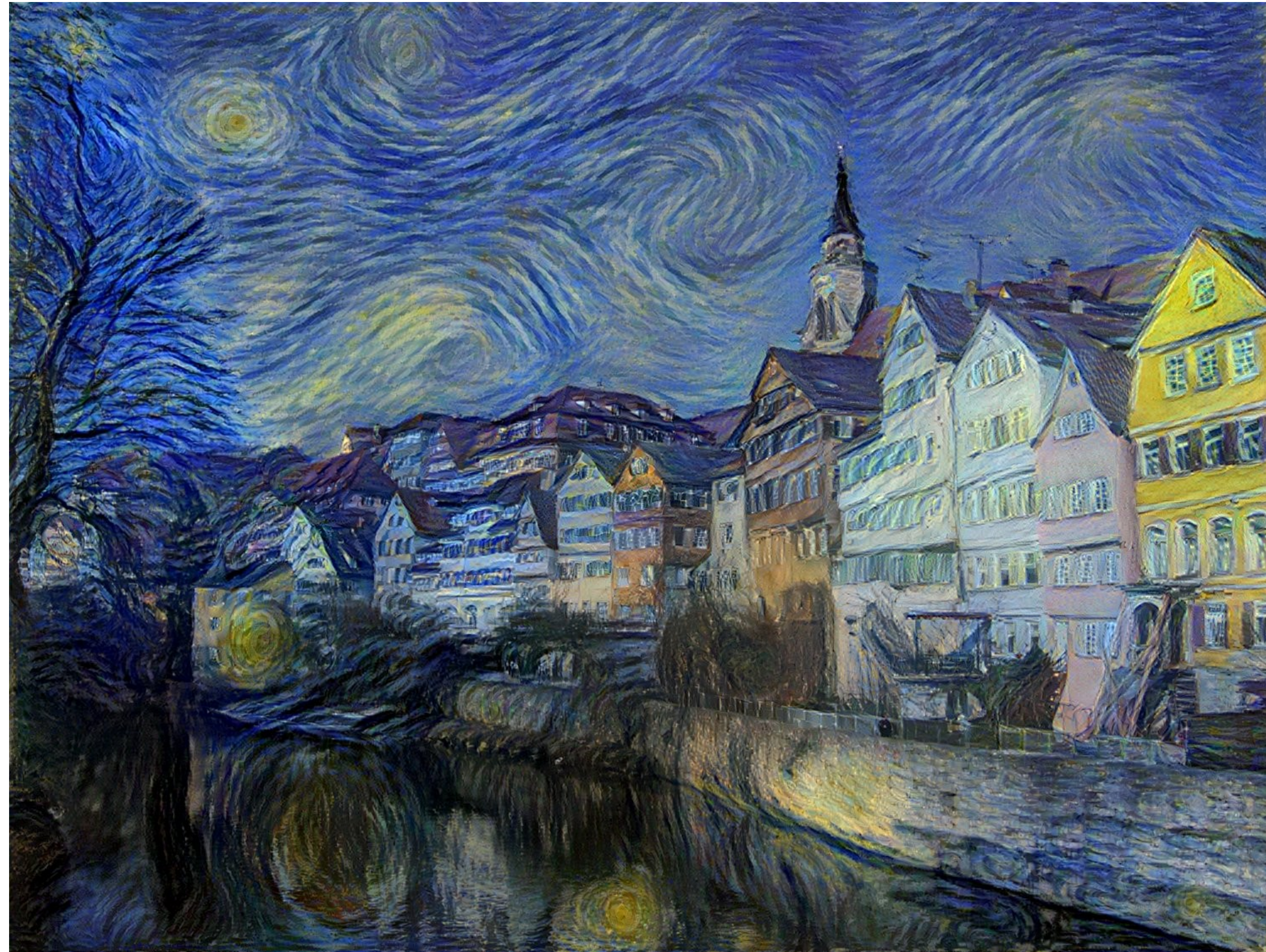


Generative Models of Images



CS180: Intro to Comp. Vision and Comp. Photo
Efros & Ng, UC Berkeley, Fall 2026

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

