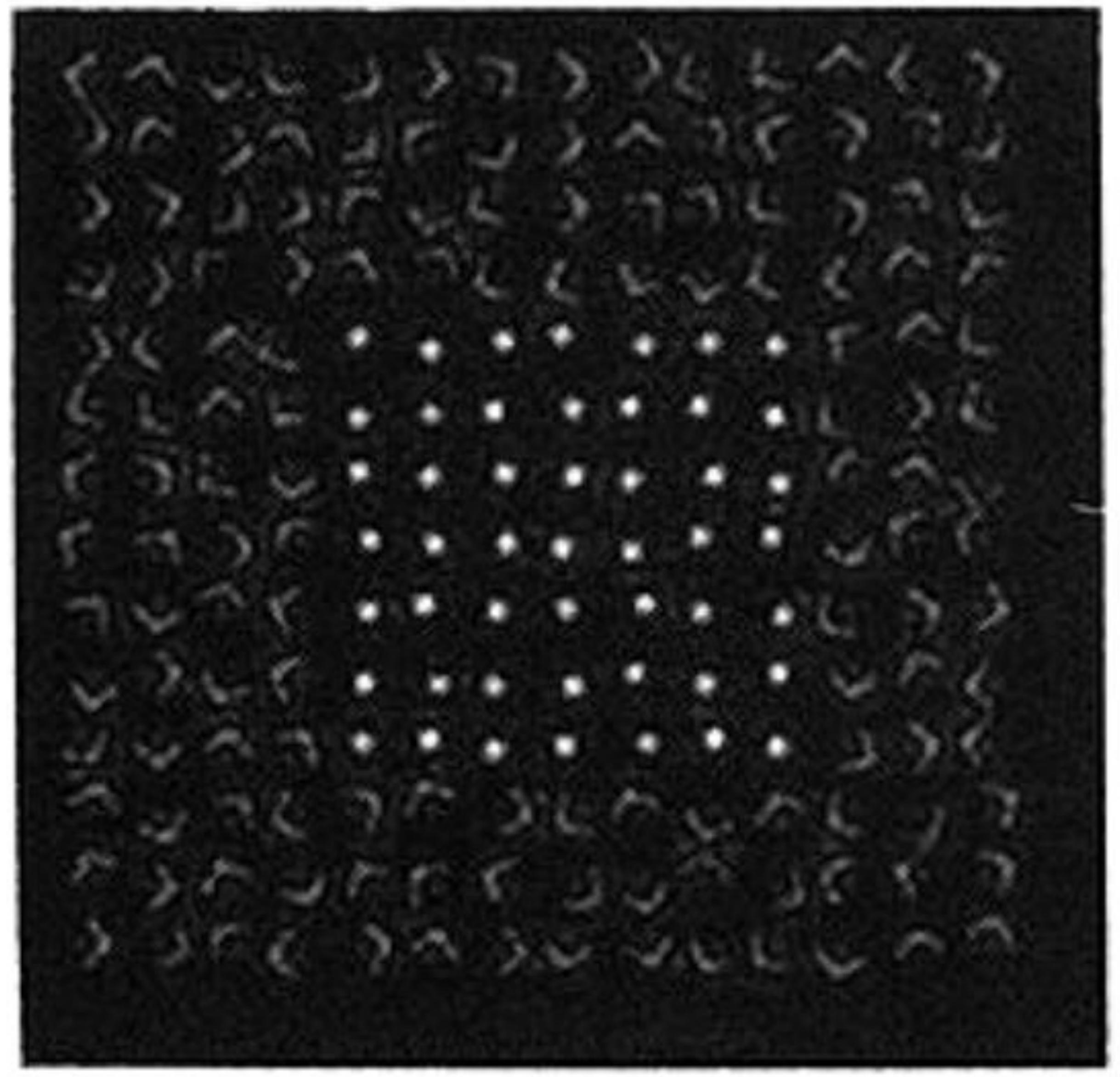


Figure 28.6: The steps of the Heeger-Bergen texture synthesis algorithm. The process starts with white noise input image. Each step takes as input the previous output, and it is modified by a function f_θ, where θ are the parameters describing a texture. At each step the output image x_t gets closer to the appearance of the reference texture (figure 28.5). The result of 500 iterations is shown in the right.

Fig. 1 *Top row*, Textures consisting of Xs within a texture composed of Ls. The micropatterns are placed at random orientations on a randomly perturbed lattice. *a*, The bars of the Xs have the same length as the bars of the Ls. *b*, The bars of the Ls have been lengthened by 25%, and the intensity adjusted for the same mean luminance. Discriminabitity is enhanced. *c*, The bars of the Ls have been shortened by 25%, and the intensity adjusted for the same mean luminance. Discriminabitity is impaired. *Bottom row*: the responses of a size-tuned mechanism *d*, response to image *a*; *e*, response to image *b*; *f*; response to image *c*.



a



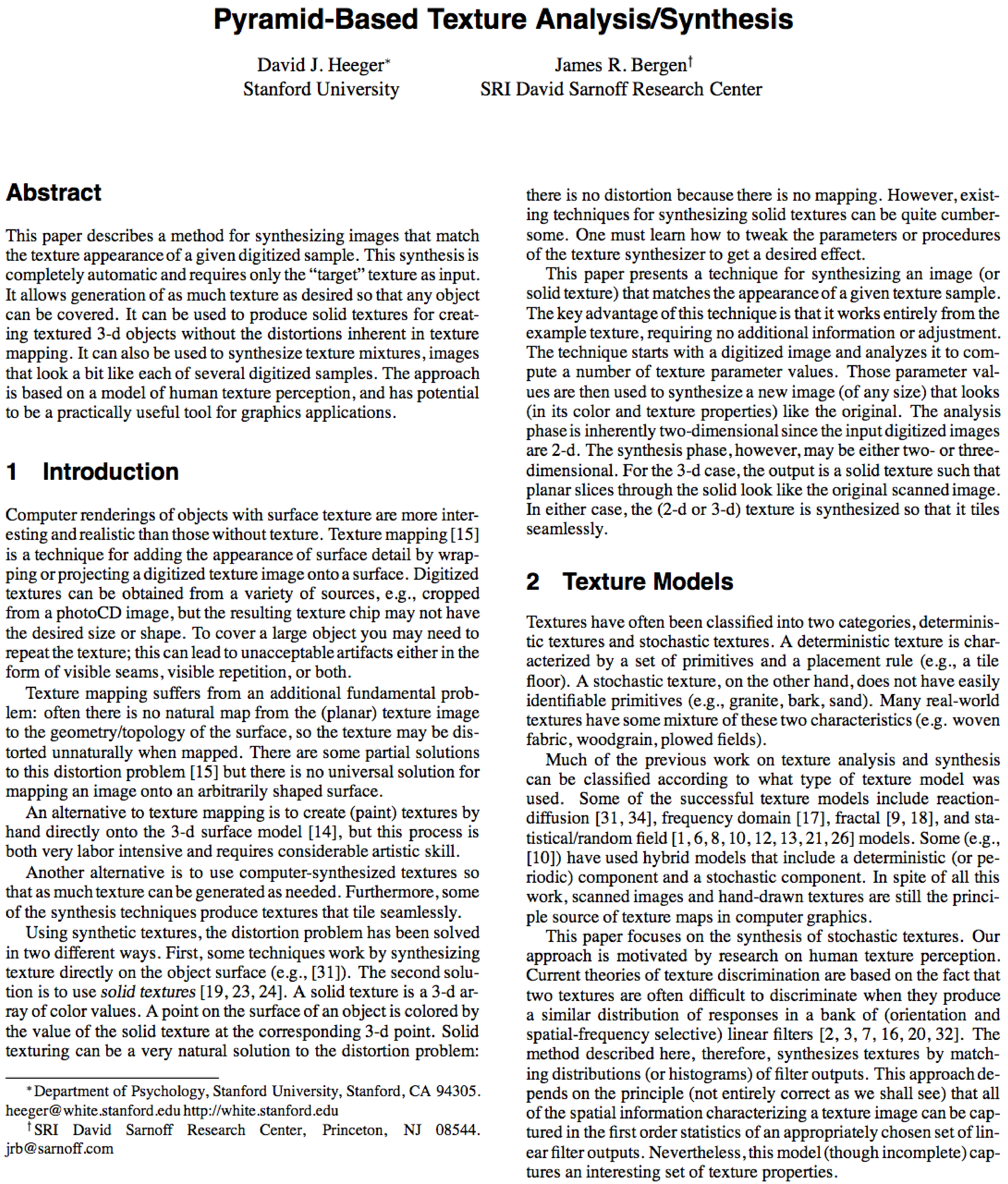
d

Early vision and texture perception

James R. Bergen* & Edward H. Adelson**

* SRI David Sarnoff Research Center, Princeton,
New Jersey 08540, USA
** Media Lab and Department of Brain and Cognitive Science,
Massachusetts Institute of Technology, Cambridge,
Massachusetts 02139, USA

Reprinted from Nature, Vol. 333. No. 6171. pp. 363-364,
© *Macmillan Magazines Ltd.,* 1988



Pyramid-Based Texture Analysis/Synthesis

David J. Heeger[*]
Stanford University

James R. Bergen[†]
SRI David Sarnoff Research Center

Abstract

This paper describes a method for synthesizing images that match the texture appearance of a given digitized sample. This synthesis is completely automatic and requires only the "target" texture as input. It allows generation of as much texture as desired so that any object can be covered. It can be used to produce solid textures for creating textured 3-d objects without the distortions inherent in texture mapping. It can also be used to synthesize texture mixtures, images that look a bit like each of several digitized samples. The approach is based on a model of human texture perception, and has potential to be a practically useful tool for graphics applications.

1 Introduction

Computer renderings of objects with surface texture are more interesting and realistic than those without texture. Texture mapping [15] is a technique for adding the appearance of surface detail by wrapping or projecting a digitized texture image onto a surface. Digitized textures can be obtained from a variety of sources, e.g., cropped from a photoCD image, but the resulting texture chip may not have the desired size or shape. To cover a large object you may need to repeat the texture; this can lead to unacceptable artifacts either in the form of visible seams, visible repetition, or both.

Texture mapping suffers from an additional fundamental problem: often there is no natural map from the (planar) texture image to the geometry/topology of the surface, so the texture may be distorted unnaturally when mapped. There are some partial solutions to this distortion problem [15] but there is no universal solution for mapping an image onto an arbitrarily shaped surface.

An alternative to texture mapping is to create (paint) textures by hand directly onto the 3-d surface model [14], but this process is both very labor intensive and requires considerable artistic skill.

Another alternative is to use computer-synthesized textures so that as much texture can be generated as needed. Furthermore, some of the synthesis techniques produce textures that tile seamlessly.

Using synthetic textures, the distortion problem has been solved in two different ways. First, some techniques work by synthesizing texture directly on the object surface (e.g., [31]). The second solution is to use *solid textures* [19, 23, 24]. A solid texture is a 3-d array of color values. A point on the surface of an object is colored by the value of the solid texture at the corresponding 3-d point. Solid texturing can be a very natural solution to the distortion problem: there is no distortion because there is no mapping. However, existing techniques for synthesizing solid textures can be quite cumbersome. One must learn how to tweak the parameters or procedures of the texture synthesizer to get a desired effect.

This paper presents a technique for synthesizing an image (or solid texture) that matches the appearance of a given texture sample. The key advantage of this technique is that it works entirely from the example texture, requiring no additional information or adjustment. The technique starts with a digitized image and analyzes it to compute a number of texture parameter values. Those parameter values are then used to synthesize a new image (of any size) that looks (in its color and texture properties) like the original. The analysis phase is inherently two-dimensional since the input digitized images are 2-d. The synthesis phase, however, may be either two- or three-dimensional. For the 3-d case, the output is a solid texture such that planar slices through the solid look like the original scanned image. In either case, the (2-d or 3-d) texture is synthesized so that it tiles seamlessly.

2 Texture Models

Textures have often been classified into two categories, deterministic textures and stochastic textures. A deterministic texture is characterized by a set of primitives and a placement rule (e.g., a tile floor). A stochastic texture, on the other hand, does not have easily identifiable primitives (e.g., granite, bark, sand). Many real-world textures have some mixture of these two characteristics (e.g. woven fabric, woodgrain, plowed fields).

Much of the previous work on texture analysis and synthesis can be classified according to what type of texture model was used. Some of the successful texture models include reaction-diffusion [31, 34], frequency domain [17], fractal [9, 18], and statistical/random field [1, 6, 8, 10, 12, 13, 21, 26] models. Some (e.g., [10]) have used hybrid models that include a deterministic (or periodic) component and a stochastic component. In spite of all this work, scanned images and hand-drawn textures are still the principle source of texture maps in computer graphics.

This paper focuses on the synthesis of stochastic textures. Our approach is motivated by research on human texture perception. Current theories of texture discrimination are based on the fact that two textures are often difficult to discriminate when they produce a similar distribution of responses in a bank of (orientation and spatial-frequency selective) linear filters [2, 3, 7, 16, 20, 32]. The method described here, therefore, synthesizes textures by matching distributions (or histograms) of filter outputs. This approach depends on the principle (not entirely correct as we shall see) that all of the spatial information characterizing a texture image can be captured in the first order statistics of an appropriately chosen set of linear filter outputs. Nevertheless, this model (though incomplete) captures an interesting set of texture properties.

[*] Department of Psychology, Stanford University, Stanford, CA 94305. heeger@white.stanford.edu http://white.stanford.edu

[†] SRI David Sarnoff Research Center, Princeton, NJ 08544. jrb@sarnoff.com

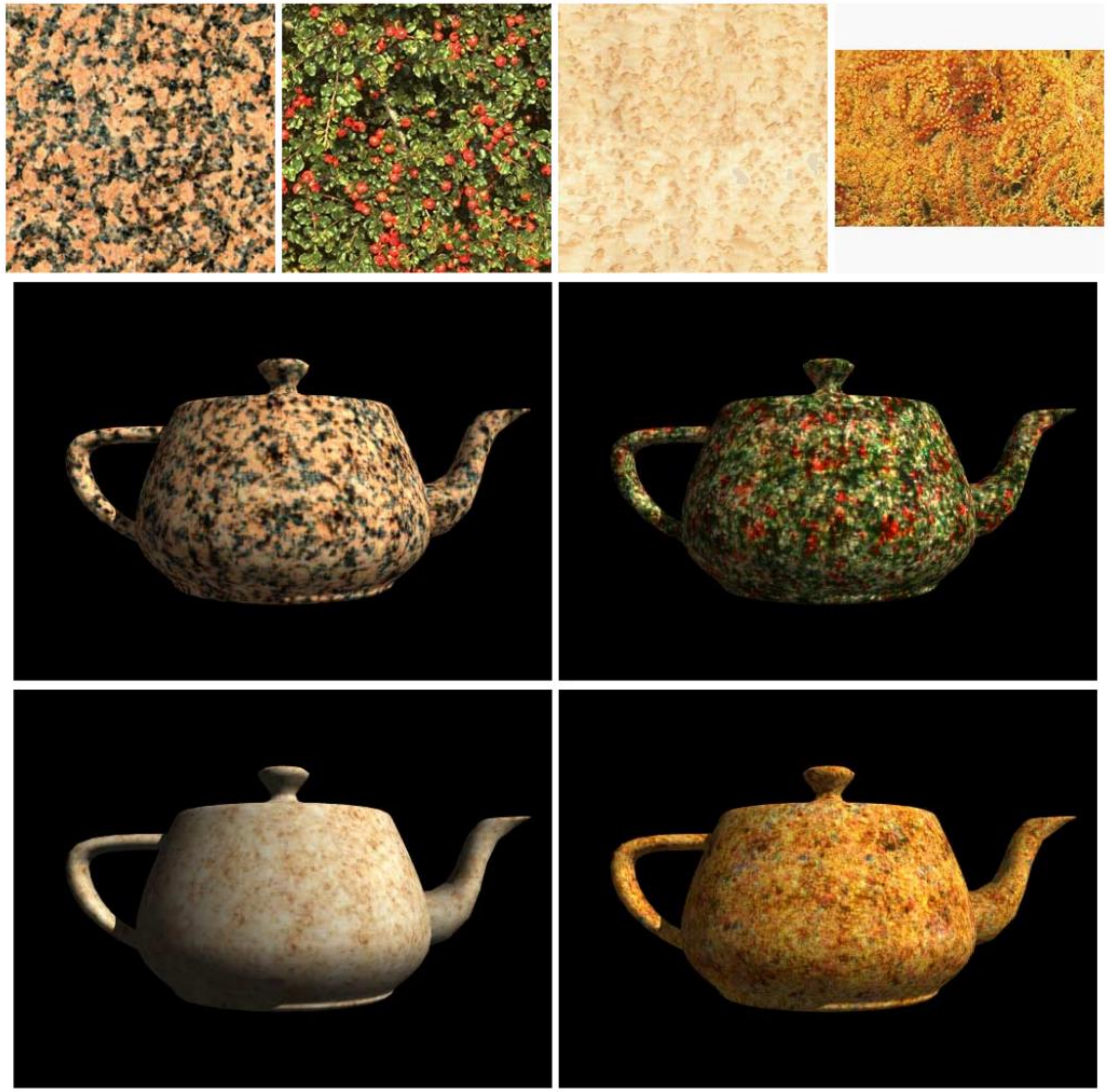


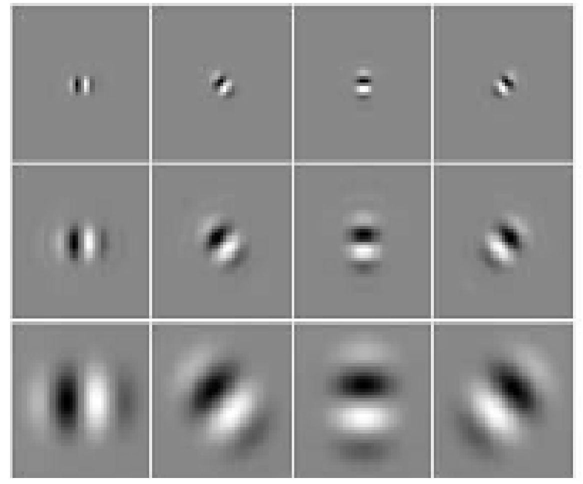
Figure 5: (Top Row) Original digitized sample textures: red granite, berry bush, figured maple, yellow coral. (Bottom Rows) Synthetic solid textured teapots.

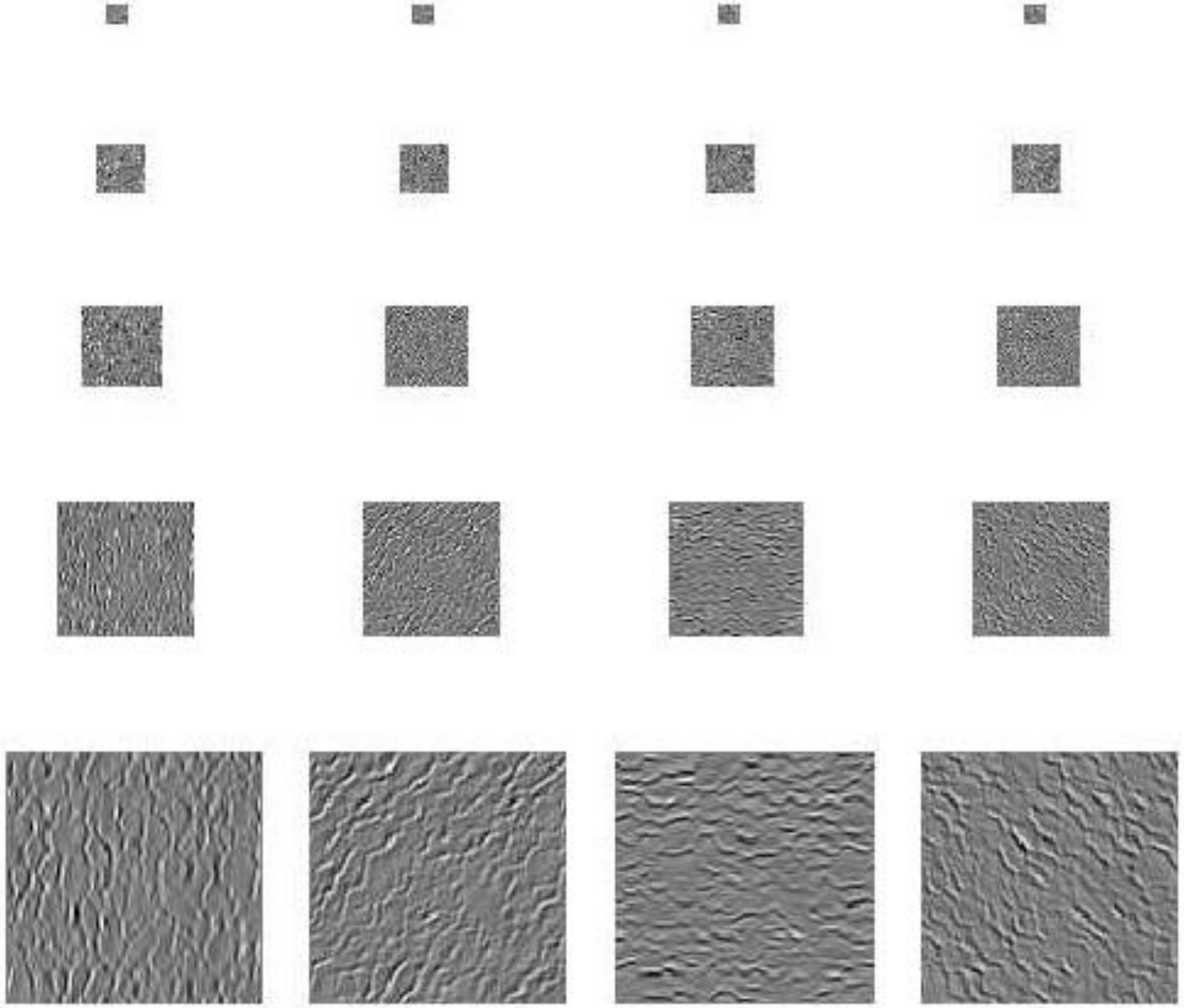
https://www.cns.nyu.edu/heegerlab/content/publications/Heeger-siggraph95.pdf

SIGGRAPH 1995

Multi-scale filter decomposition (steerable pyramid)

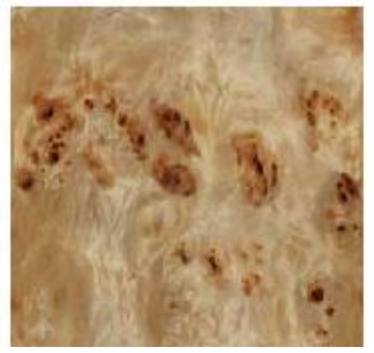
Filter



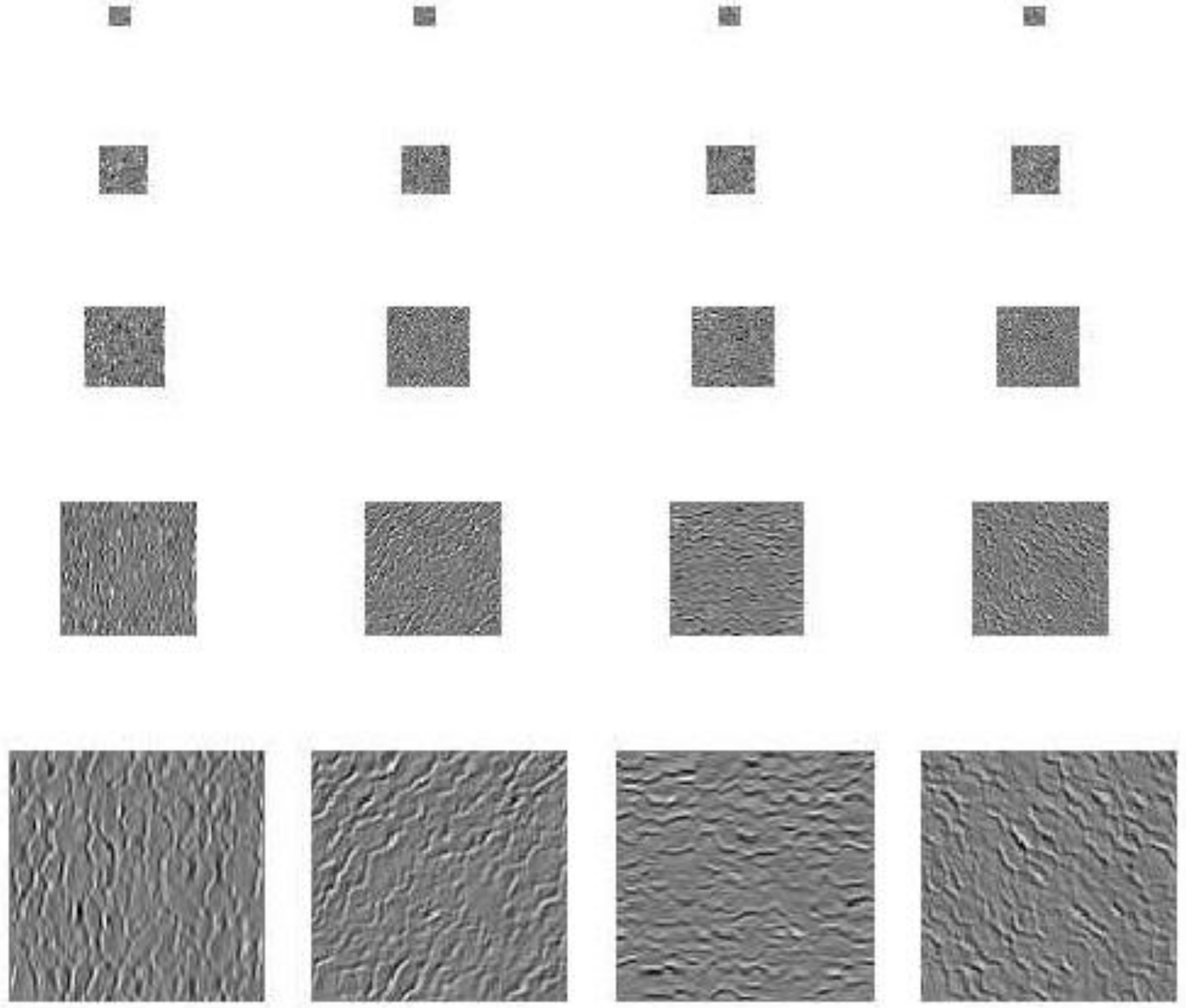


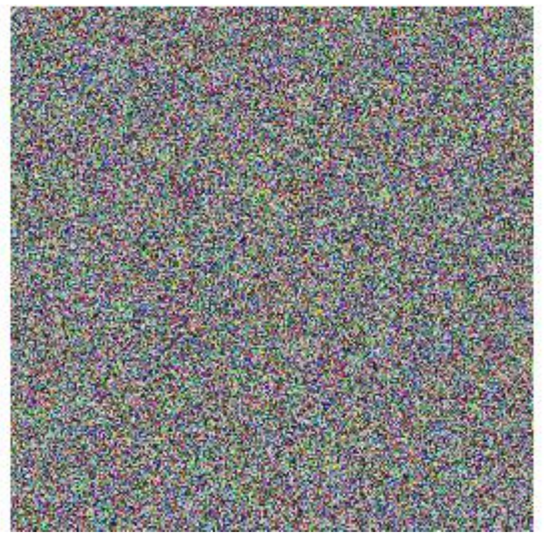
Input image

Step 1: Convolve with filterbank

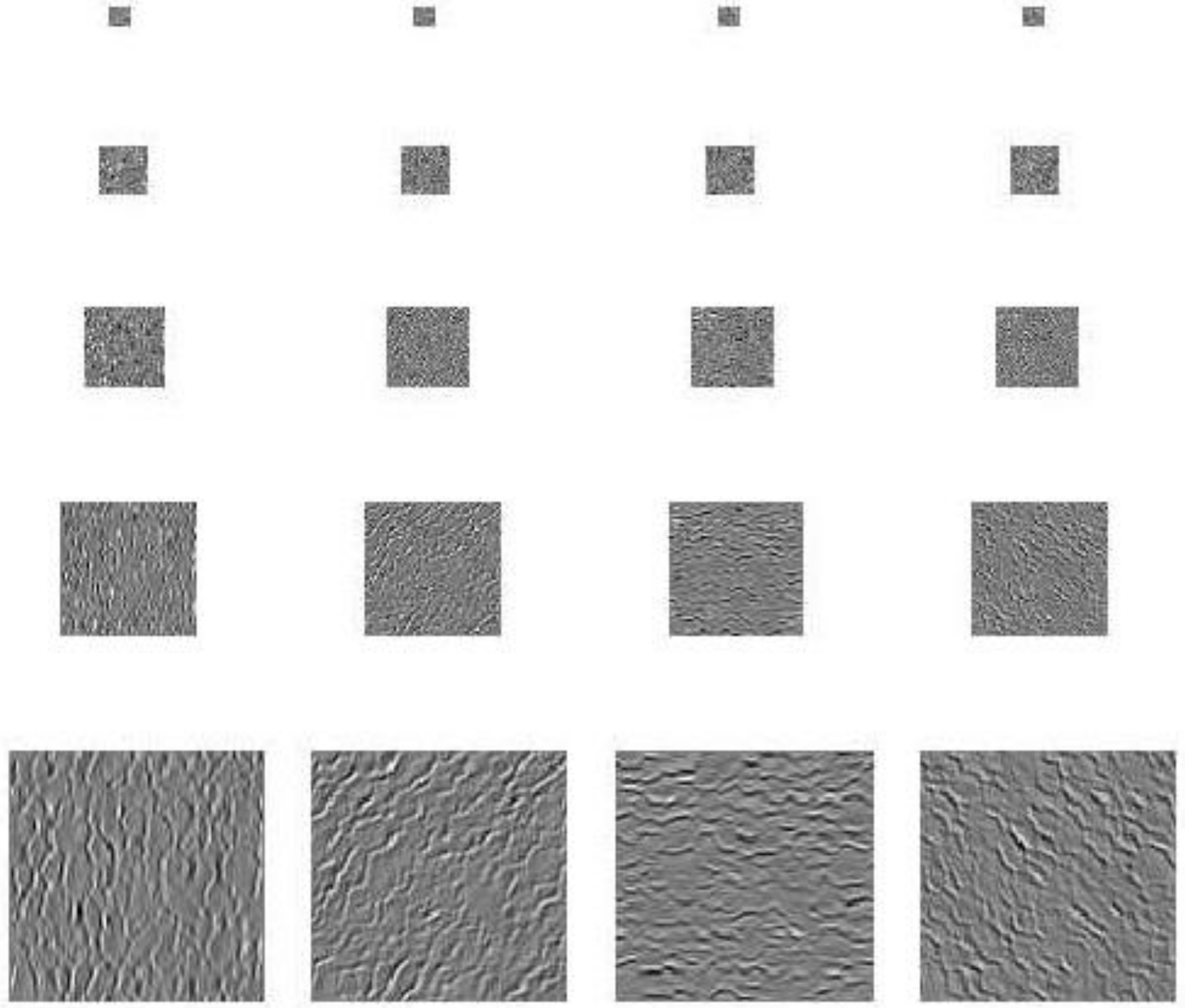


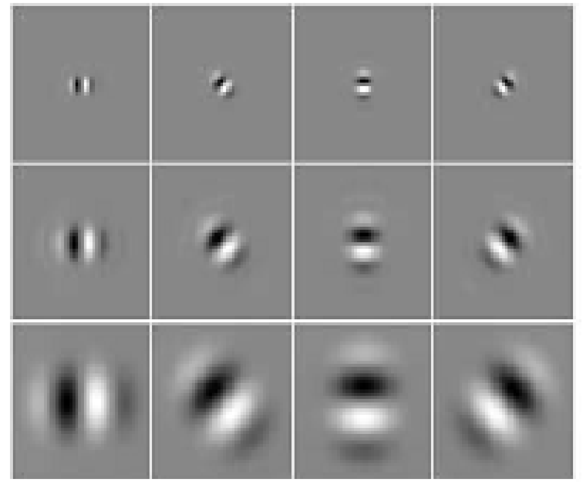
input



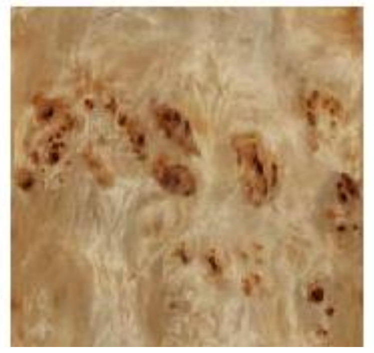


Noise image

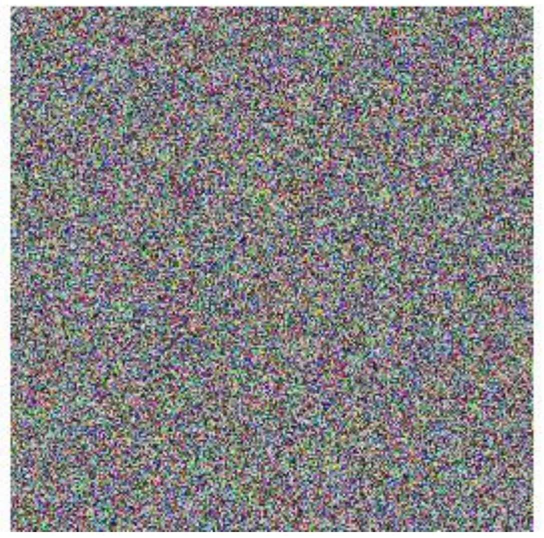




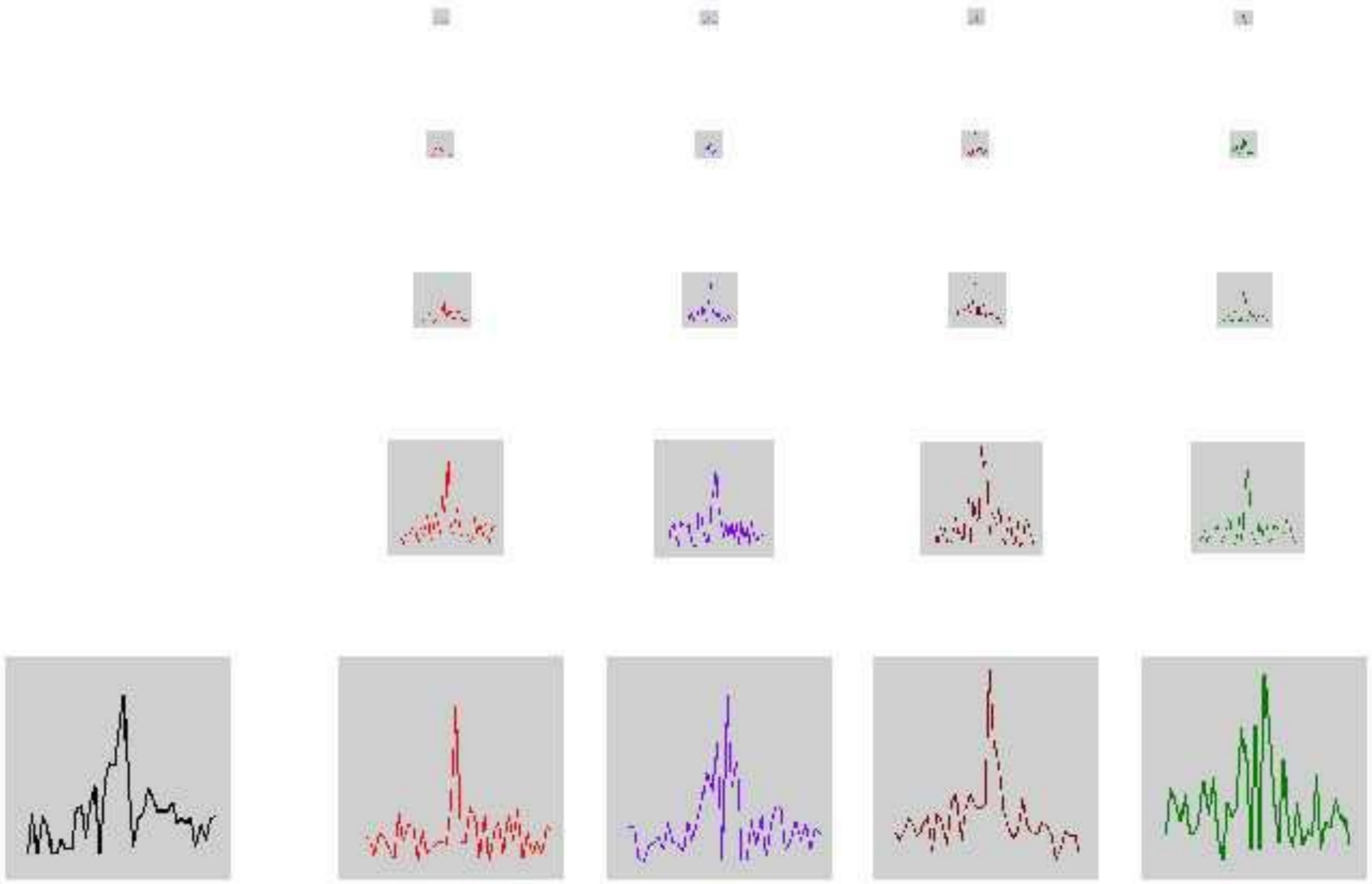
Step 2: match per--channel histograms

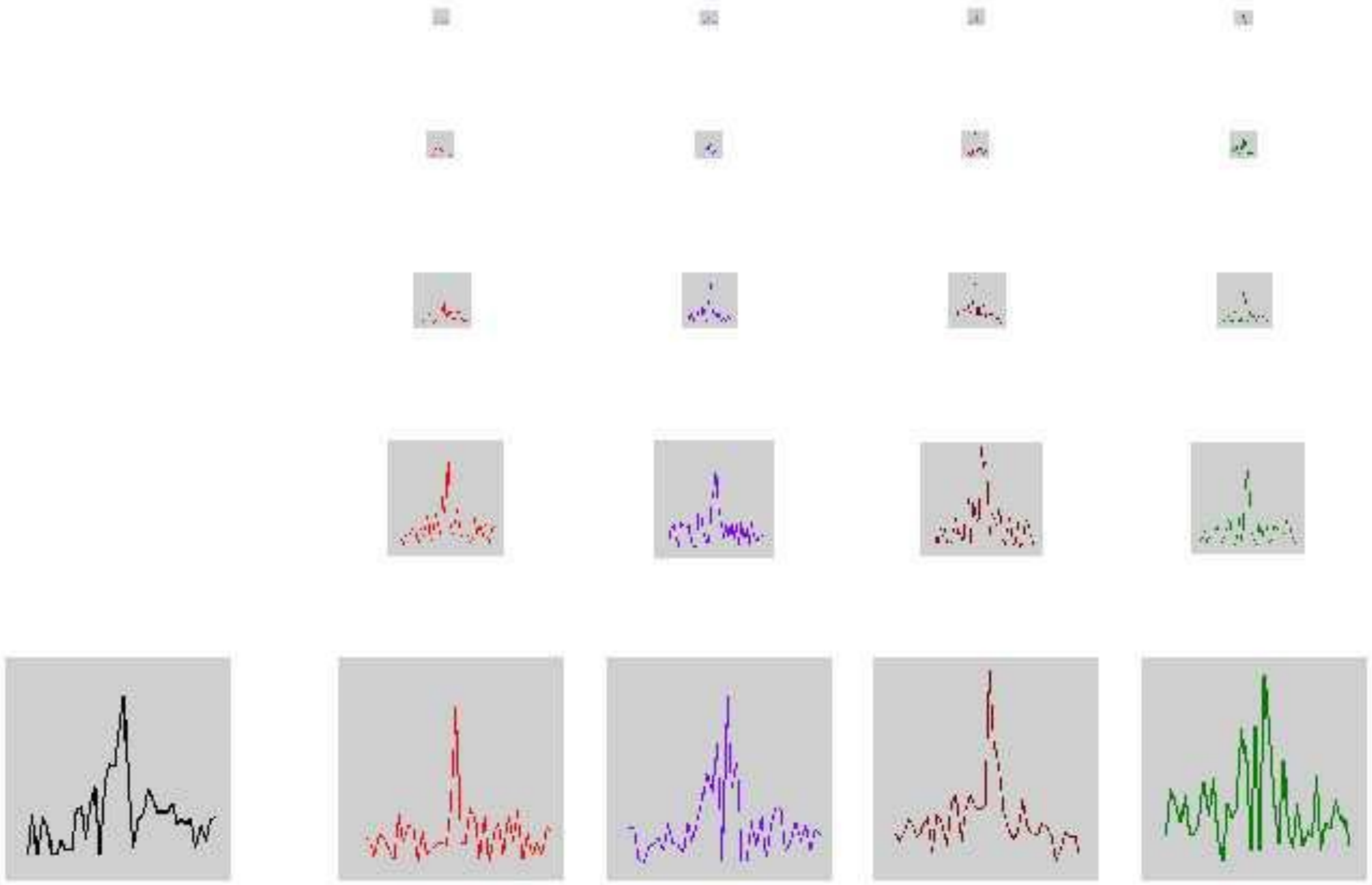


input

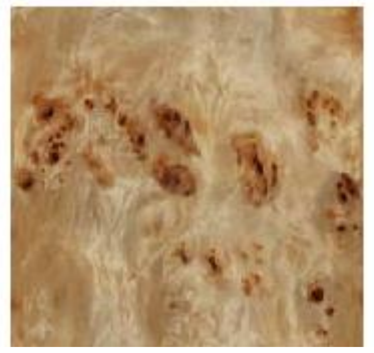


Noise image

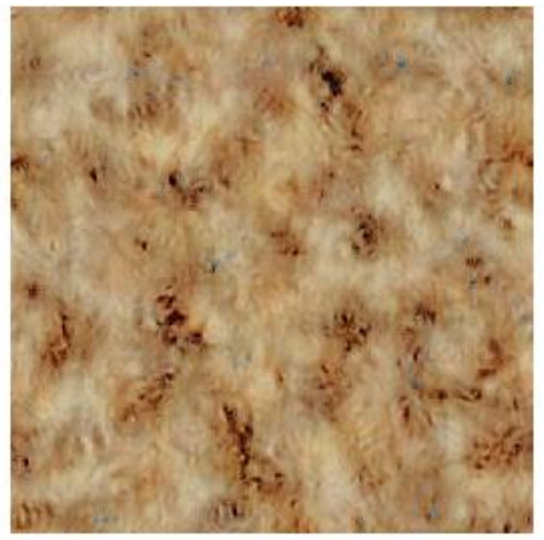




Step 3: collapse pyramid and repeat!

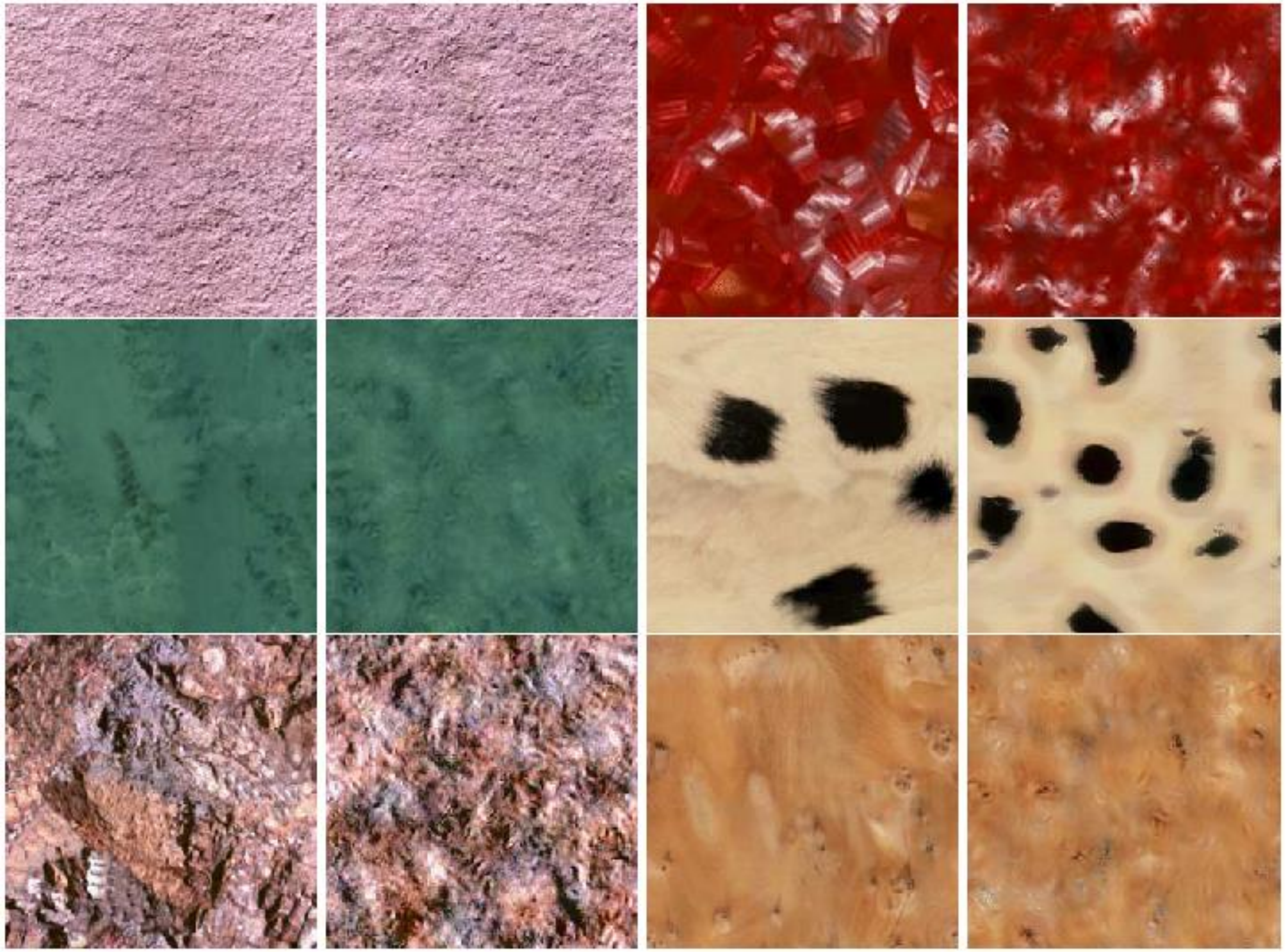


input

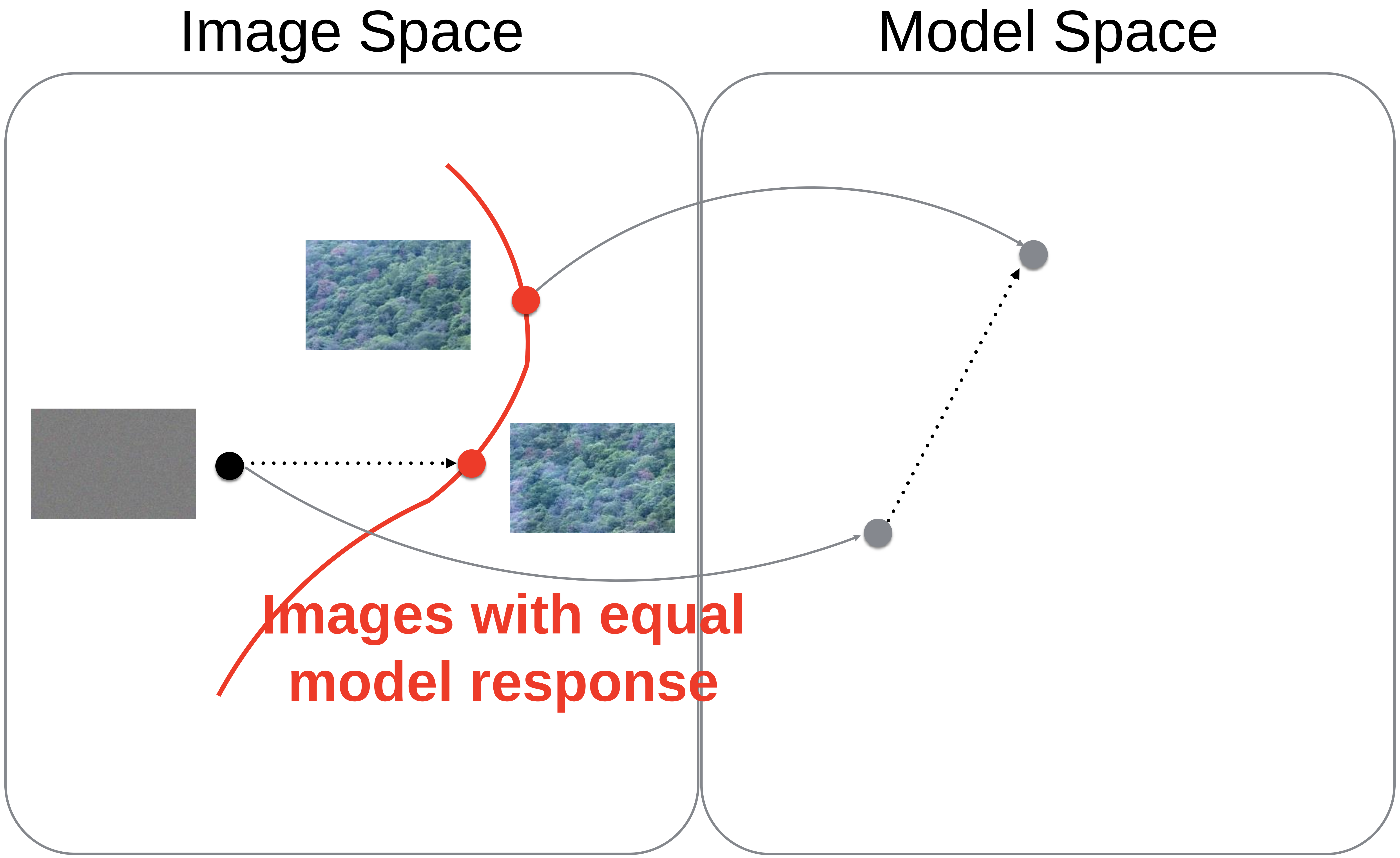


Noise image

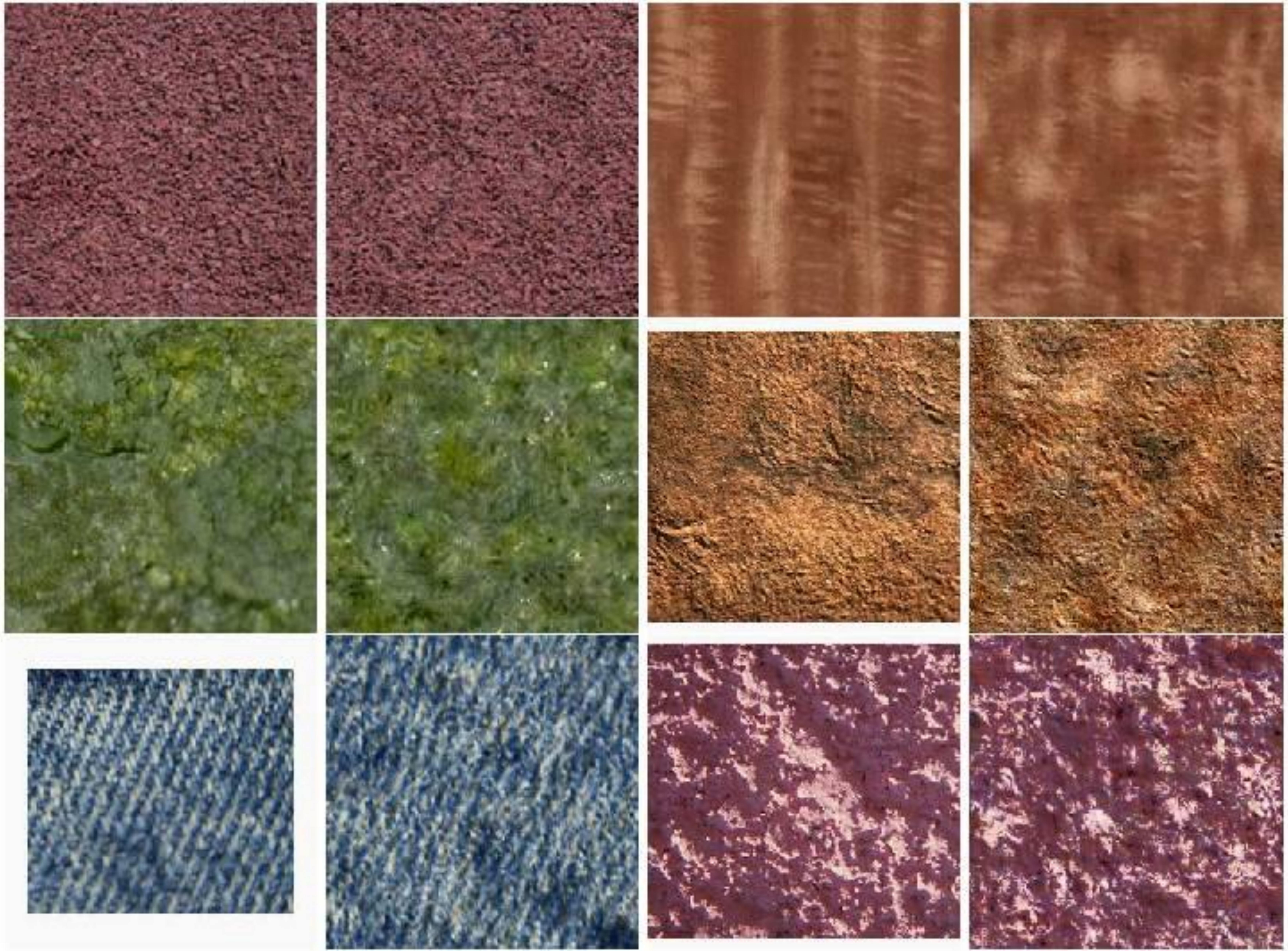
Heeger & Bergen, SIGGRAPH‘95



Texture Synthesis



Portilla & Simoncelli (2000)



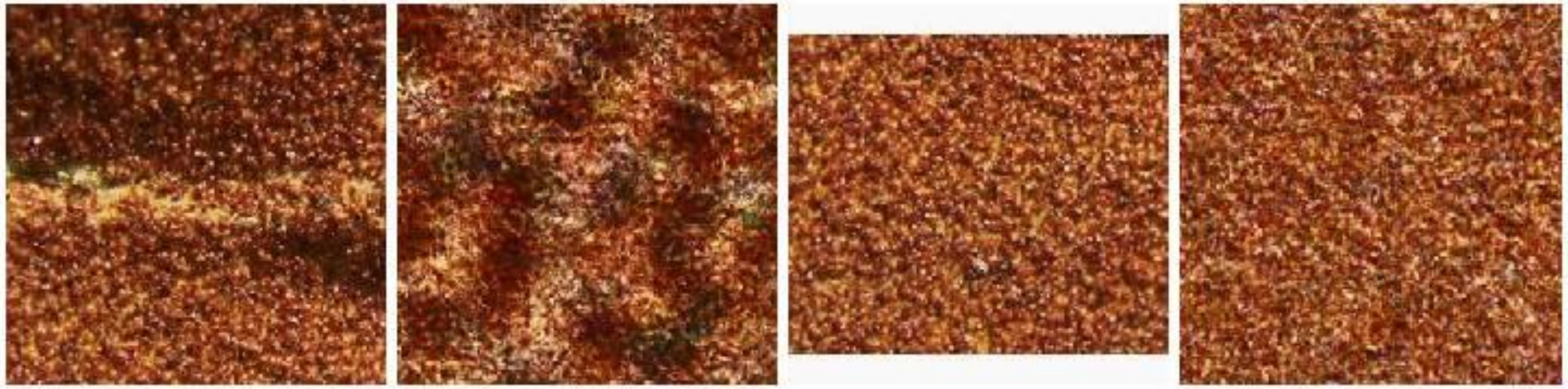


Figure 7: (Left pair) Inhomogeneous input texture produces blotchy synthetic texture. (Right pair) Homogenous input.

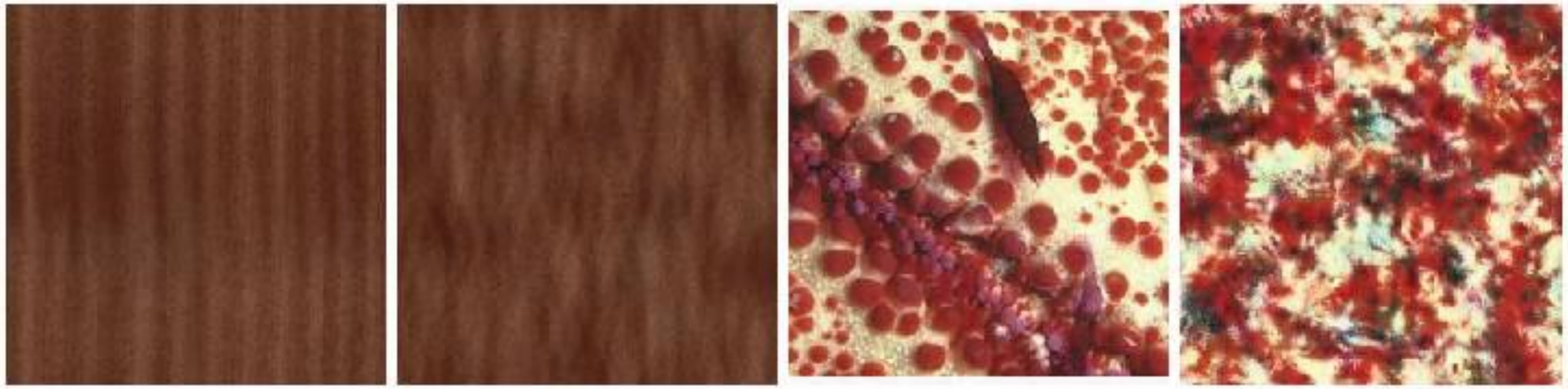


Figure 8: Examples of failures: wood grain and red coral.

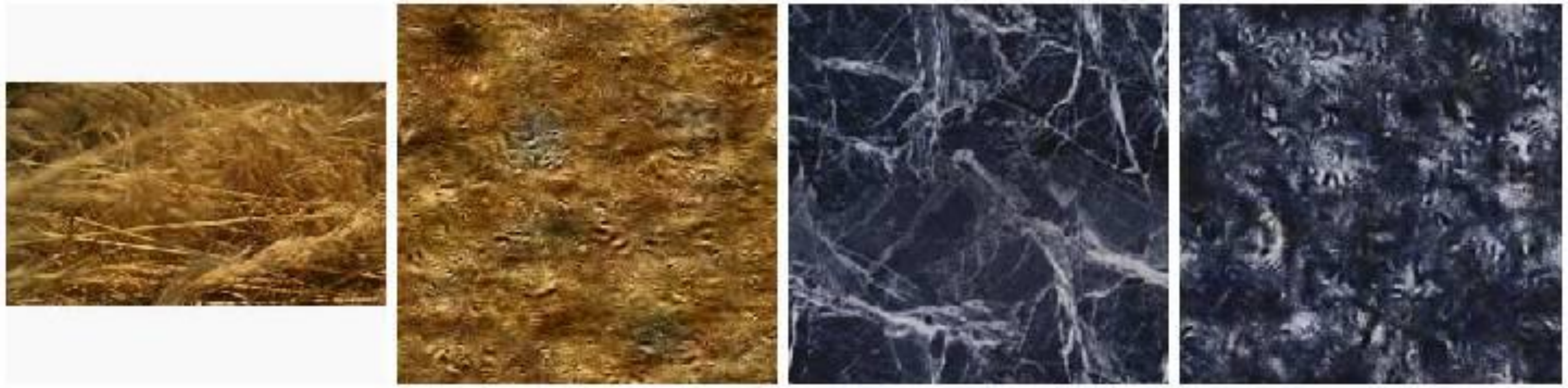


Figure 9: More failures: hay and marble.

Simoncelli & Portilla ’98+

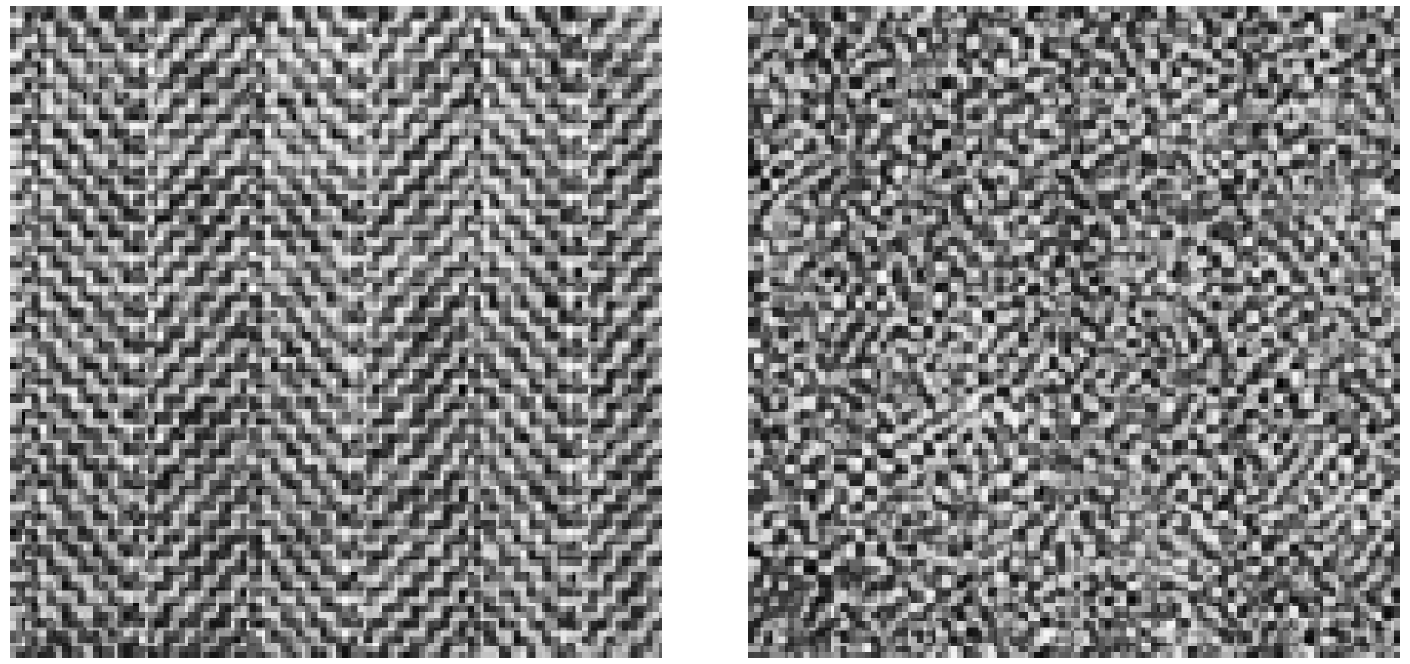
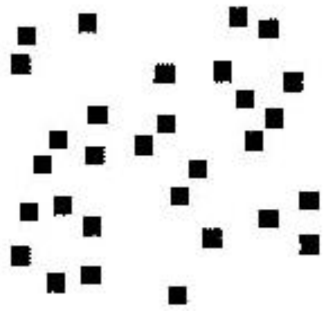


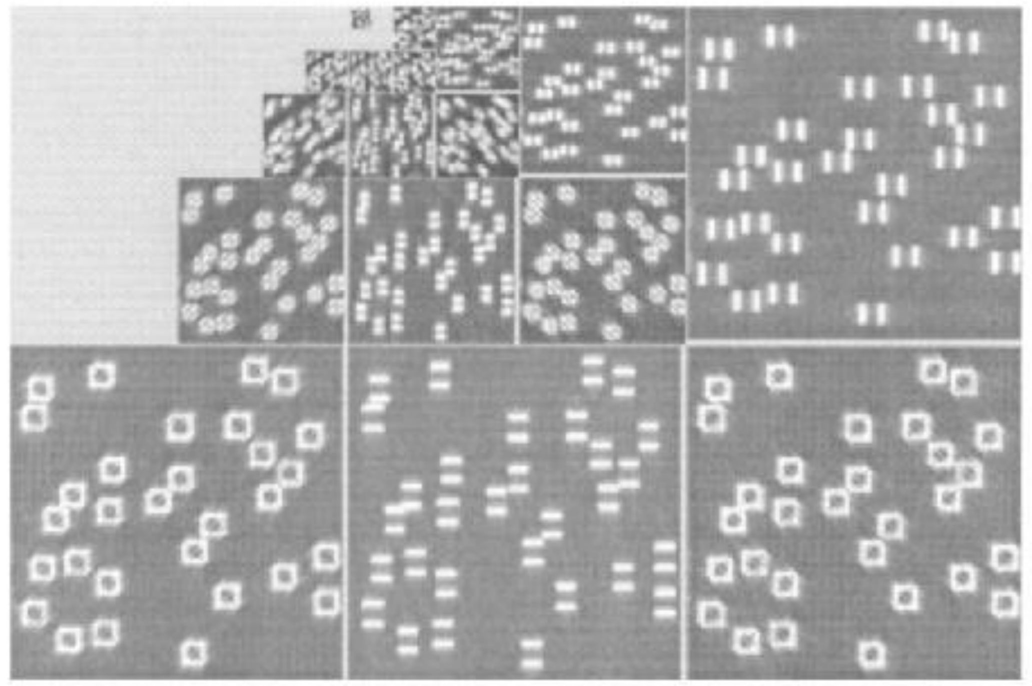
Figure 1. Textures with matching marginal statistics.

- Marginal statistics are not enough
- Neighboring filter responses are highly correlated
  - an edge at low-res will cause an edge at high-res
- Let’s match 2nd order statistics too!

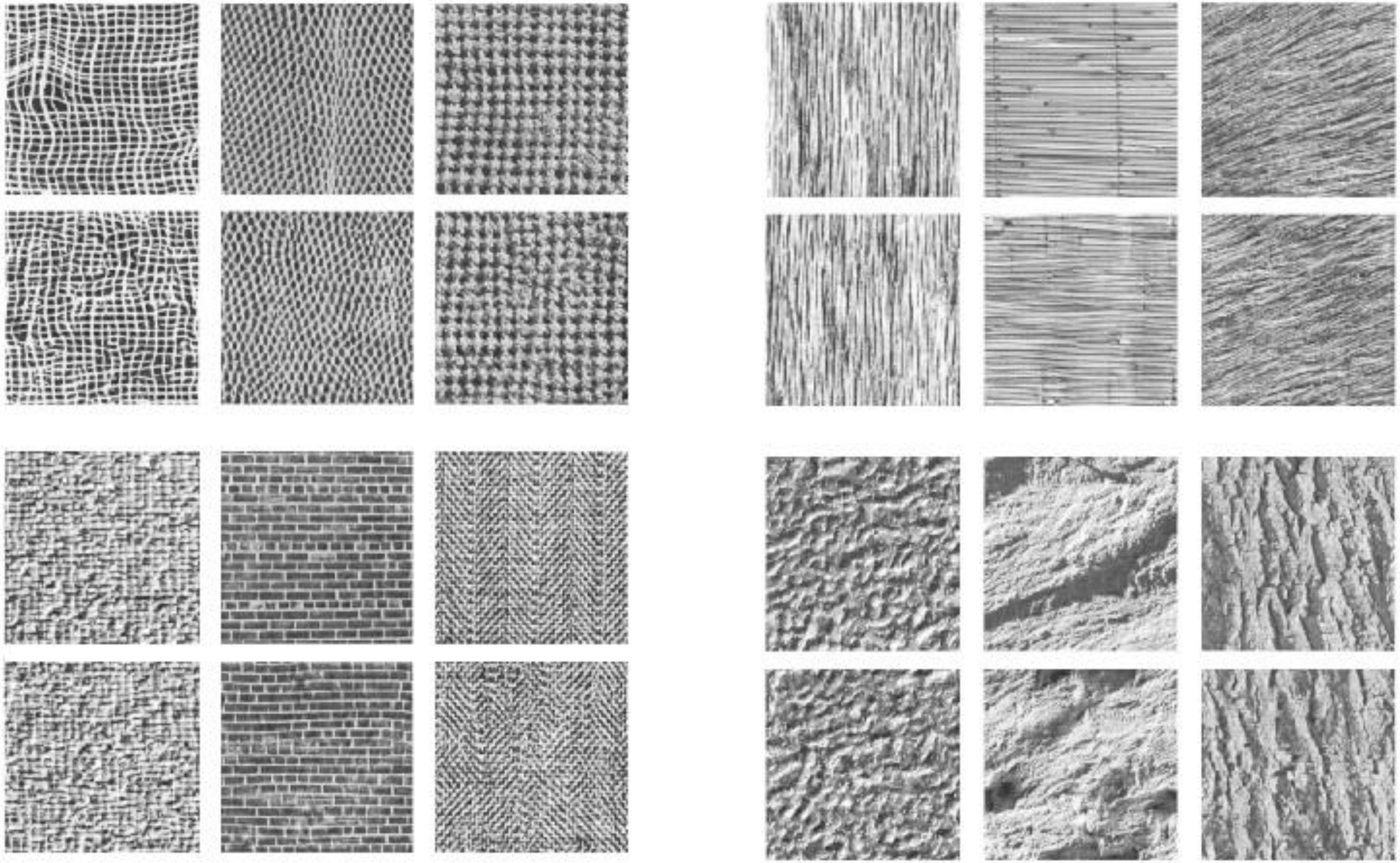
- J Portilla and E P Simoncelli. *A Parametric Texture Model based on Joint Statistics of Complex Wavelet Coefficients.* Int'l Journal of Computer Vision. 40(1):49-71, October, 2000.

Simoncelli & Portilla ’98+

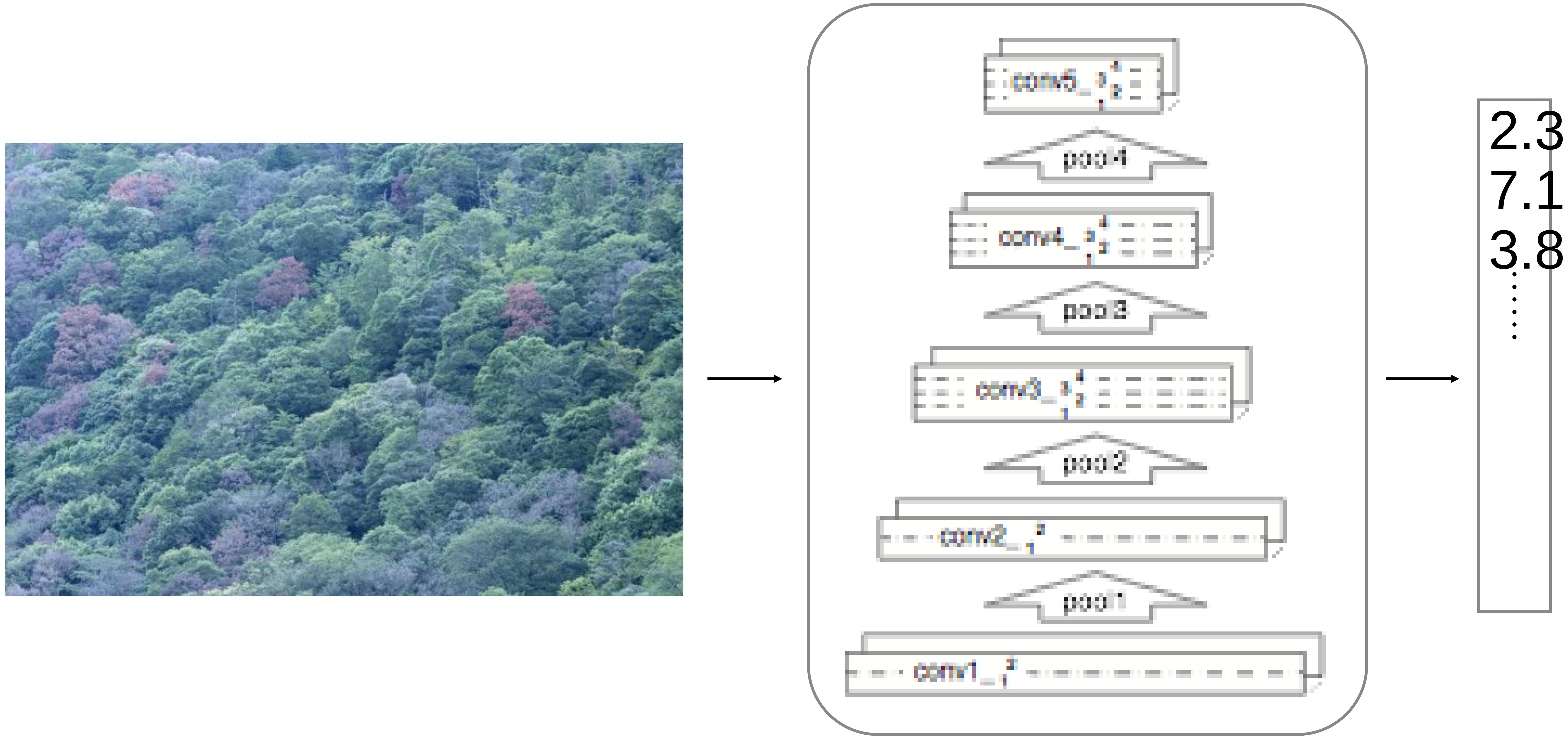




- Match joint histograms of pairs of filter responses at adjacent spatial locations, orientations, and scales.
- Optimize using repeated projections onto statistical constraint sufraces



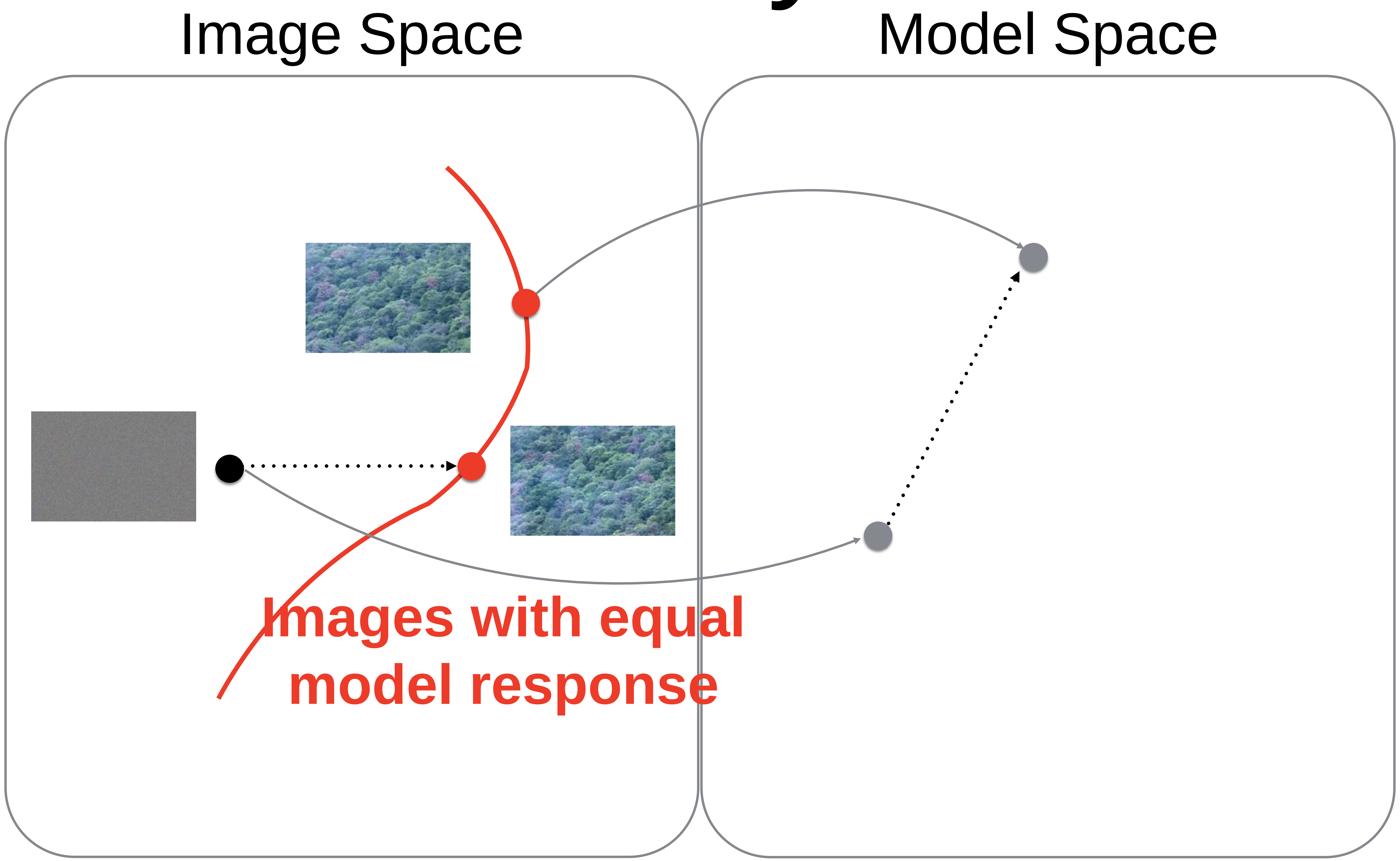
Convolutional Neural Network Texture Model



Convolutional Neural Network

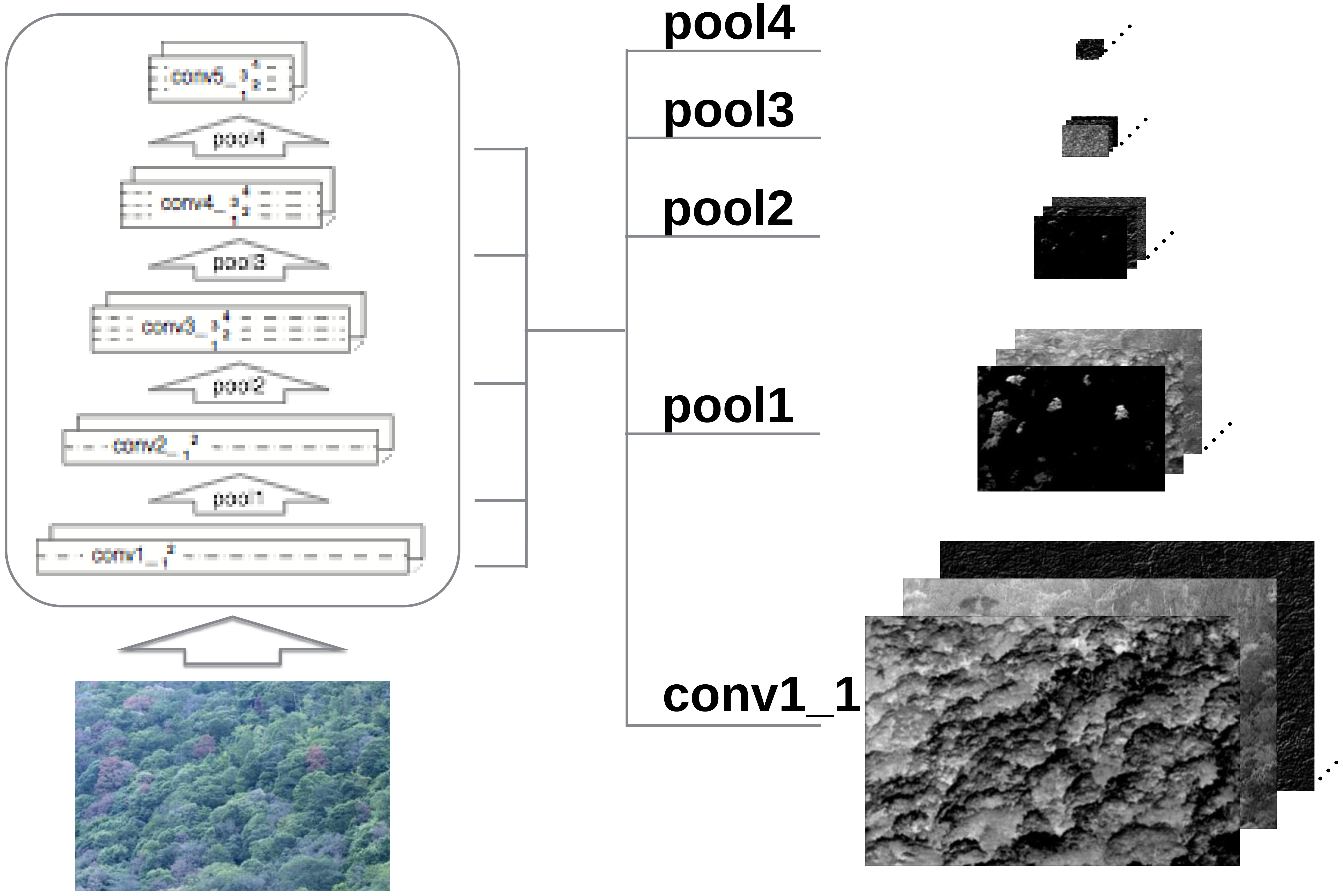
Gatys et al. (NIPS 2015)

Texture Synthesis



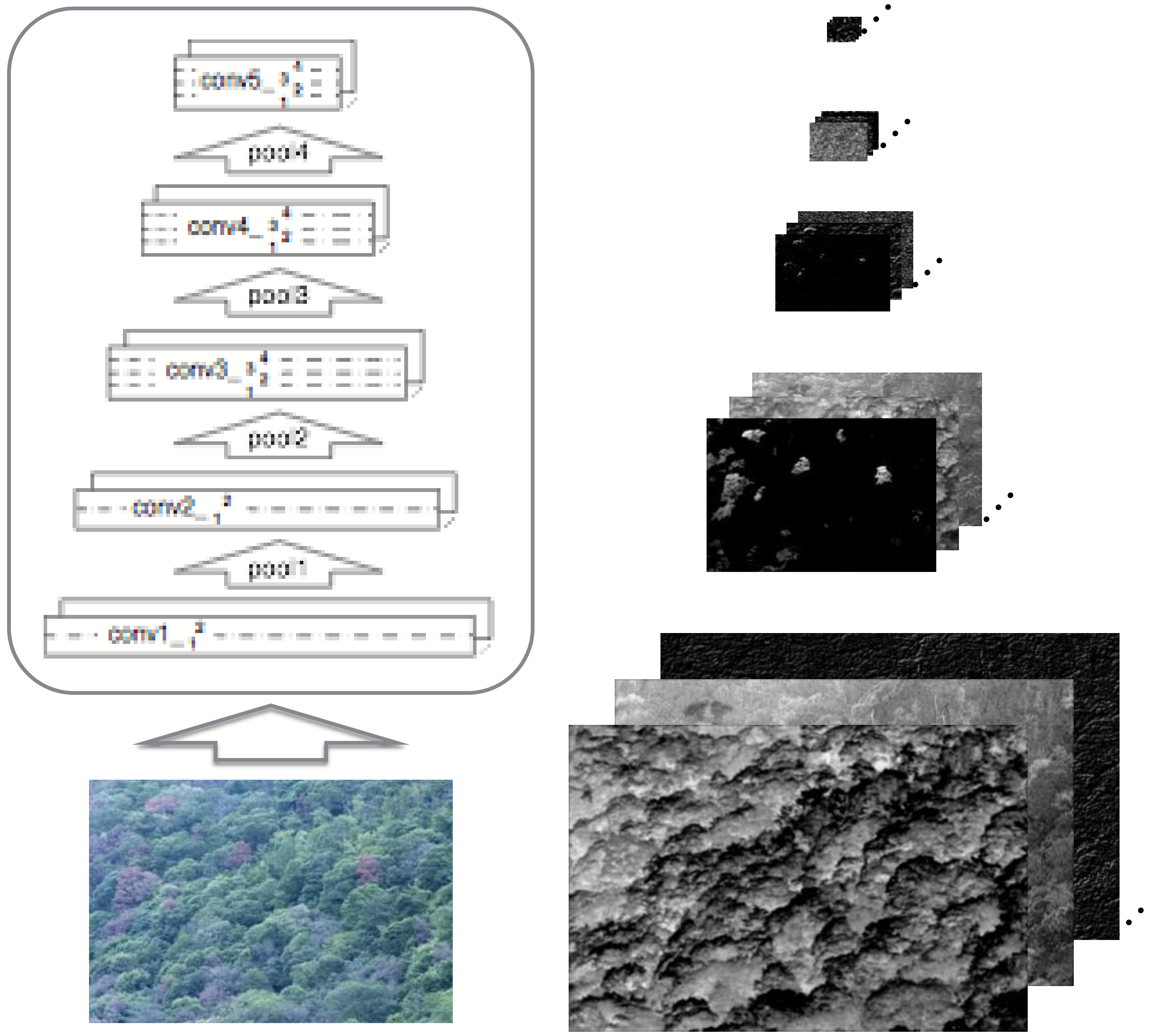
Portilla & Simoncelli (2000)

CNN - Multiscale Filter Bank



	# features
pool4	512
pool3	256
pool2	128
pool1	64
conv1_1	64

CNN - Texture Features



$$F = \left[\bar{f}_1, \bar{f}_2, \bar{f}_3, ..., \bar{f}_N\right]^T$$

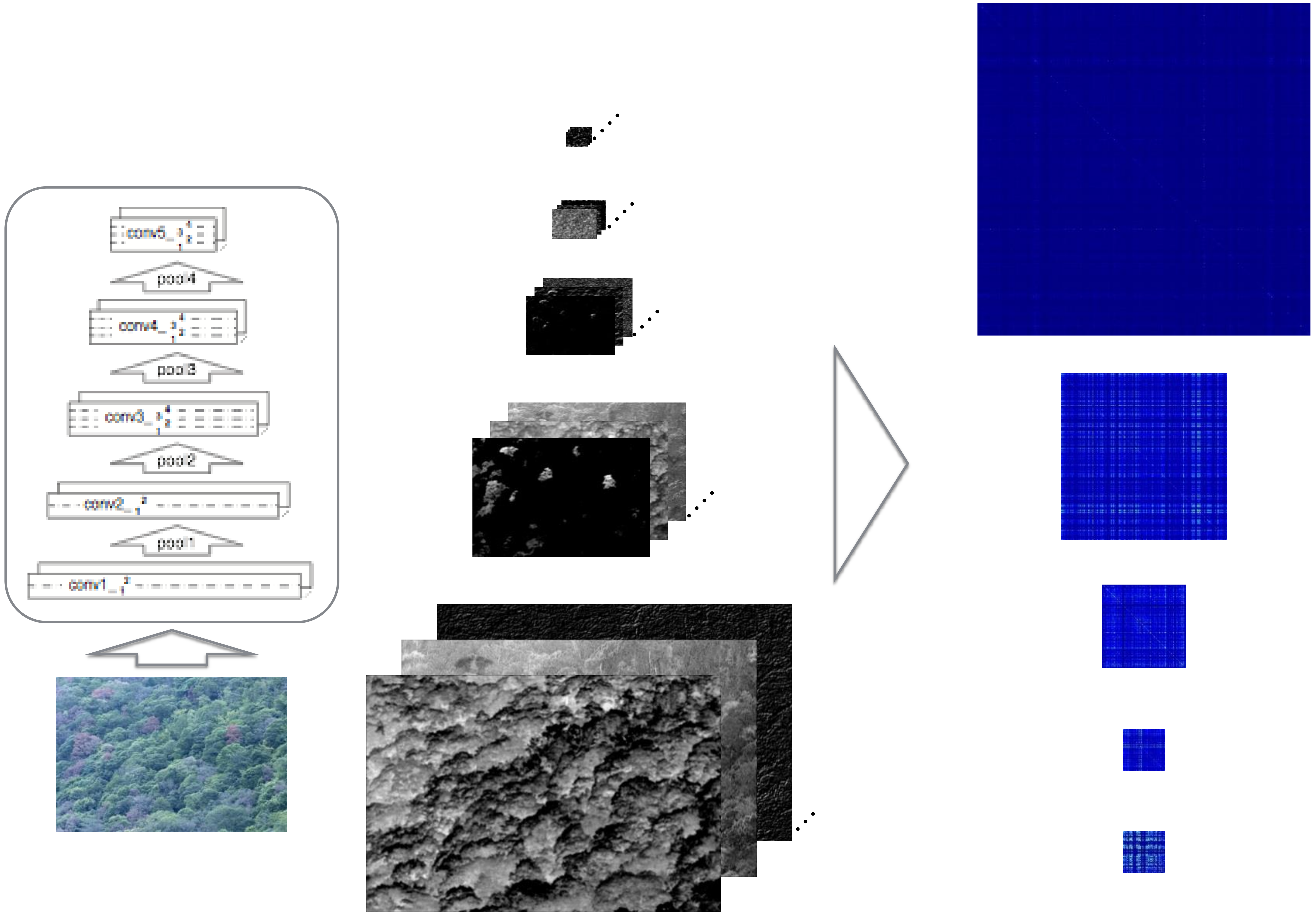
$$G = FF^T$$

$$= \begin{pmatrix} \langle \bar{f}_1, \bar{f}_1 \rangle & \cdots & \langle \bar{f}_1, \bar{f}_N \rangle \\ \langle \bar{f}_2, \bar{f}_1 \rangle & & \vdots \\ \vdots & \ddots & \vdots \\ \langle \bar{f}_N, \bar{f}_1 \rangle & \cdots & \langle \bar{f}_N, \bar{f}_N \rangle \end{pmatrix}$$

$$\langle \bar{f}_i, \bar{f}_j \rangle = \sum_k F_{ik} F_{jk}$$

CNN-Texture Features

Gram Matrices



# features

512

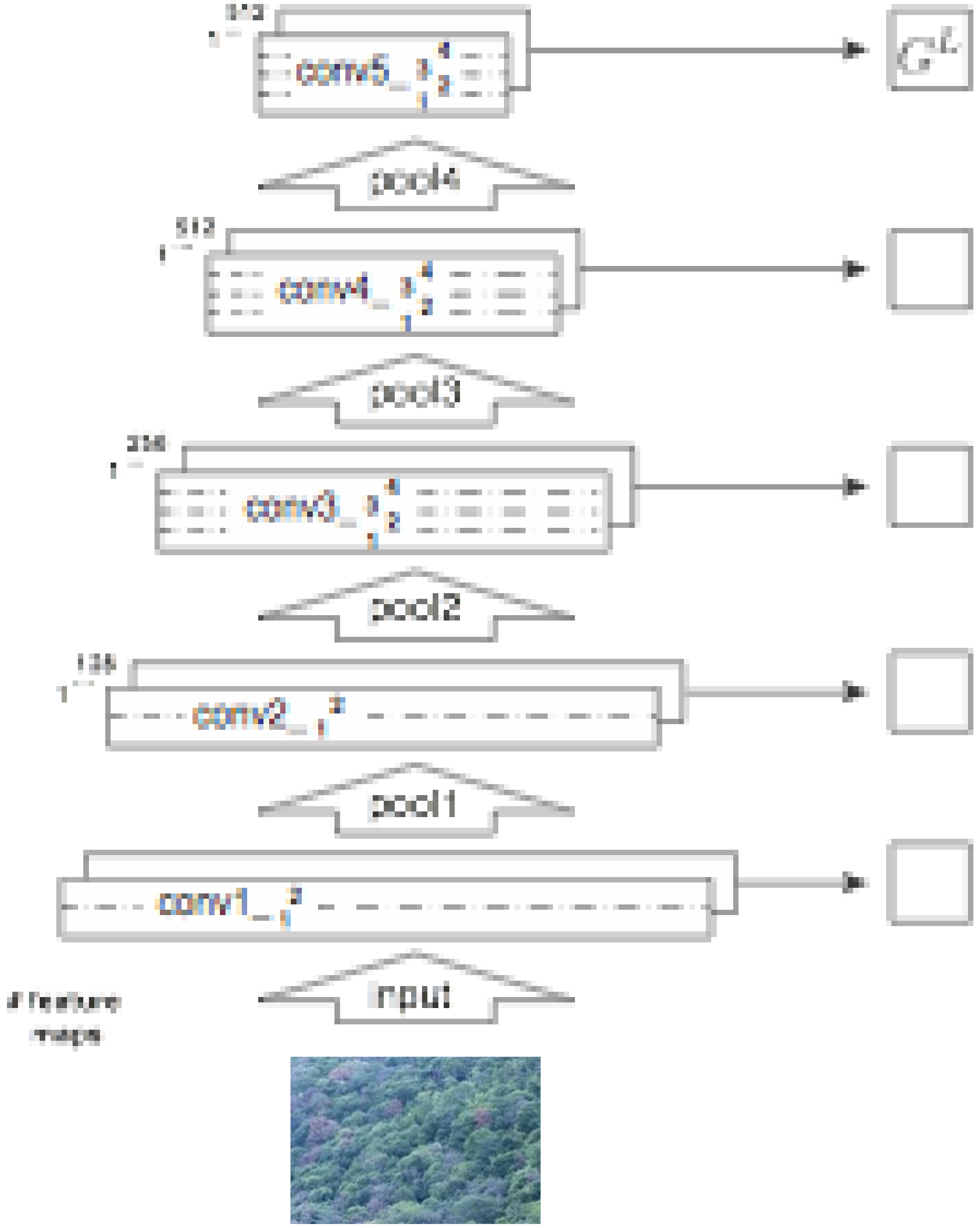
256

128

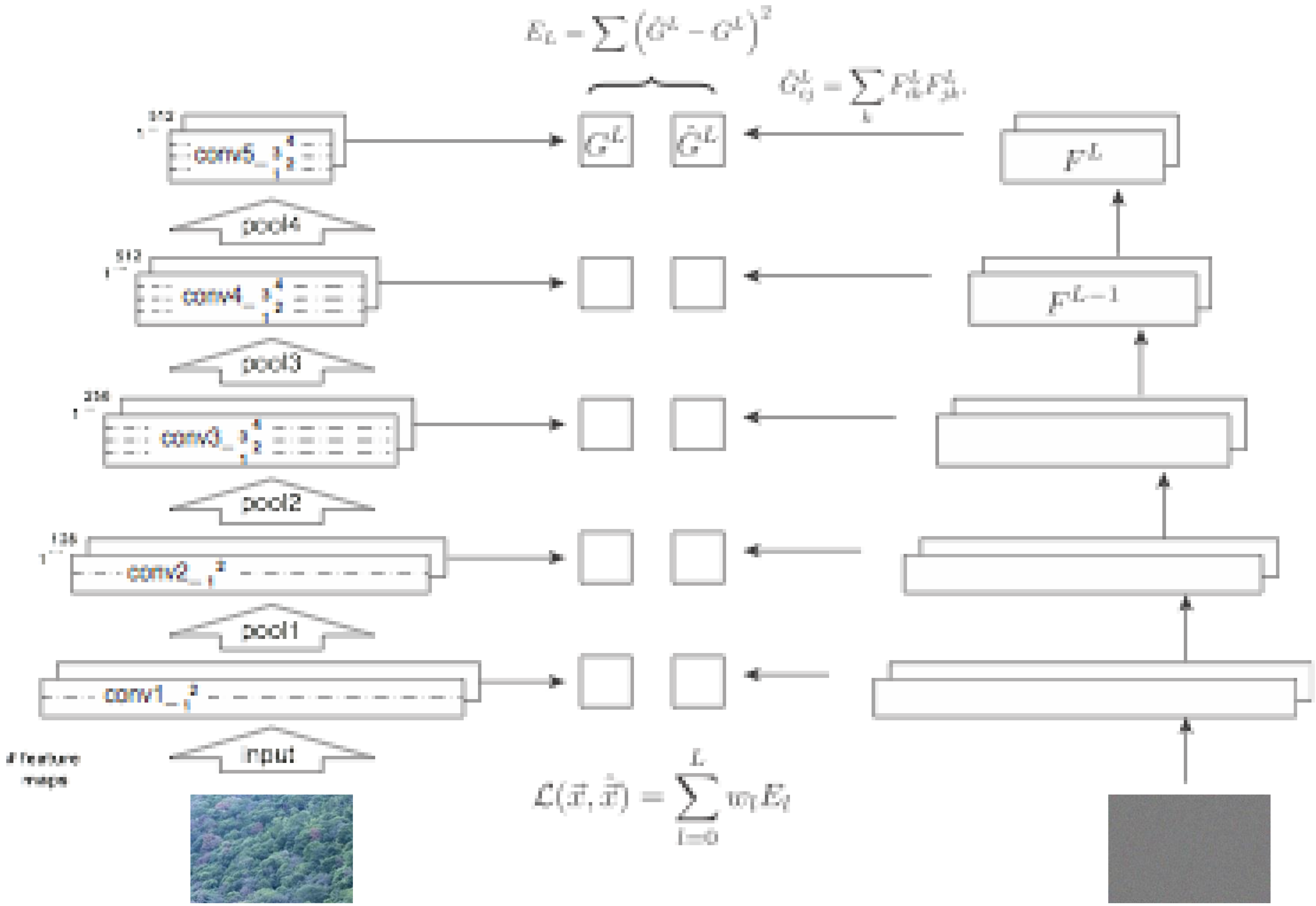
64

64

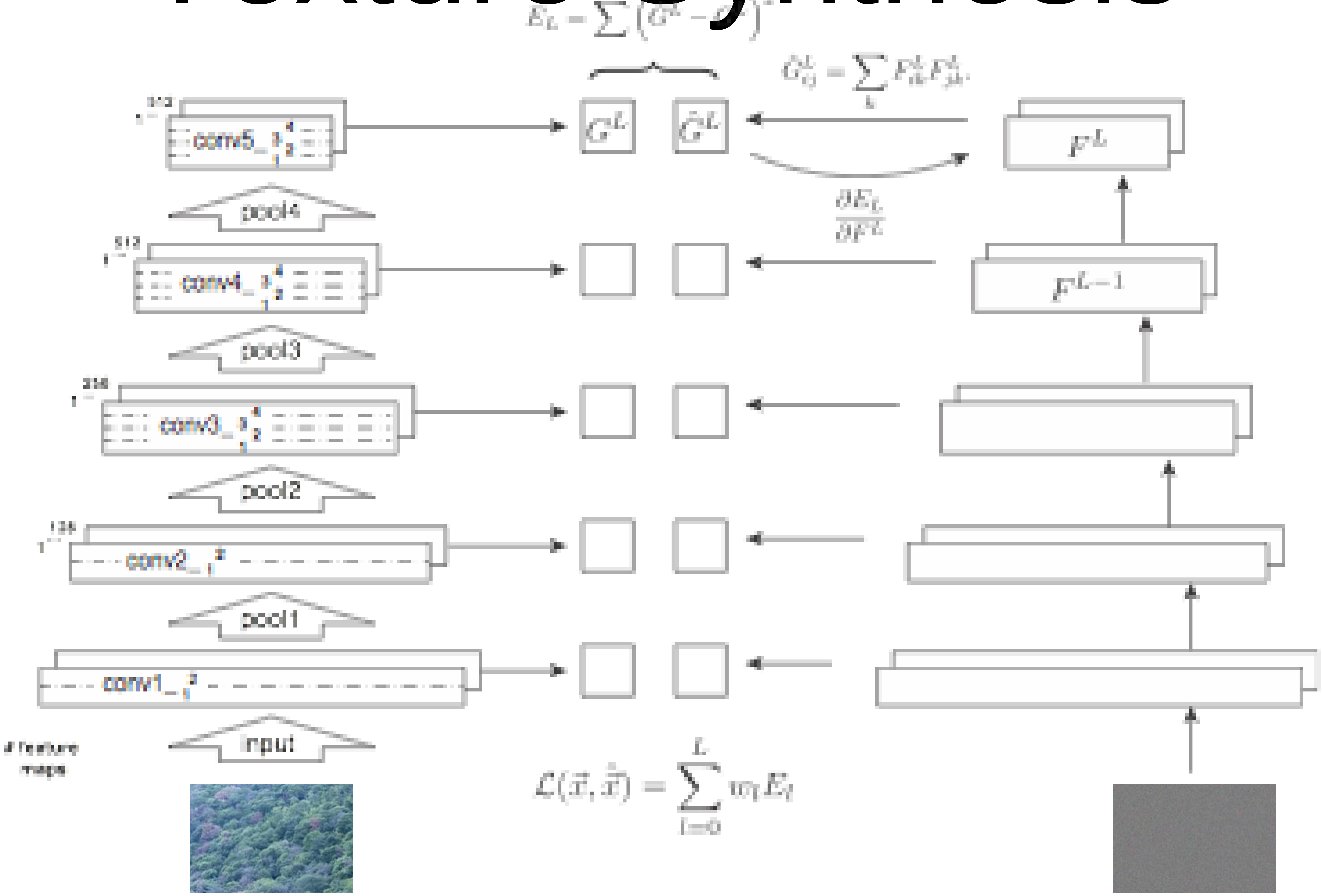
Texture Synthesis



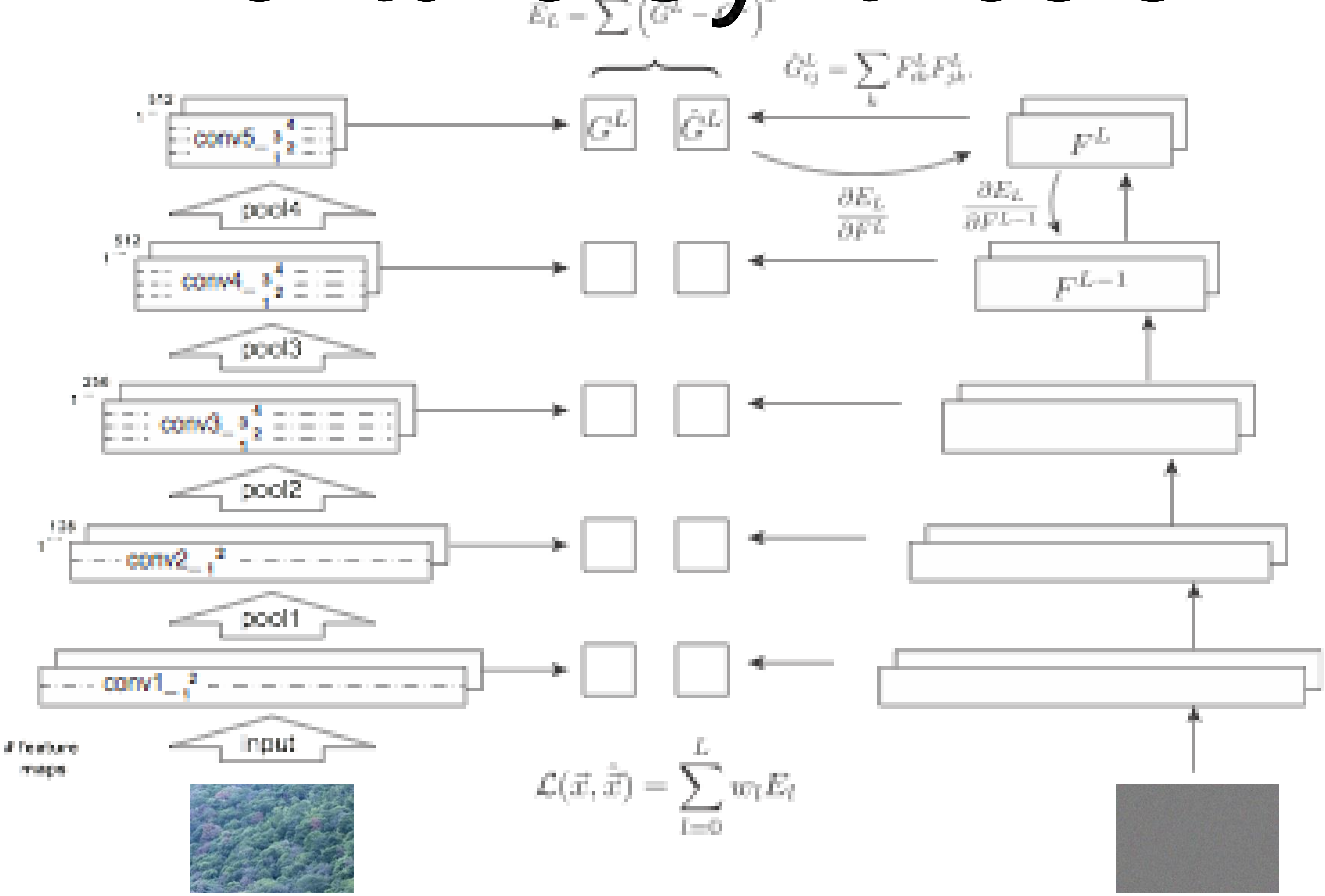
Texture Synthesis


$E_L = \sum \left(\hat{G}^L - G^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik} \hat{F}^L_{jk}$
512
conv5_ 1 2 3 4
G^L
$\hat{G}^L$
F^L
pool4
512
conv4_ 1 2 3 4
F^{L-1}
pool3
256
conv3_ 1 2 3 4
pool2
128
conv2_ 1 2
pool1
conv1_ 1 2
# feature maps
input
$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$

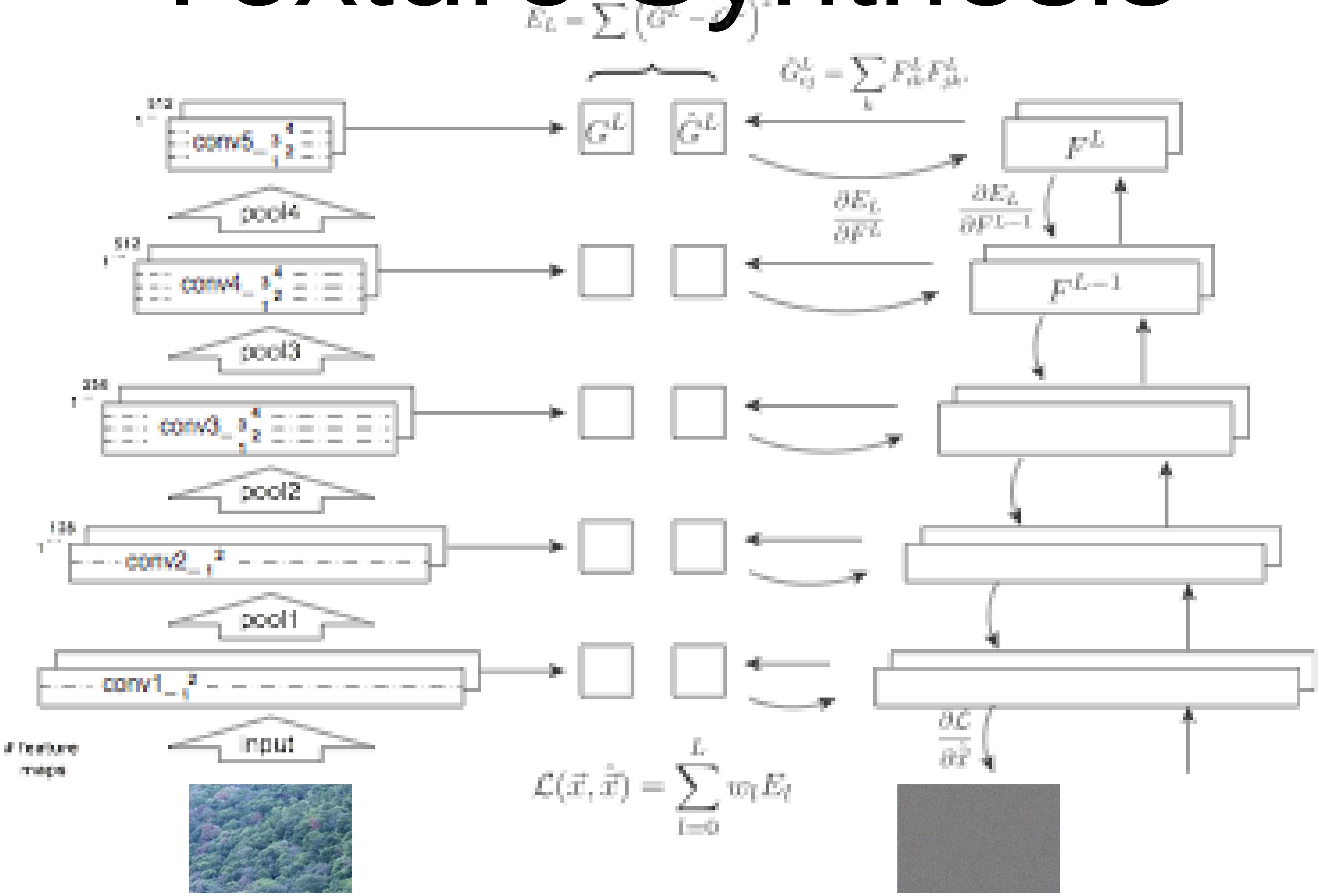
Texture Synthesis


$E_L = \sum \left(G^L - \hat{G}^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik}\hat{F}^L_{jk}$
G^L
$\hat{G}^L$
F^L
$\frac{\partial E_L}{\partial \hat{F}^L}$
F^{L-1}
512
conv5_ 3 4 2 1
pool4
512
conv4_ 3 4 2 1
pool3
256
conv3_ 3 4 2 1
pool2
128
conv2_ 1 2
pool1
conv1_ 1 2
input
# feature maps
$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$