The Space of All Images

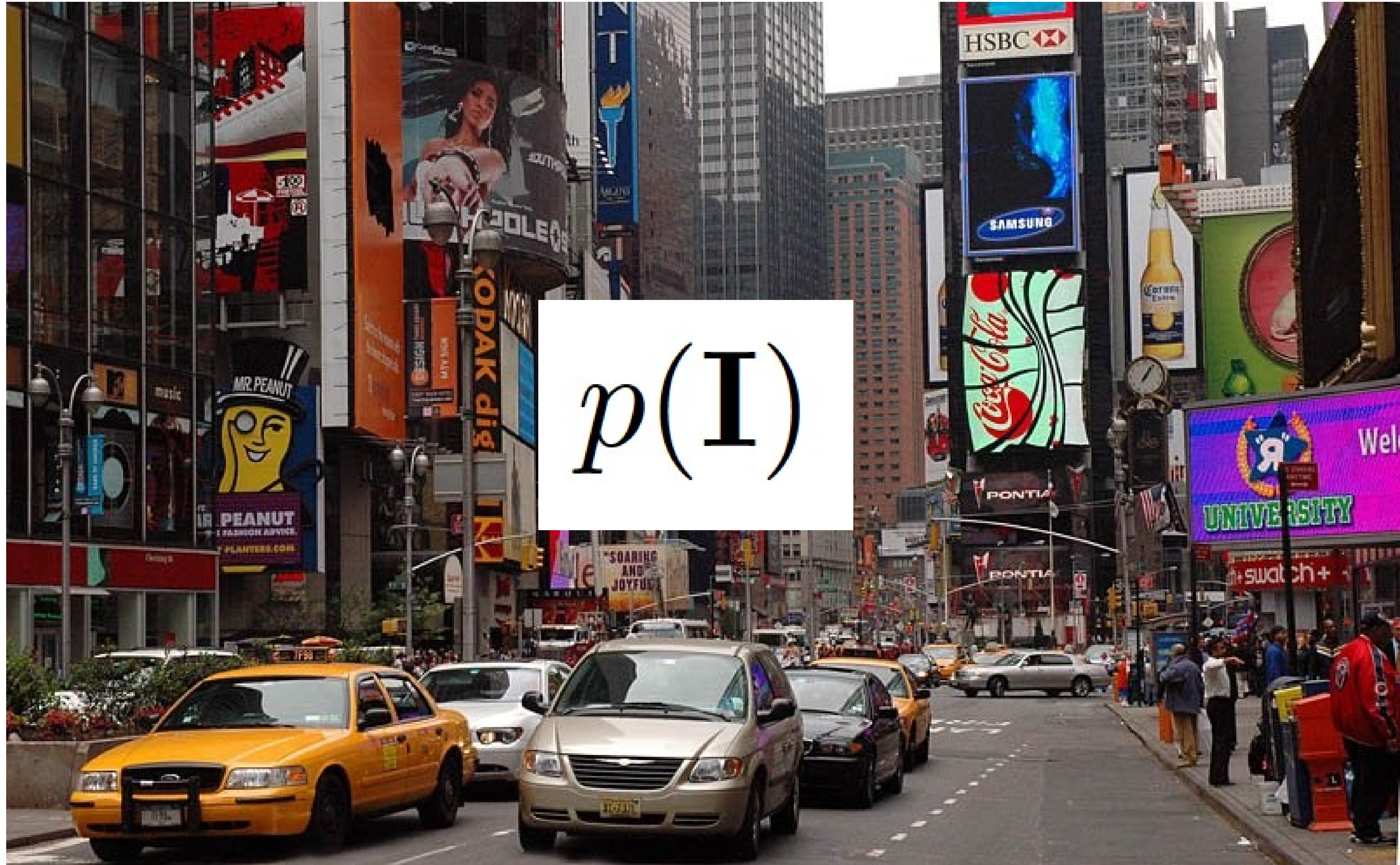
- Lets consider the space of all 100x100 images
- Now lets randomly sample that space...
- Conclusion: Most images are noise