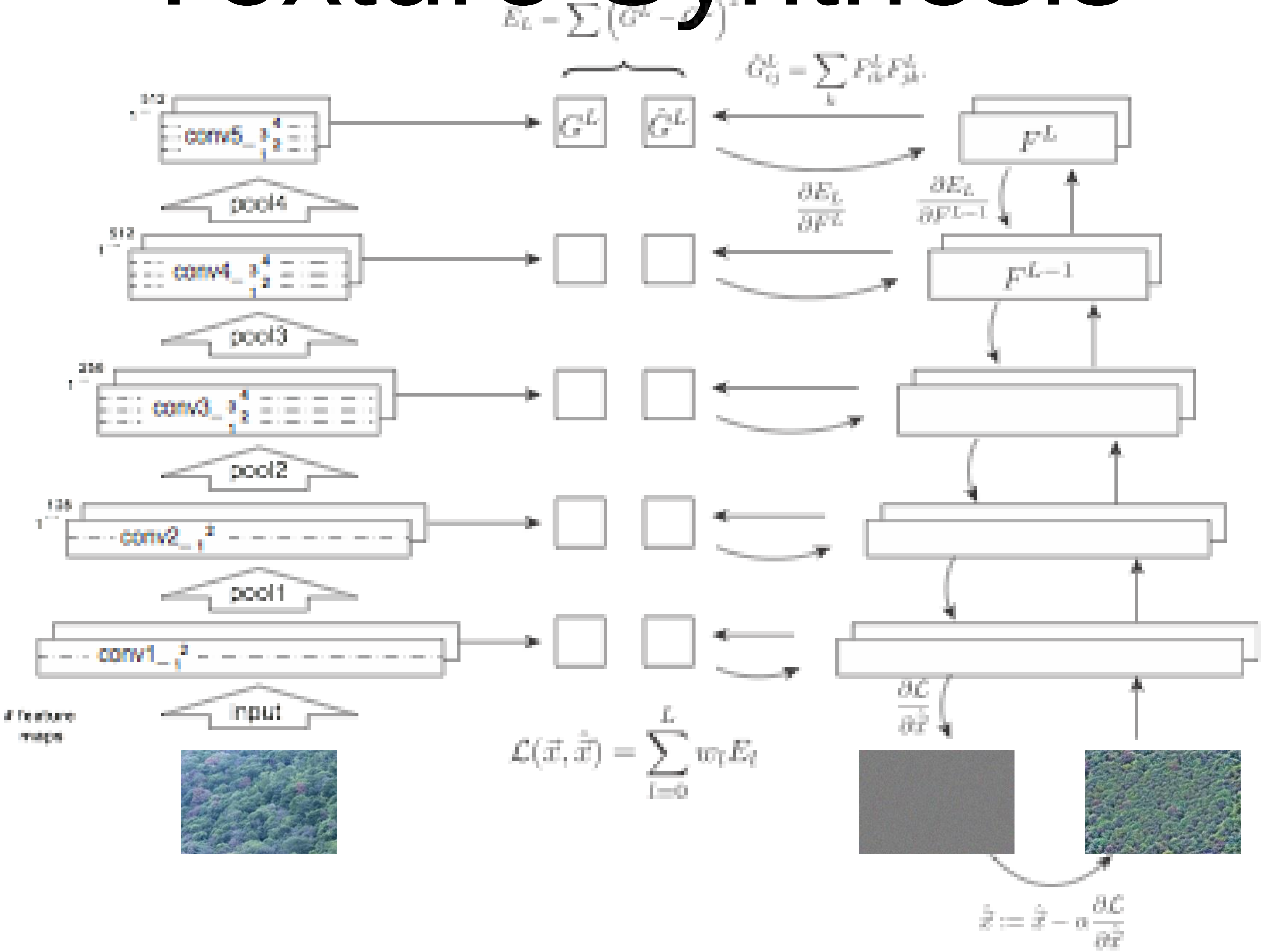
Texture Synthesis


$E_L = \sum \left(G^L - \hat{G}^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik} \hat{F}^L_{jk}$
G^L
$\hat{G}^L$
F^L
$\frac{\partial E_L}{\partial \hat{F}^L}$
$\frac{\partial E_L}{\partial \hat{F}^{L-1}}$
F^{L-1}
512
conv5_1 2 3 4
pool4
512
conv4_1 2 3 4
pool3
256
conv3_1 2 3 4
pool2
128
conv2_1 2
pool1
conv1_1 2
input
# feature maps
1
$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$

Texture Synthesis

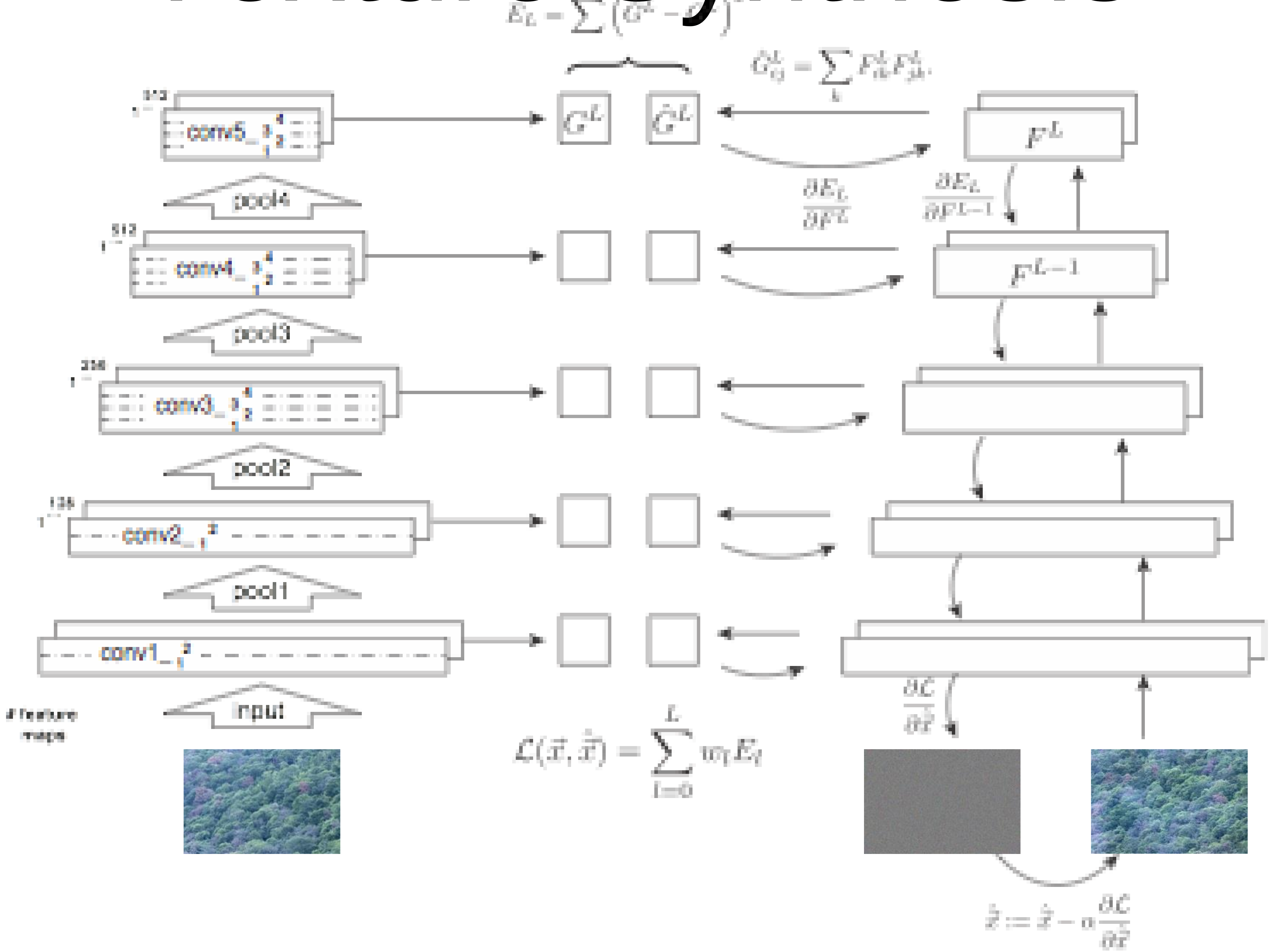

$E_L = \sum \left(G^L - \hat{G}^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik}\hat{F}^L_{jk}$
512
conv5_ 1 2 3 4
pool4
512
conv4_ 1 2 3 4
pool3
256
conv3_ 1 2 3 4
pool2
128
conv2_ 1 2
pool1
conv1_ 1 2
input
# feature maps
G^L
$\hat{G}^L$
F^L
F^{L-1}
$\frac{\partial E_L}{\partial \hat{F}^L}$
$\frac{\partial E_L}{\partial \hat{F}^{L-1}}$
$\frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}$
$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$

Texture Synthesis

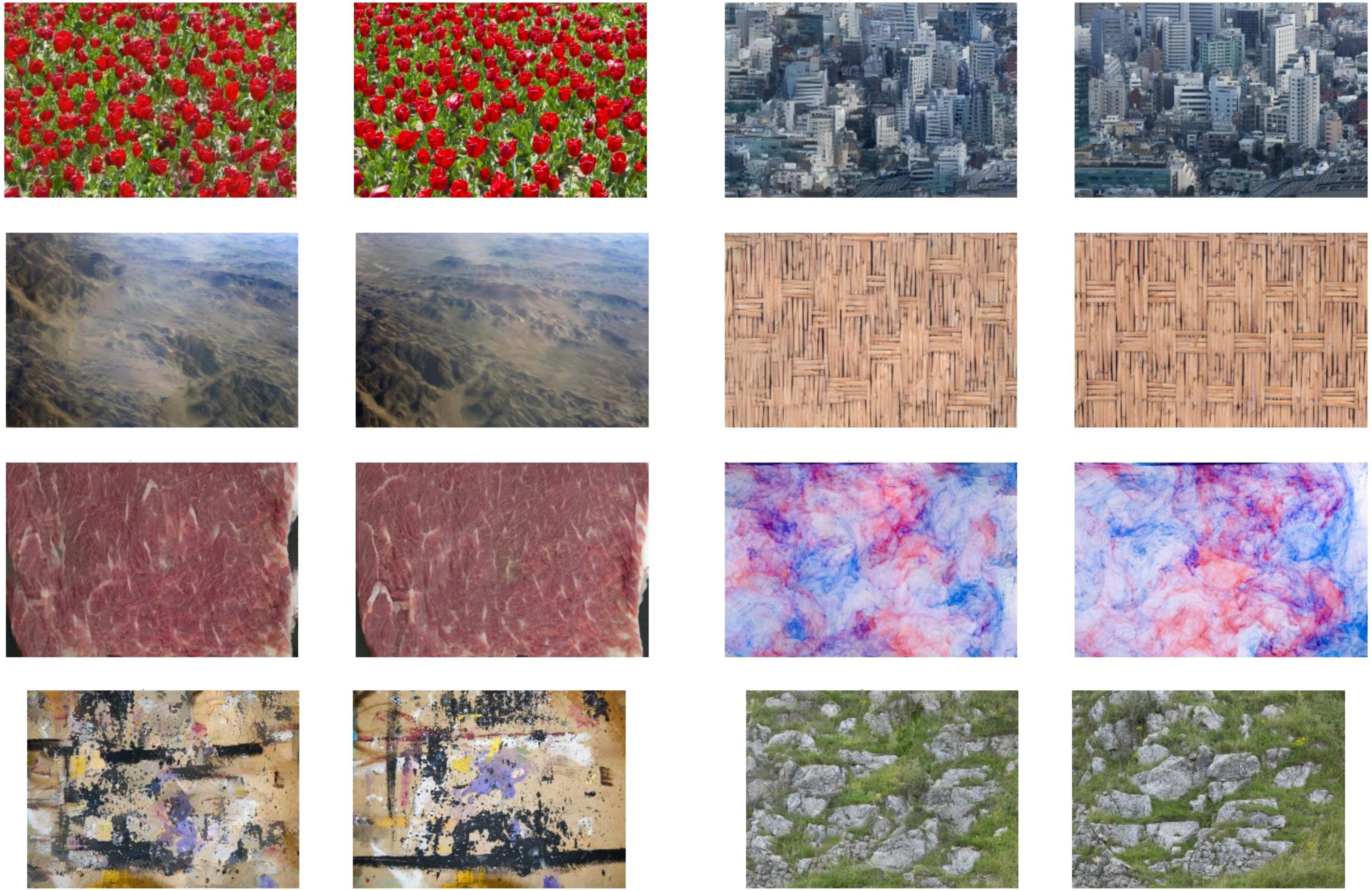


$E_L = \sum \left(G^L - \hat{G}^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik} \hat{F}^L_{jk}$
G^L
$\hat{G}^L$
F^L
$\frac{\partial E_L}{\partial \hat{F}^L}$
$\frac{\partial E_L}{\partial \hat{F}^{L-1}}$
F^{L-1}
512
conv5_ 1 2 3 4
pool4
512
conv4_ 1 2 3 4
pool3
256
conv3_ 1 2 3 4
pool2
128
conv2_ 1 2
pool1
conv1_ 1 2
input
# feature maps
$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$
$\frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}$
$\hat{\vec{x}} := \hat{\vec{x}} - \alpha \frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}$

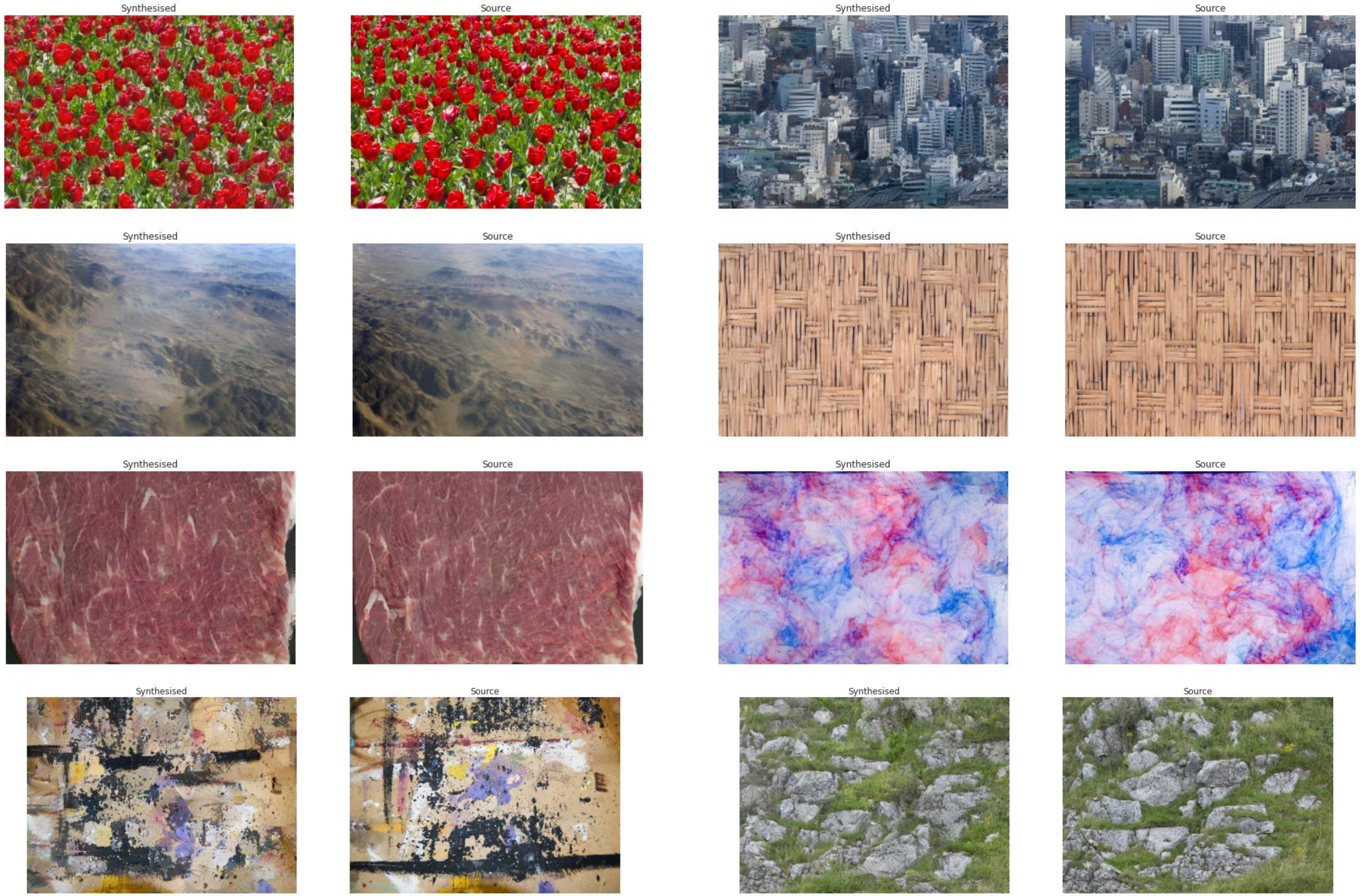
Texture Synthesis


$E_L = \sum \left(G^L - \hat{G}^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik} \hat{F}^L_{jk}$
512
conv5_ 1 2 3 4
pool4
512
conv4_ 1 2 3 4
pool3
256
conv3_ 1 2 3 4
pool2
128
conv2_ 1 2
pool1
64
conv1_ 1 2
input
# feature maps
G^L
$\hat{G}^L$
F^L
F^{L-1}
$\frac{\partial E_L}{\partial \hat{F}^L}$
$\frac{\partial E_L}{\partial \hat{F}^{L-1}}$
$\mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$
$\frac{\partial \mathcal{L}}{\partial \hat{x}}$
$\hat{x} := \hat{x} - \alpha \frac{\partial \mathcal{L}}{\partial \hat{x}}$

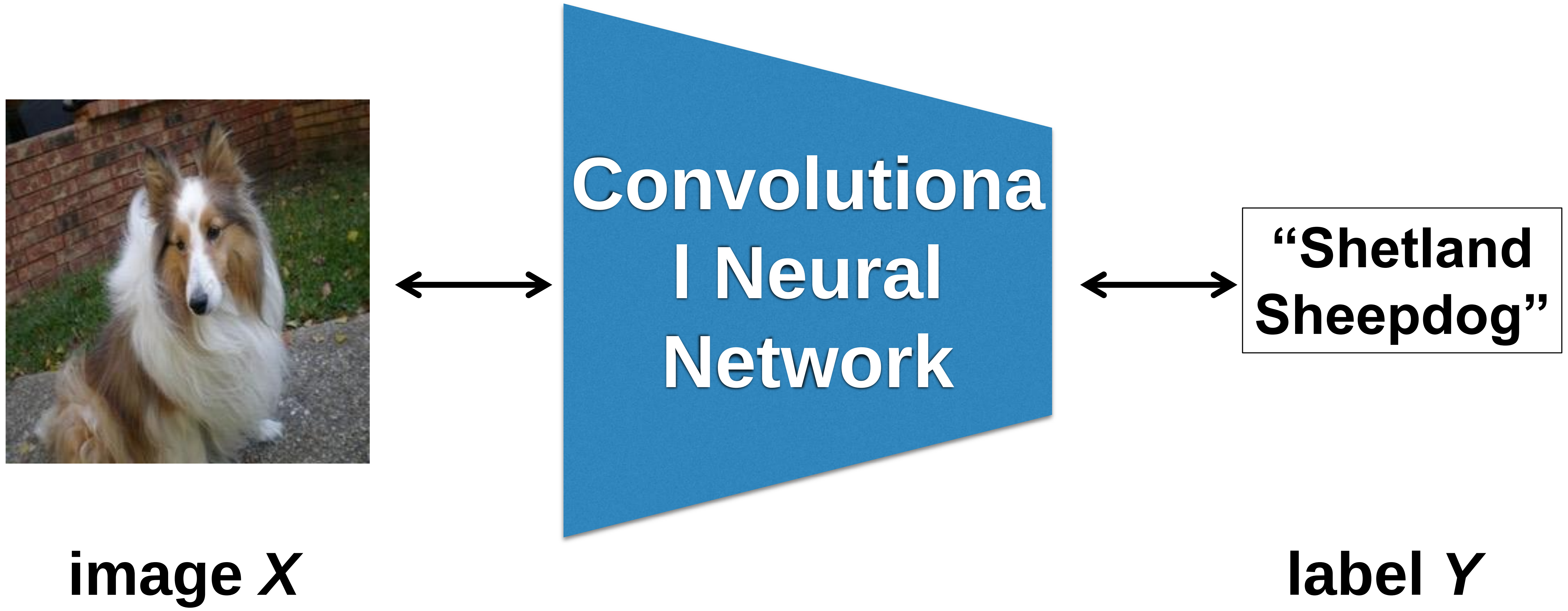
Test Julesz' Conjecture



Test Julesz’ Conjecture



ImageNet Recognition is just Texture Recognition



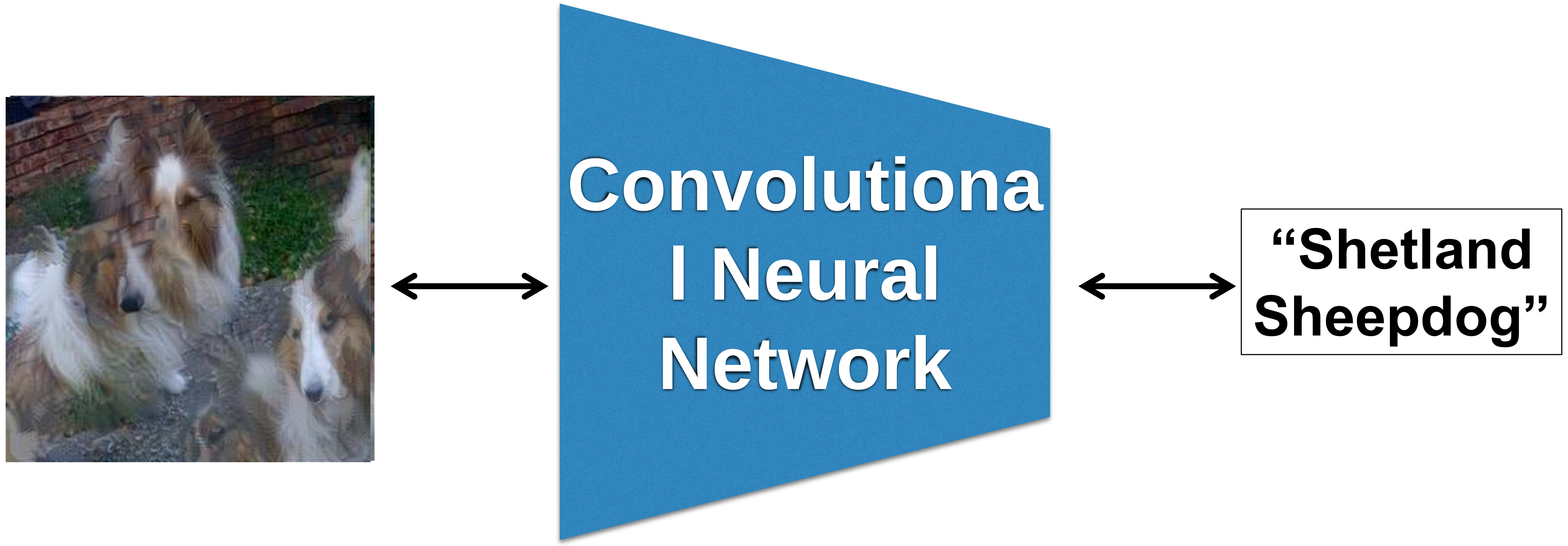
Convolutional Neural Network

“Shetland Sheepdog”

image *X*

label *Y*

ImageNet Recognition
is just Texture Recognition



Convolutiona
l Neural
Network

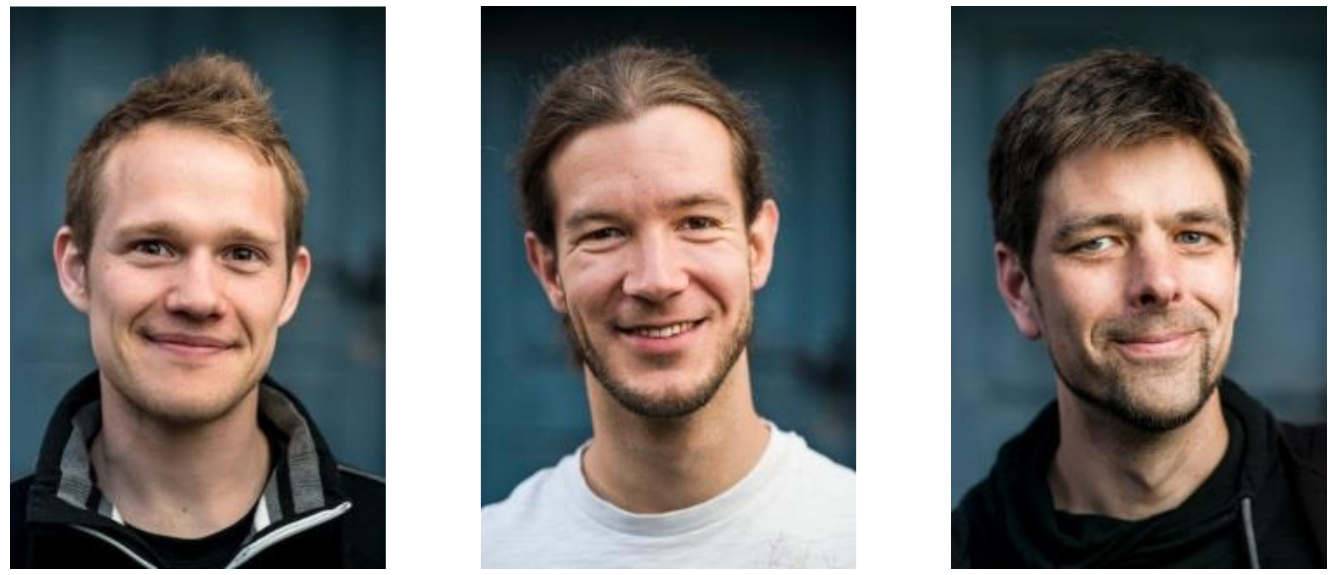
“Shetland
Sheepdog”

image *X*

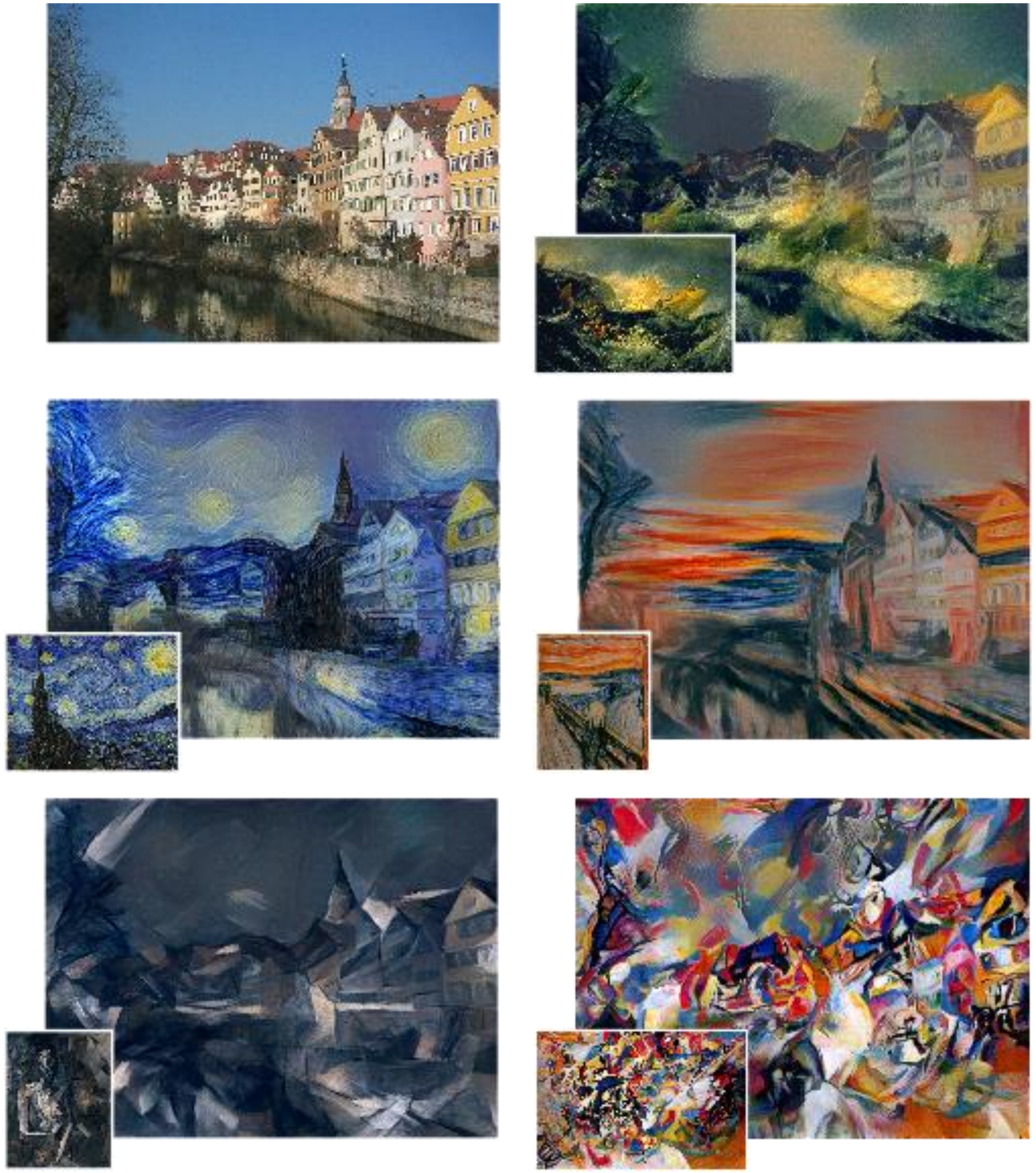
label *Y*

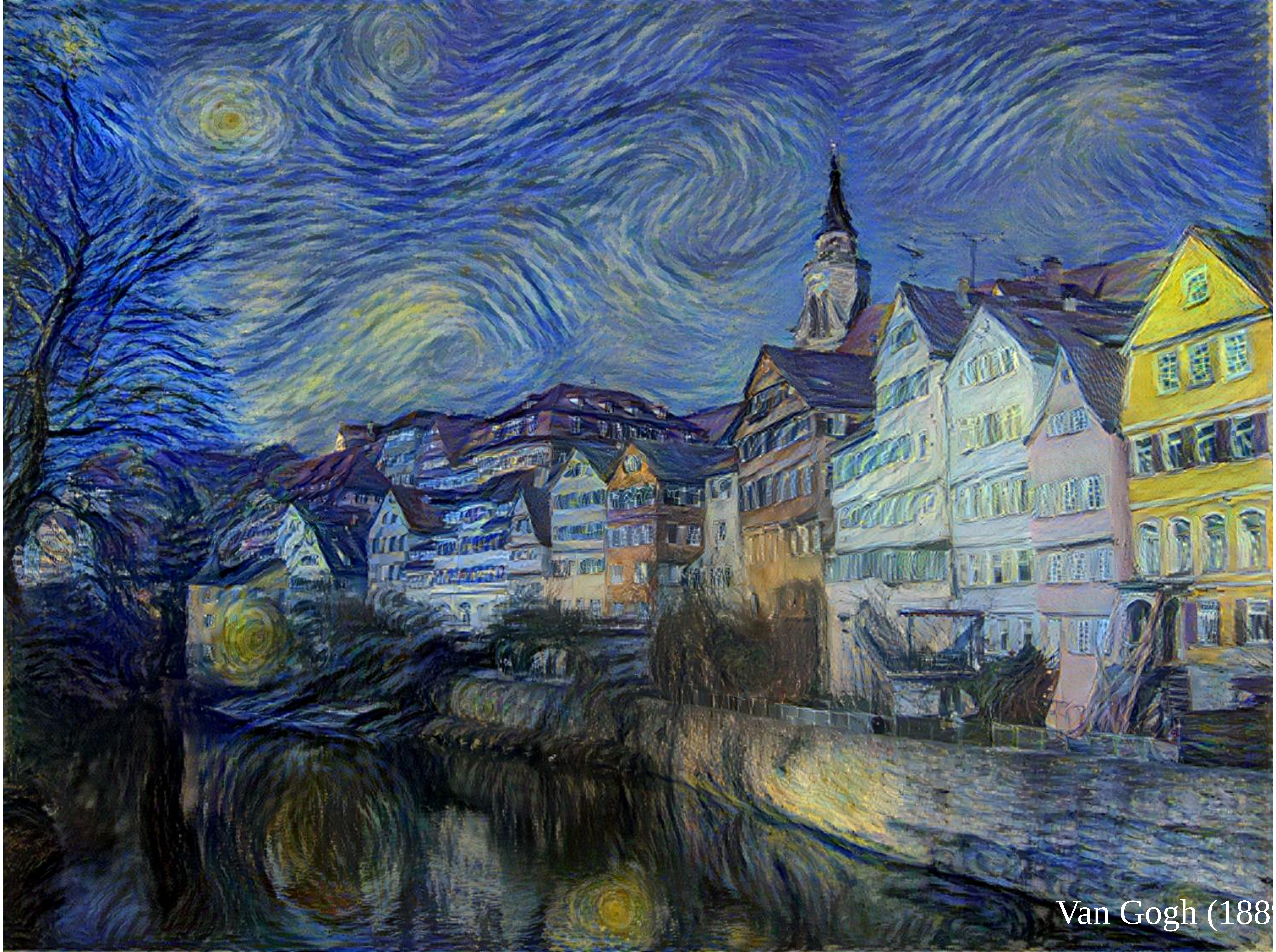
Gatys et al, 2017

A Neural Algorithm of Artistic Style

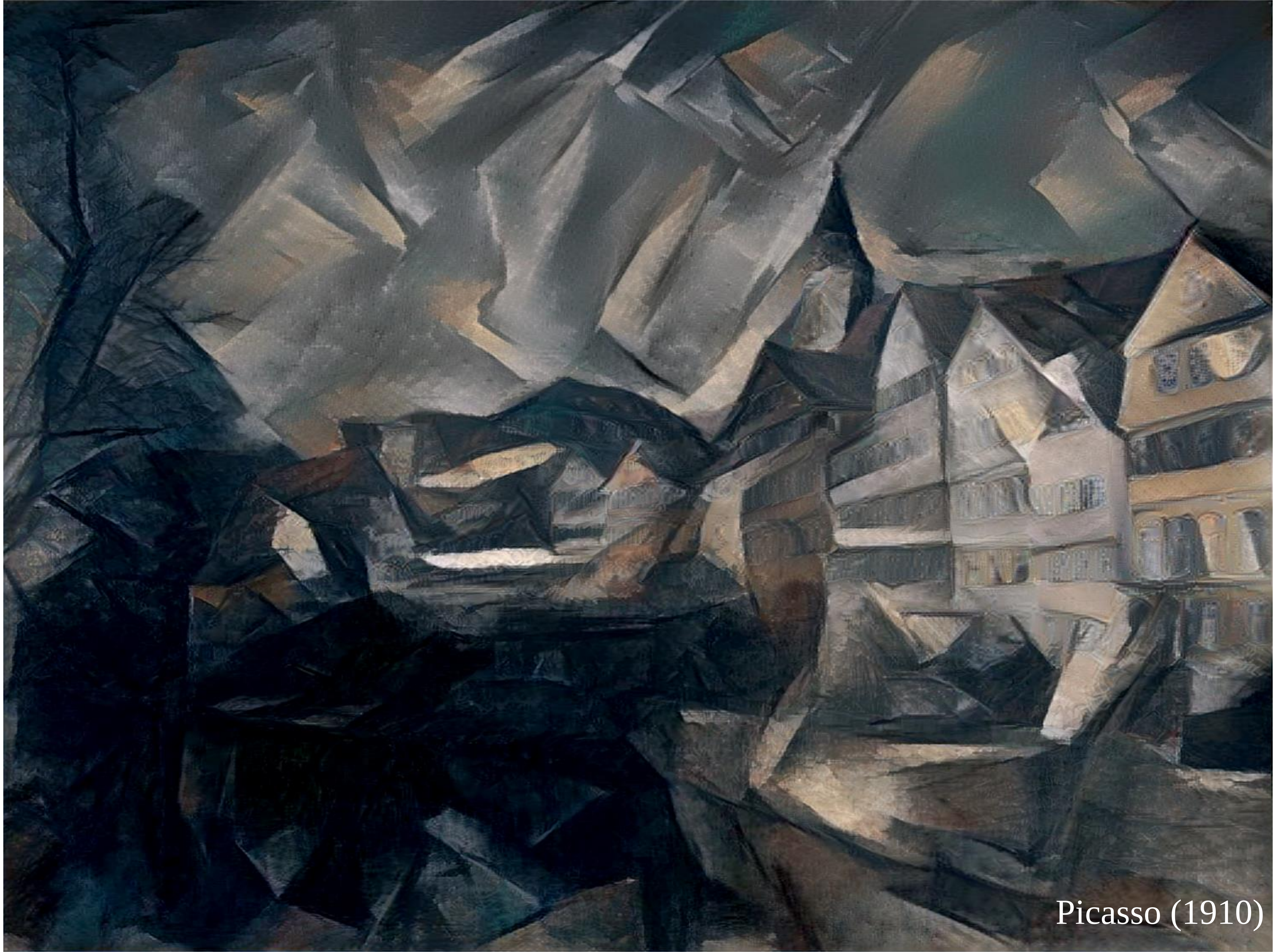


Gatys, Ecker, Bethge (arXiv 2015)

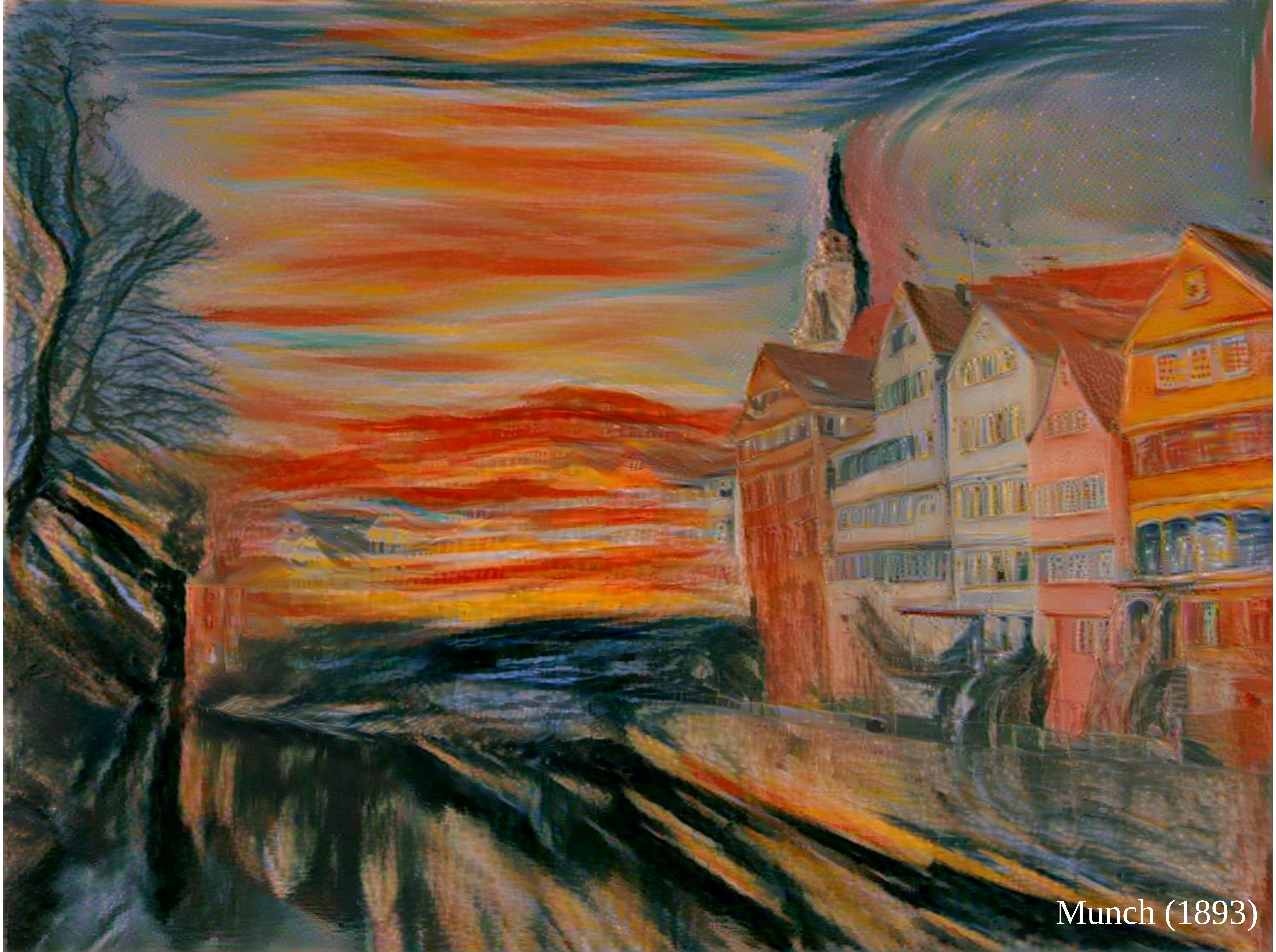




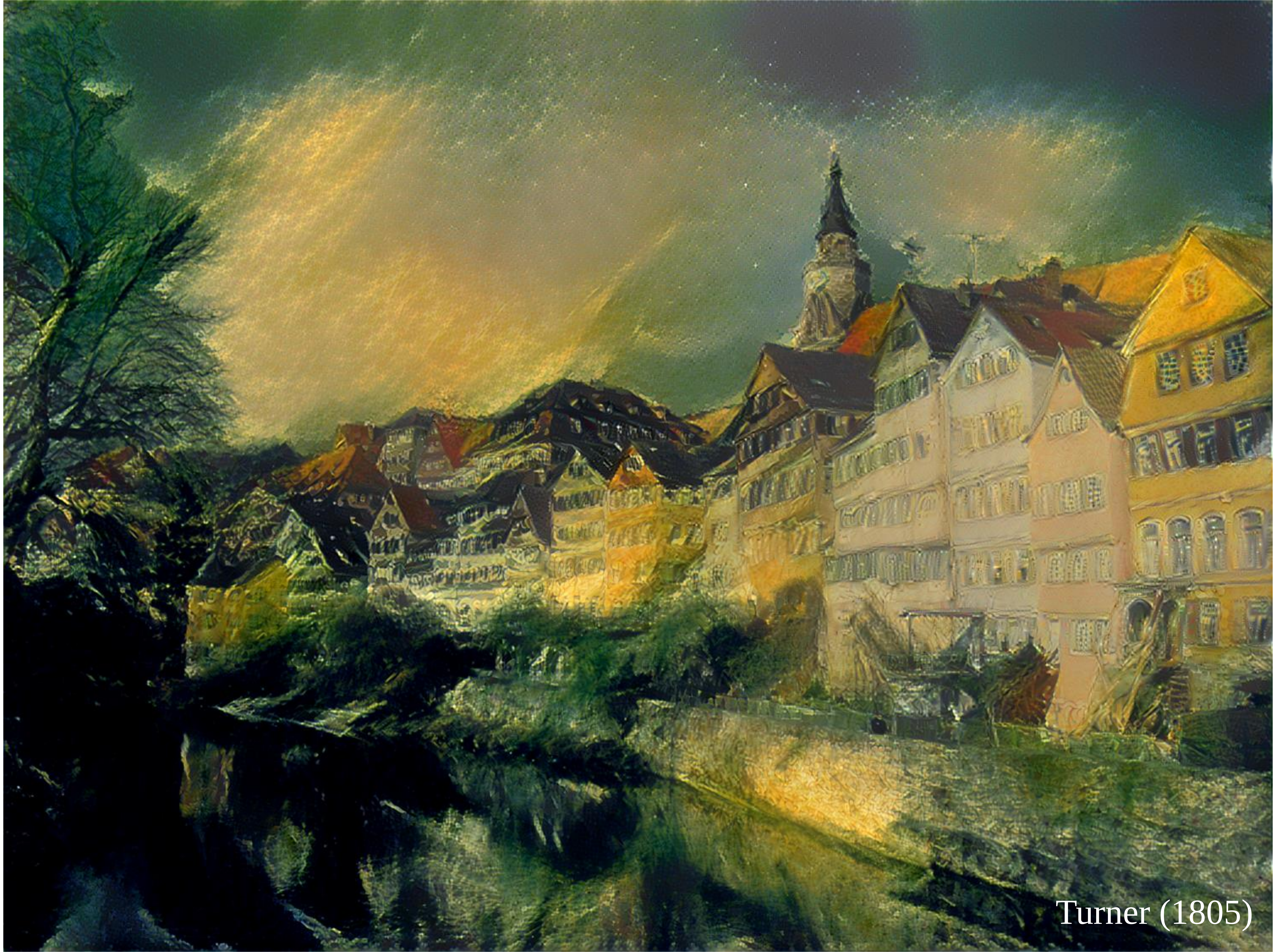
Van Gogh (188



Picasso (1910)

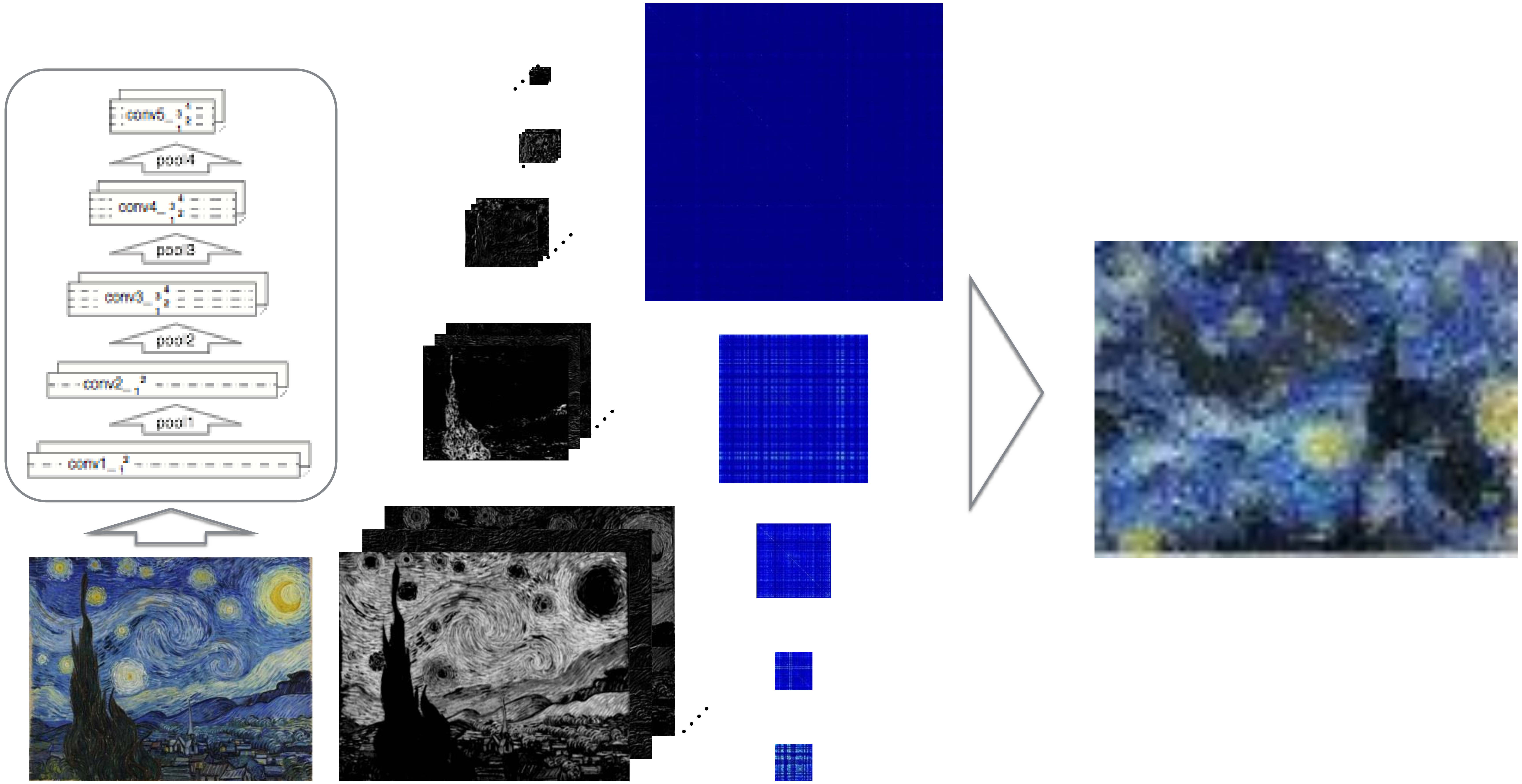


Munch (1893)


Turner (1805)

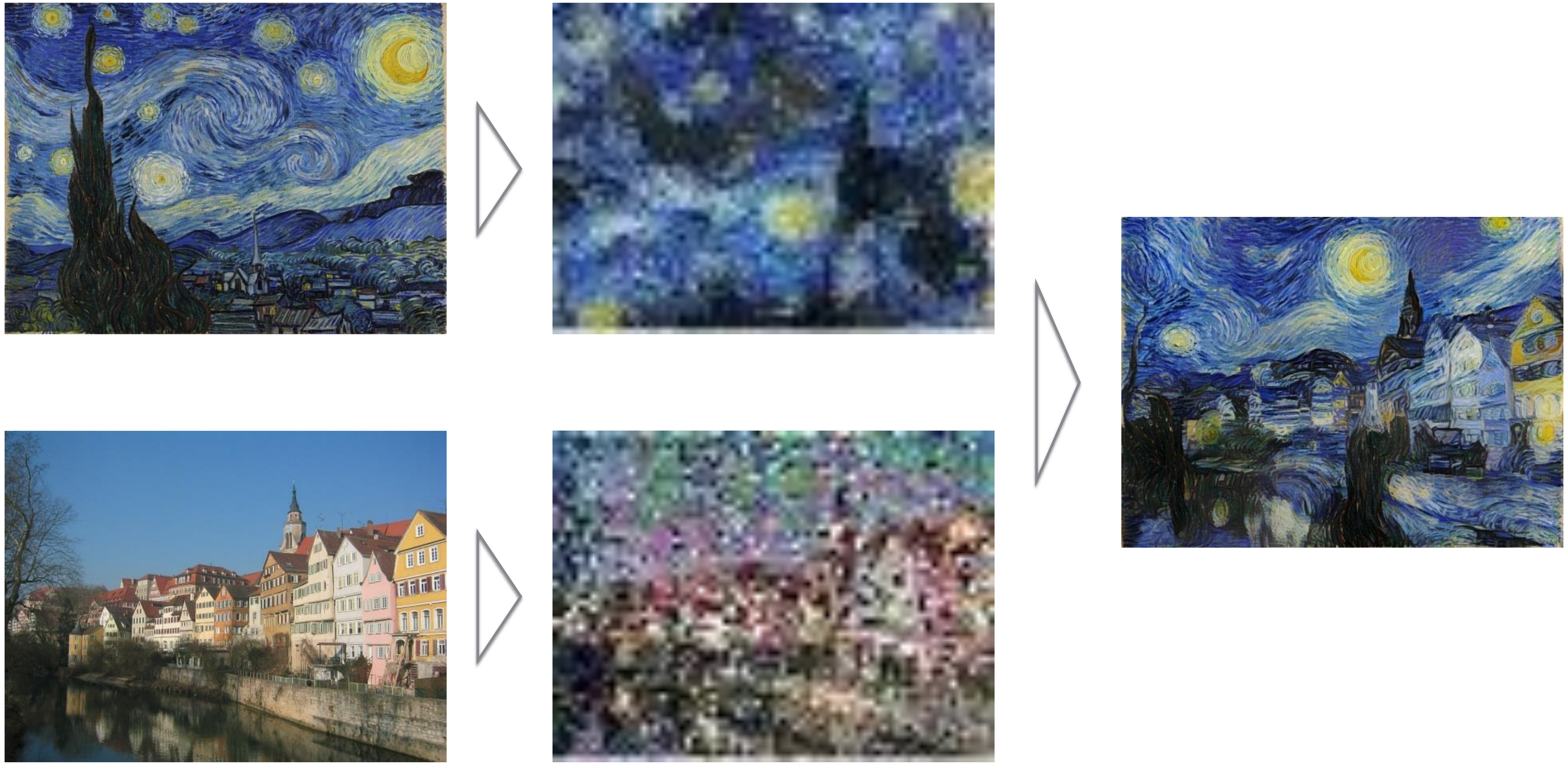
Kandinsky (191

CNN - Texture Synthesis

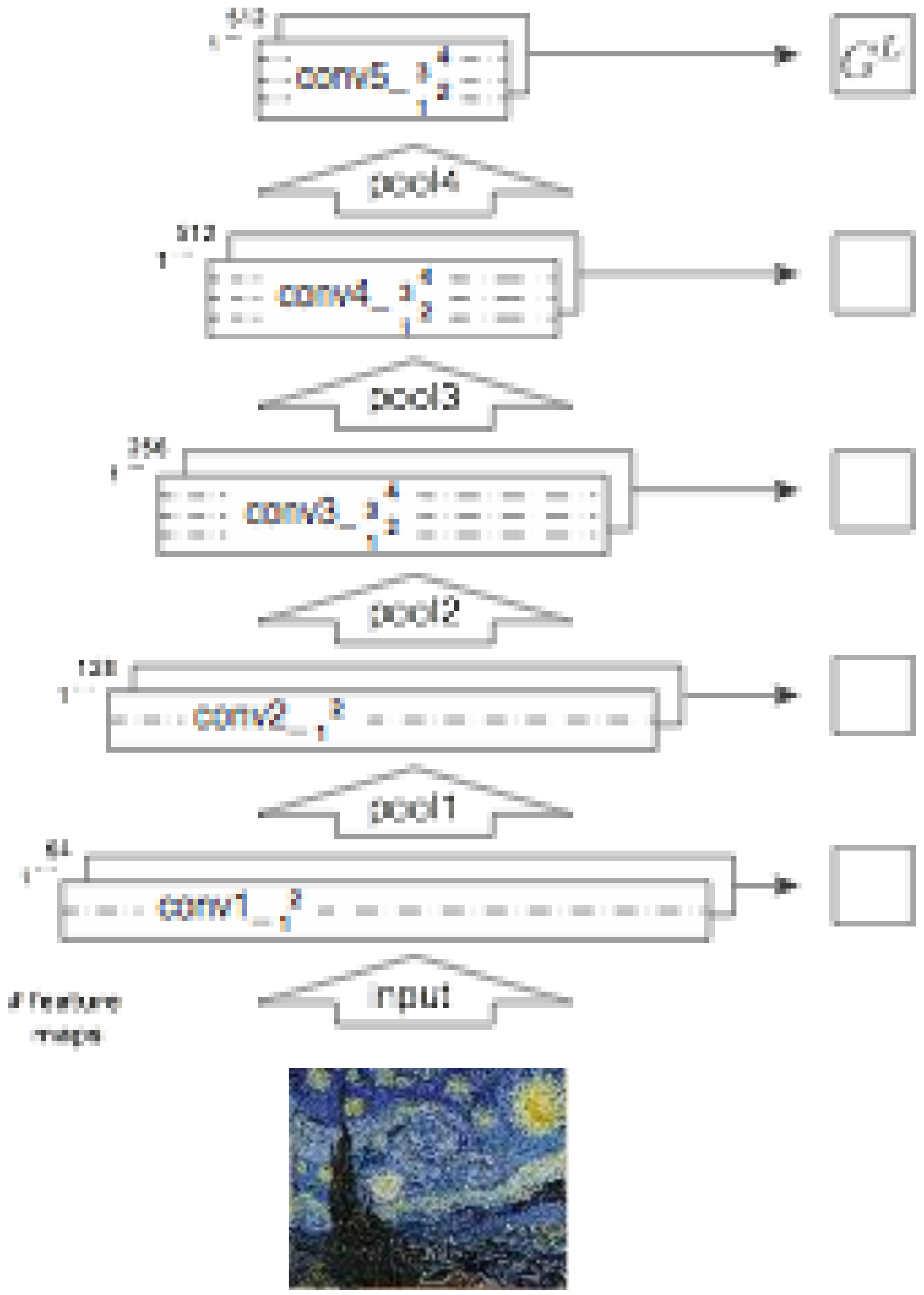


Gatys et al. (NIPS 2015)

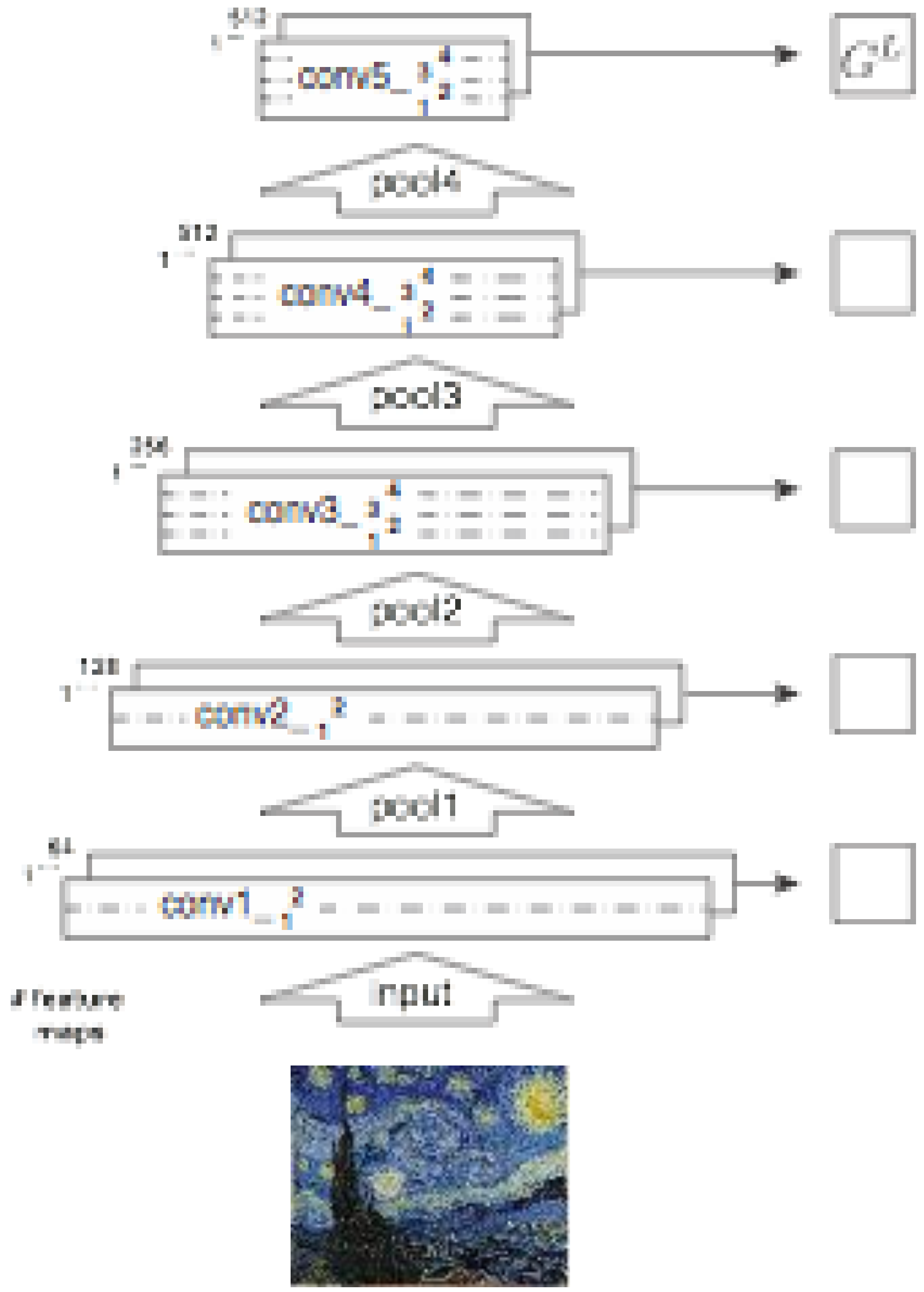
Artistic Style Transfer

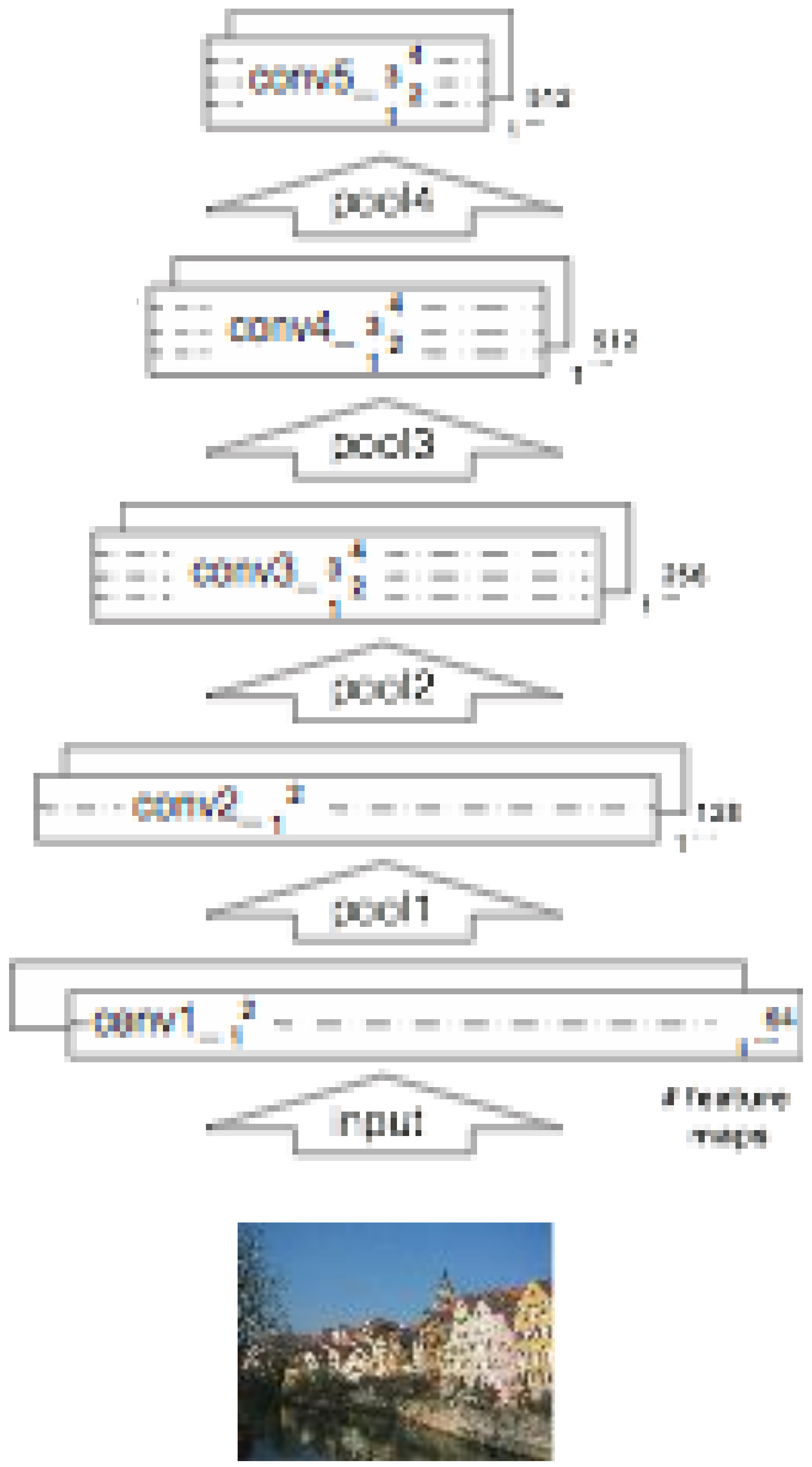


Artistic Style Transfer

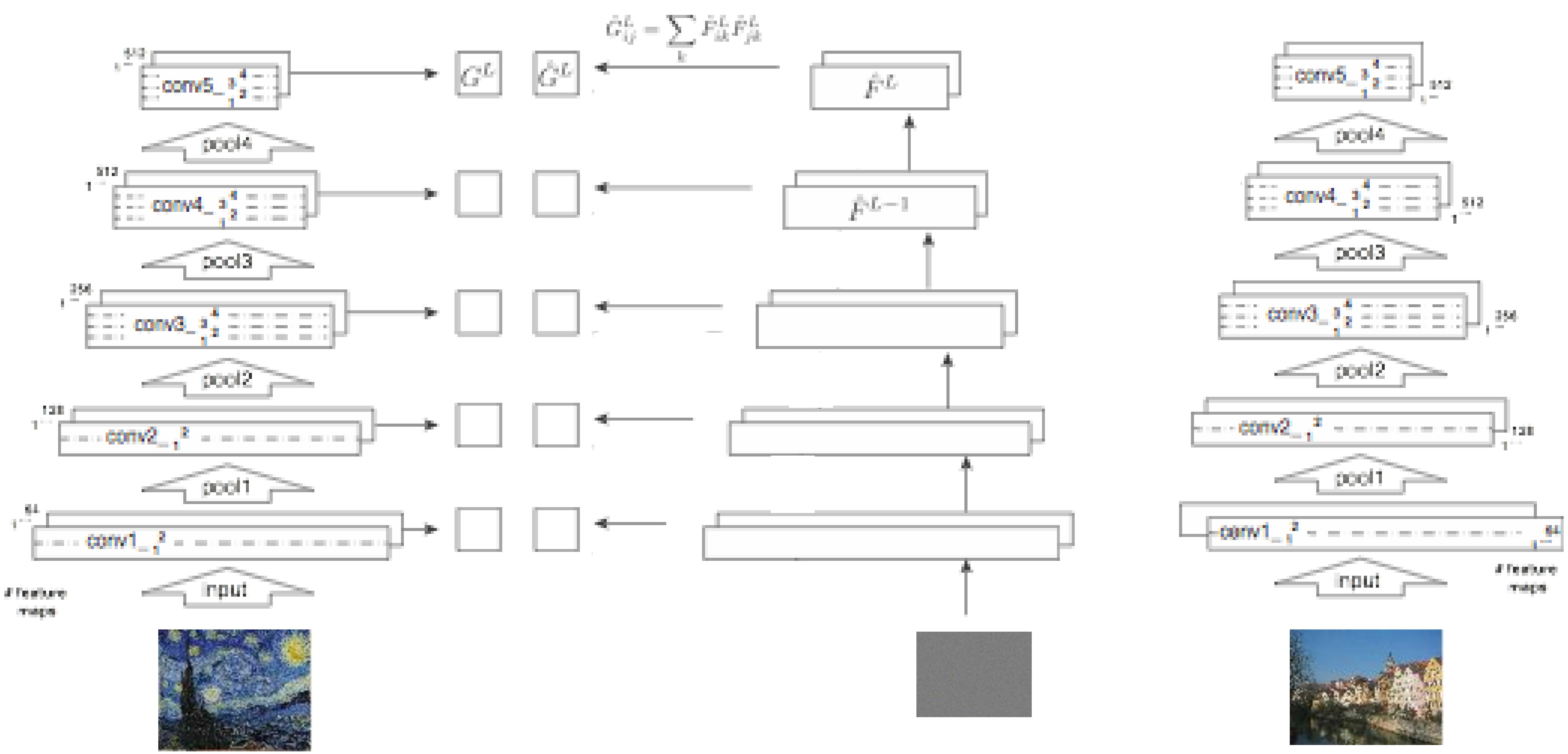


Artistic Style Transfer

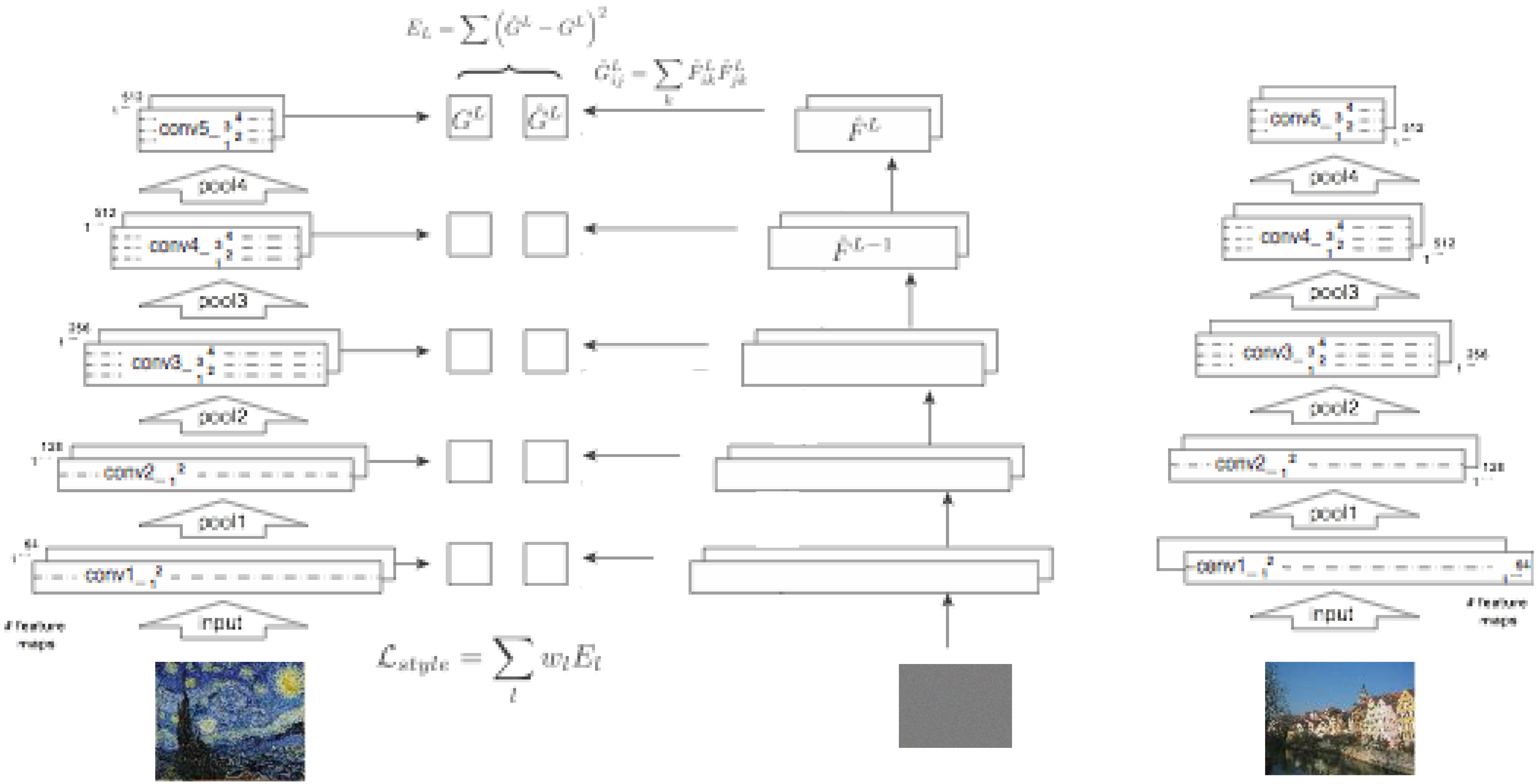




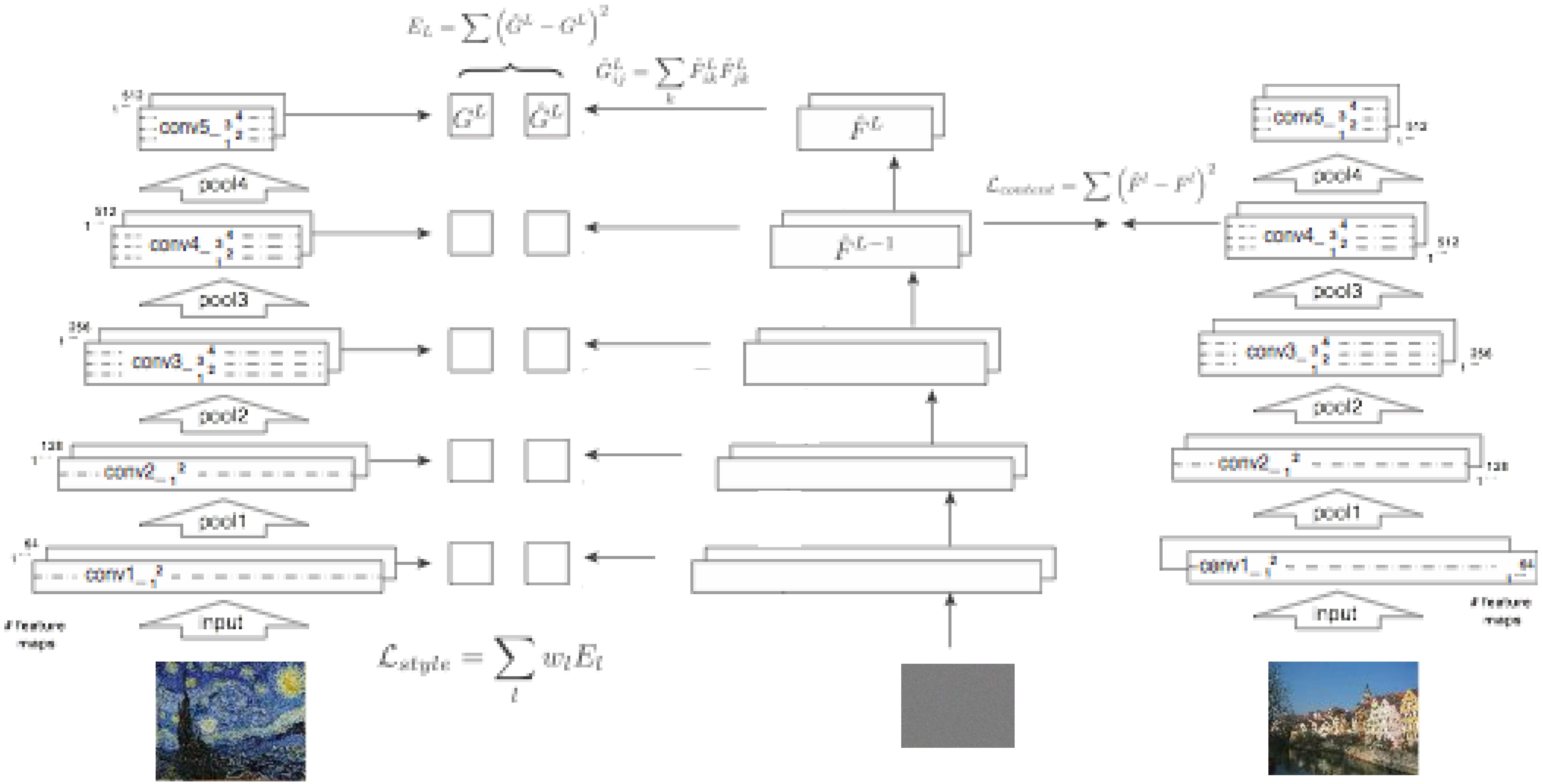
Artistic Style Transfer



Artistic Style Transfer



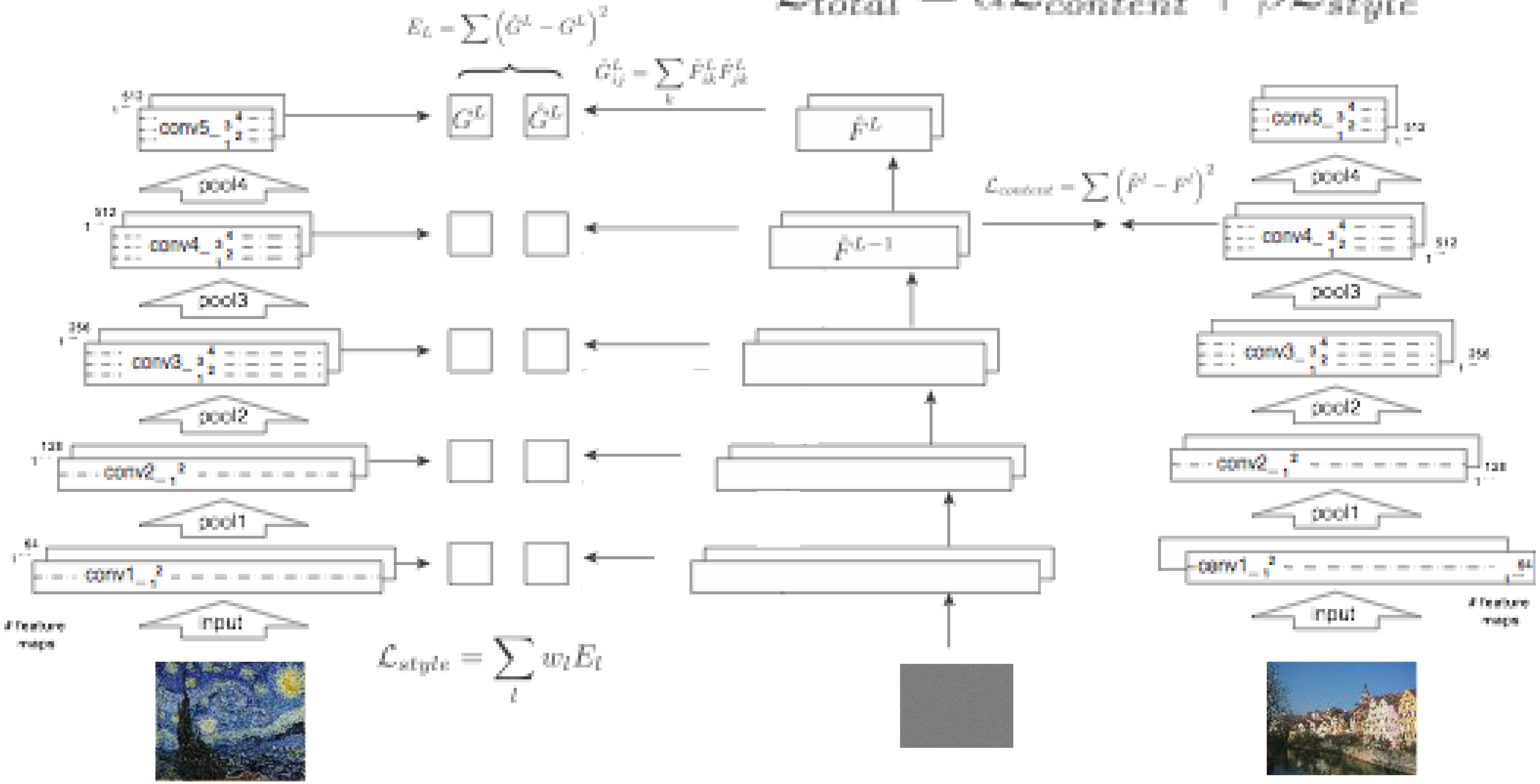
Artistic Style Transfer



$E_L = \sum \left(\hat{G}^L - G^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik} \hat{F}^L_{jk}$
G^L
$\hat{G}^L$
$\hat{F}^L$
$\hat{F}^{L-1}$
$\mathcal{L}_{content} = \sum \left(\hat{F}^l - F^l\right)^2$
$\mathcal{L}_{style} = \sum_l w_l E_l$
conv5_1 2 3 4
pool4
conv4_1 2 3 4
pool3
conv3_1 2 3 4
pool2
conv2_1 2
pool1
conv1_1 2
input
512
256
128
64
# feature maps

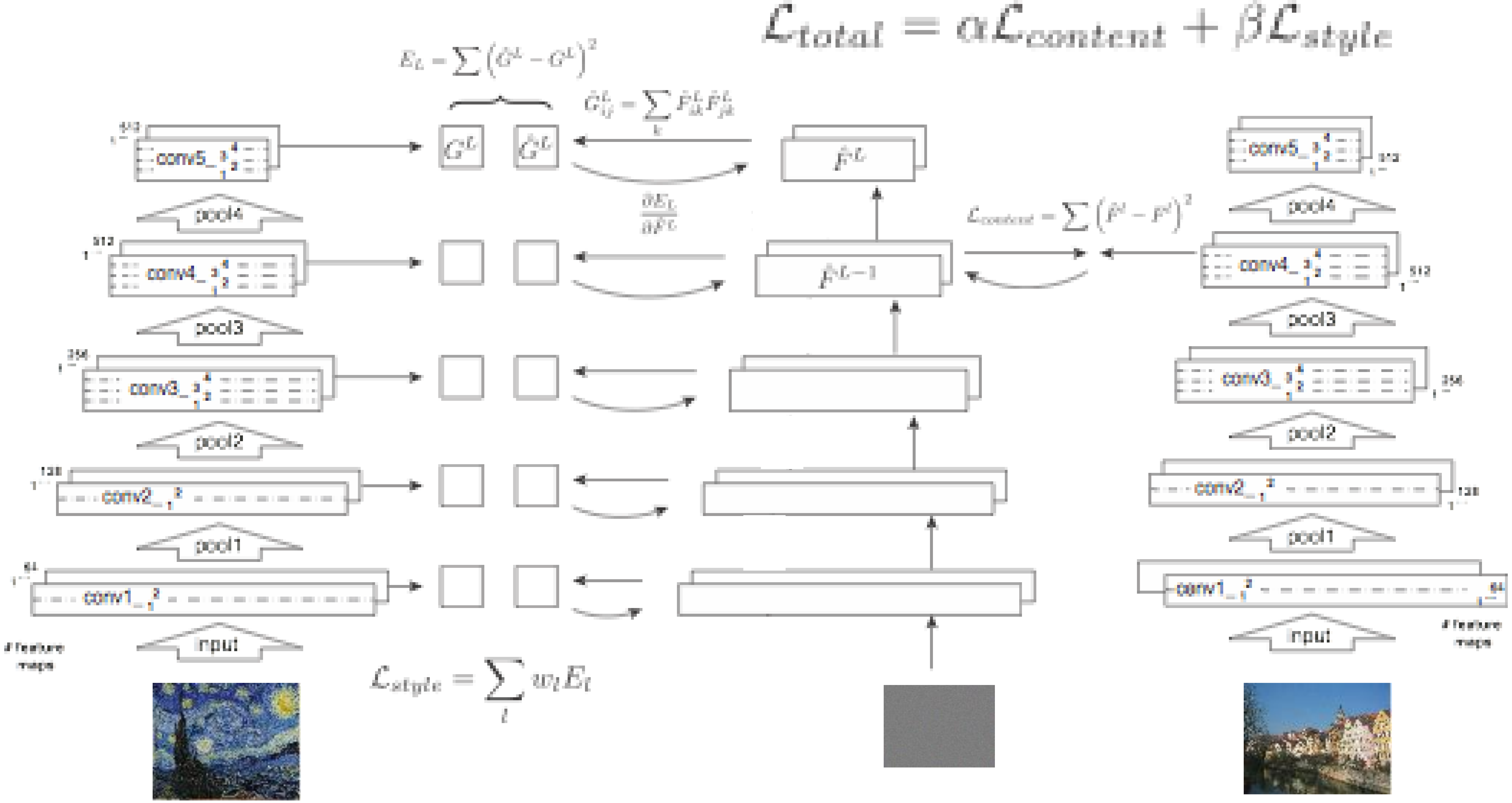
Artistic Style Transfer

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$$



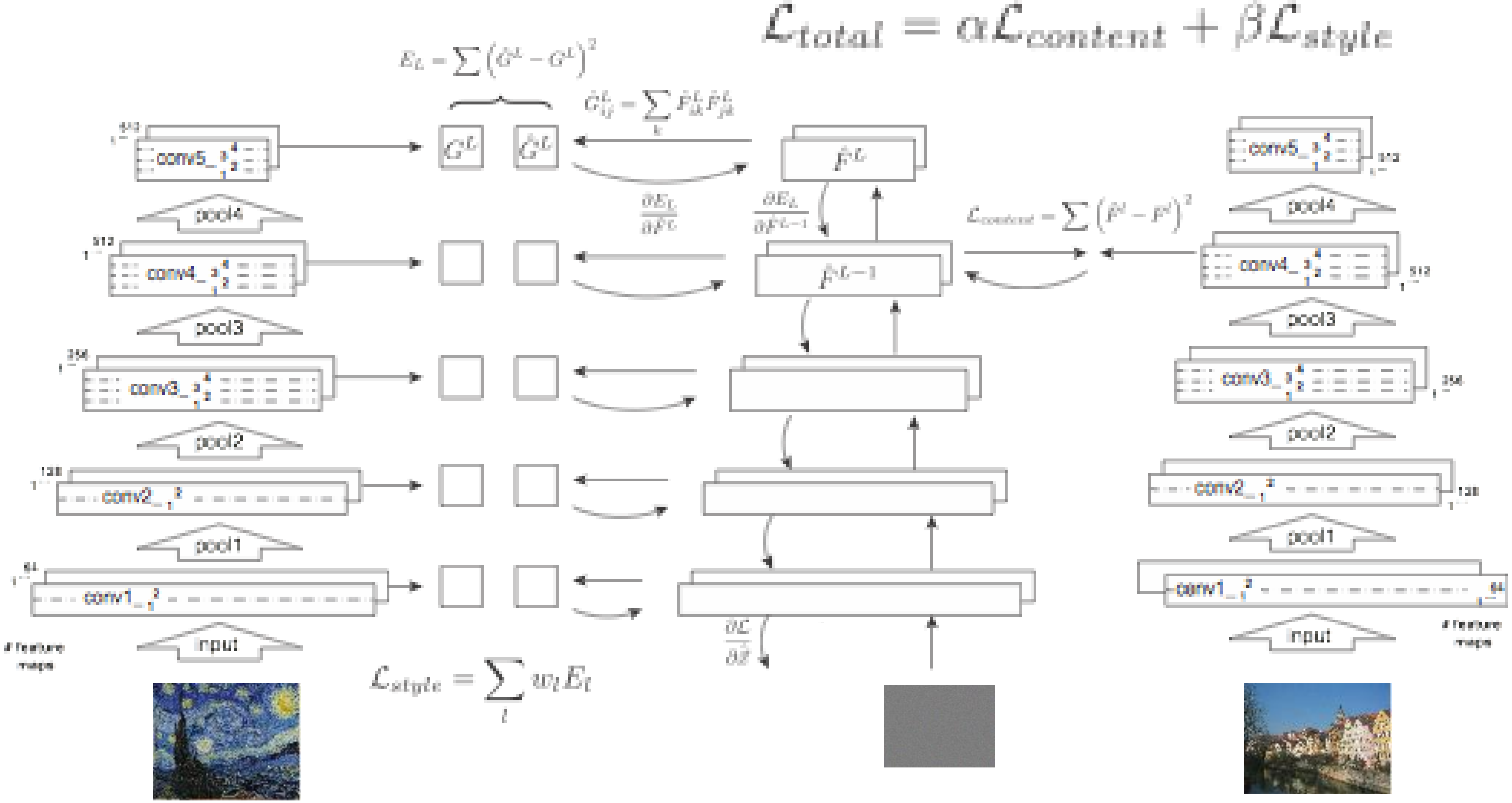
$E_L = \sum \left(\hat{G}^L - G^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik} \hat{F}^L_{jk}$
G^L
$\hat{G}^L$
$\hat{F}^L$
$\hat{F}^{L-1}$
$\mathcal{L}_{content} = \sum \left(\hat{F}^l - F^l\right)^2$
$\mathcal{L}_{style} = \sum_l w_l E_l$
conv5_
pool4
conv4_
pool3
conv3_
pool2
conv2_
pool1
conv1_
input
512
256
128
64
# feature maps

Artistic Style Transfer



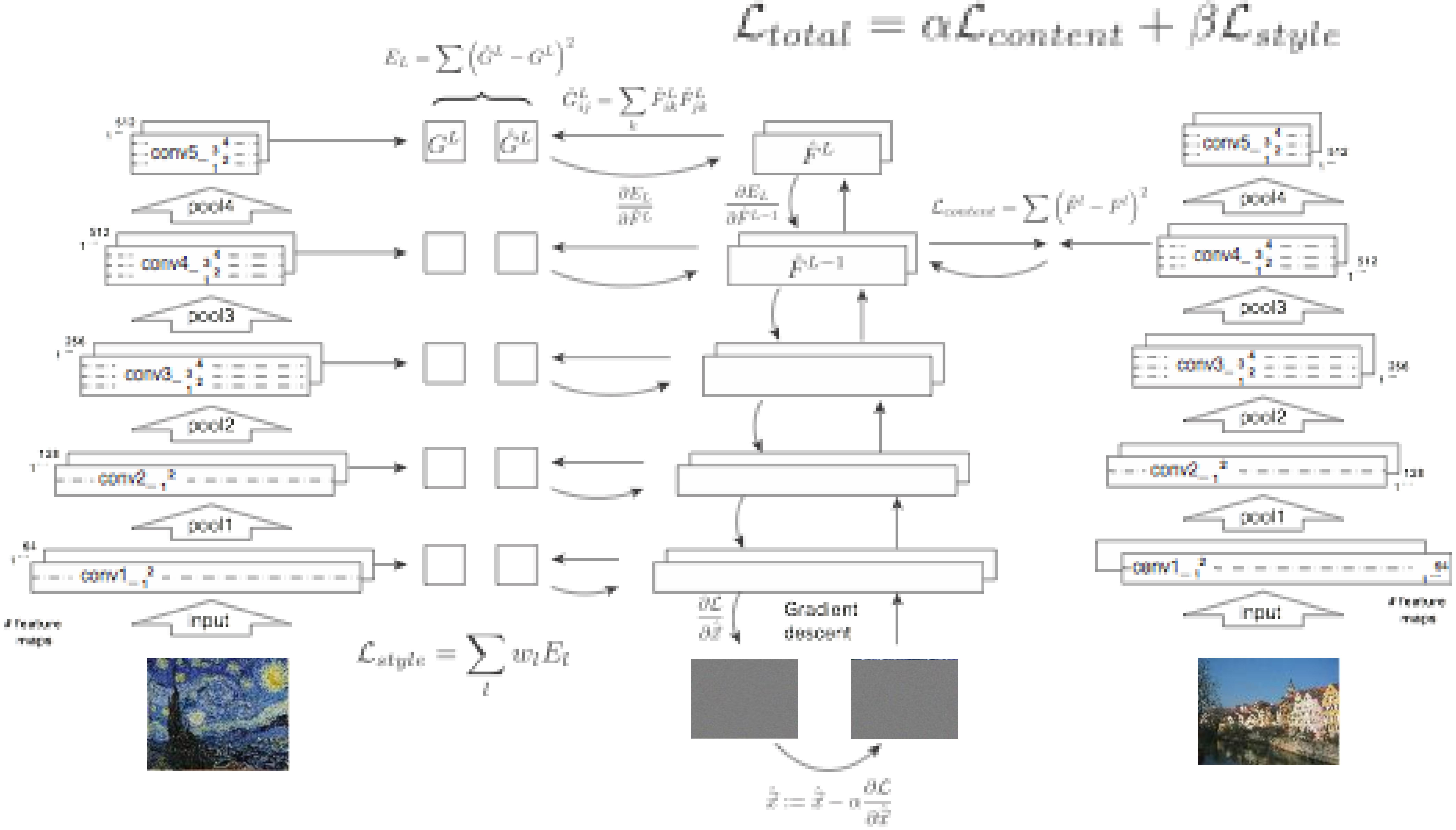
$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$
$E_L = \sum (\hat{G}^L - G^L)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik} \hat{F}^L_{jk}$
G^L
$\hat{G}^L$
$\frac{\partial E_L}{\partial \hat{F}^L}$
$\hat{F}^L$
$\hat{F}^{L-1}$
$\mathcal{L}_{content} = \sum (\hat{F}^l - F^l)^2$
$\mathcal{L}_{style} = \sum_l w_l E_l$
conv5_1 2 3 4
conv4_1 2 3 4
conv3_1 2 3 4
conv2_1 2
conv1_1 2
pool4
pool3
pool2
pool1
input
512
256
128
64
1
# feature maps

Artistic Style Transfer

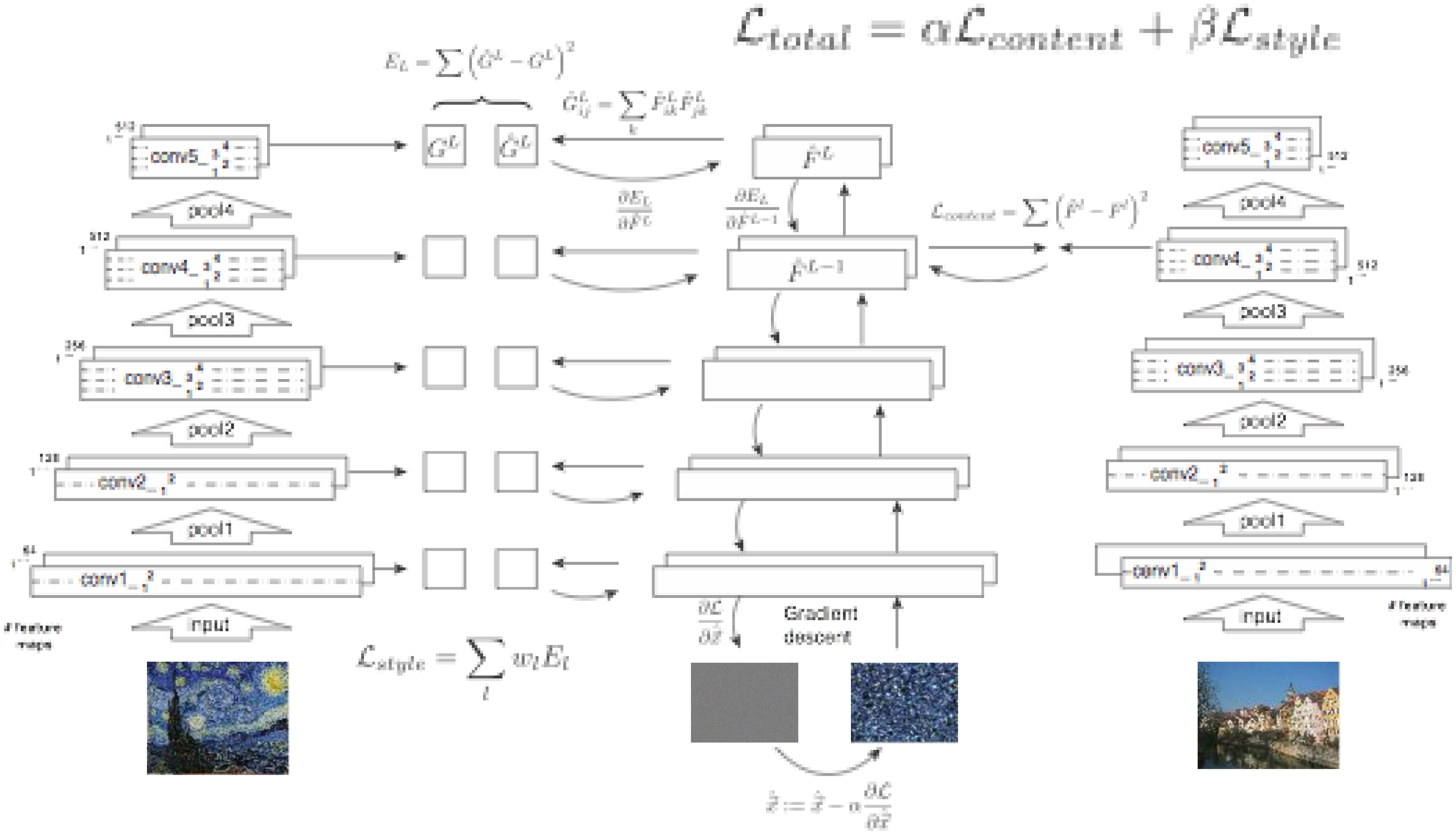


$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$
$E_L = \sum \left(\hat{G}^L - G^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik}\hat{F}^L_{jk}$
G^L
$\hat{G}^L$
$\frac{\partial E_L}{\partial \hat{F}^L}$
$\frac{\partial E_L}{\partial \hat{F}^{L-1}}$
$\hat{F}^L$
$\hat{F}^{L-1}$
$\mathcal{L}_{content} = \sum \left(\hat{F}^l - F^l\right)^2$
$\frac{\partial \mathcal{L}}{\partial \hat{x}}$
$\mathcal{L}_{style} = \sum_l w_l E_l$
conv5_1 2 3 4
pool4
conv4_1 2 3 4
pool3
conv3_1 2 3 4
pool2
conv2_1 2
pool1
conv1_1 2
input
512
256
128
64
1
# feature maps

Artistic Style Transfer

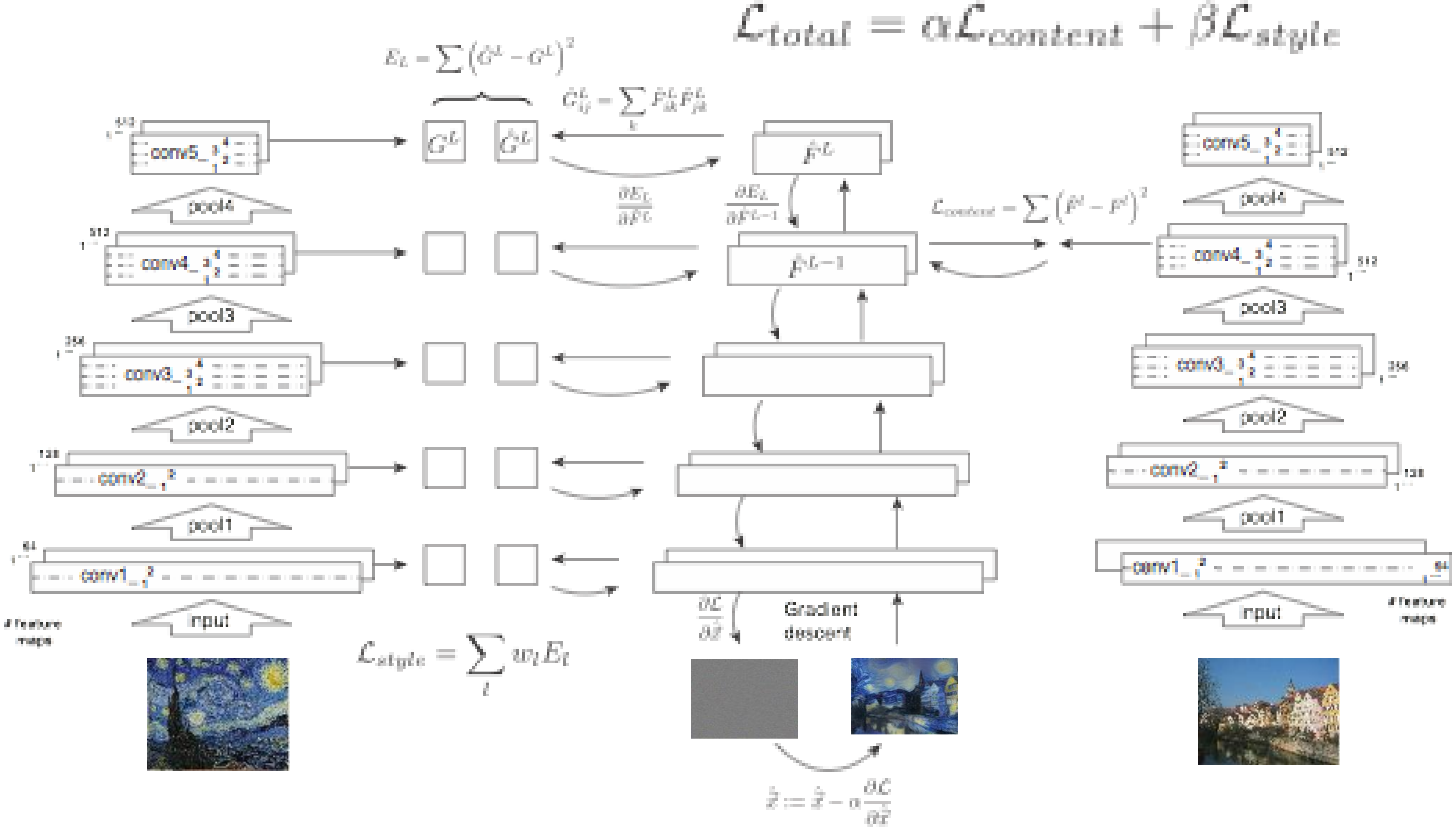

$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$
$E_L = \sum\left(\hat{G}^L - G^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik}\hat{F}^L_{jk}$
G^L
$\hat{G}^L$
$\hat{F}^L$
$\hat{F}^{L-1}$
$\frac{\partial E_L}{\partial \hat{F}^L}$
$\frac{\partial E_L}{\partial \hat{F}^{L-1}}$
$\mathcal{L}_{content} = \sum\left(\hat{F}^l - F^l\right)^2$
conv5_
conv4_
conv3_
conv2_
conv1_
pool4
pool3
pool2
pool1
input
512
256
128
64
# feature maps
$\mathcal{L}_{style} = \sum_l w_l E_l$
$\frac{\partial \mathcal{L}}{\partial \hat{x}}$
Gradient descent
$\hat{x} := \hat{x} - \alpha\frac{\partial \mathcal{L}}{\partial \hat{x}}$

Artistic Style Transfer



$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$
$E_L = \sum\left(\hat{G}^L - G^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik}\hat{F}^L_{jk}$
G^L
$\hat{G}^L$
$\hat{F}^L$
$\hat{F}^{L-1}$
$\frac{\partial E_L}{\partial \hat{F}^L}$
$\frac{\partial E_L}{\partial \hat{F}^{L-1}}$
$\mathcal{L}_{content} = \sum\left(\hat{F}^l - F^l\right)^2$
conv5_1 2 3 4
pool4
conv4_1 2 3 4
pool3
conv3_1 2 3 4
pool2
conv2_1 2
pool1
conv1_1 2
input
512
256
128
64
1
# feature maps
$\mathcal{L}_{style} = \sum_l w_l E_l$
$\frac{\partial \mathcal{L}}{\partial \hat{x}}$
Gradient descent
$\hat{x} := \hat{x} - \alpha\frac{\partial \mathcal{L}}{\partial \hat{x}}$

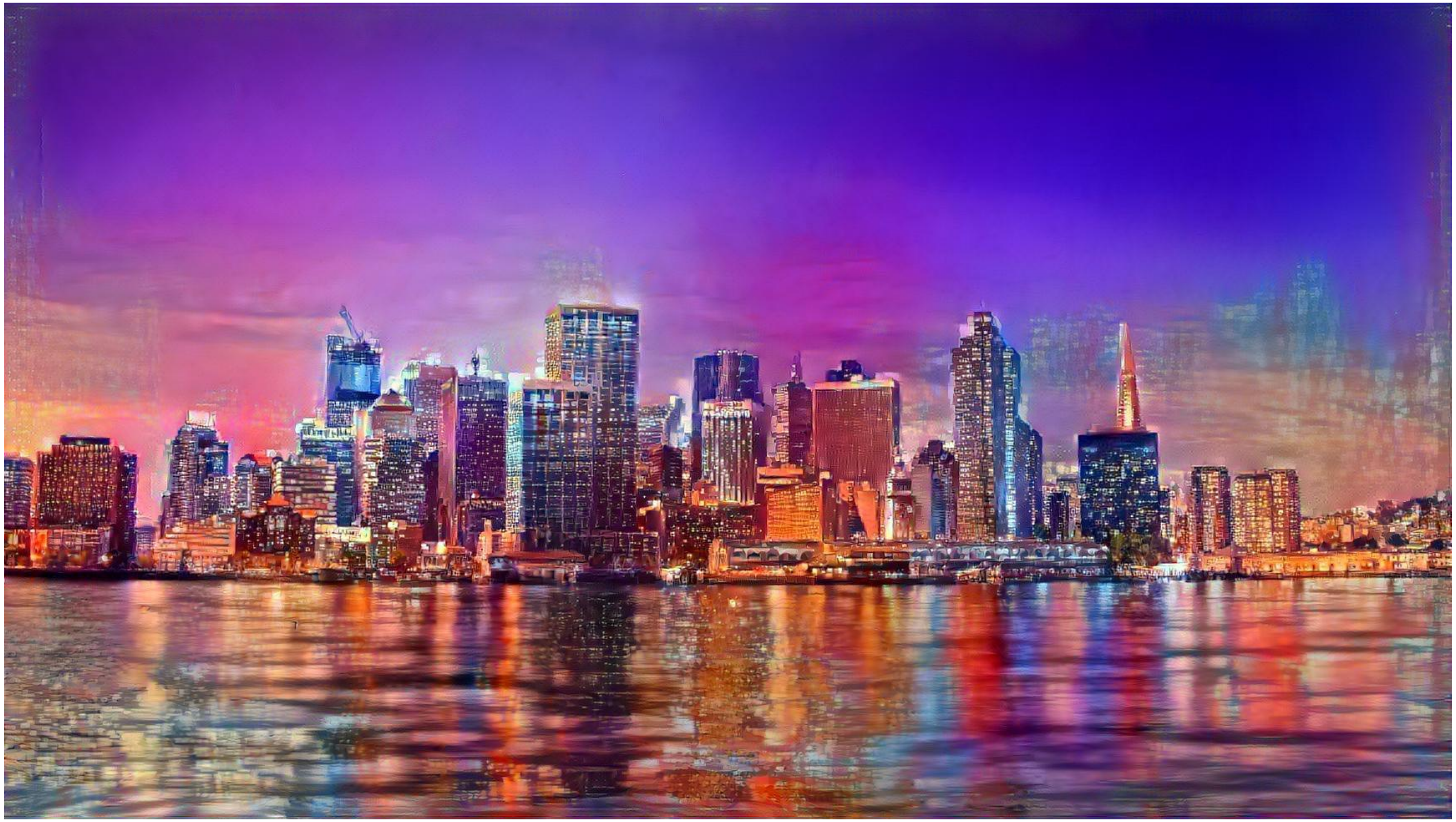
Artistic Style Transfer

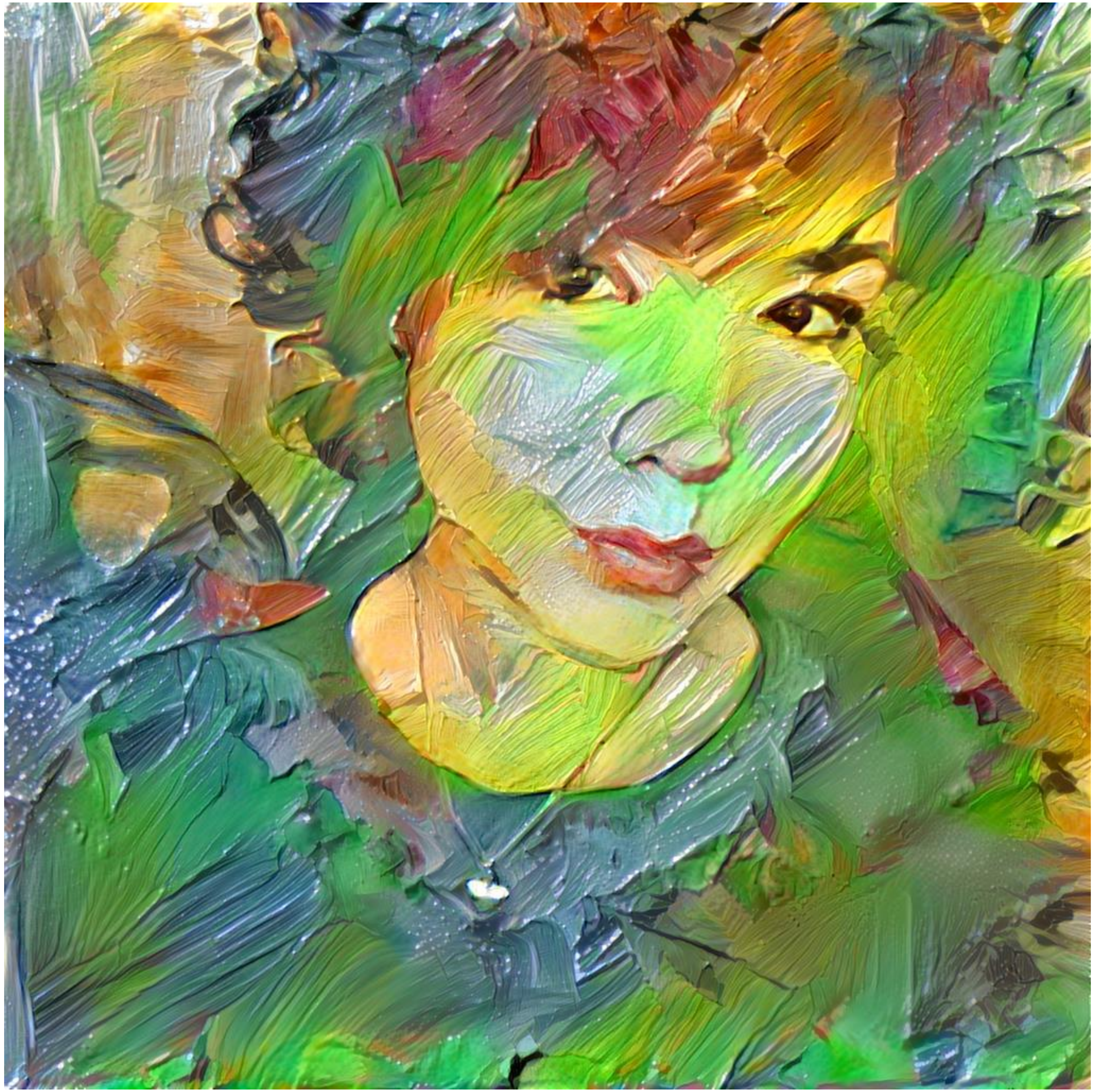


$\mathcal{L}_{total} = \alpha\mathcal{L}_{content} + \beta\mathcal{L}_{style}$
$E_L = \sum\left(\hat{G}^L - G^L\right)^2$
$\hat{G}^L_{ij} = \sum_k \hat{F}^L_{ik}\hat{F}^L_{jk}$
G^L
$\hat{G}^L$
$\hat{F}^L$
$\frac{\partial E_L}{\partial \hat{F}^L}$
$\frac{\partial E_L}{\partial \hat{F}^{L-1}}$
$\hat{F}^{L-1}$
$\mathcal{L}_{content} = \sum\left(\hat{F}^l - F^l\right)^2$
conv5_1 2 3 4
pool4
conv4_1 2 3 4
pool3
conv3_1 2 3 4
pool2
conv2_1 2
pool1
conv1_1 2
input
512
256
128
64
# feature maps
$\mathcal{L}_{style} = \sum_l w_l E_l$
$\frac{\partial \mathcal{L}}{\partial \hat{x}}$
Gradient descent
$\hat{x} := \hat{x} - \alpha\frac{\partial \mathcal{L}}{\partial \hat{x}}$