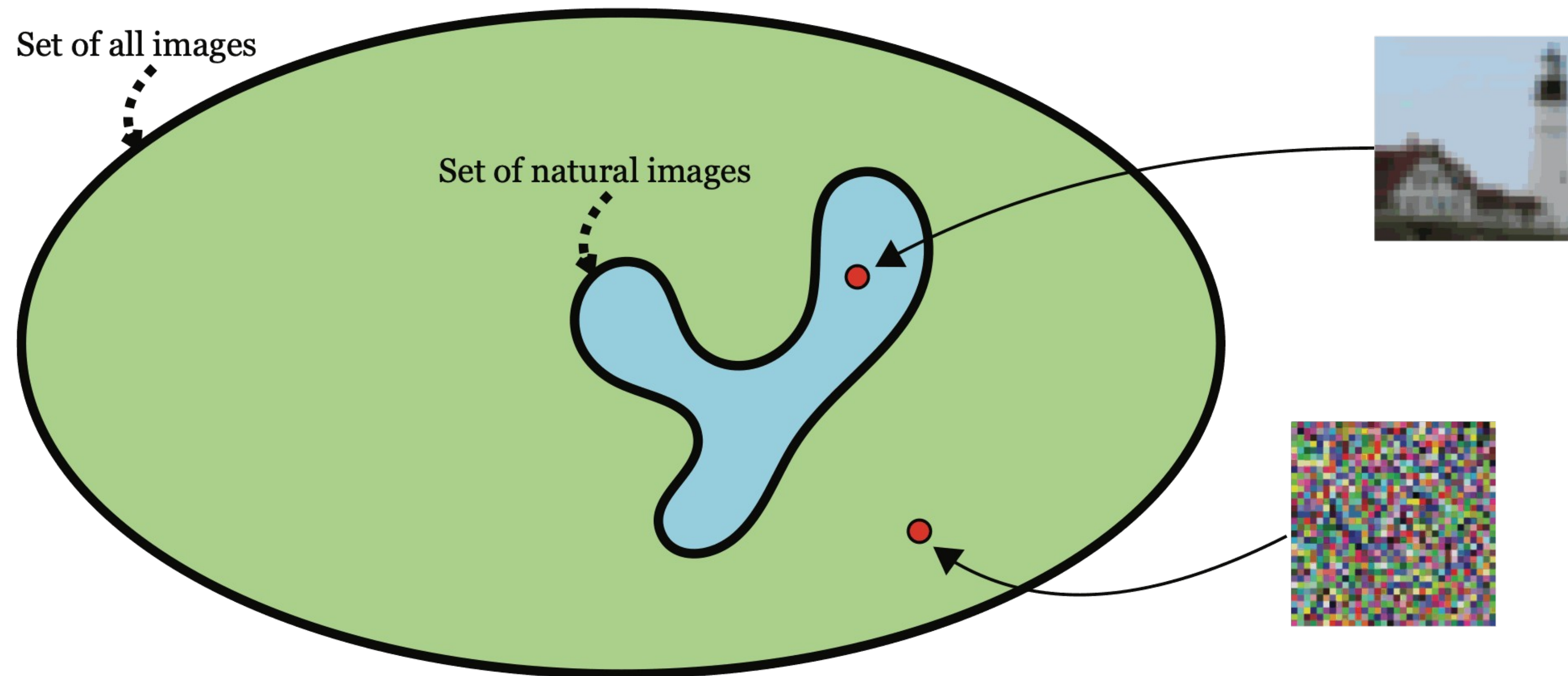
Question:
What do we expect a random uniform sample of all images to look like?