General Style Transfer

General Style Transfer

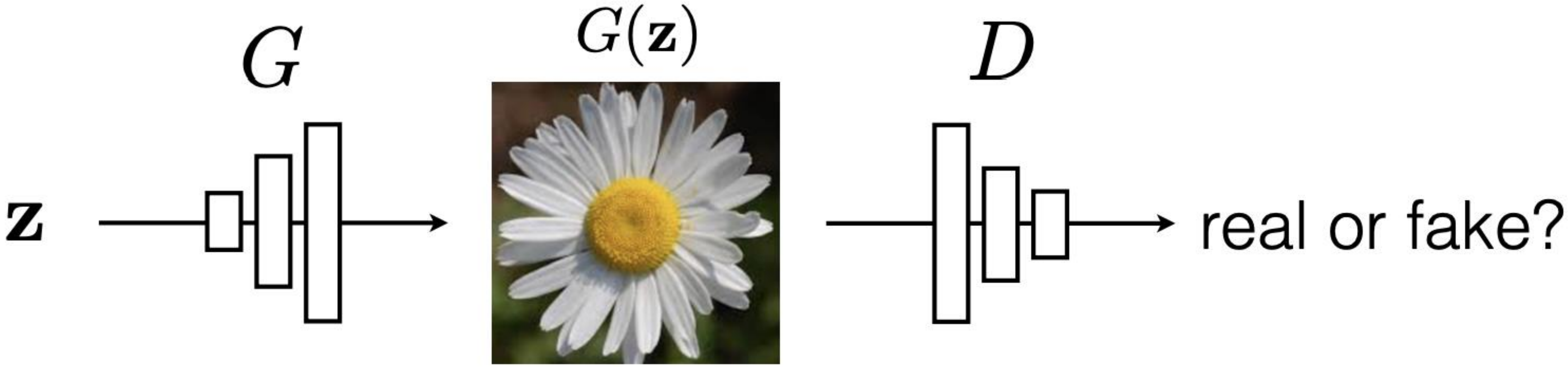




© Wayne Tilcock

GANs as generative models

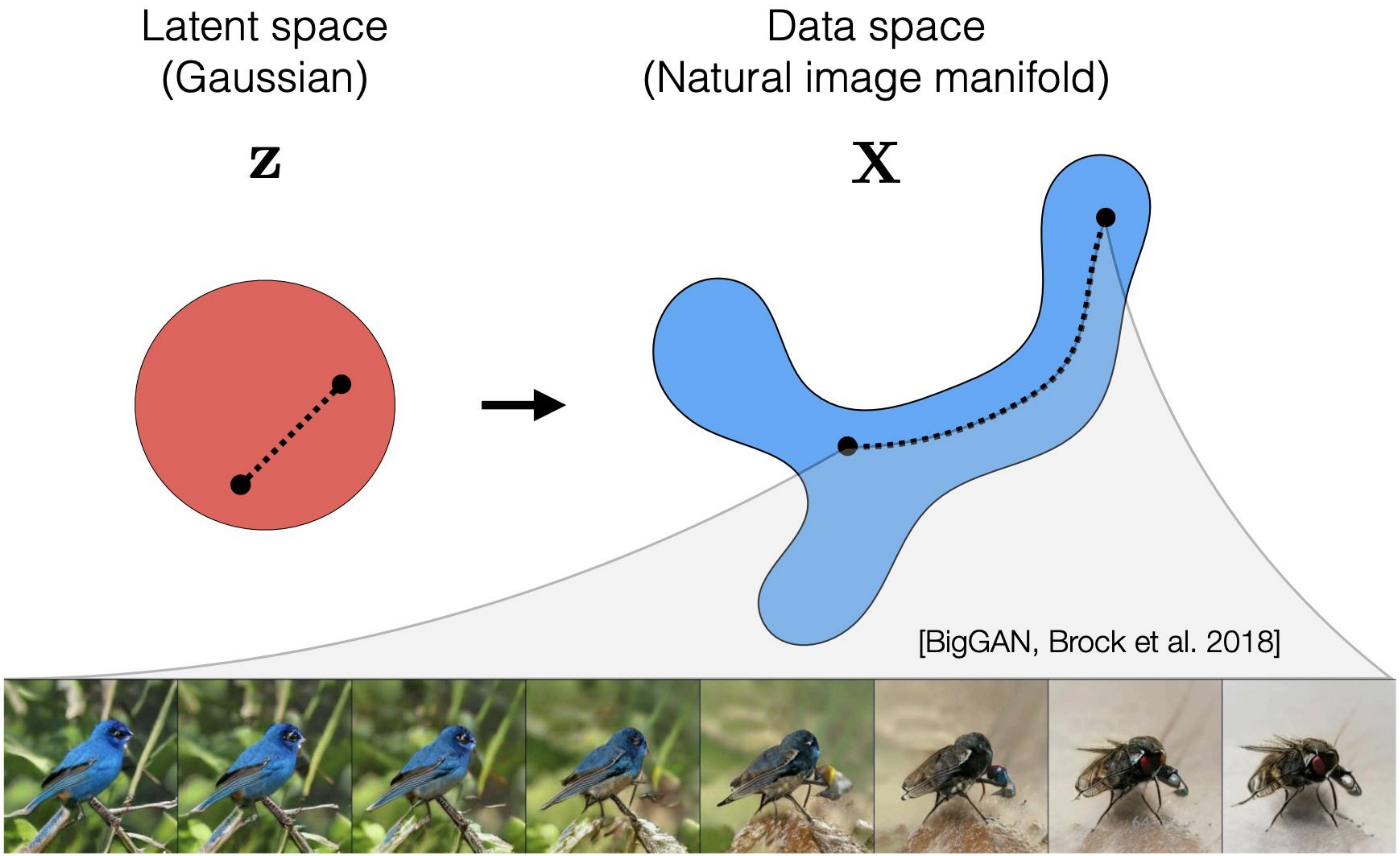
- **G** tries to synthesize fake images that ***fool*** the ***best*** **D**
- **D** tries to identify the fakes



$$\arg \boxed{\min_{G}} \boxed{\max_{D}} \; \mathbb{E}_{\mathbf{z},\mathbf{x}}[\; \log D(G(\mathbf{z})) \; + \; \log(1 - D(\mathbf{x})) \;]$$

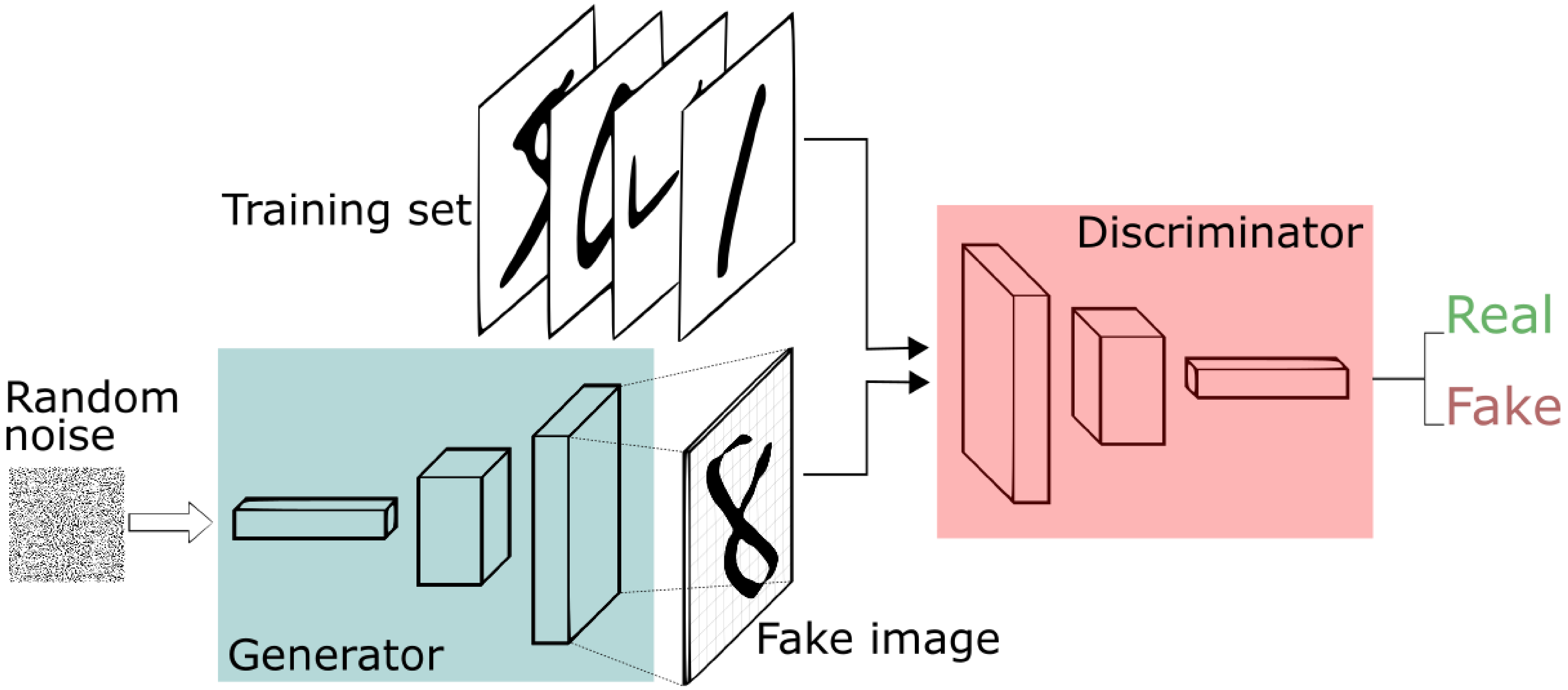
Source: Isola, Freeman, Torralba

GANs can “walk” on the manifold



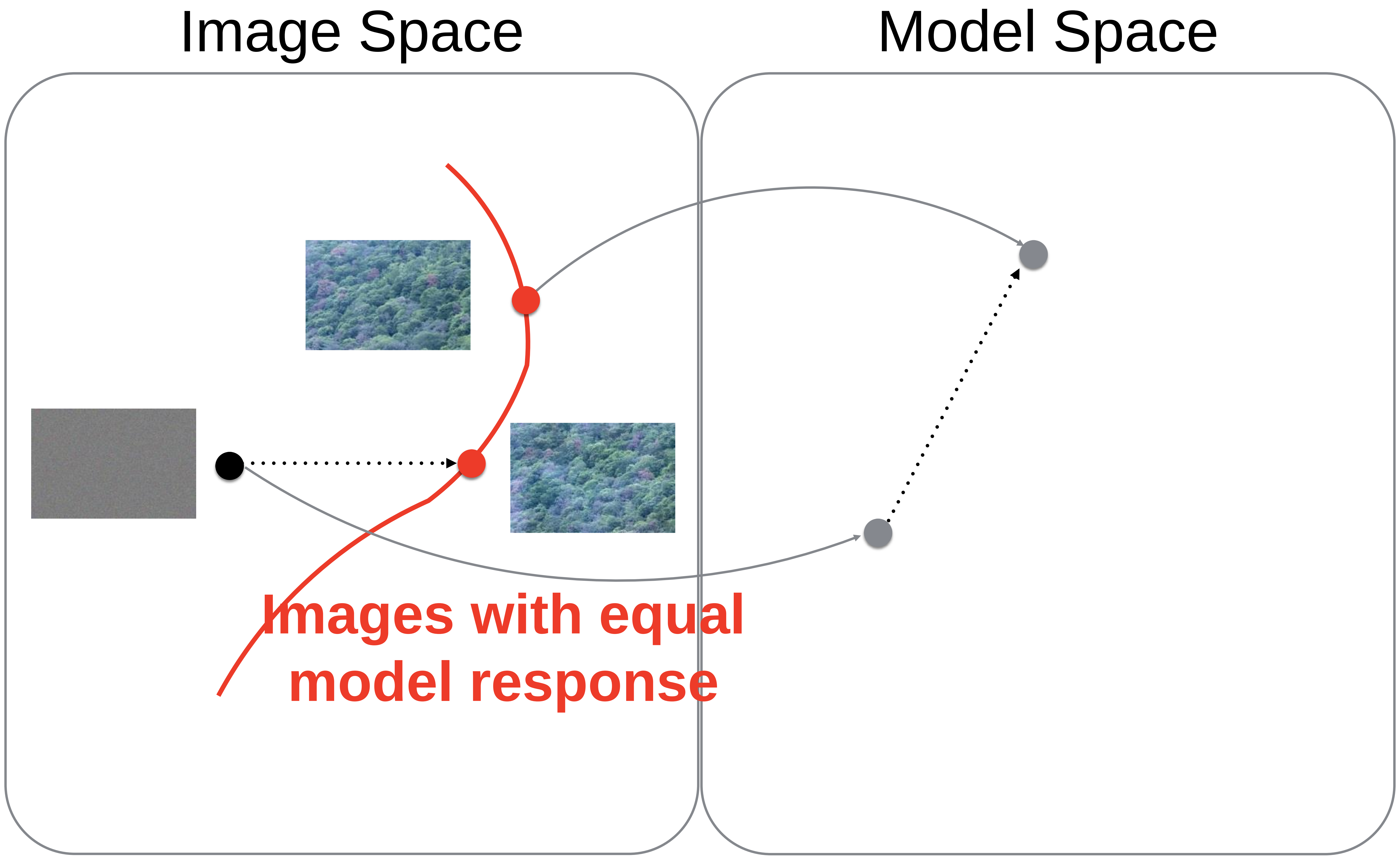
[BigGAN, Brock et al. 2018]

GANs as Texture Synthesis?



- **Conjecture:** GANs might be learning the “right” features to match for natural images

Projecting onto the “Image Manifold”



Portilla & Simoncelli (2000)

diffusion

Generate 100 images

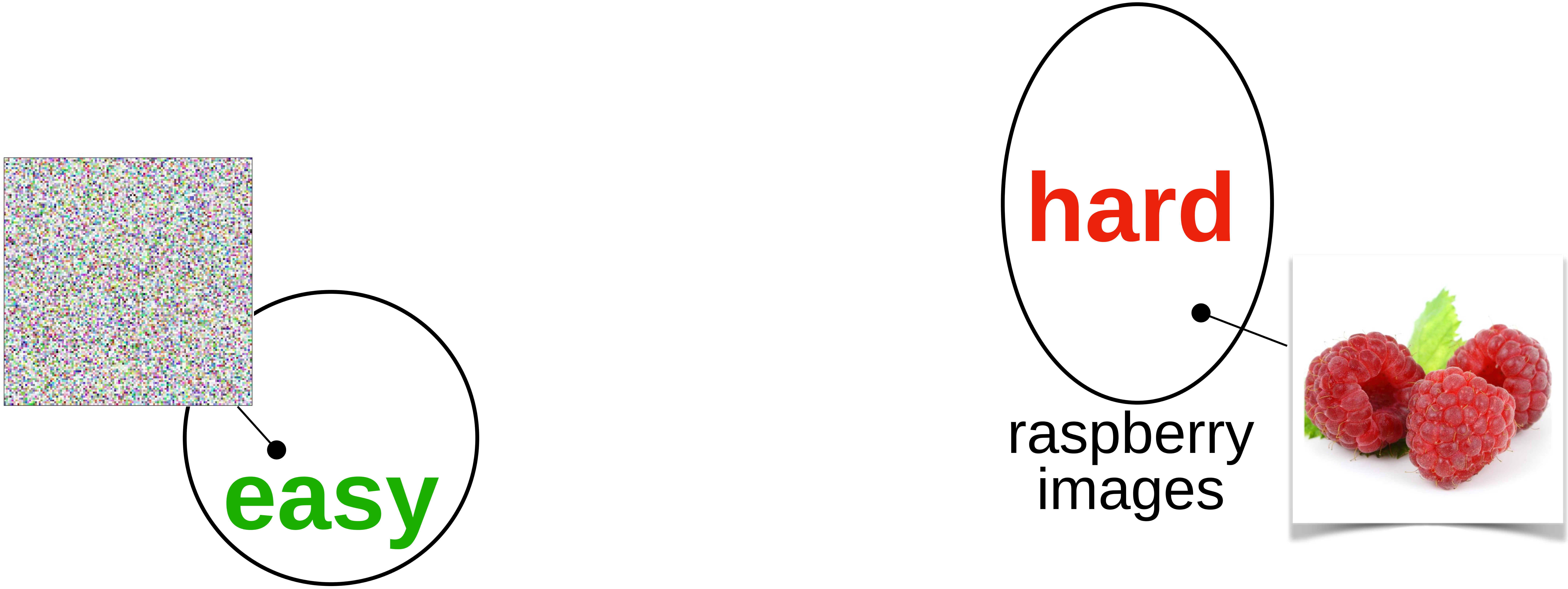
slide from Steve Seitz’s video

Generate 100 images

slide from Steve Seitz's video

Generate 100 images of raspberries

slide from Steve Seitz’s video



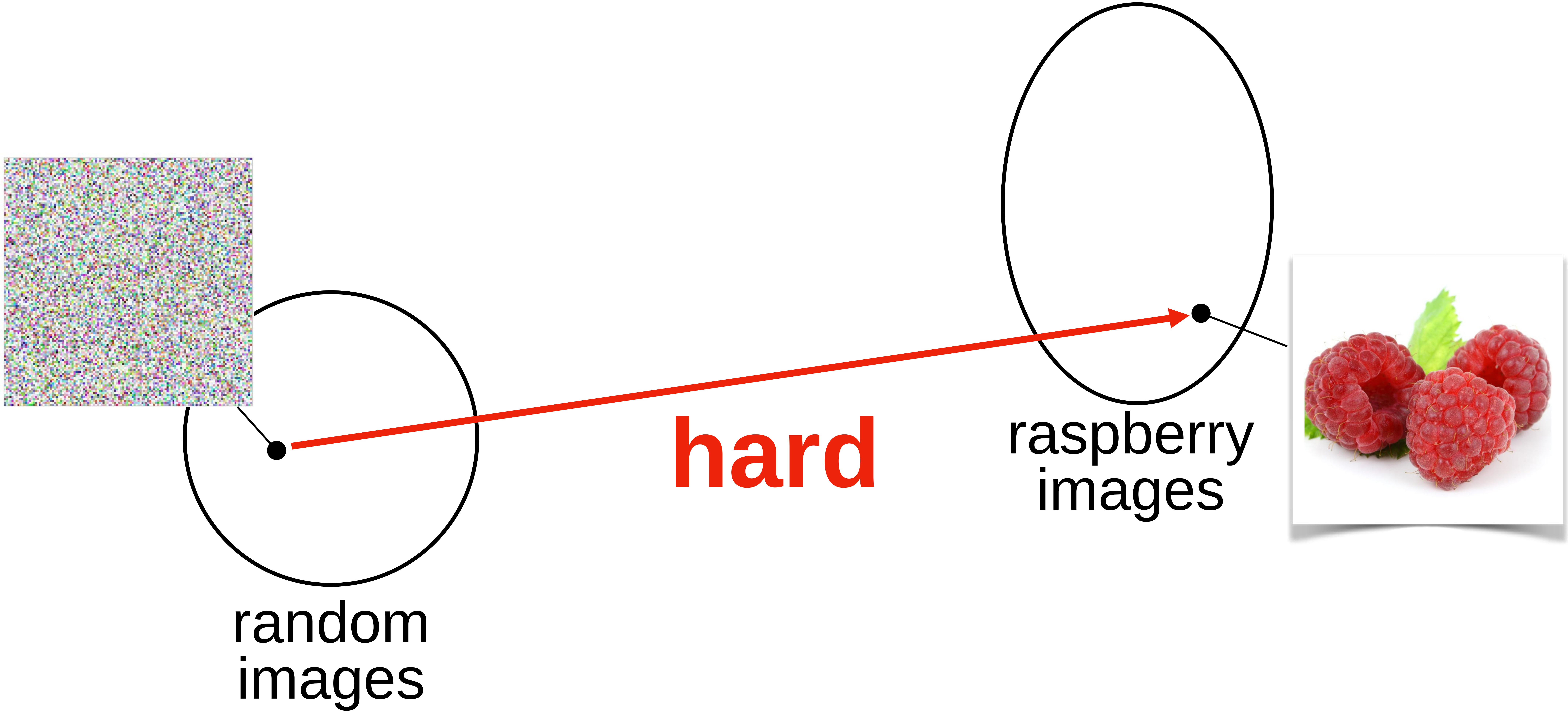
easy

random images

hard

raspberry images

slide from Steve Seitz's video

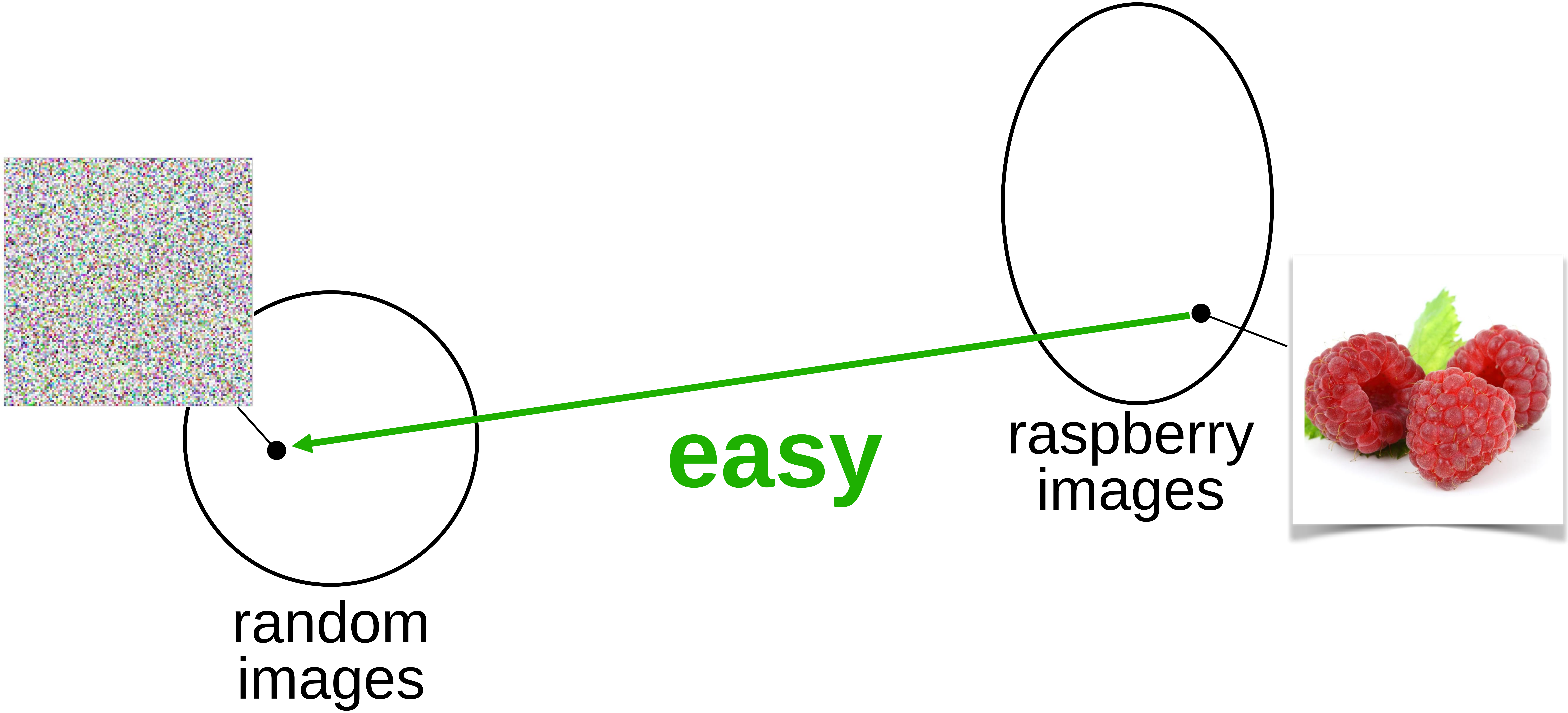


hard

random images

raspberry images

slide from Steve Seitz's video

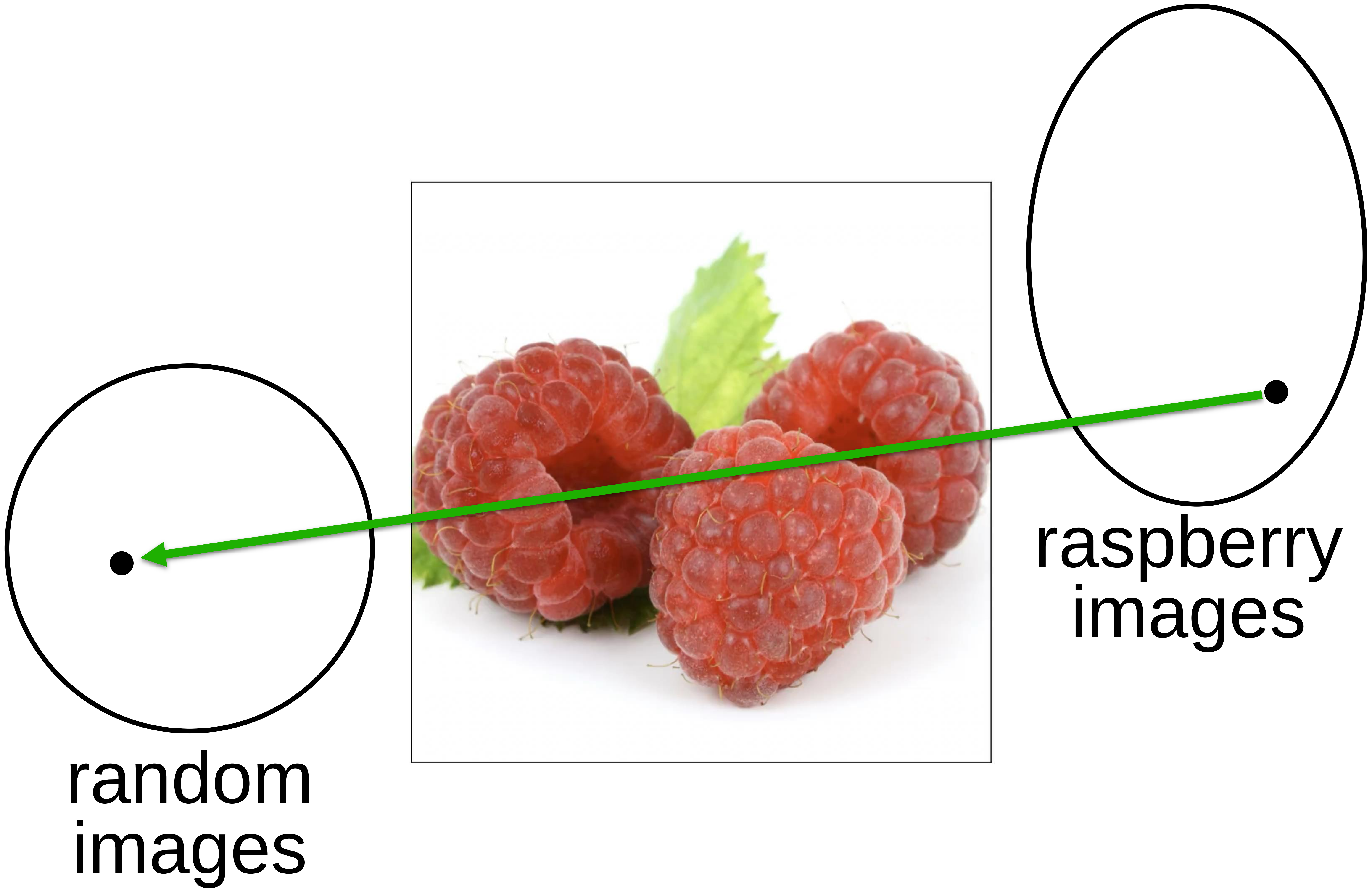


easy

raspberry
images

random
images

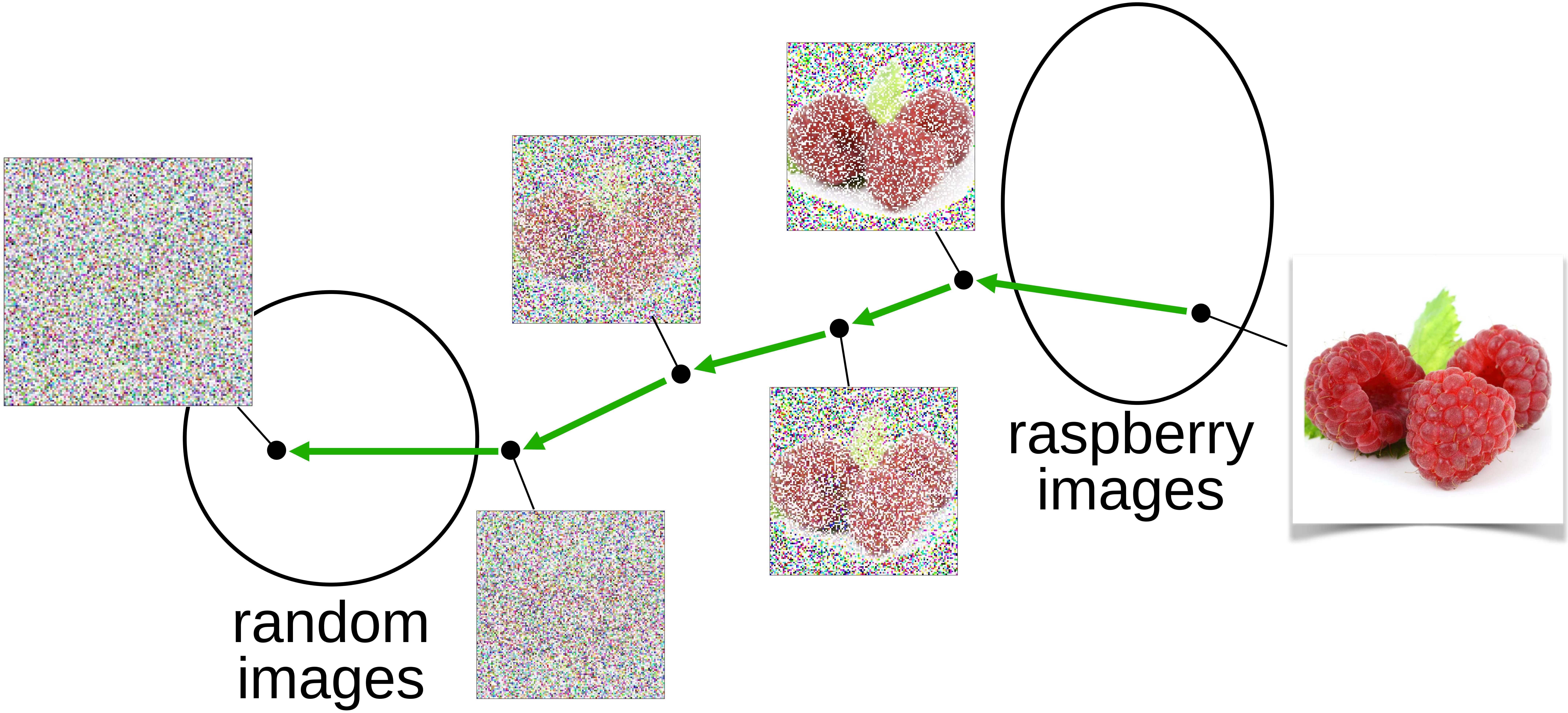
slide from Steve Seitz’s video



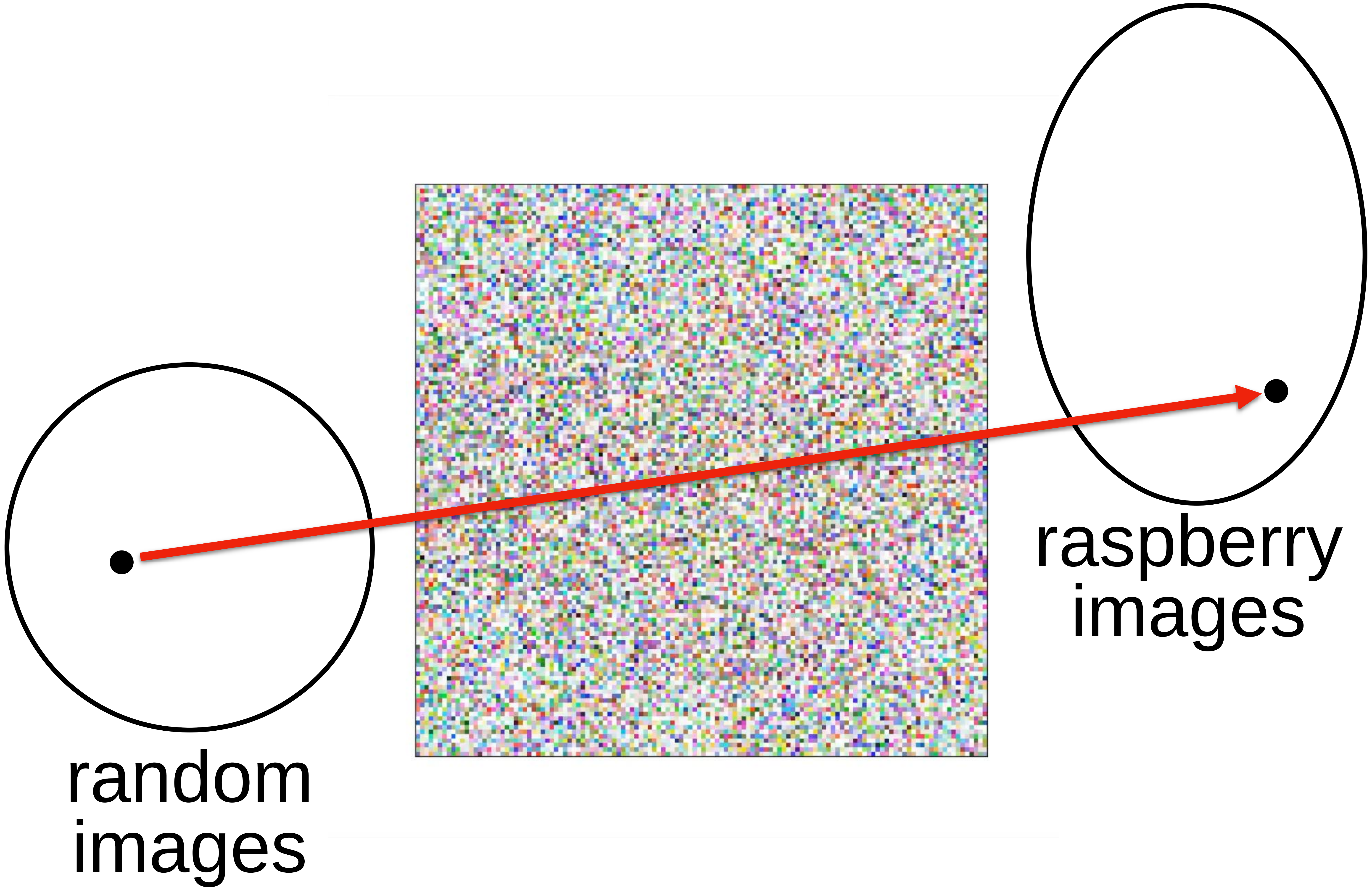
random images

raspberry images

slide from Steve Seitz’s video



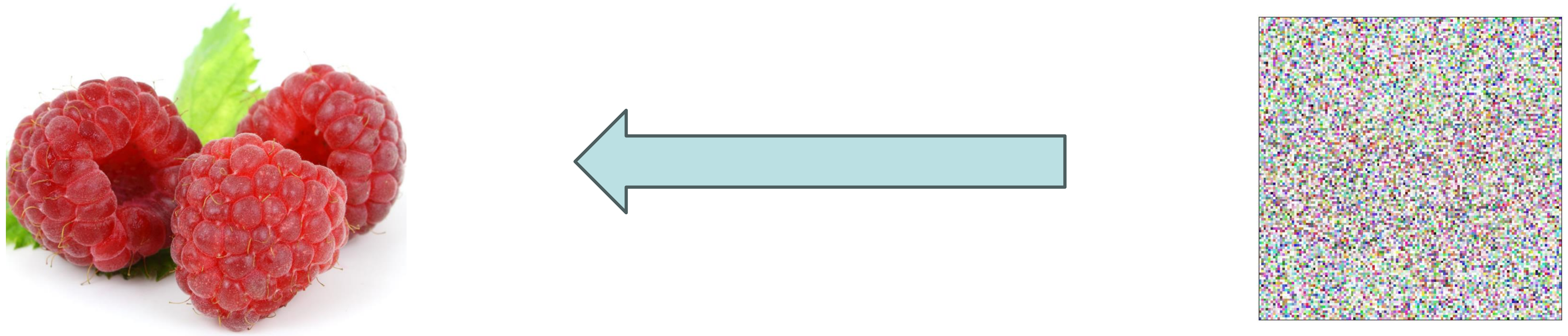
slide from Steve Seitz's video



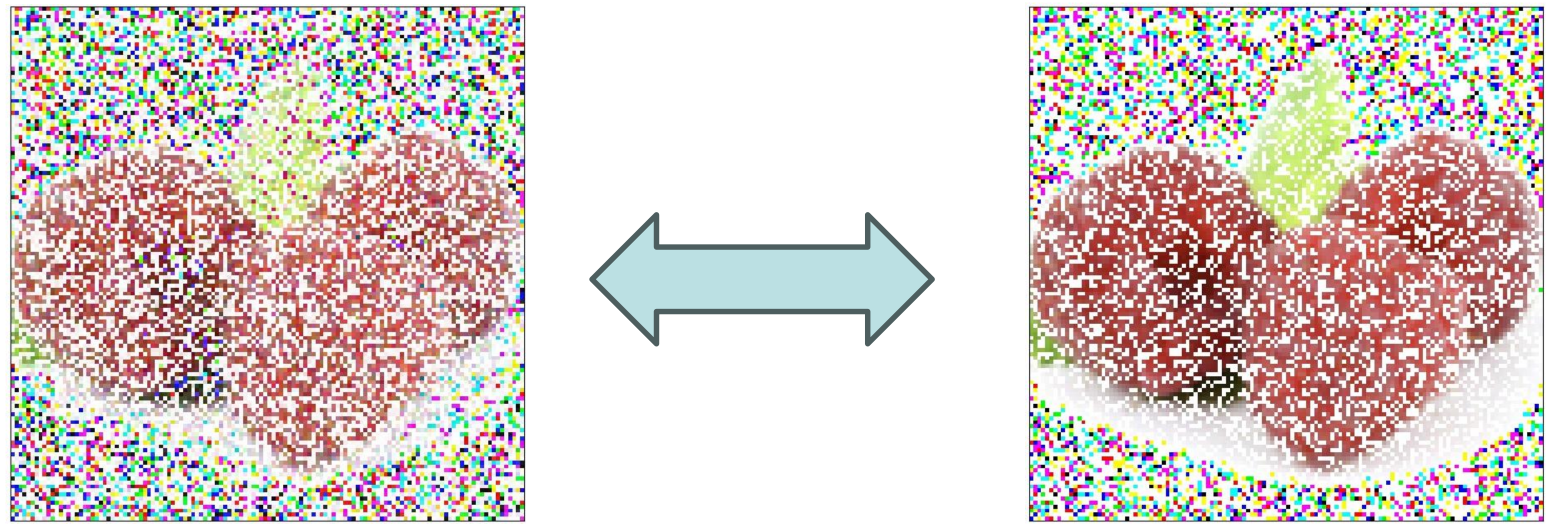
slide from Steve Seitz's video

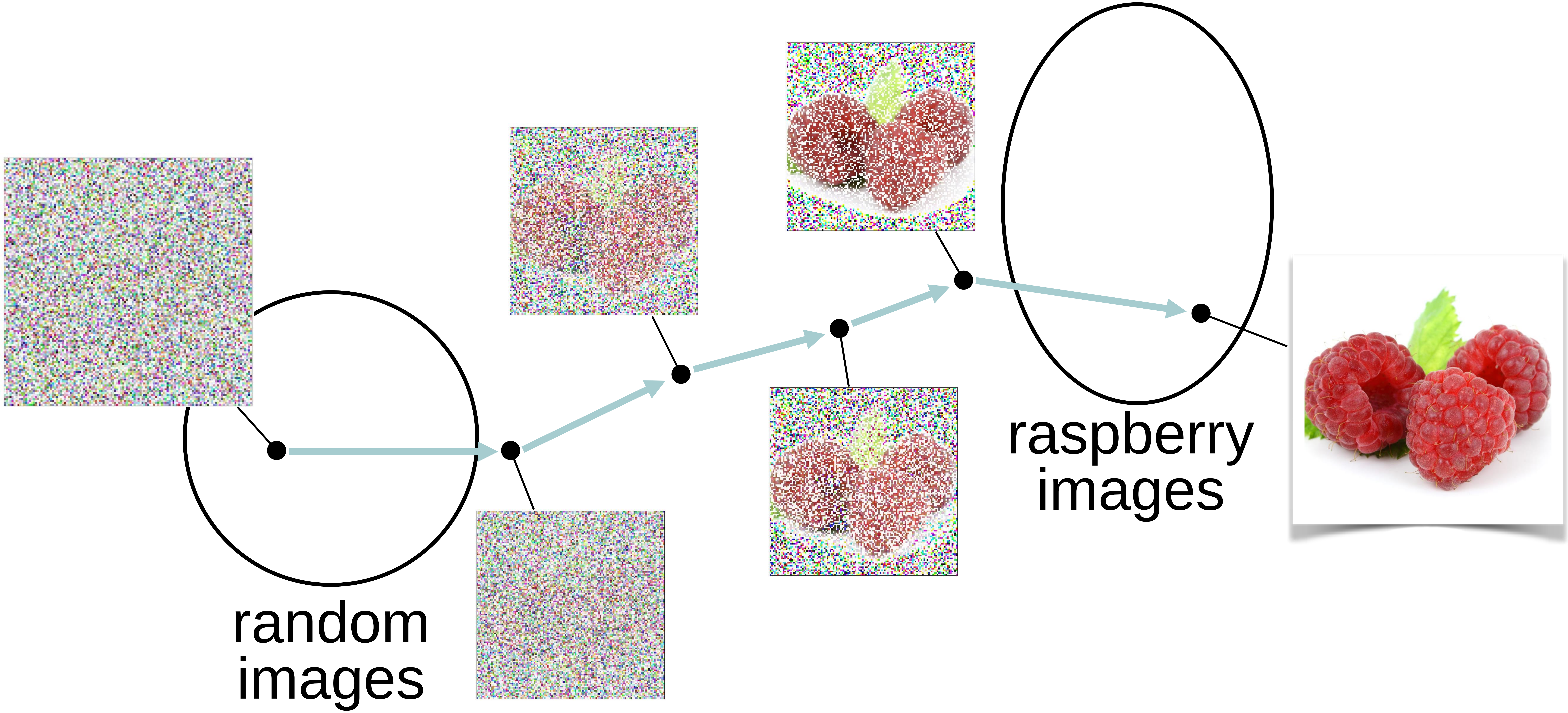
Key insight

- Globally, creation is much harder than destruction



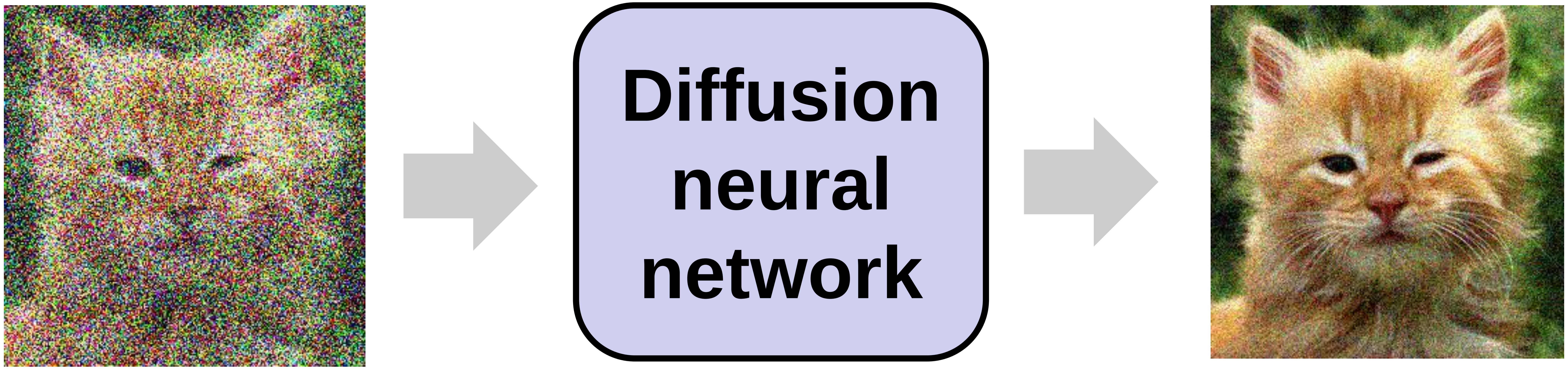
- But locally, they are almost reversable!





slide from Steve Seitz's video

Denoising diffusion neural network



Diffusion neural network

This network can be a U-Net or other suitable image-to-image network

Slide source: Steve Seitz

Recall: U-Net for image denoising

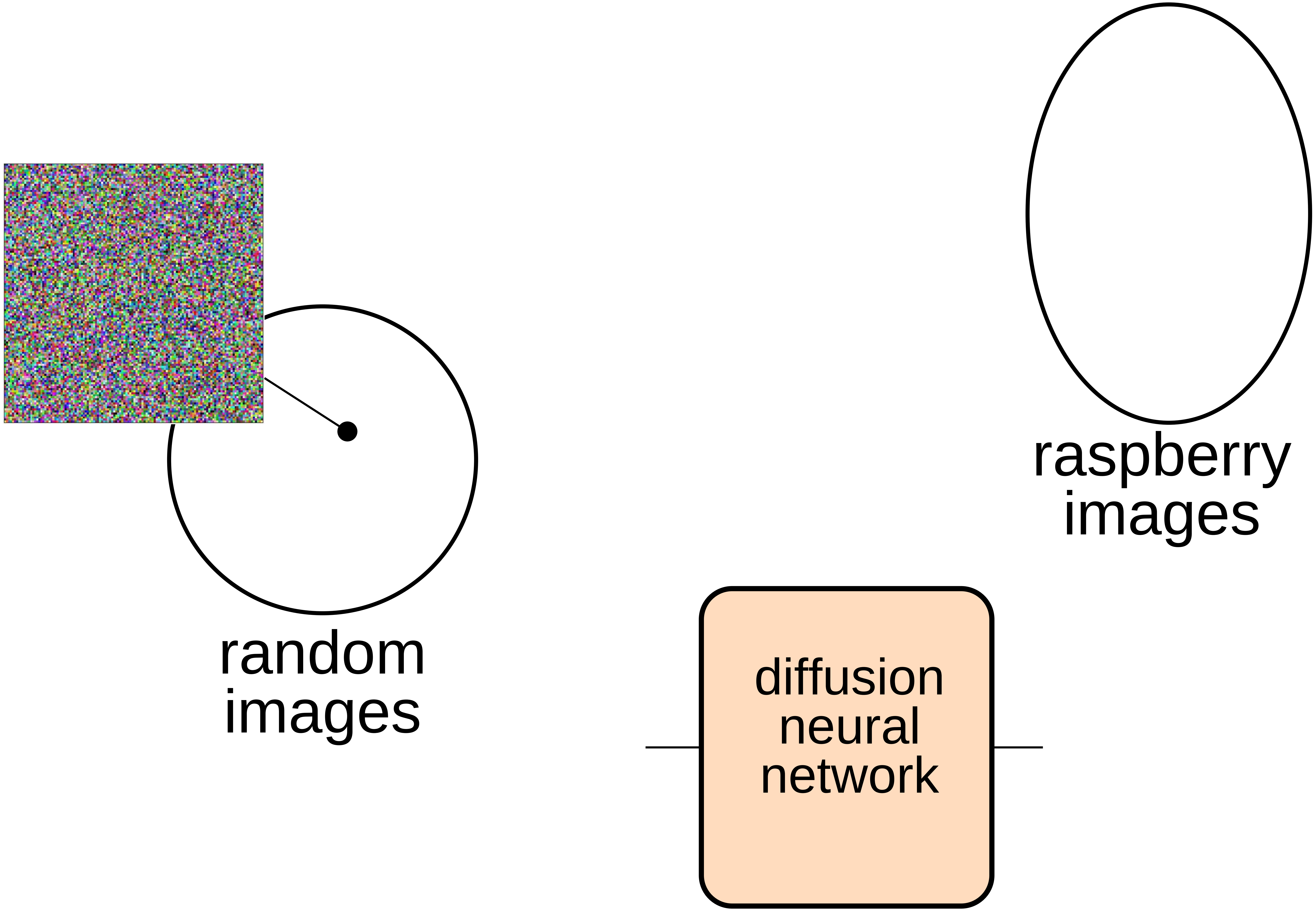
=
$\mathbf{x}_{\text{clean}}$
+
random noise
$\hat{\mathbf{x}}$

Loss: $\|\mathbf{x}_{\text{clean}} - \hat{\mathbf{x}}\|^2$

U-Net for diffusion

=
+
$\mathbf{x}_{\text{clean}}$
random noise
. . .
random noise

Loss: $\|\mathbf{x}_{\text{clean}} - \hat{\mathbf{x}}\|^2$

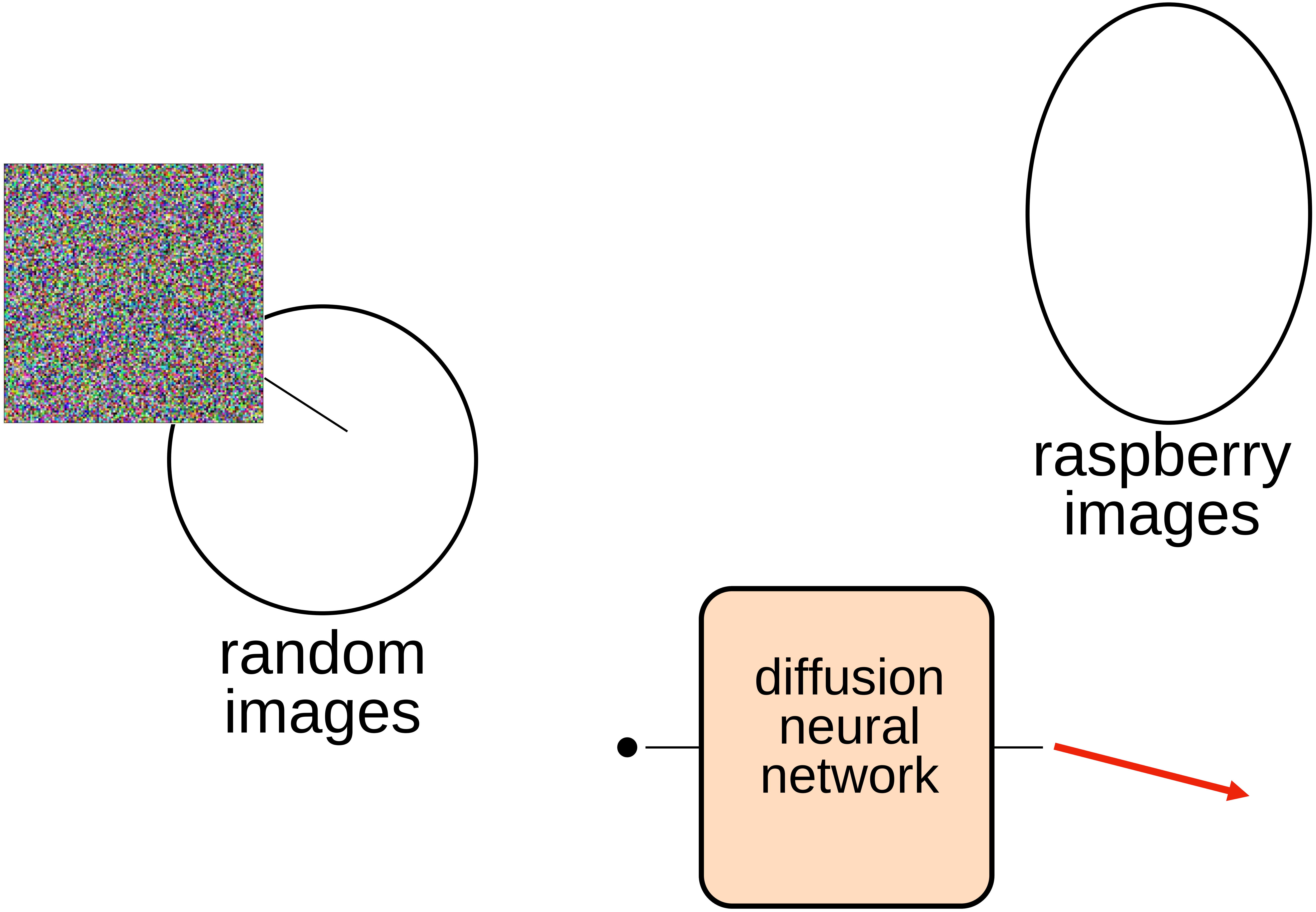


raspberry images

random images

diffusion neural network

slide from Steve Seitz's video

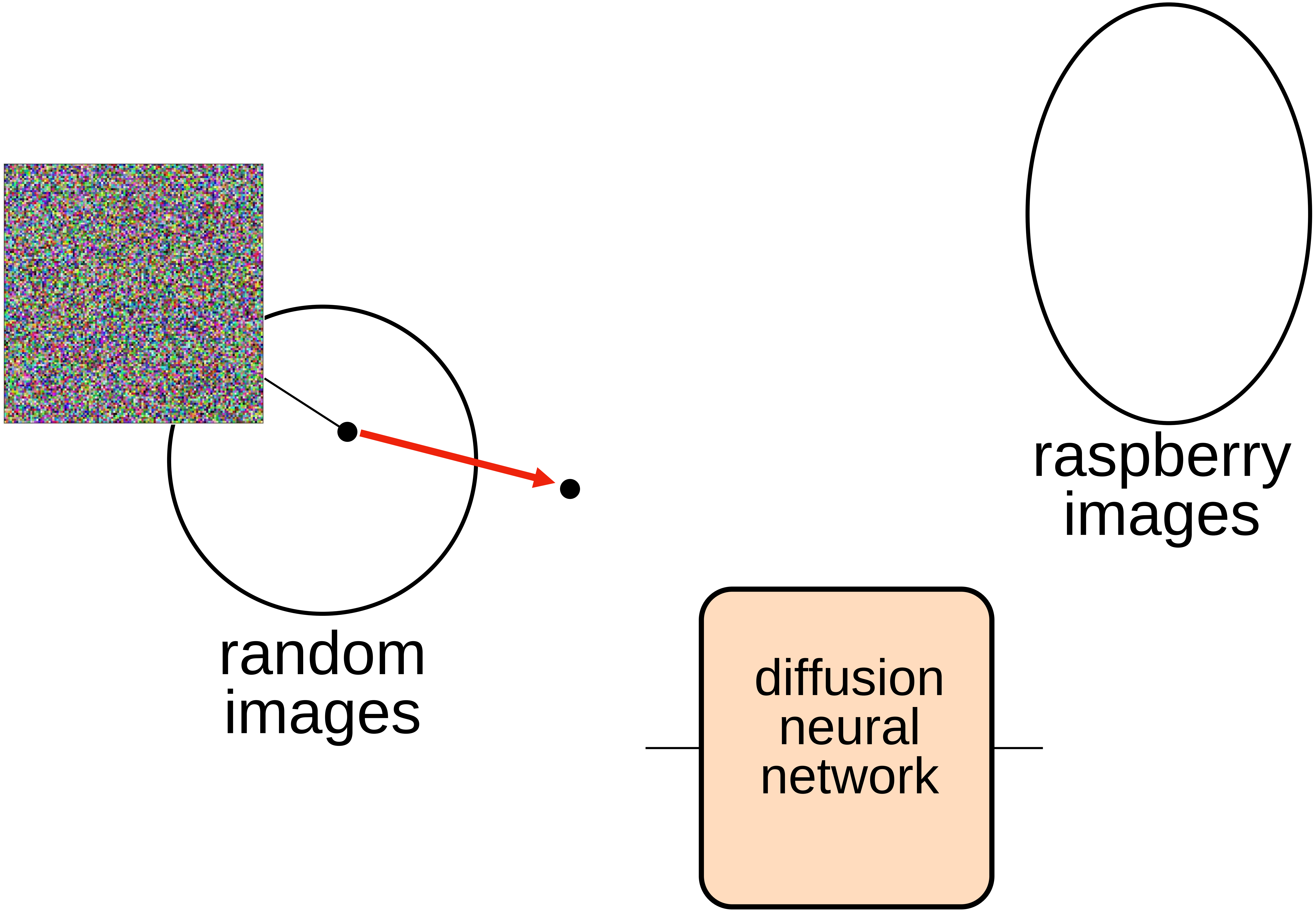


raspberry images

random images

diffusion neural network

slide from Steve Seitz’s video

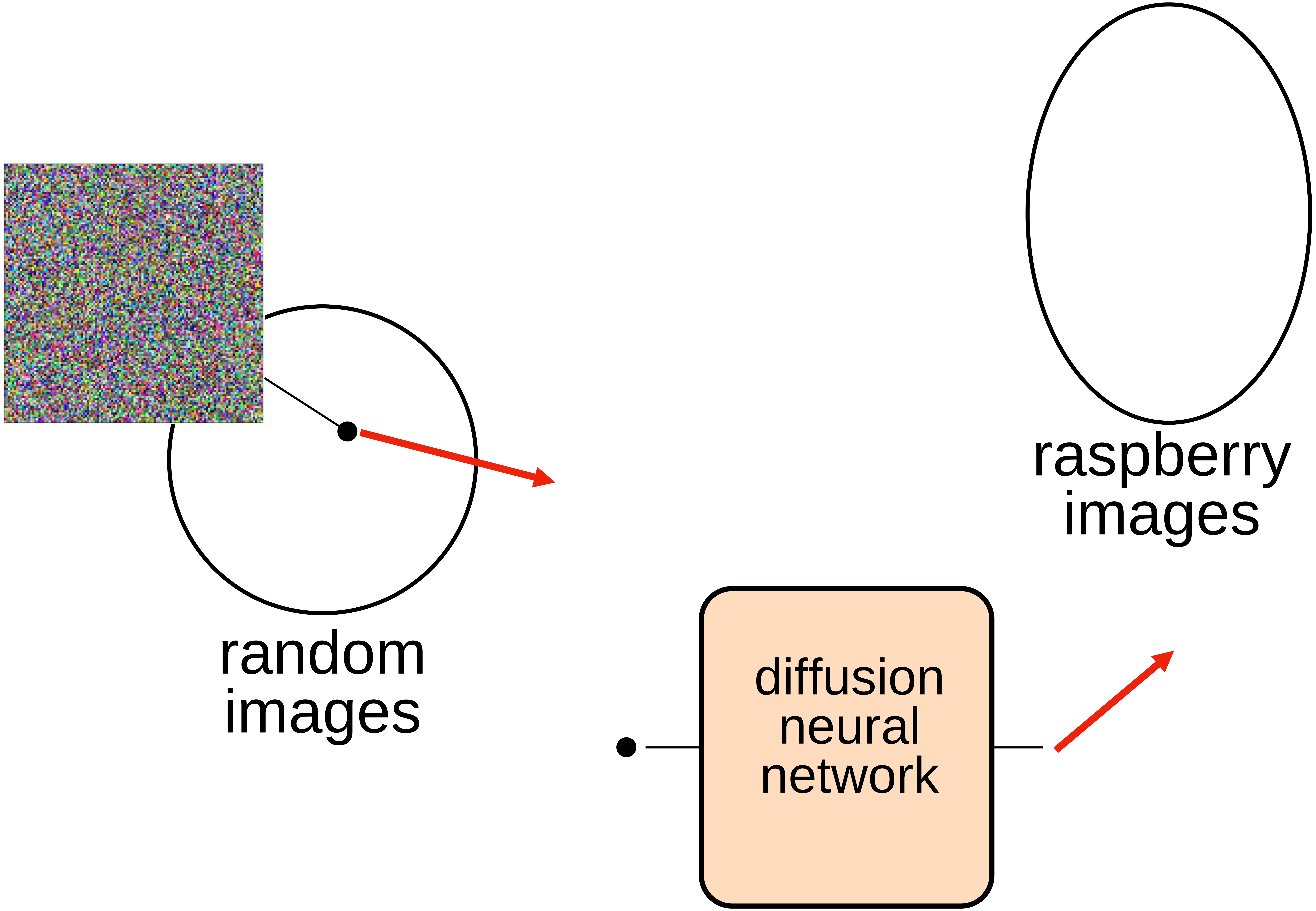


random images

raspberry images

diffusion neural network

slide from Steve Seitz’s [video]

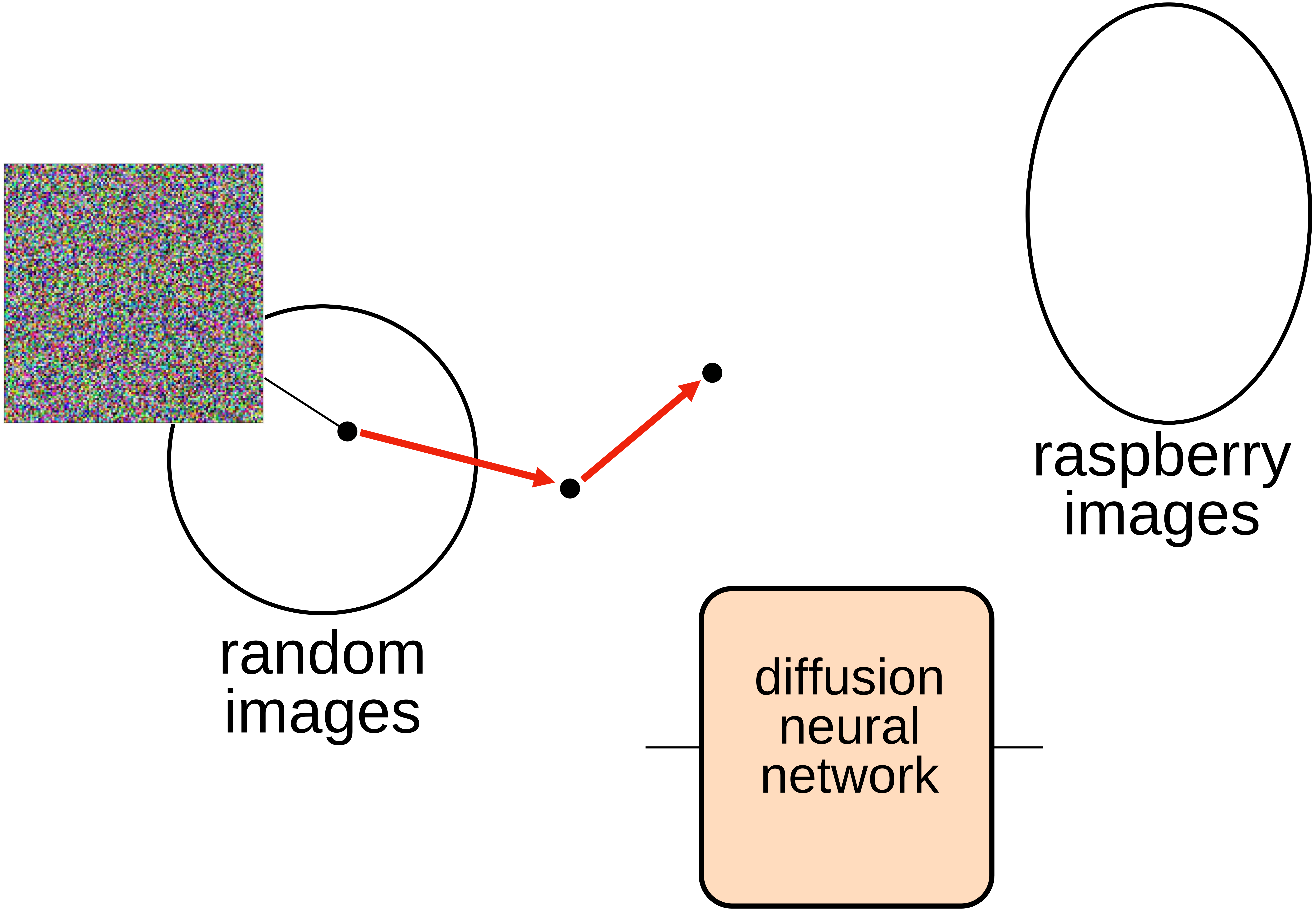


random images

raspberry images

diffusion neural network

slide from Steve Seitz's video

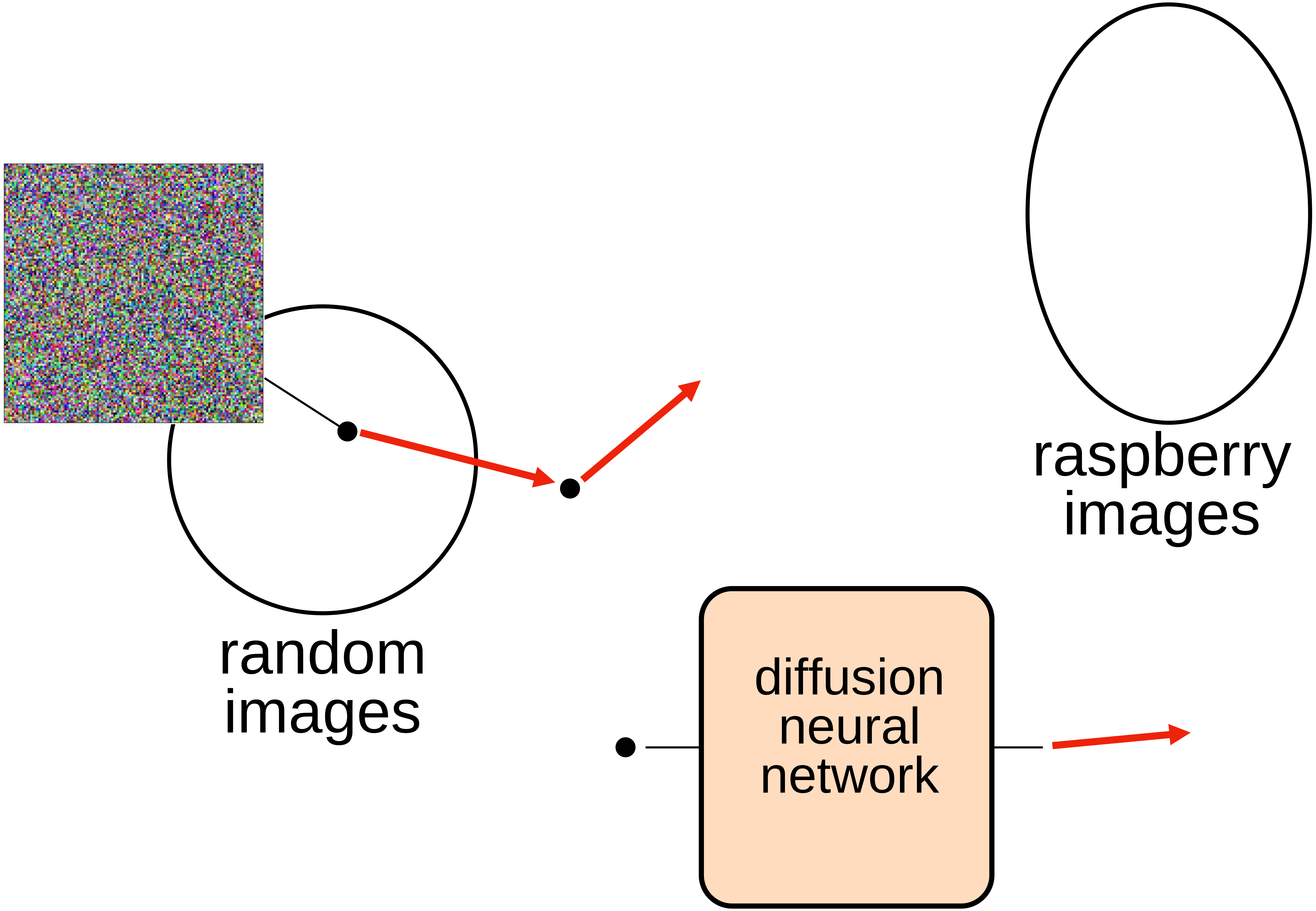


random images

raspberry images

diffusion neural network

slide from Steve Seitz’s [video]

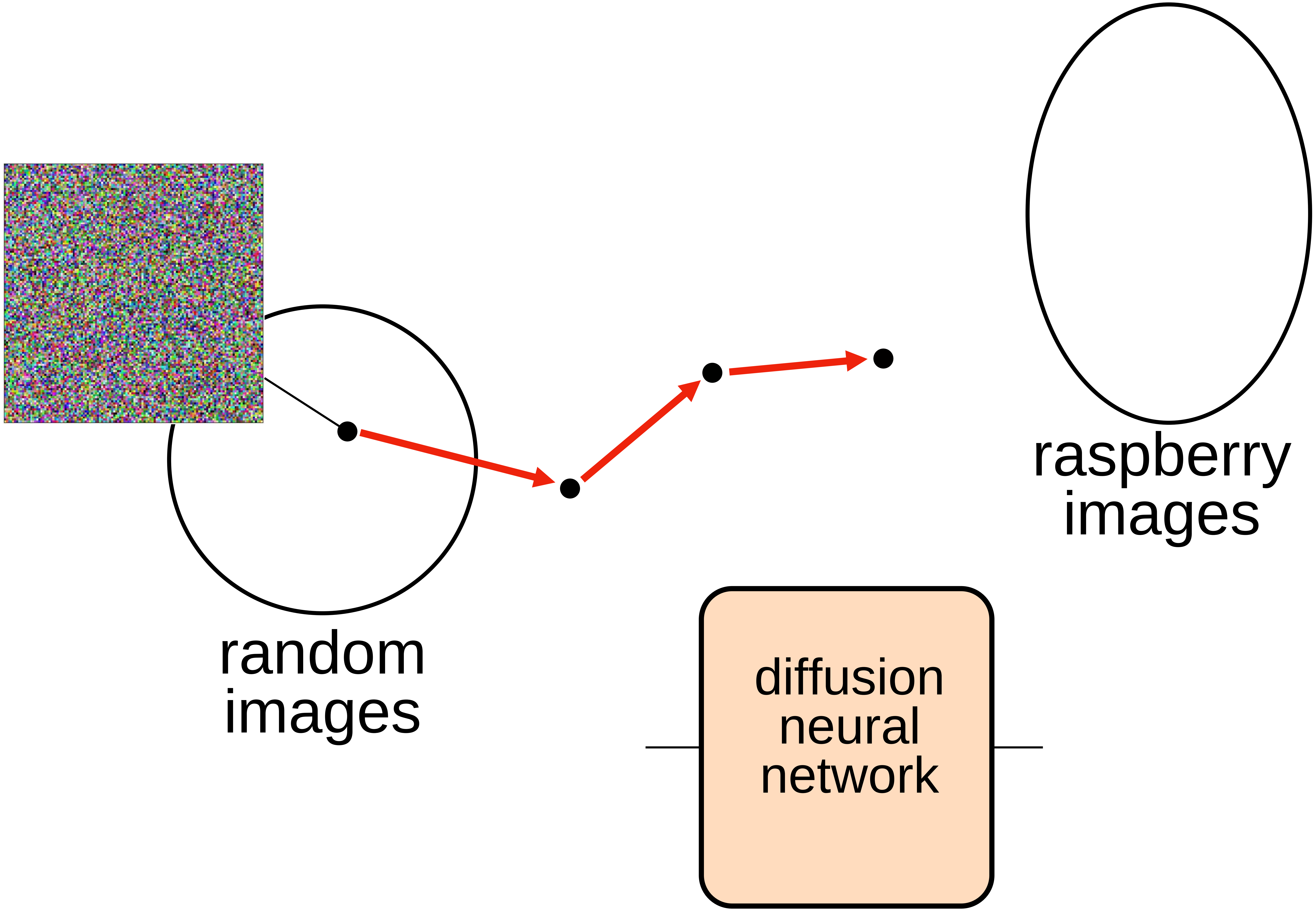


random images

raspberry images

diffusion neural network

slide from Steve Seitz's video

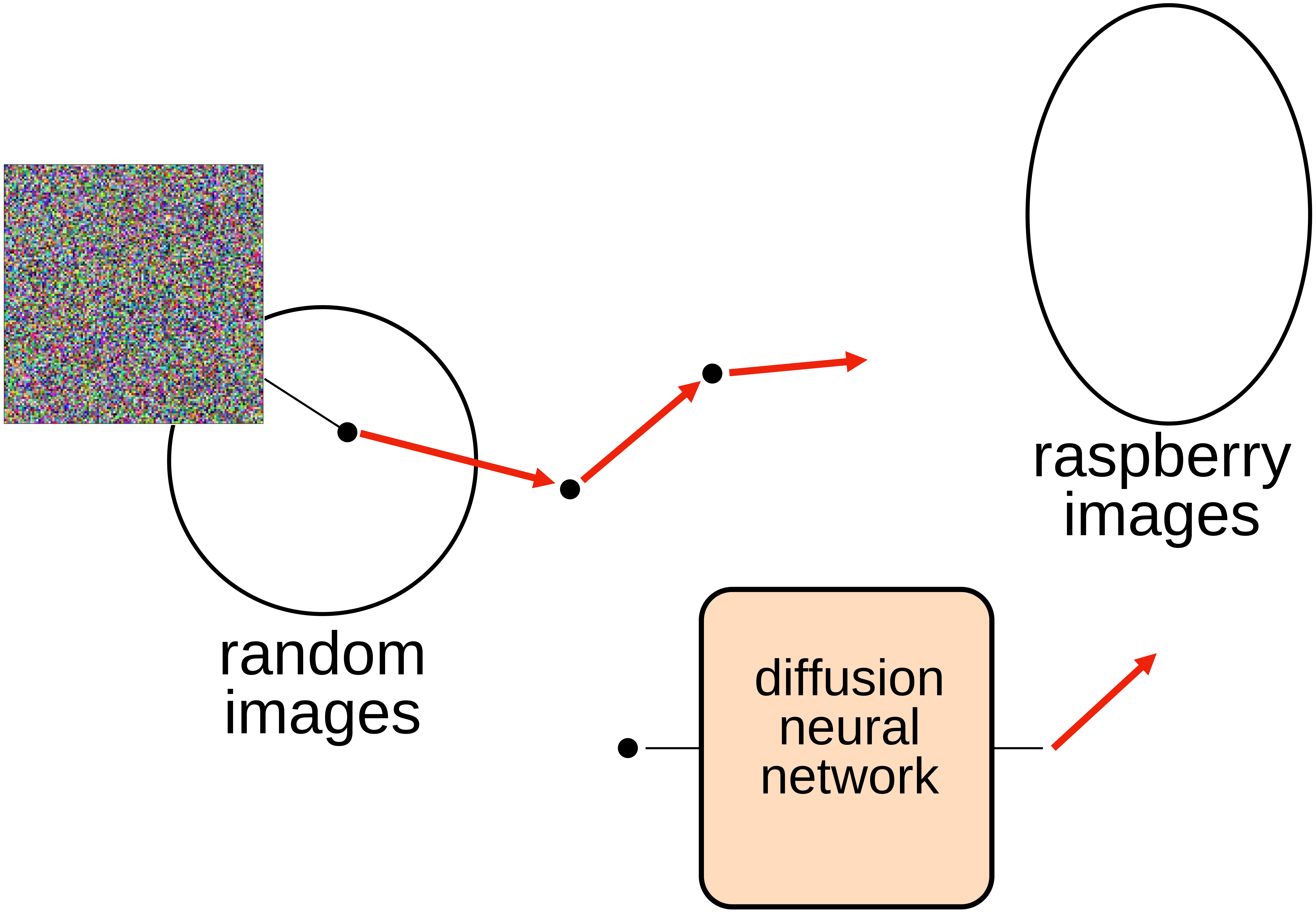


random images

raspberry images

diffusion neural network

slide from Steve Seitz's [video]

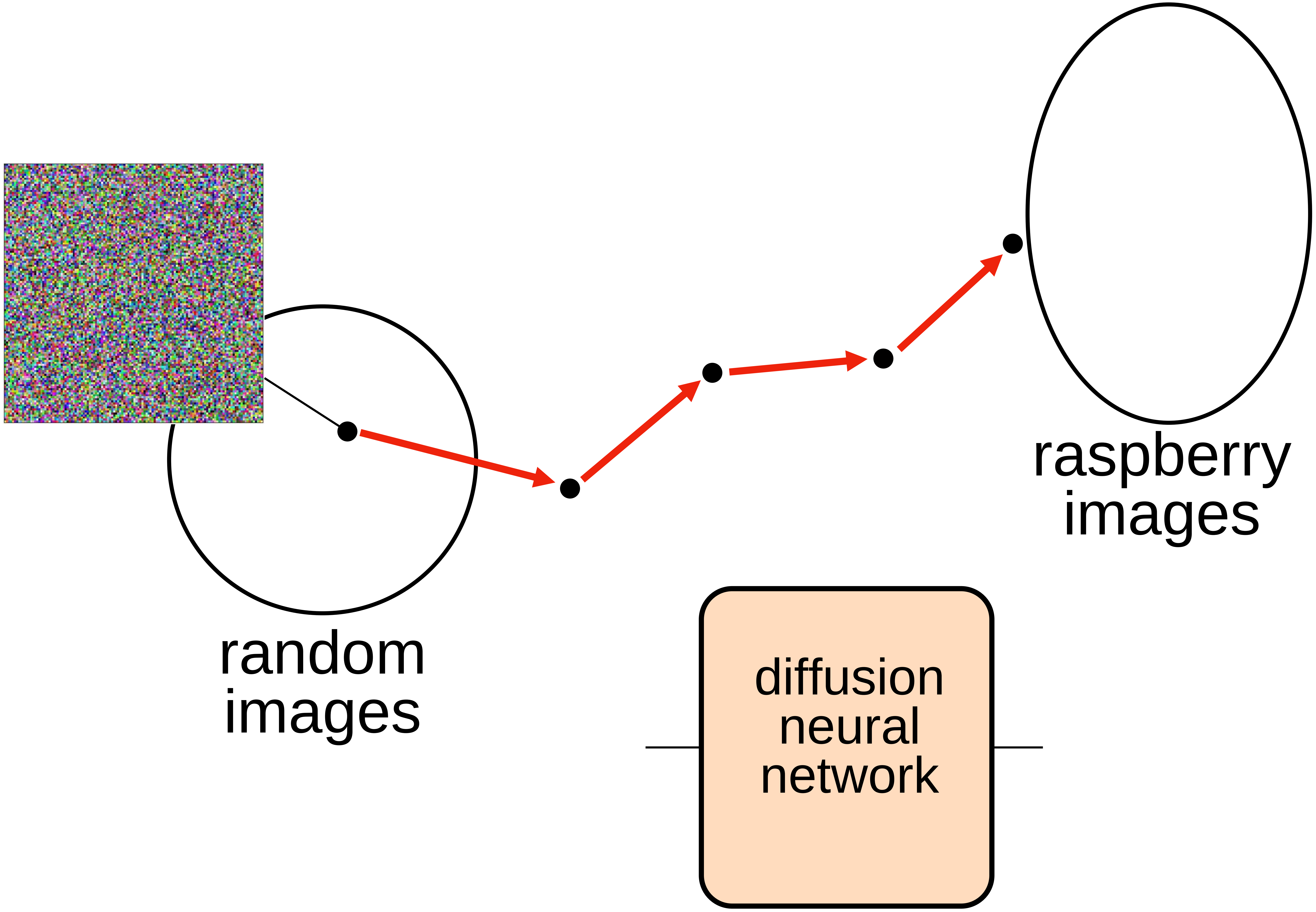


random images

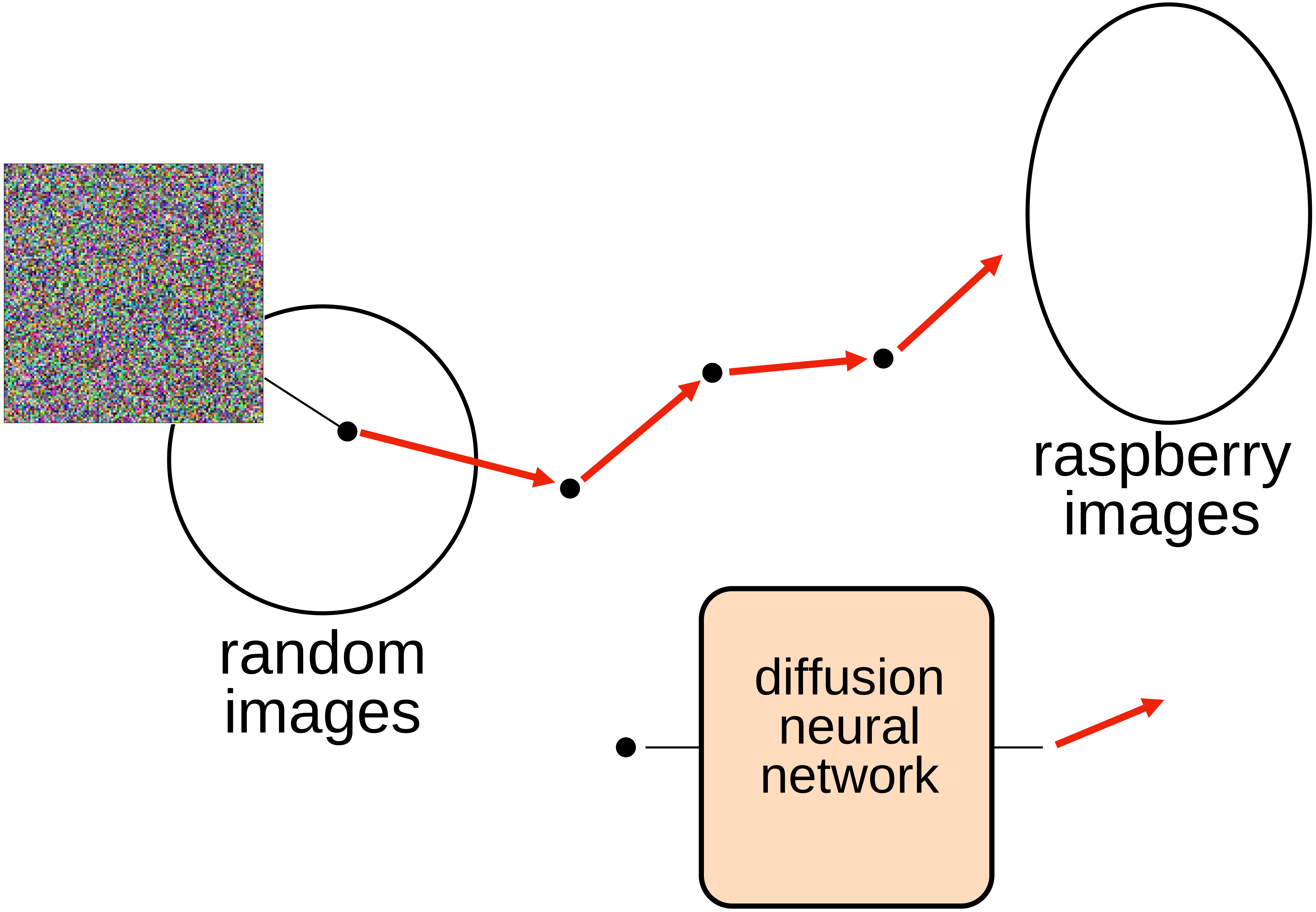
raspberry images

diffusion neural network

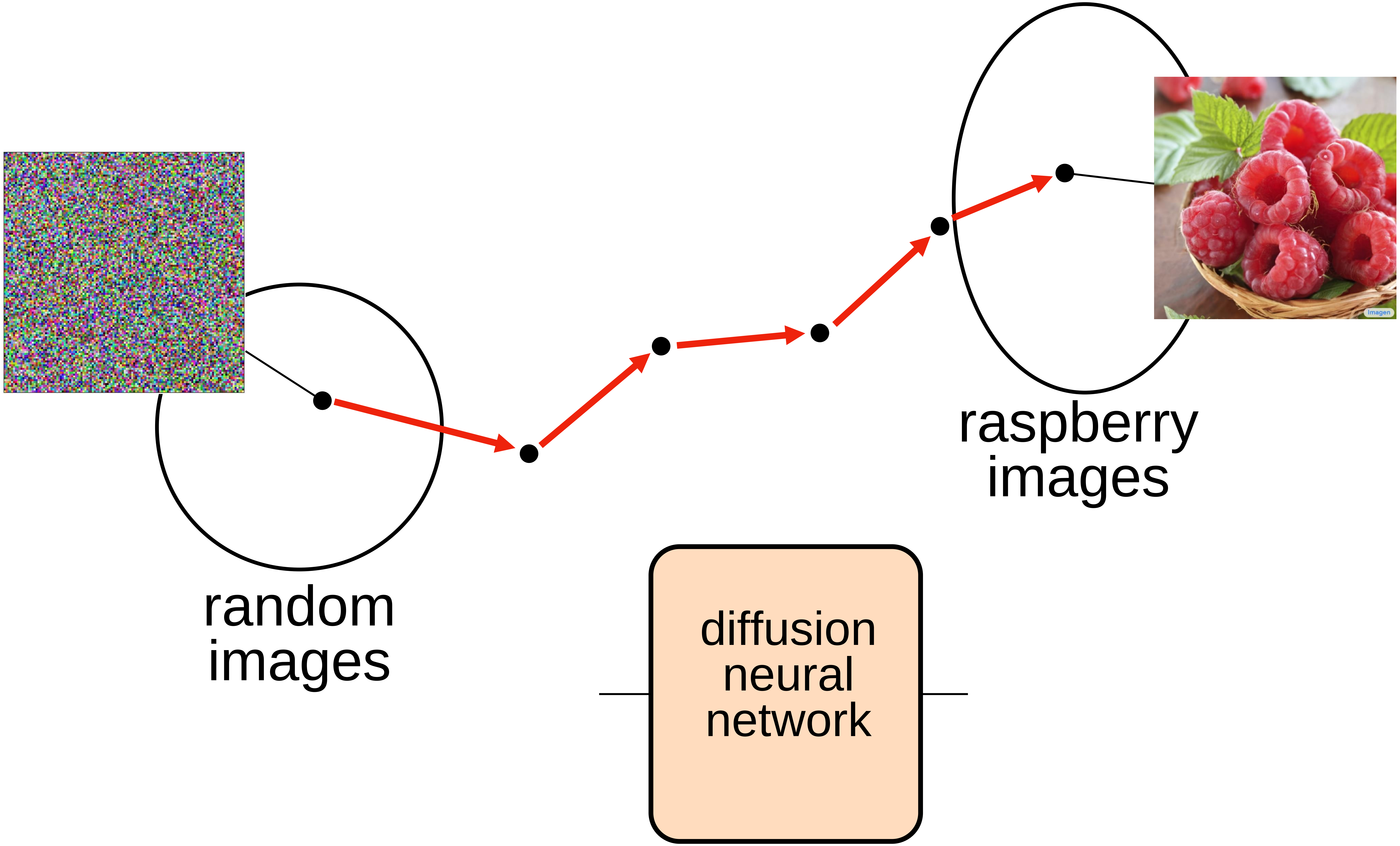
slide from Steve Seitz's video



slide from Steve Seitz's [video]



slide from Steve Seitz's video



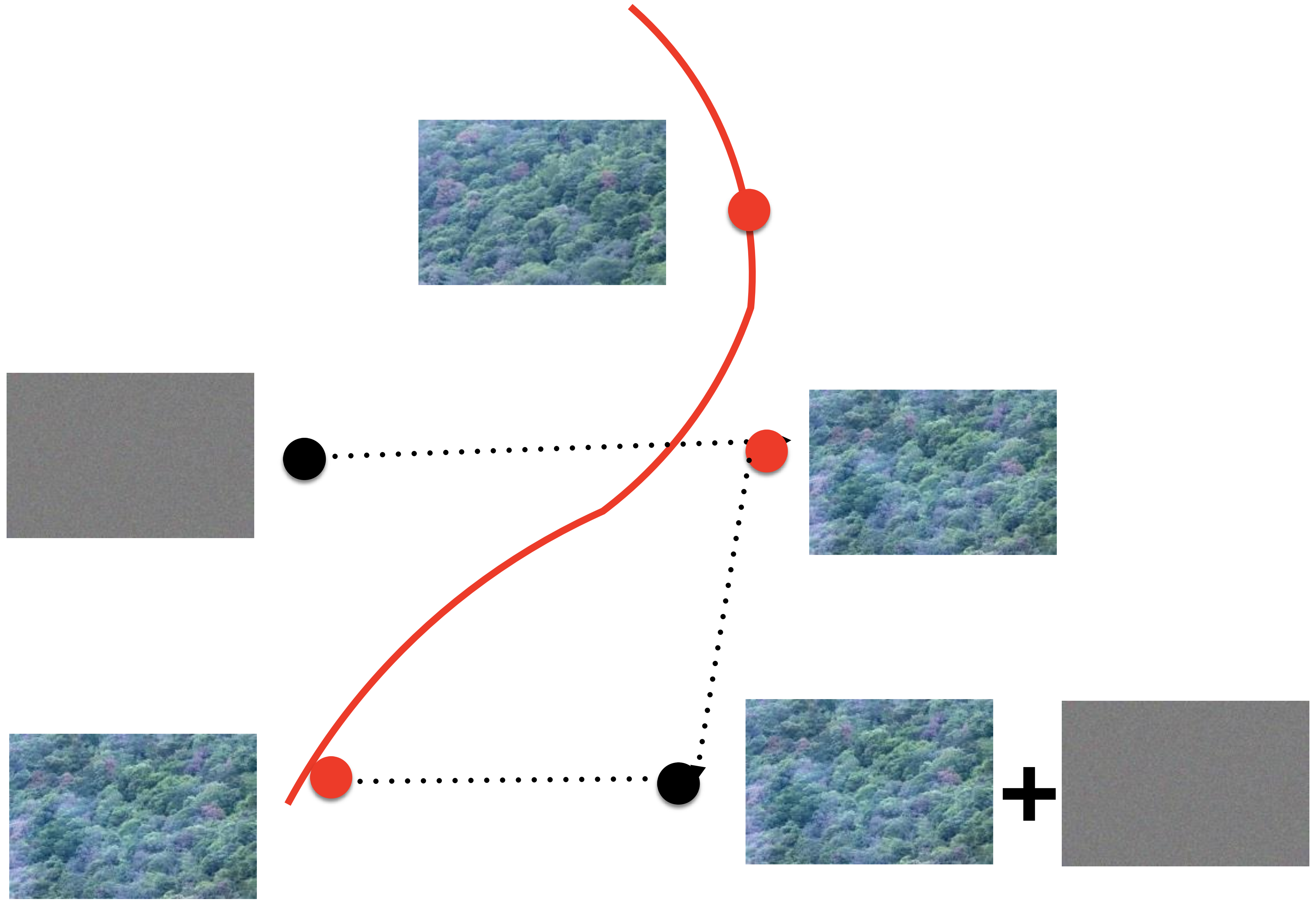
random images

raspberry images

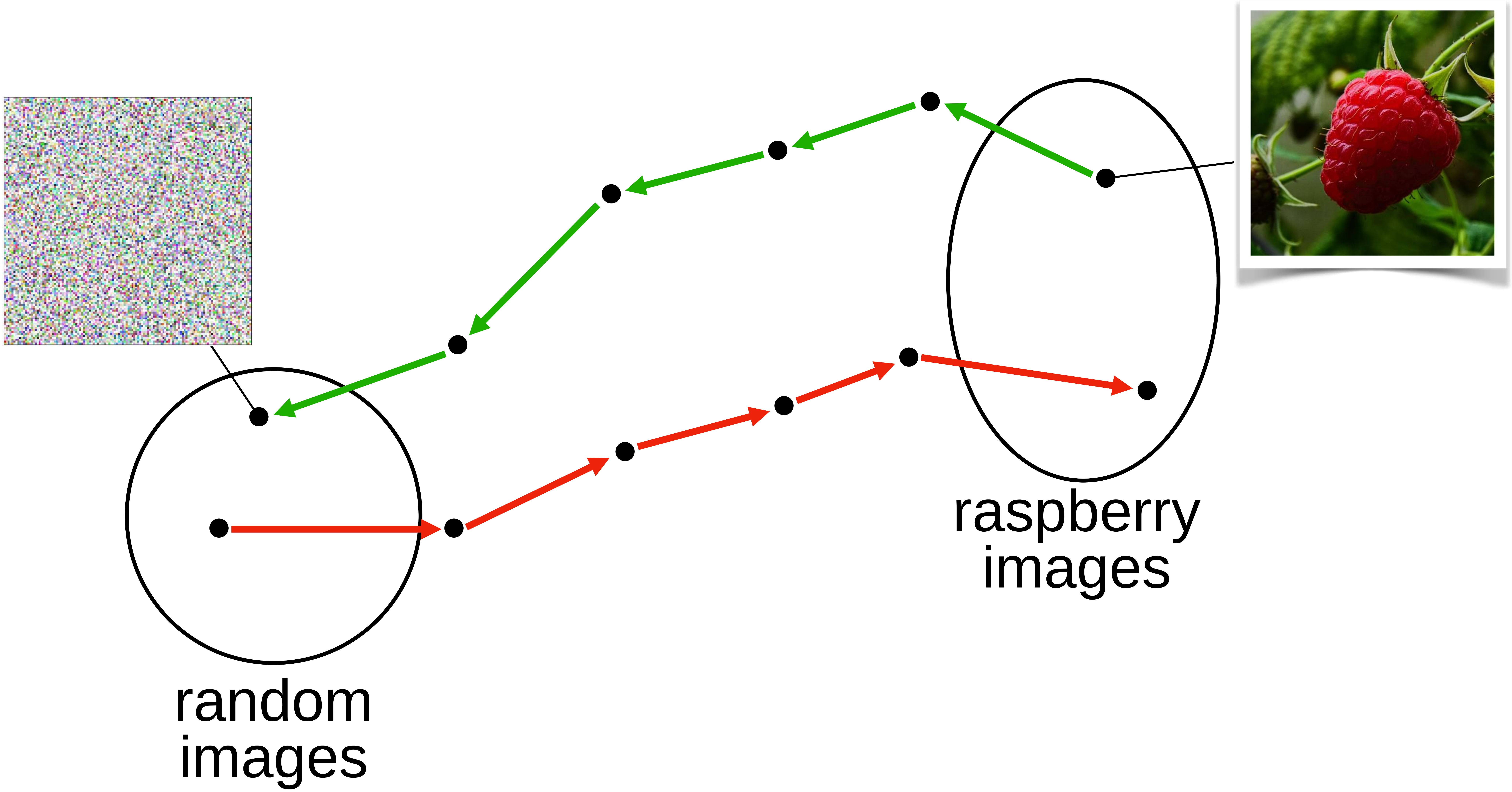
diffusion neural network

slide from Steve Seitz's video

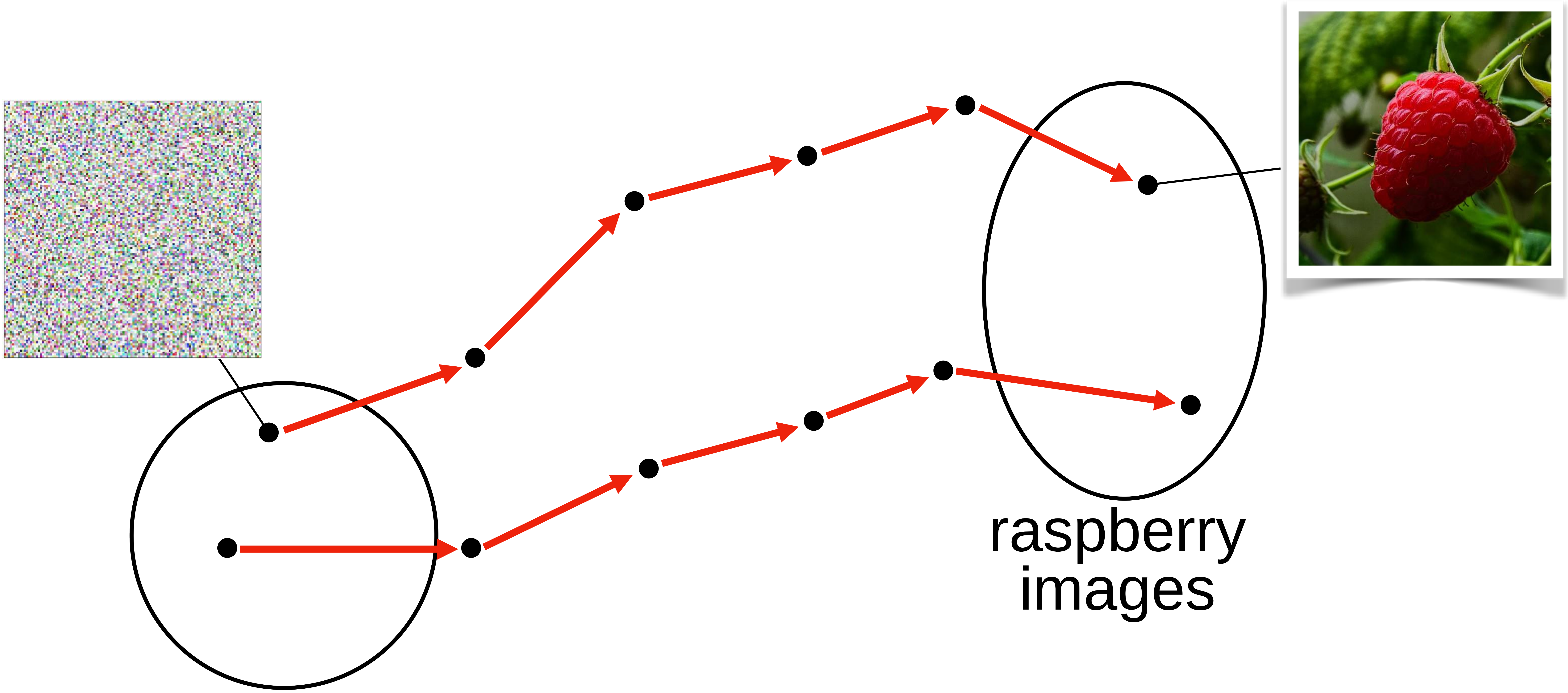
Projecting onto the “Image Manifold”