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


Statistical modeling of images

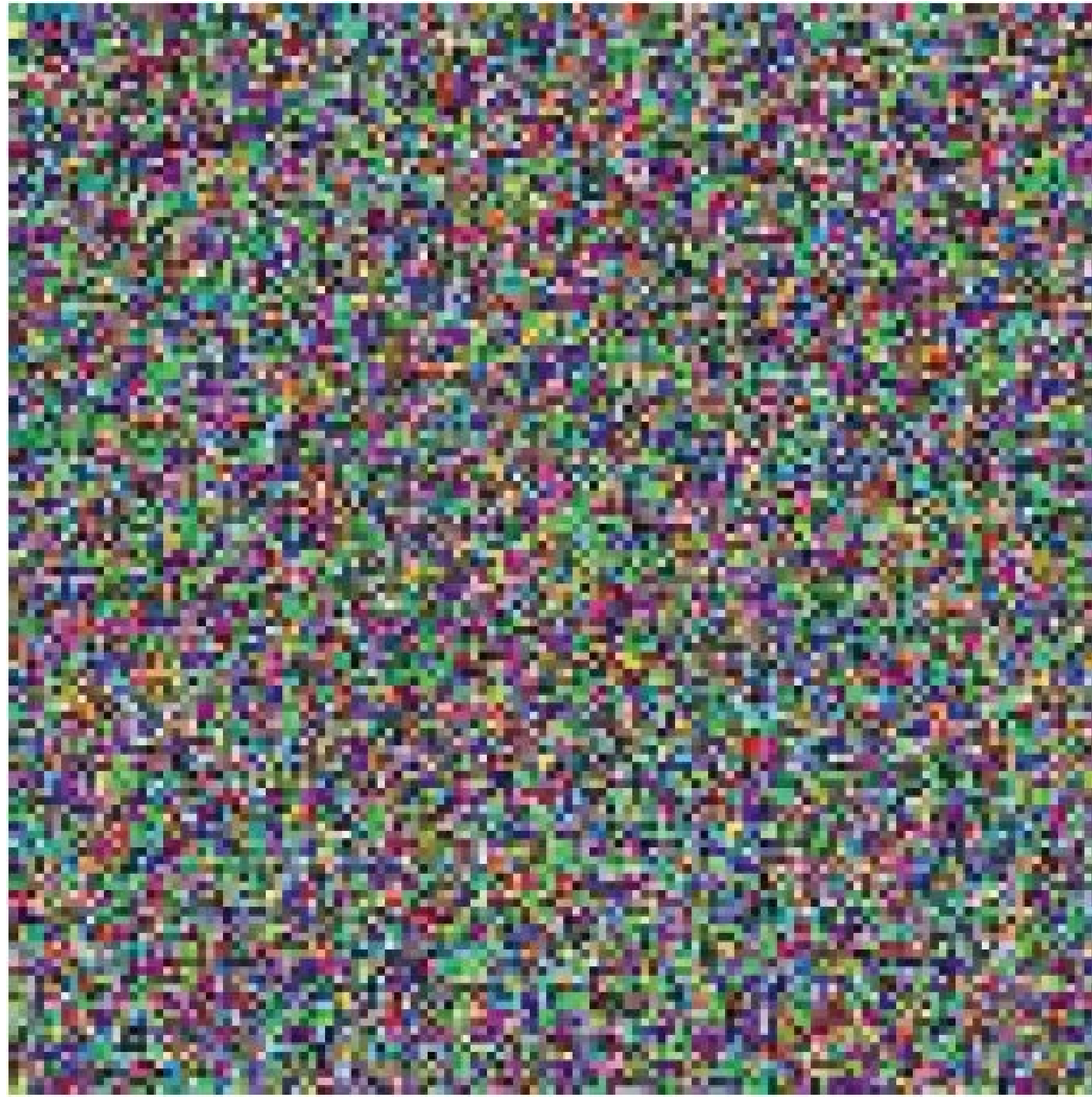


Statistical modeling of images