Images with equal
model response



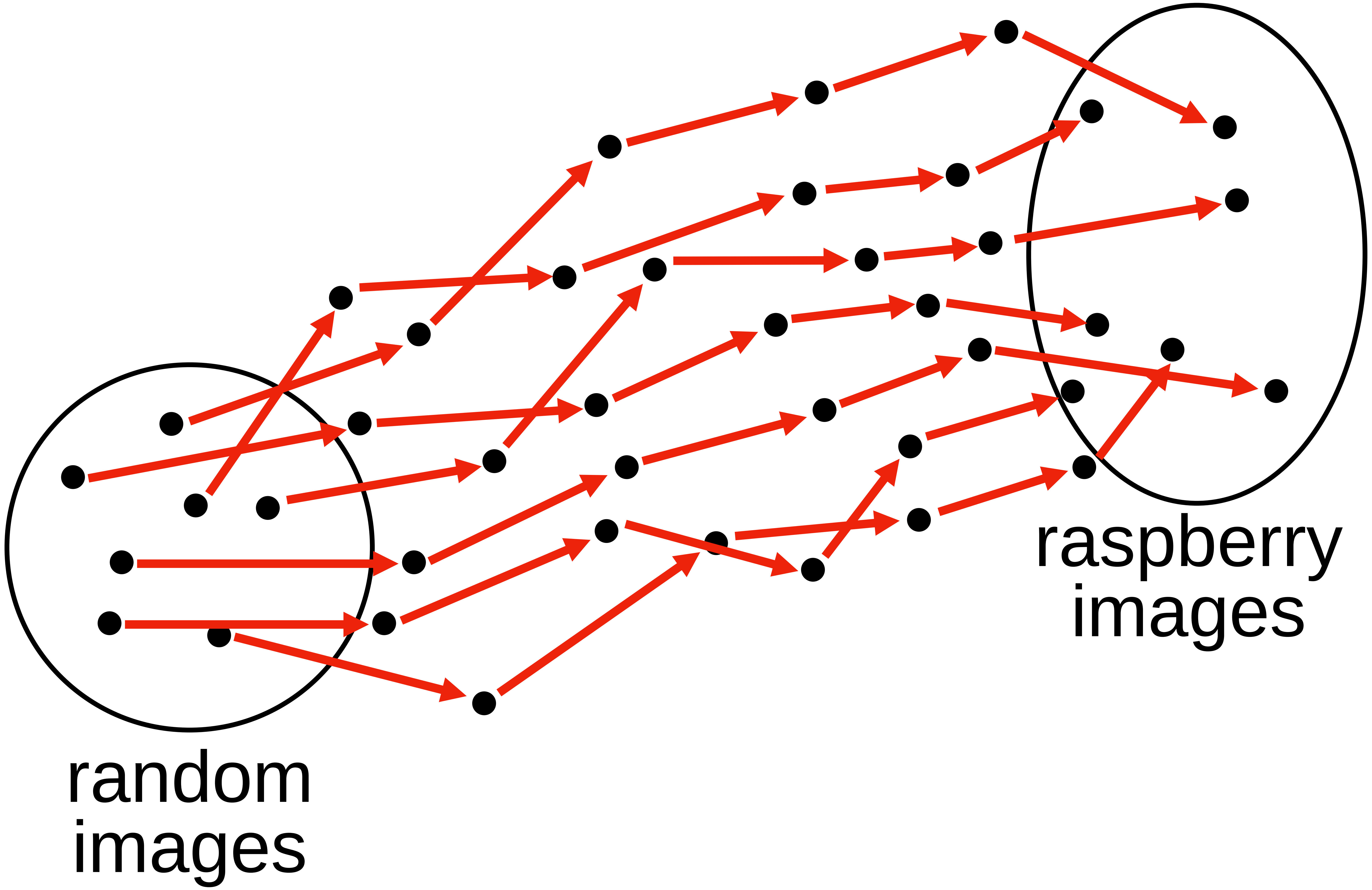
slide from Steve Seitz's video



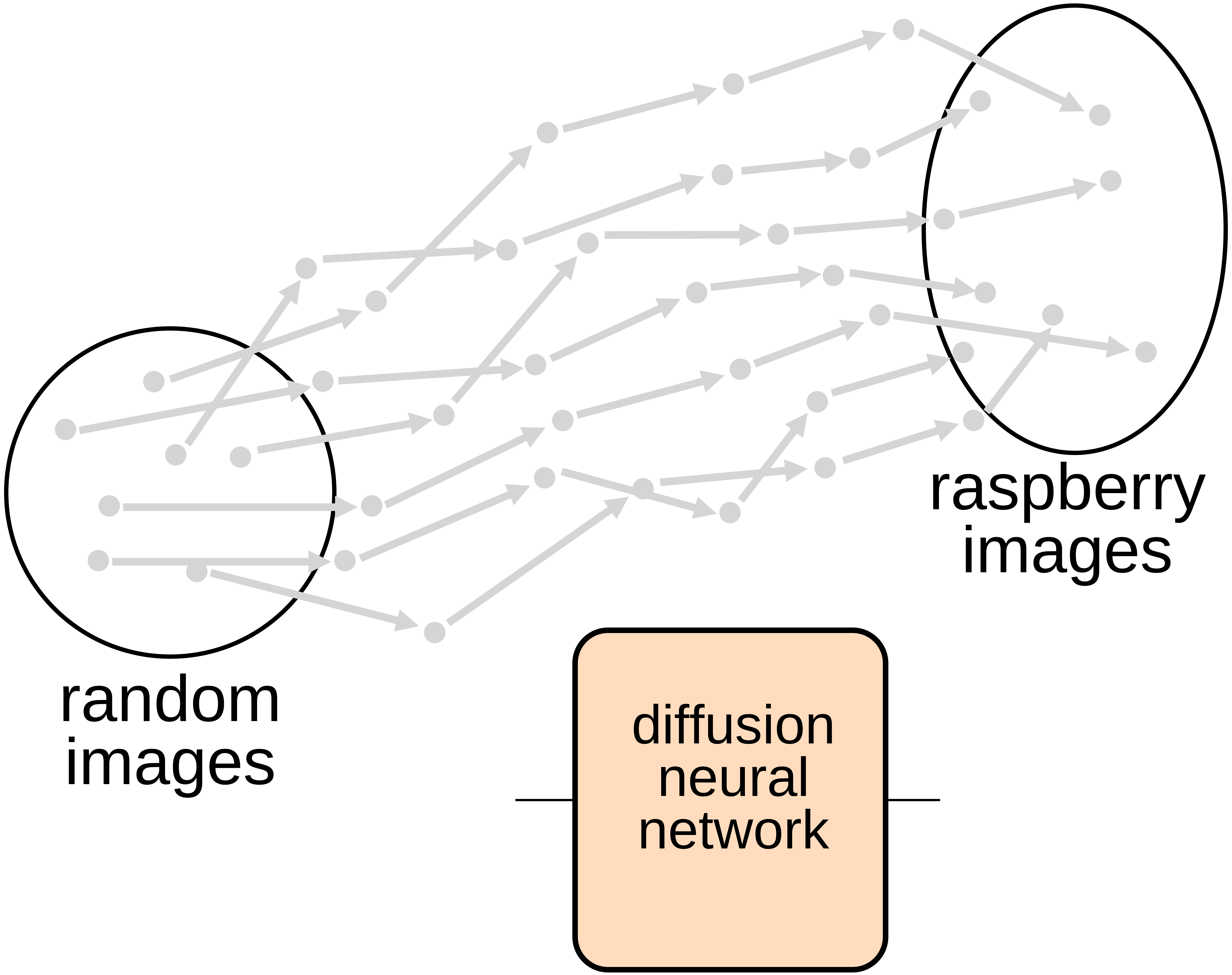
raspberry images

random images

slide from Steve Seitz's video



slide from Steve Seitz's video

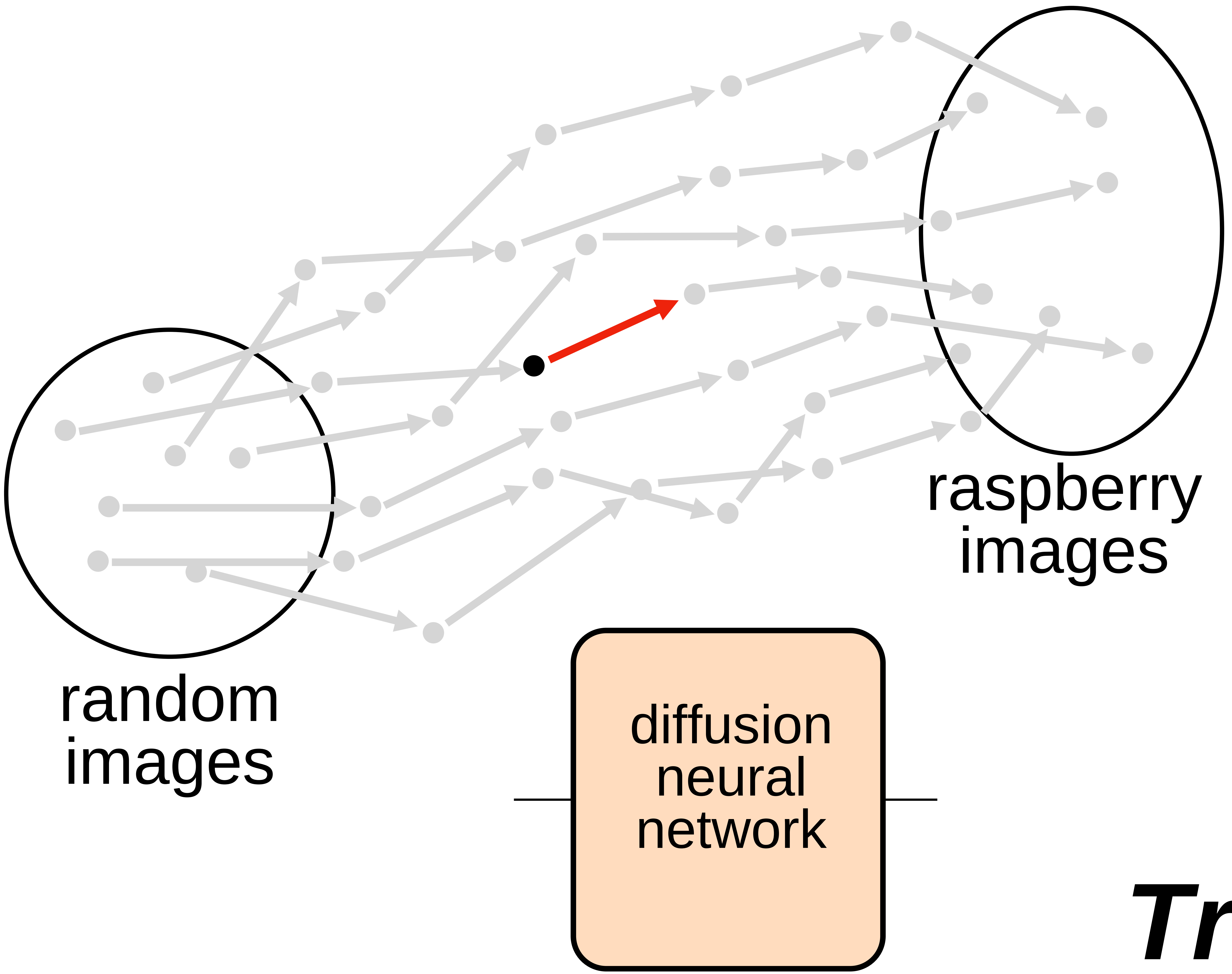


random images

raspberry images

diffusion neural network

slide from Steve Seitz's video



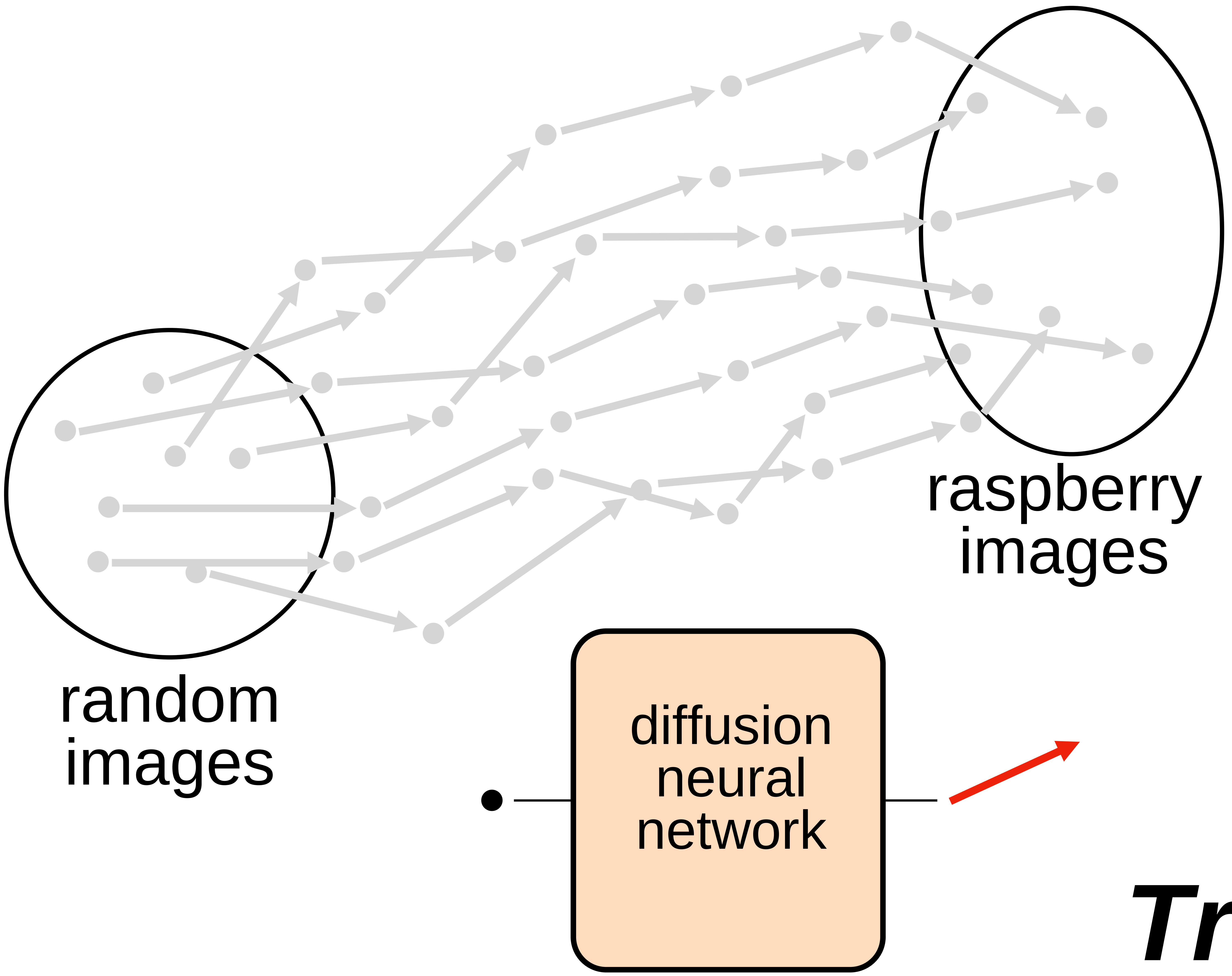
random images

raspberry images

diffusion neural network

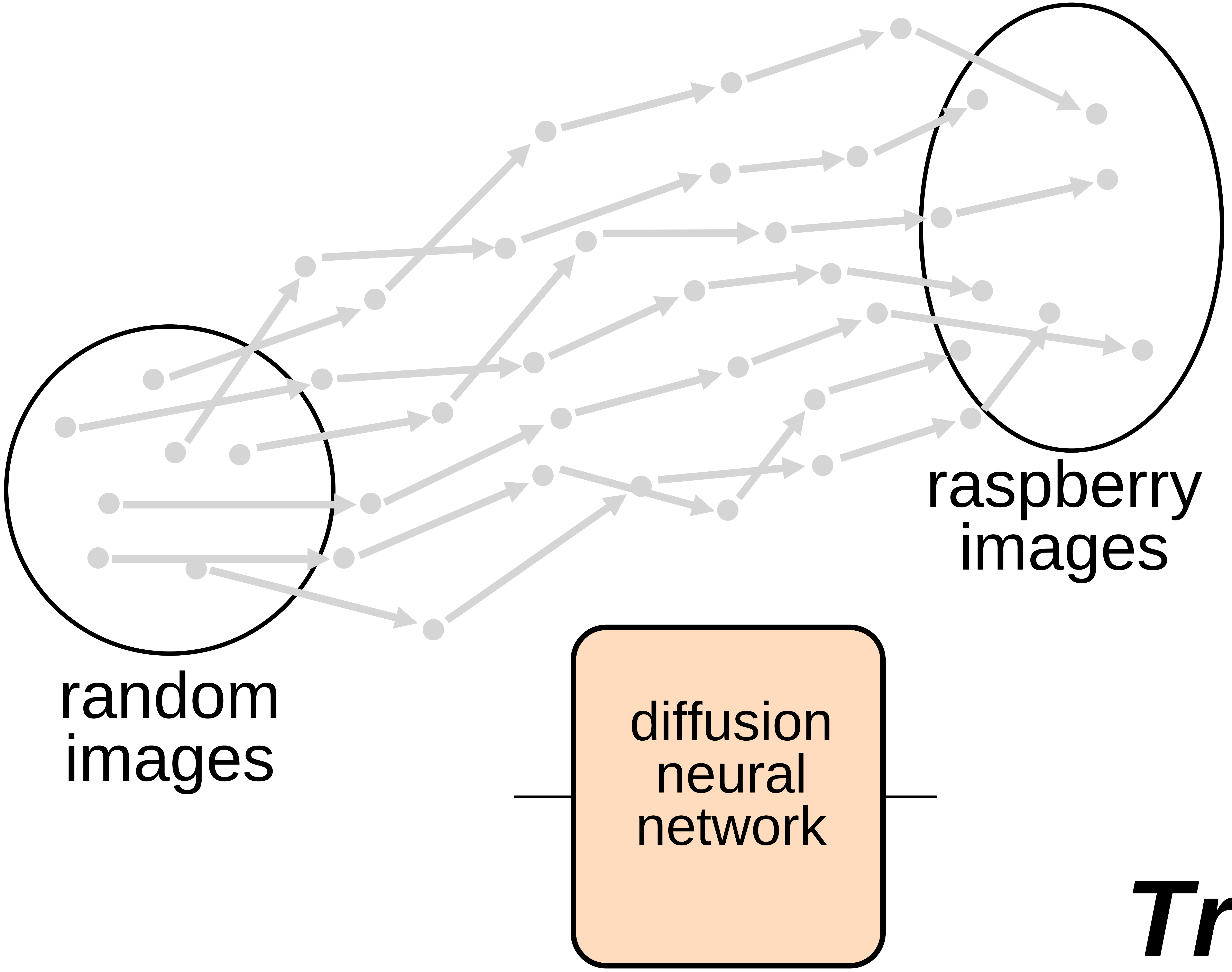
Training

slide from Steve Seitz’s video



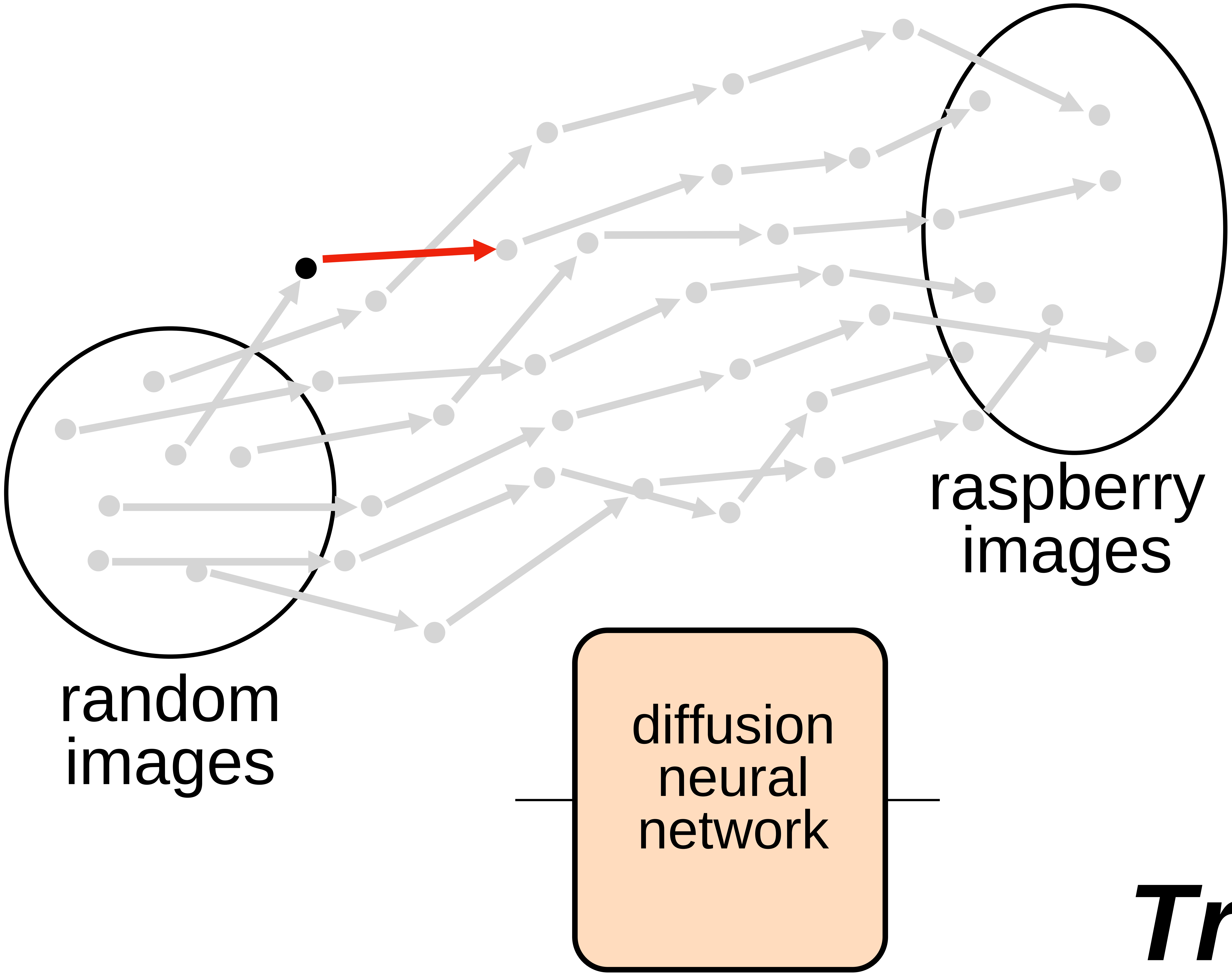
Training

slide from Steve Seitz's video



Training

slide from Steve Seitz's video



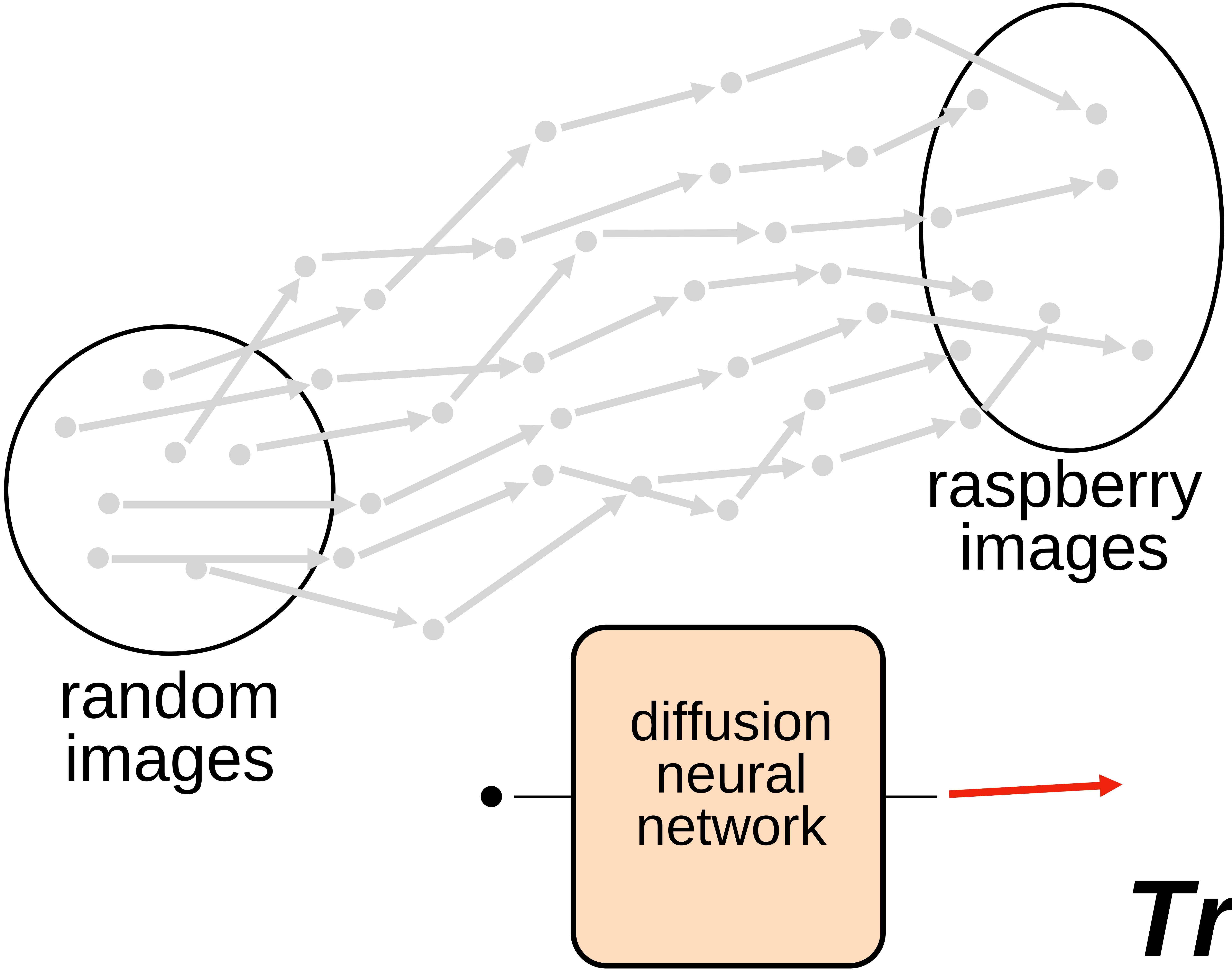
random images

raspberry images

diffusion neural network

Training

slide from Steve Seitz's video



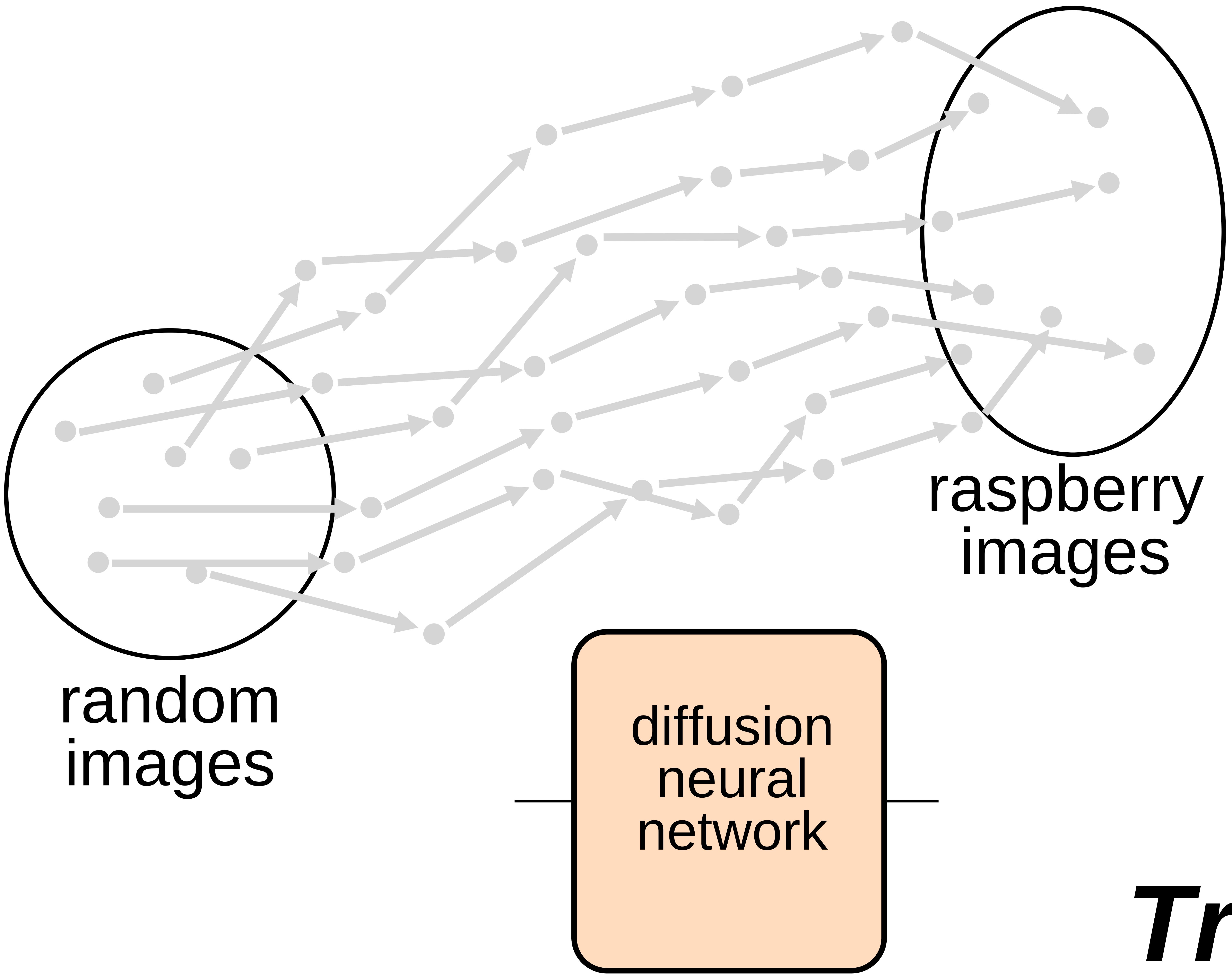
random images

raspberry images

diffusion neural network

Training

slide from Steve Seitz’s video



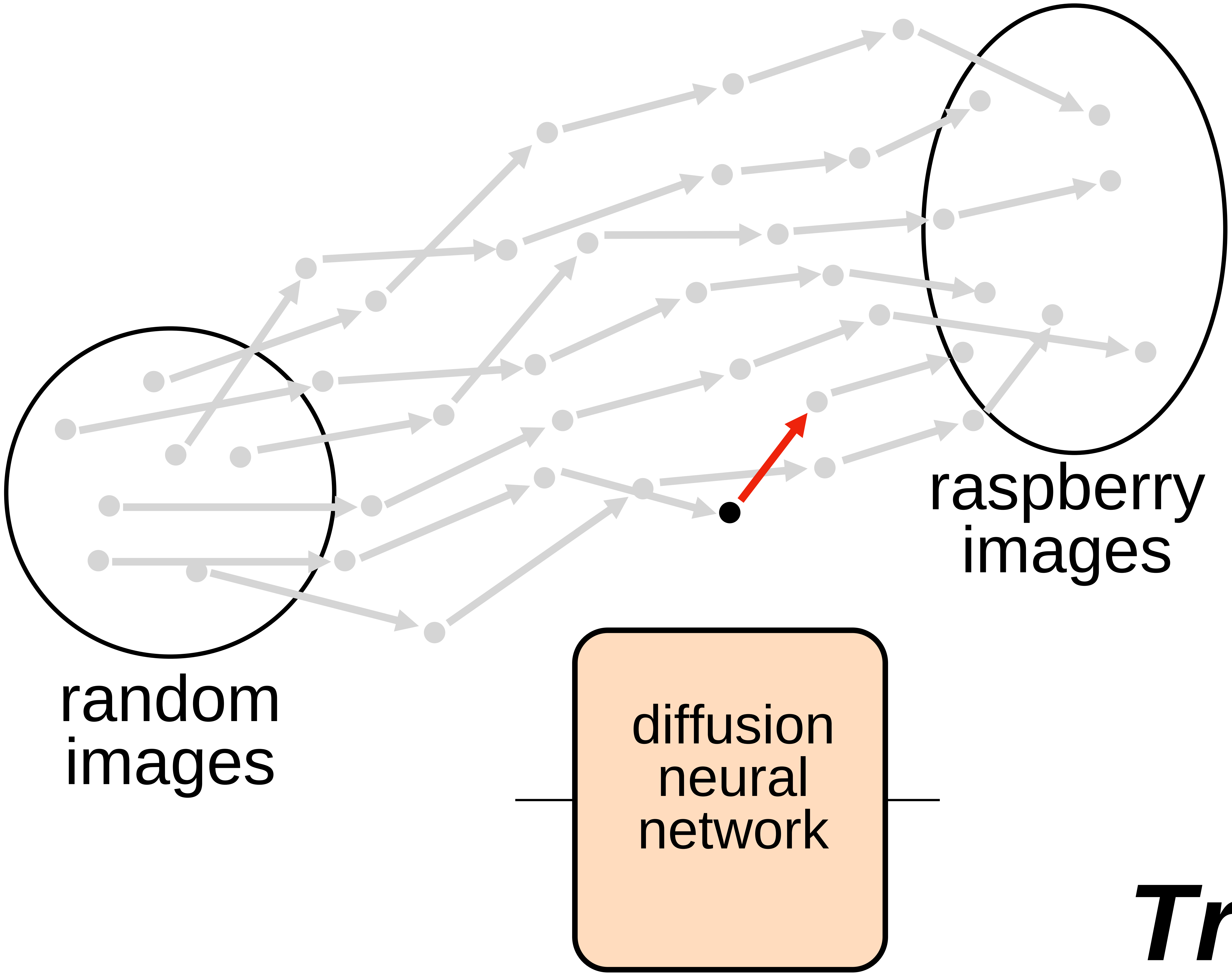
random images

raspberry images

diffusion neural network

Training

slide from Steve Seitz's video



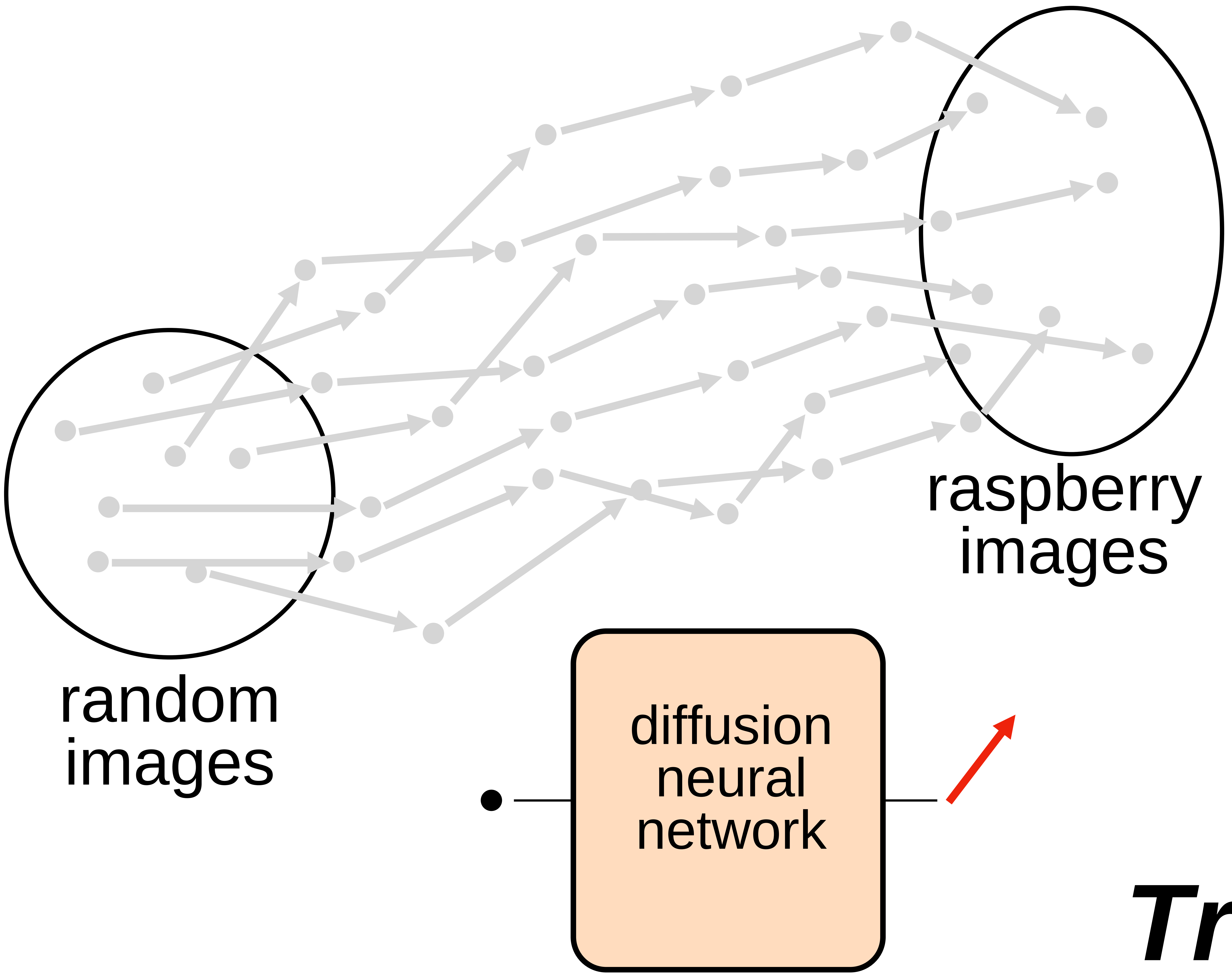
random images

raspberry images

diffusion neural network

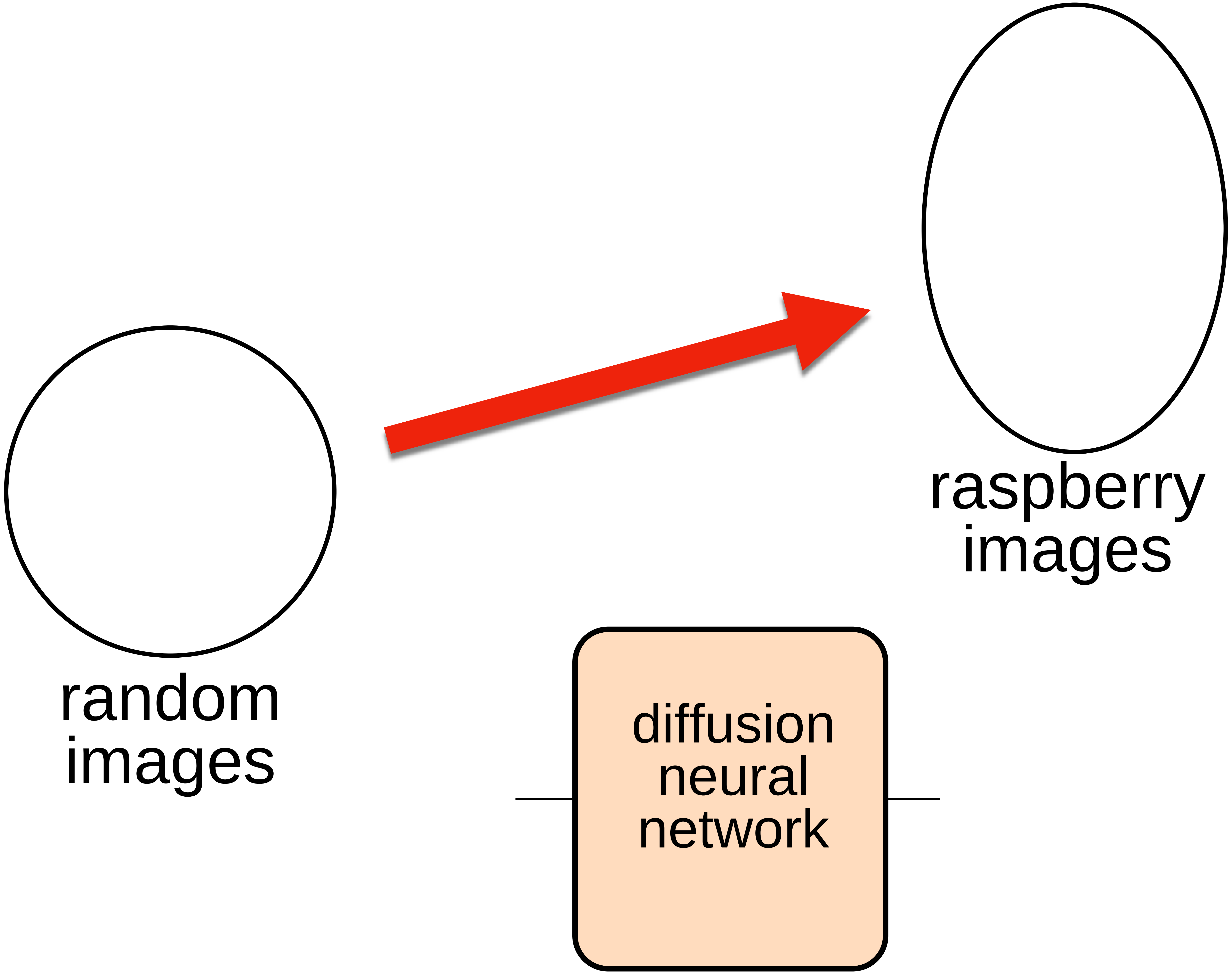
Training

slide from Steve Seitz's video



Training

slide from Steve Seitz's video



random images

raspberry images

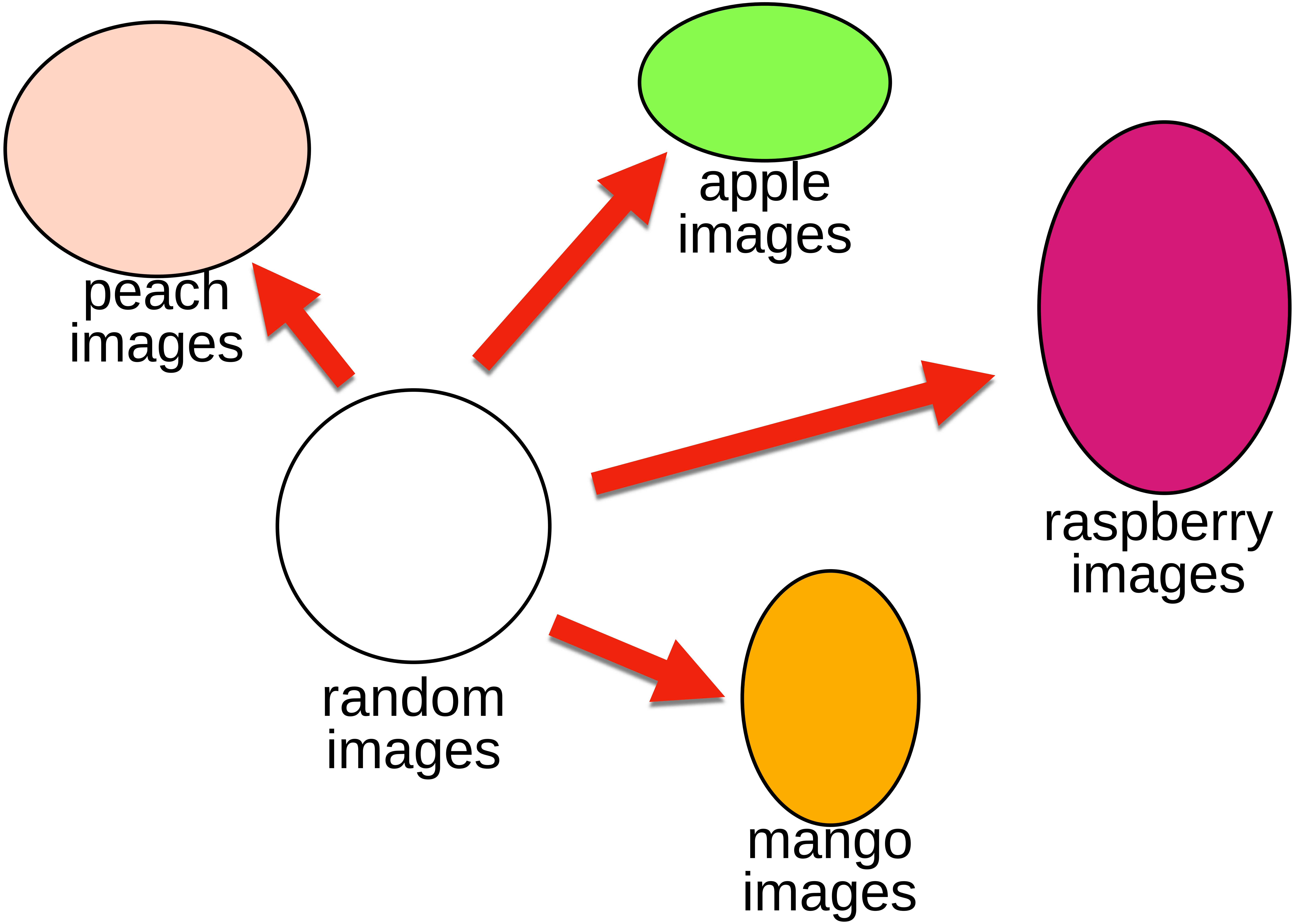
diffusion neural network

slide from Steve Seitz's [video]

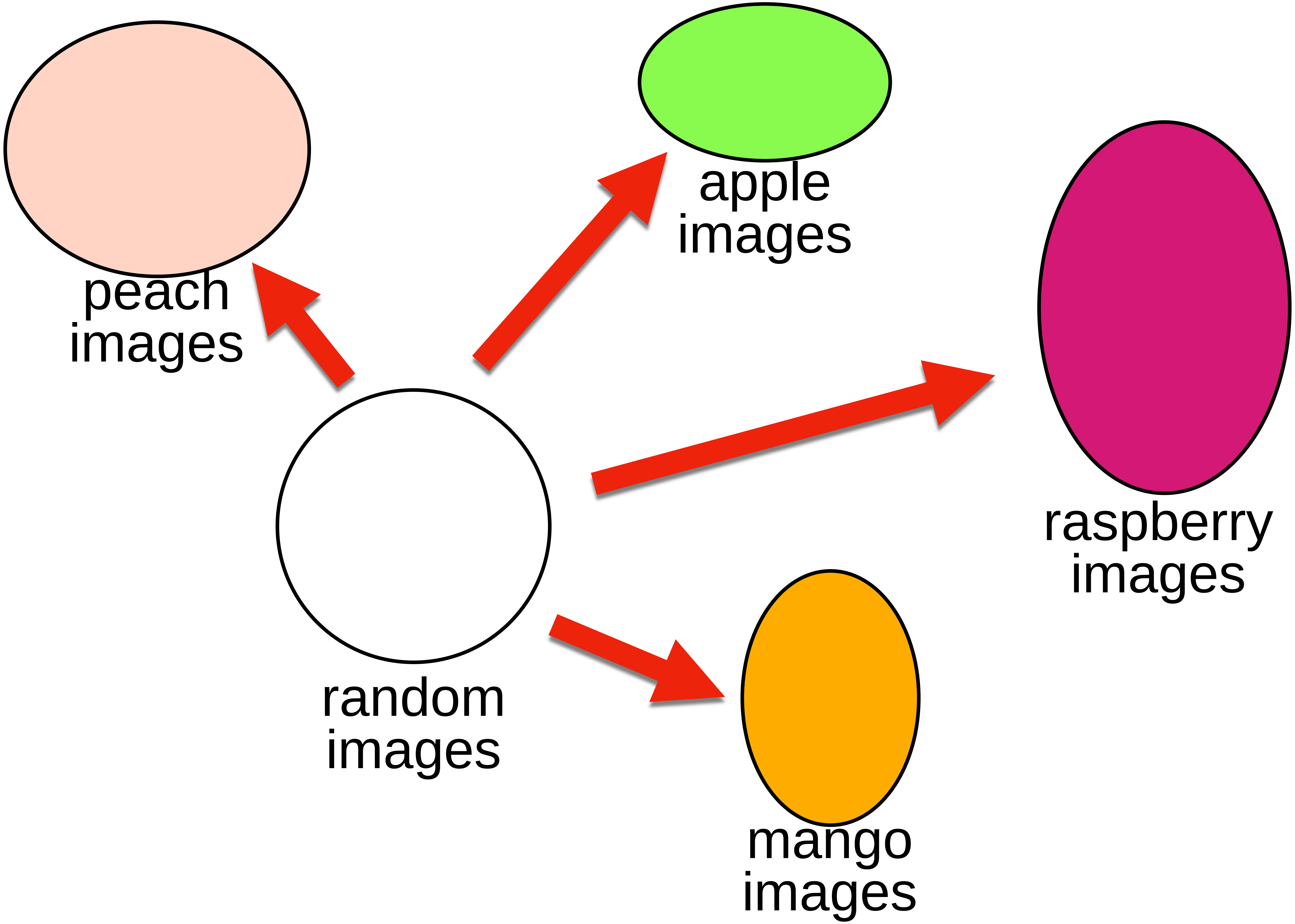
Curious property of Diffusion

- We are training the model to reconstruct the training set
  - But it fails!
  - Instead, it generate novel images
  - Which is what makes it great

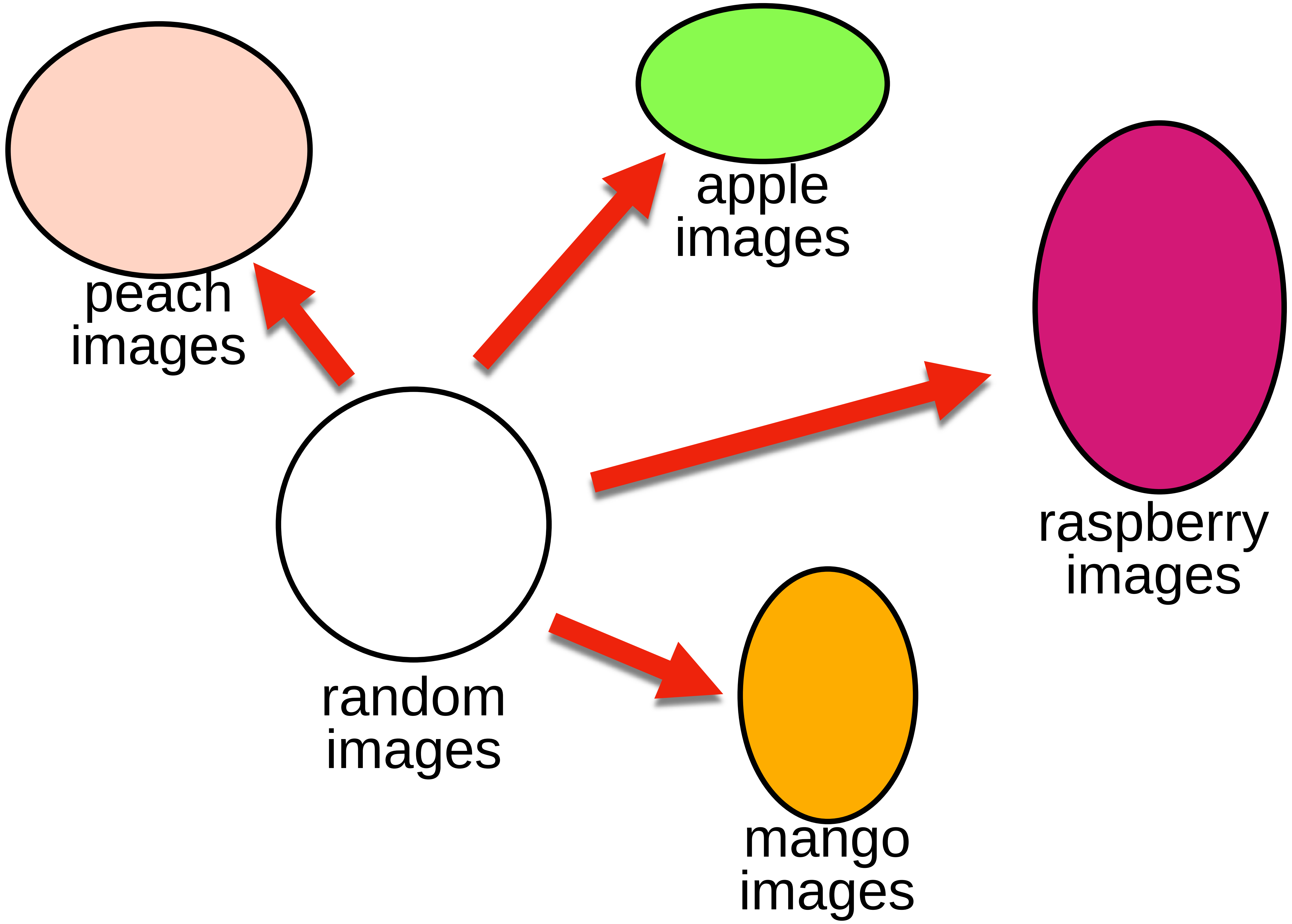
- Perhaps it models images as textures
  - Keeping important correlations and throwing away the rest
  - But we don’t know the “model space” of these textures

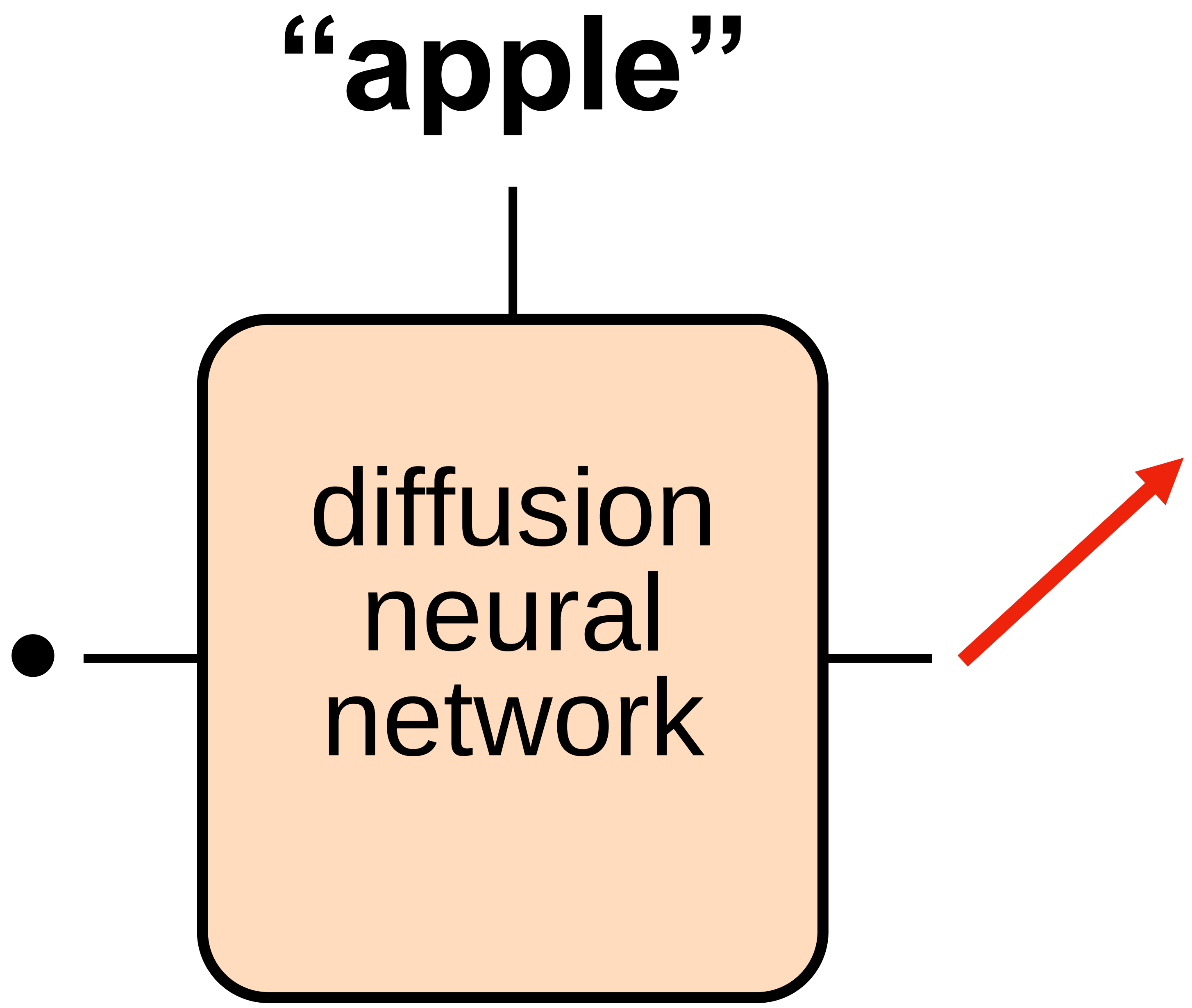


slide from Steve Seitz's video

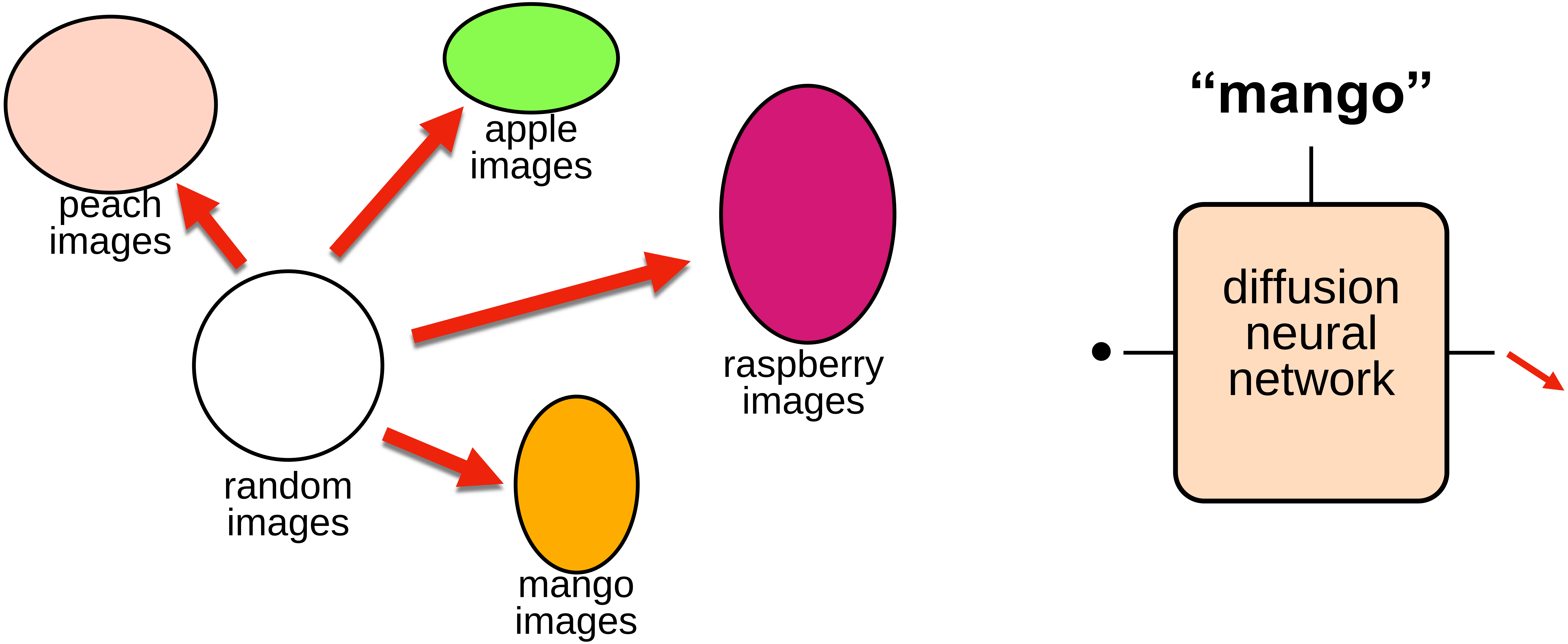


slide from Steve Seitz's video

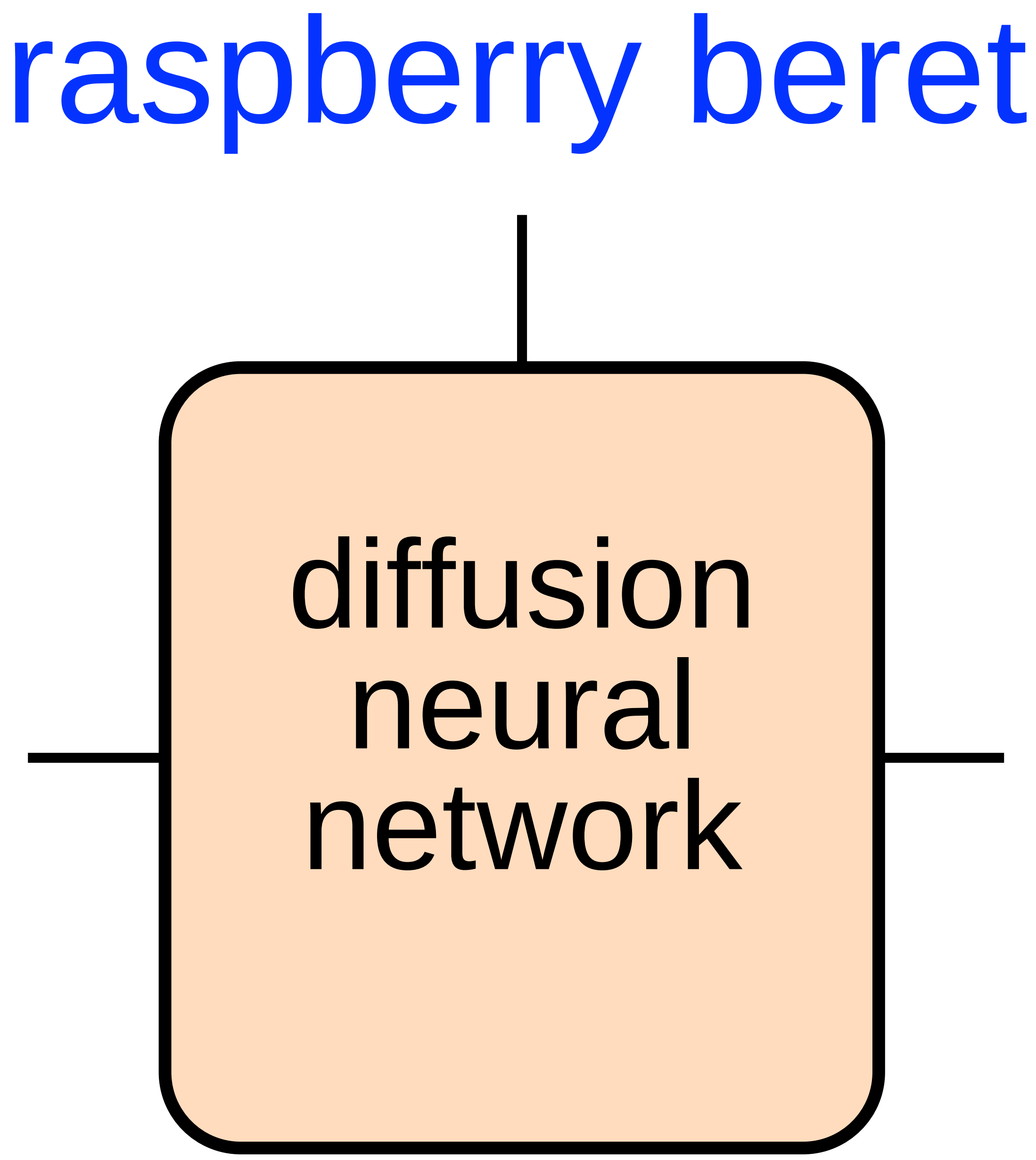




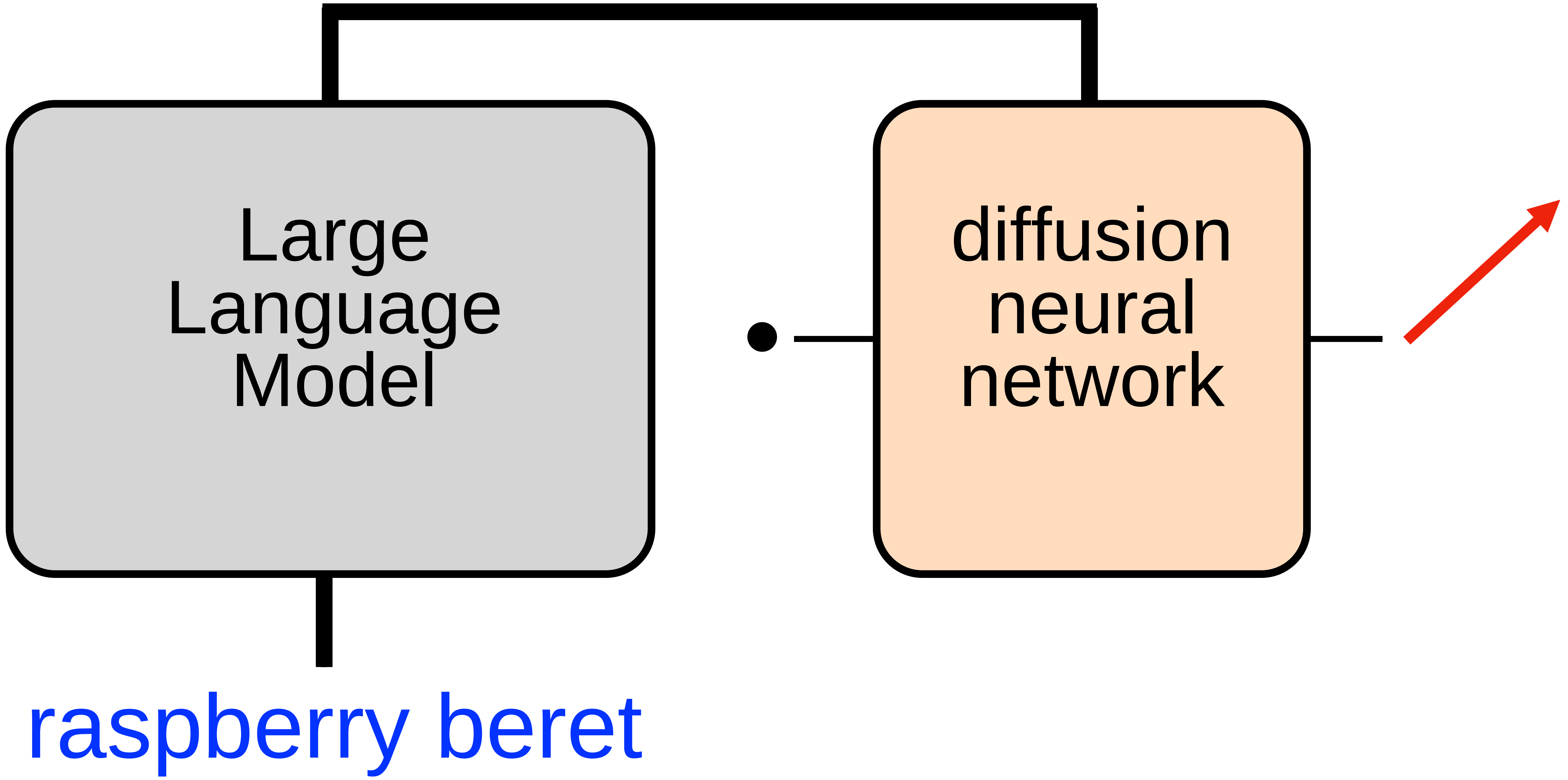
slide from Steve Seitz's video



slide from Steve Seitz's video



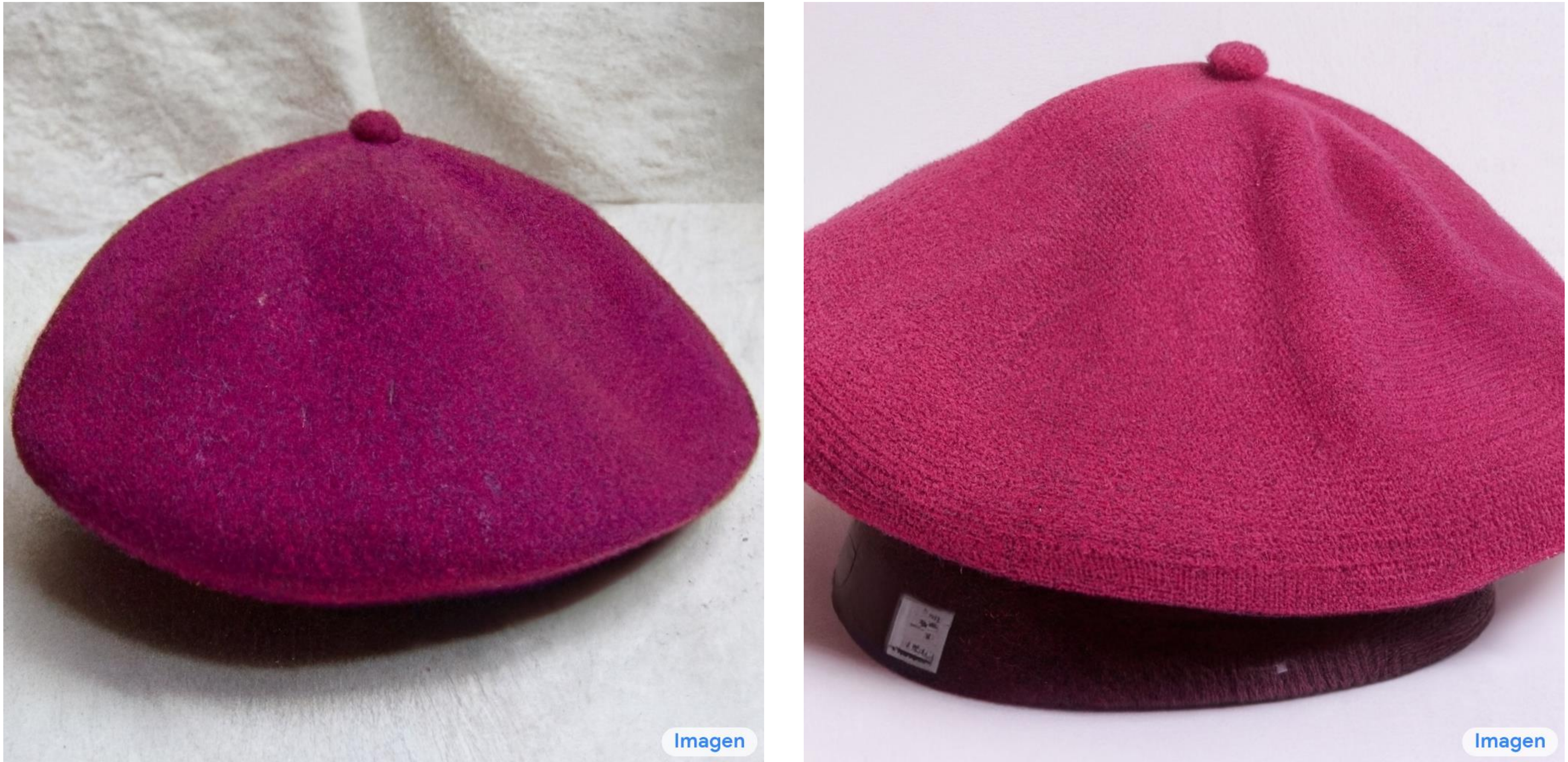
slide from Steve Seitz's video



slide from Steve Seitz's video

raspberry beret

slide from Steve Seitz's video

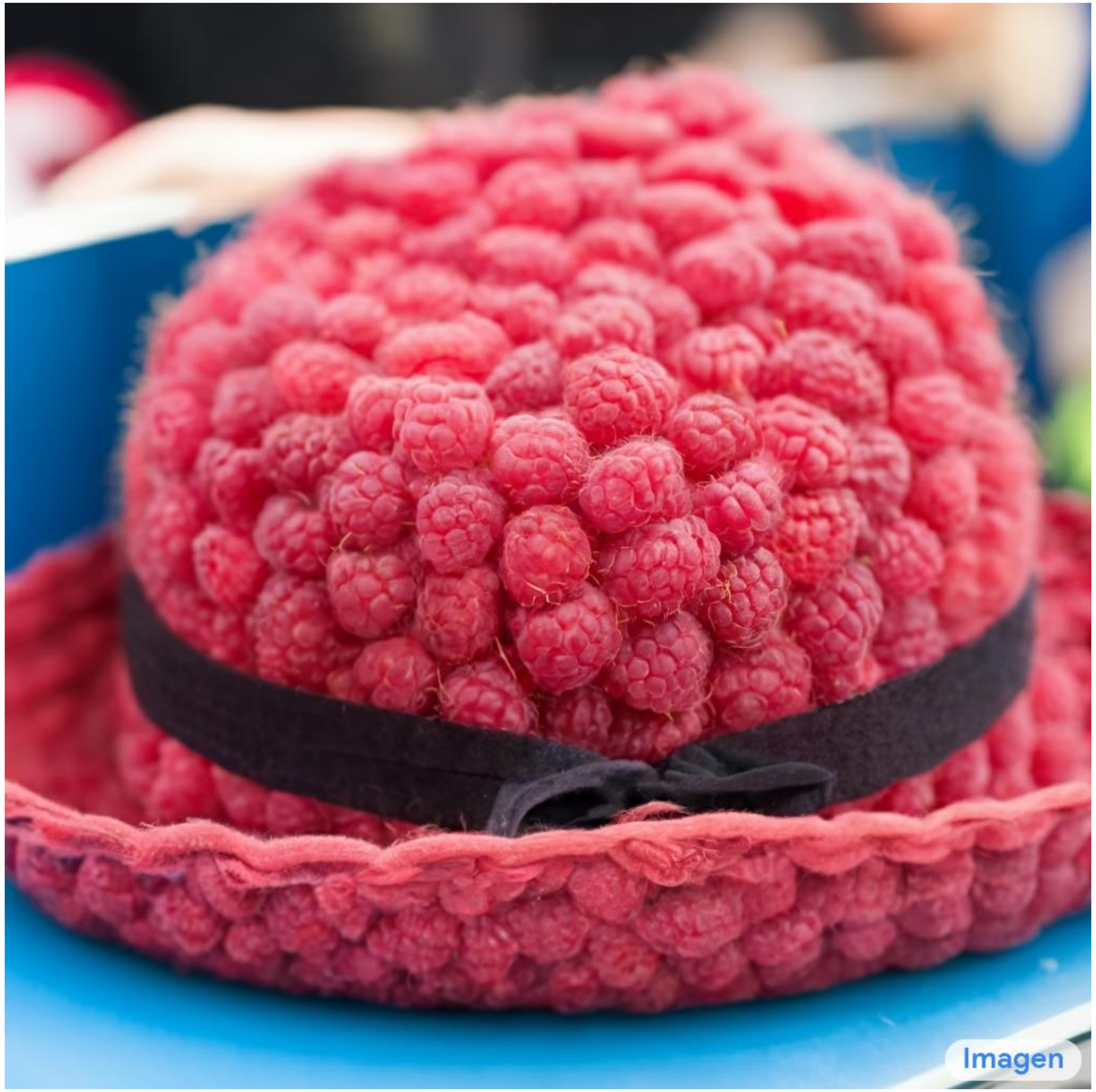


raspberry beret

slide from Steve Seitz's video

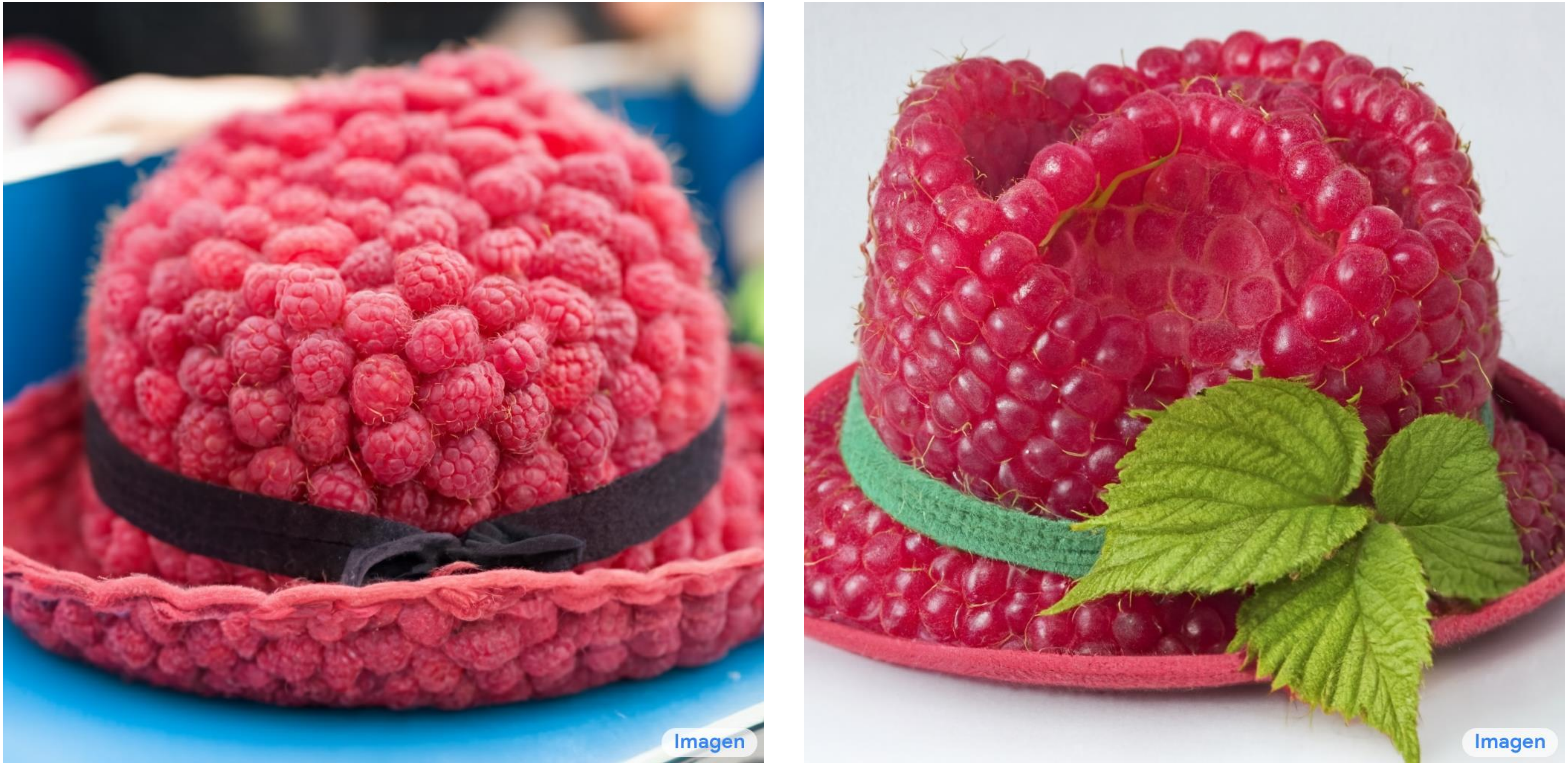
beret of raspberries

slide from Steve Seitz's video



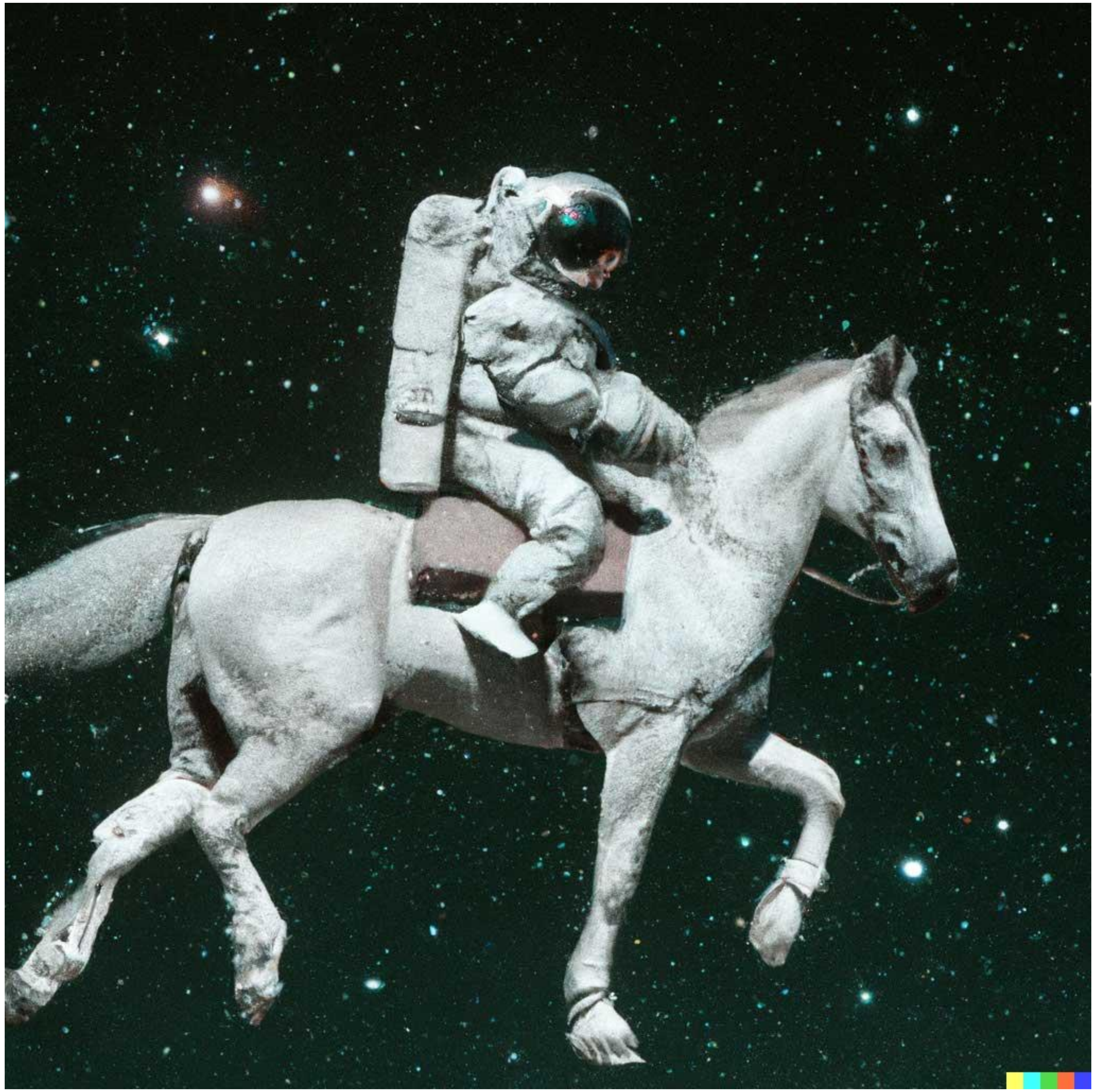
beret of raspberries

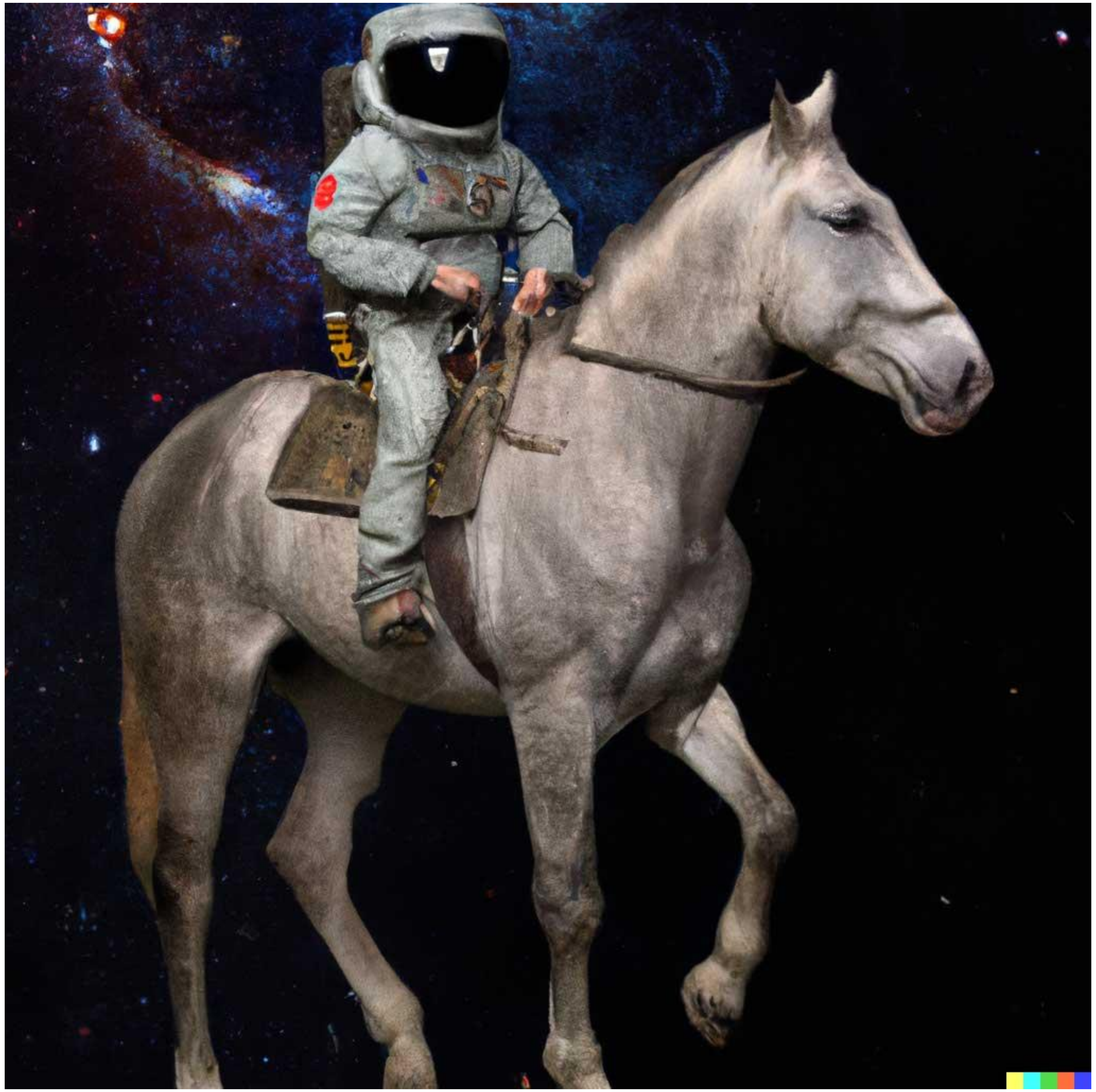
slide from Steve Seitz’s video



beret of raspberries

slide from Steve Seitz's video

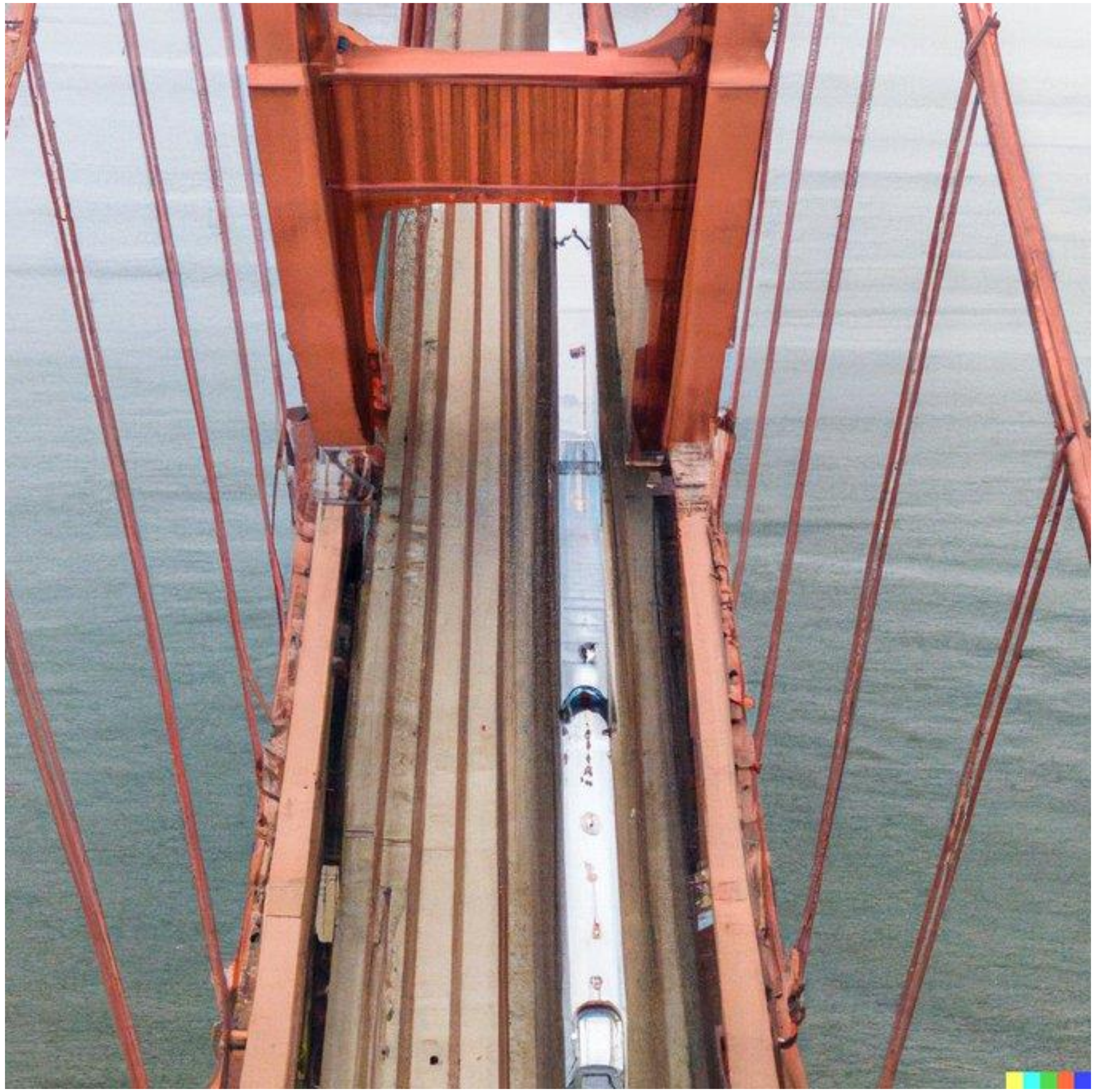


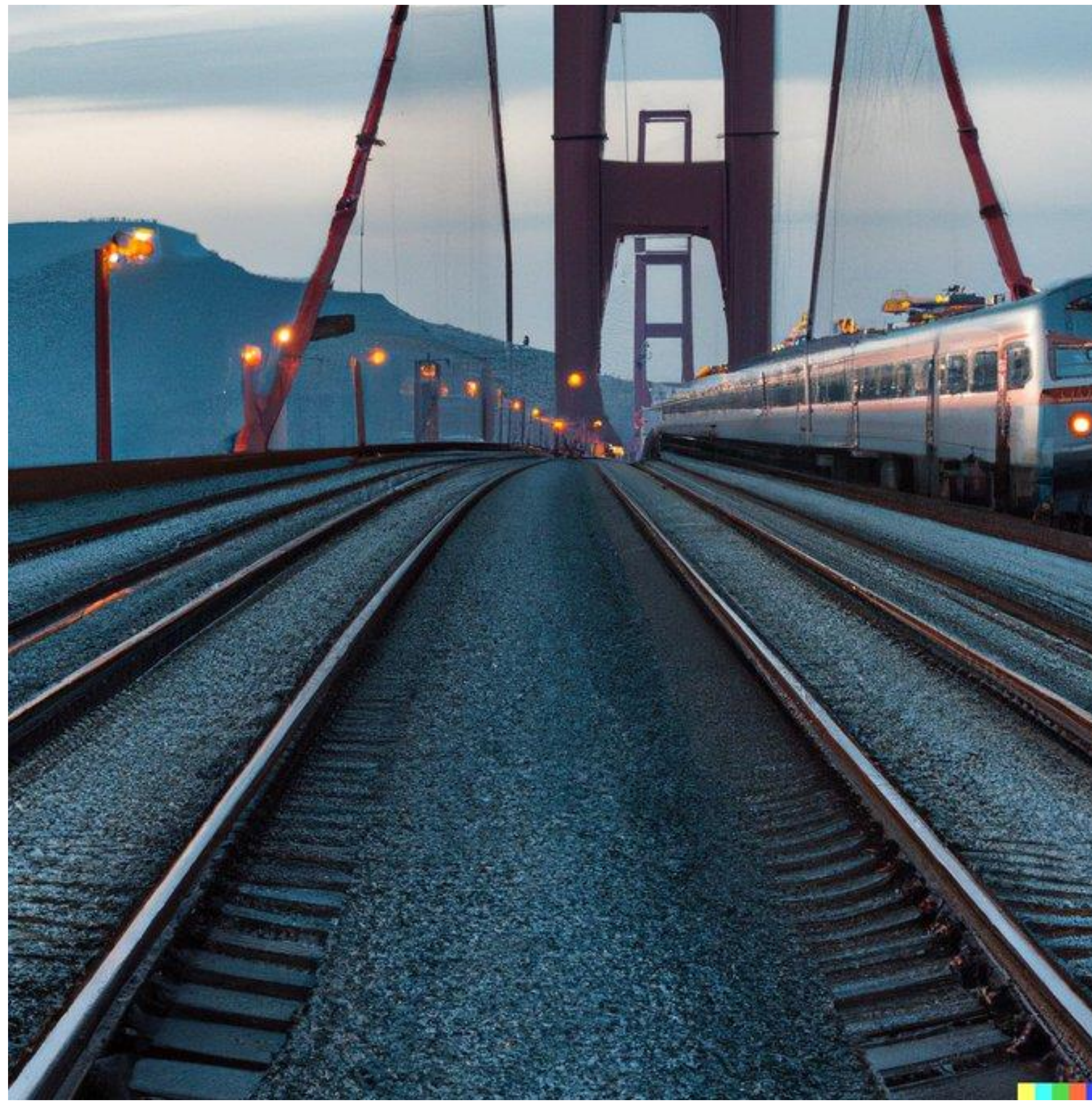


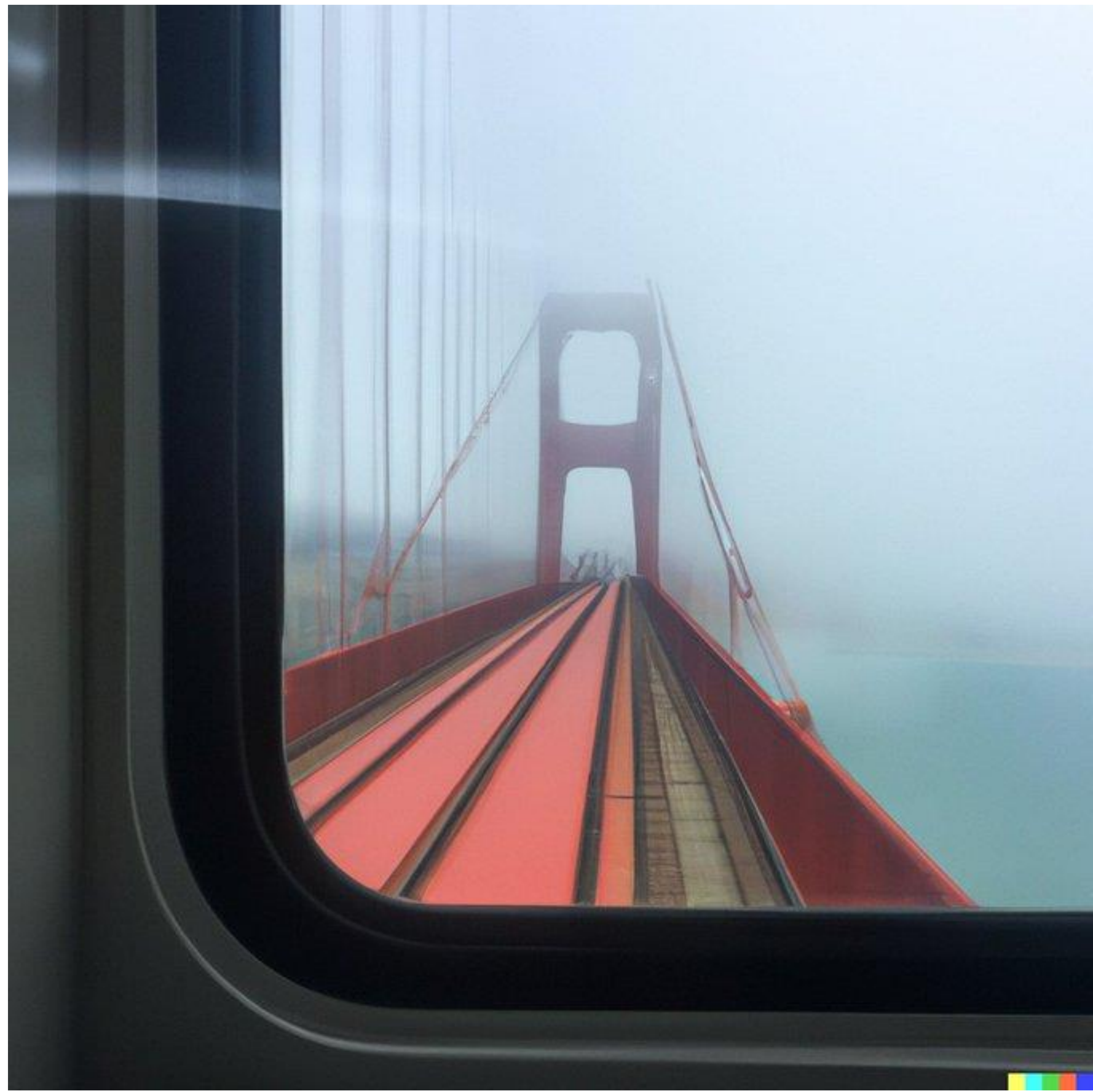
An astronaut riding a horse in a photorealistic style (Dall-E 2)

slide from Steve Seitz's video

Impressive compositionality:



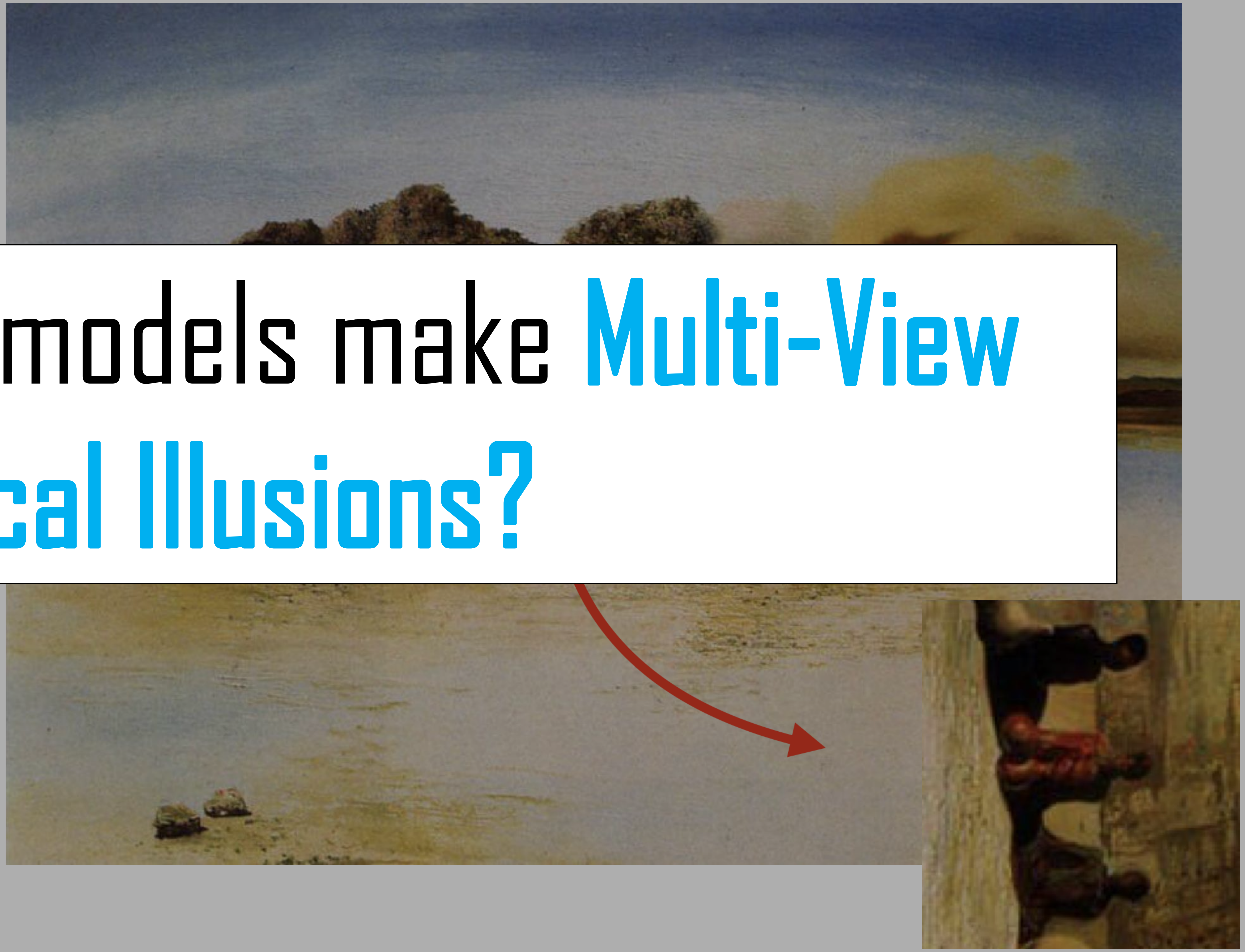




DALL-E + Danielle Baskin

Project 5!

Can generative models make Multi-View Optical Illusions?



Salvador Dalí,
Paranoiac Face. 1937.

Visual Anagrams: Generating Multi-View Optical Illusions with Diffusion Models

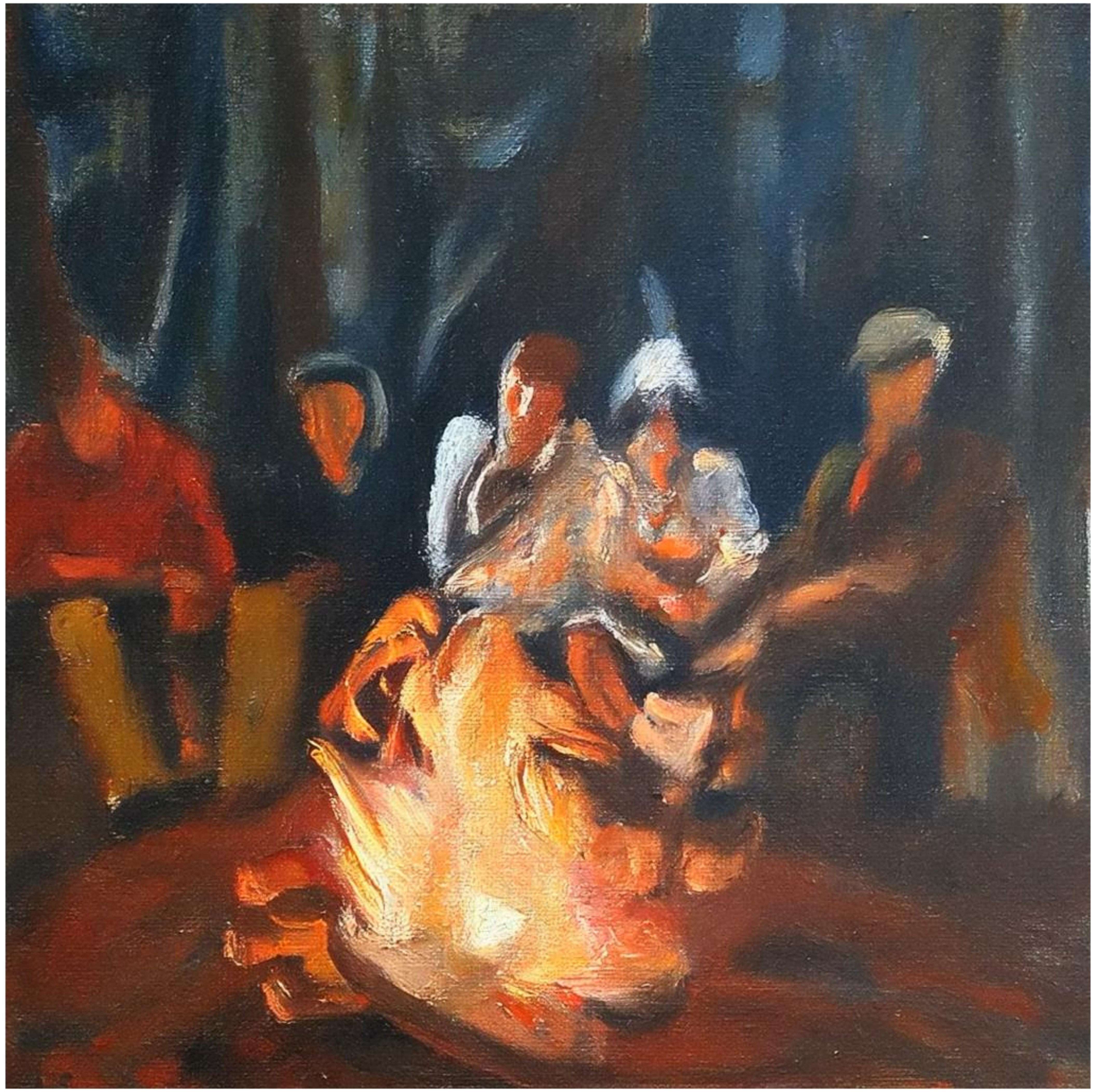
Daniel Geng Aaron Park Andrew Owens

University of Michigan

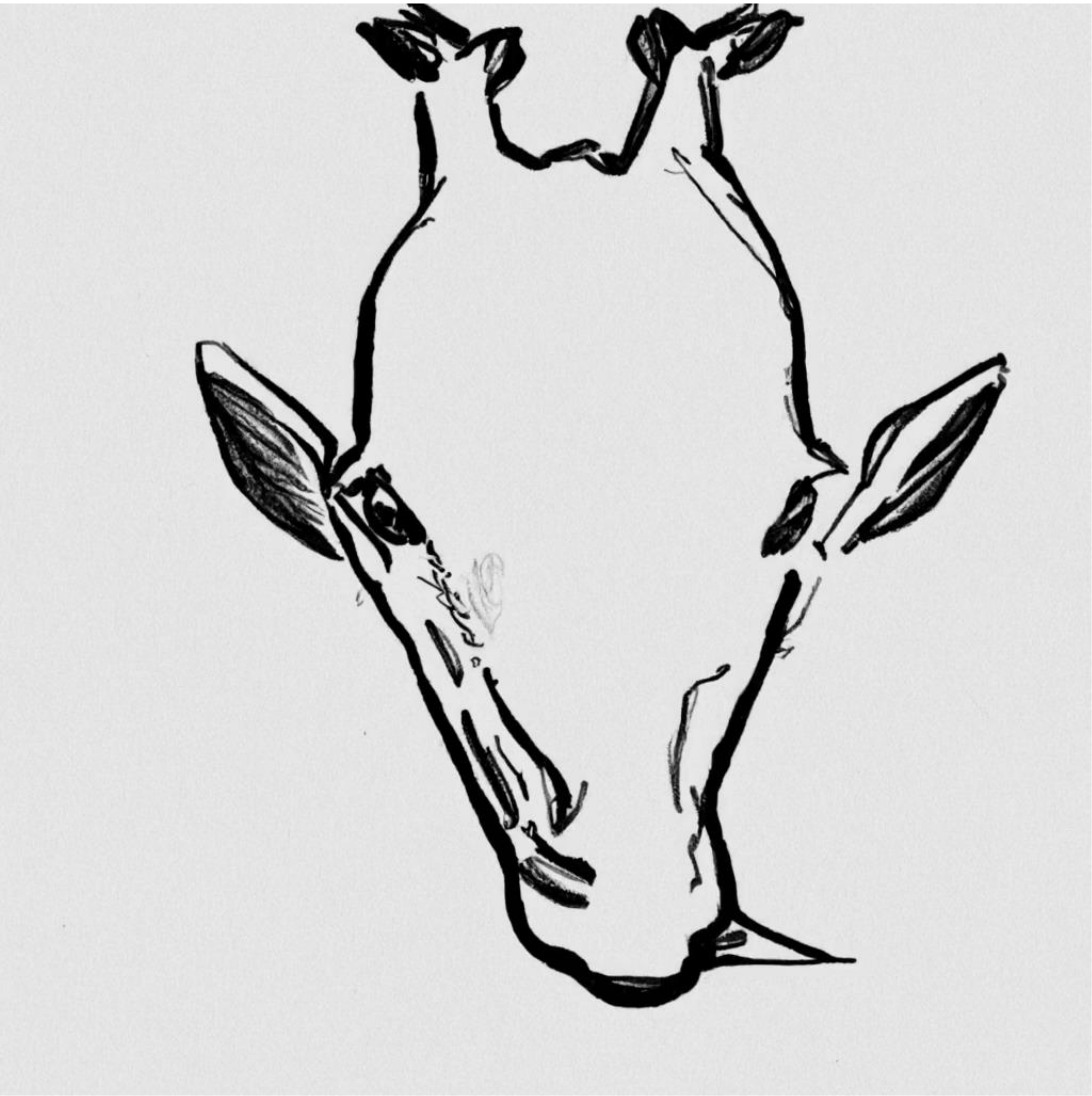
CVPR 2024

https://dangeng.github.io/visual_anagrams/

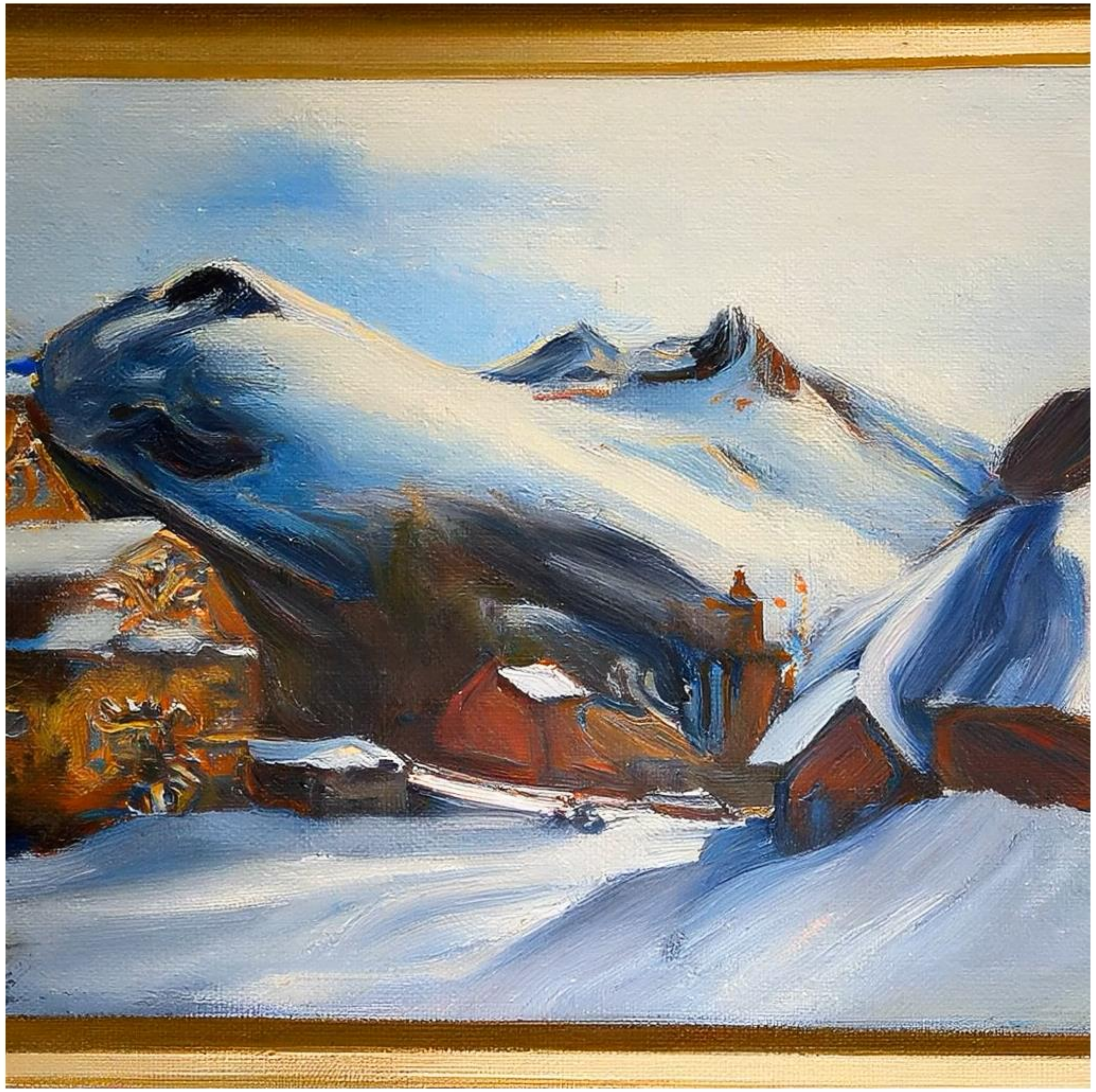
a watercolor of mount rainier



an oil painting of people
around a campfire



a drawing of a giraffe

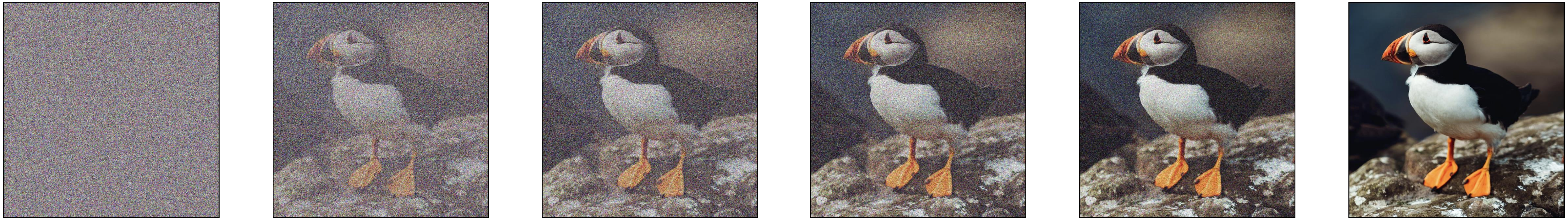


an oil painting of a
snowy mountain village

Diffusion Models

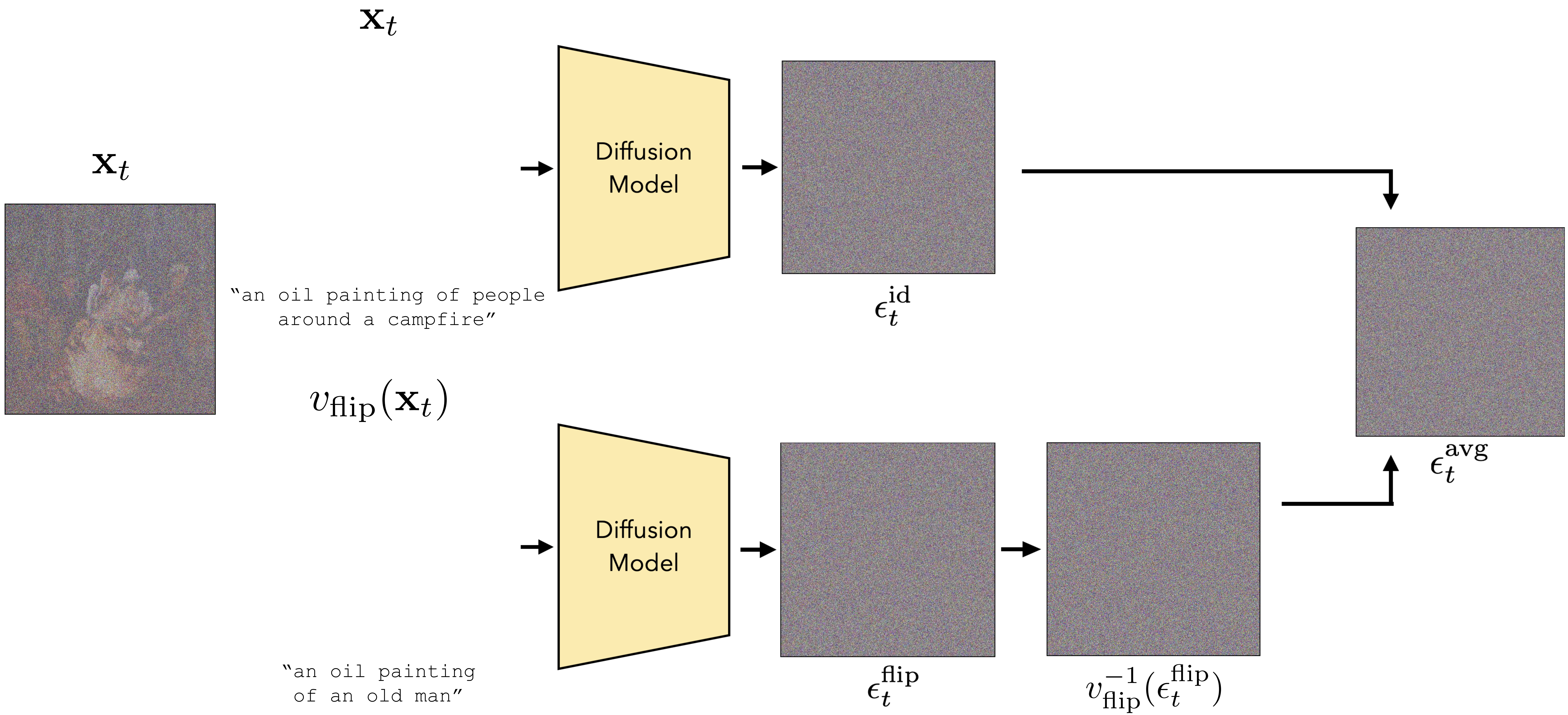
$\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$

$\mathbf{x}_0$

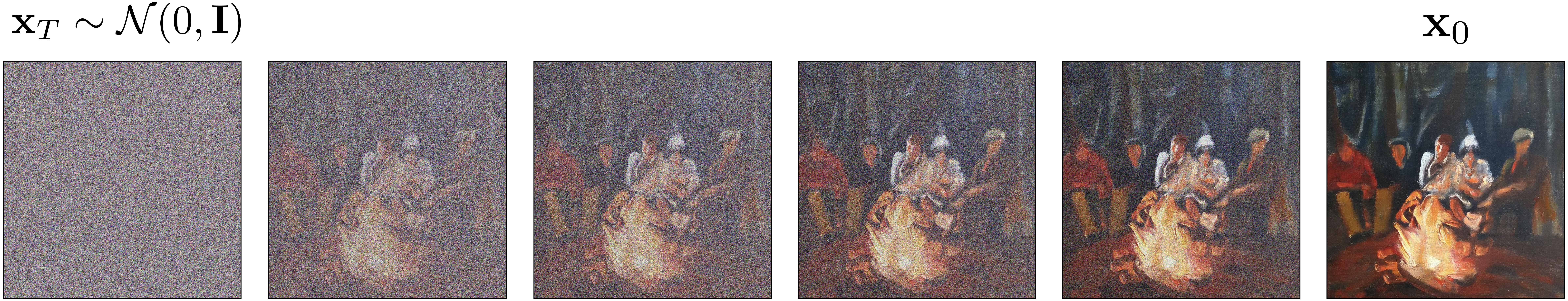


“a photo of a puffin”

Method


$\mathbf{x}_t$
$\mathbf{x}_t$
"an oil painting of people around a campfire"
$v_{\text{flip}}(\mathbf{x}_t)$
"an oil painting of an old man"
Diffusion Model
Diffusion Model
ϵ_t^{id}
ϵ_t^{flip}
$v_{\text{flip}}^{-1}(\epsilon_t^{\text{flip}})$
ϵ_t^{avg}

Method


$\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$
$\mathbf{x}_0$