Visual Worlds

Visual Worlds



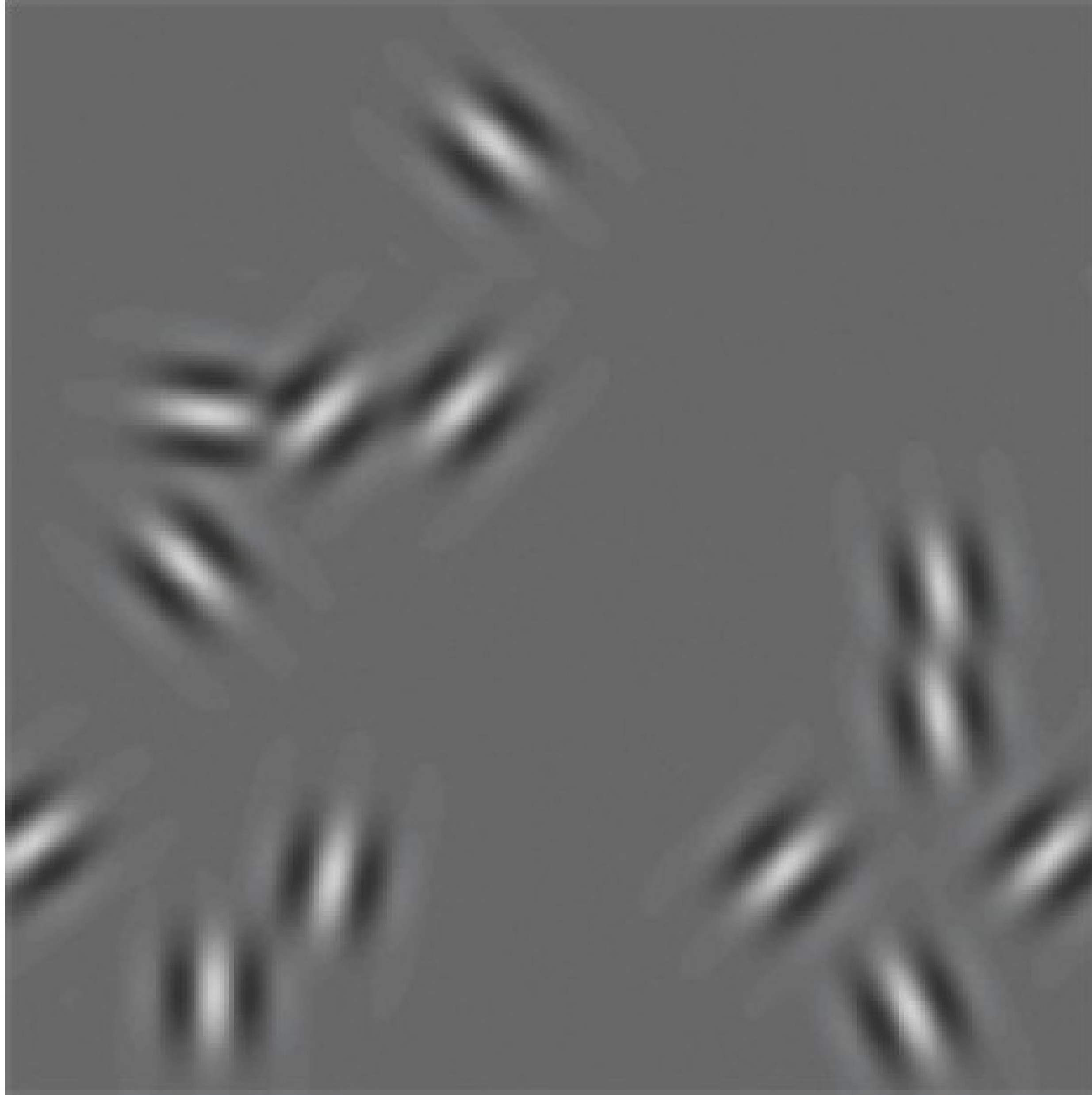
Visual Worlds



Visual Worlds



Visual Worlds



Visual Worlds



Visual Worlds



Visual Worlds

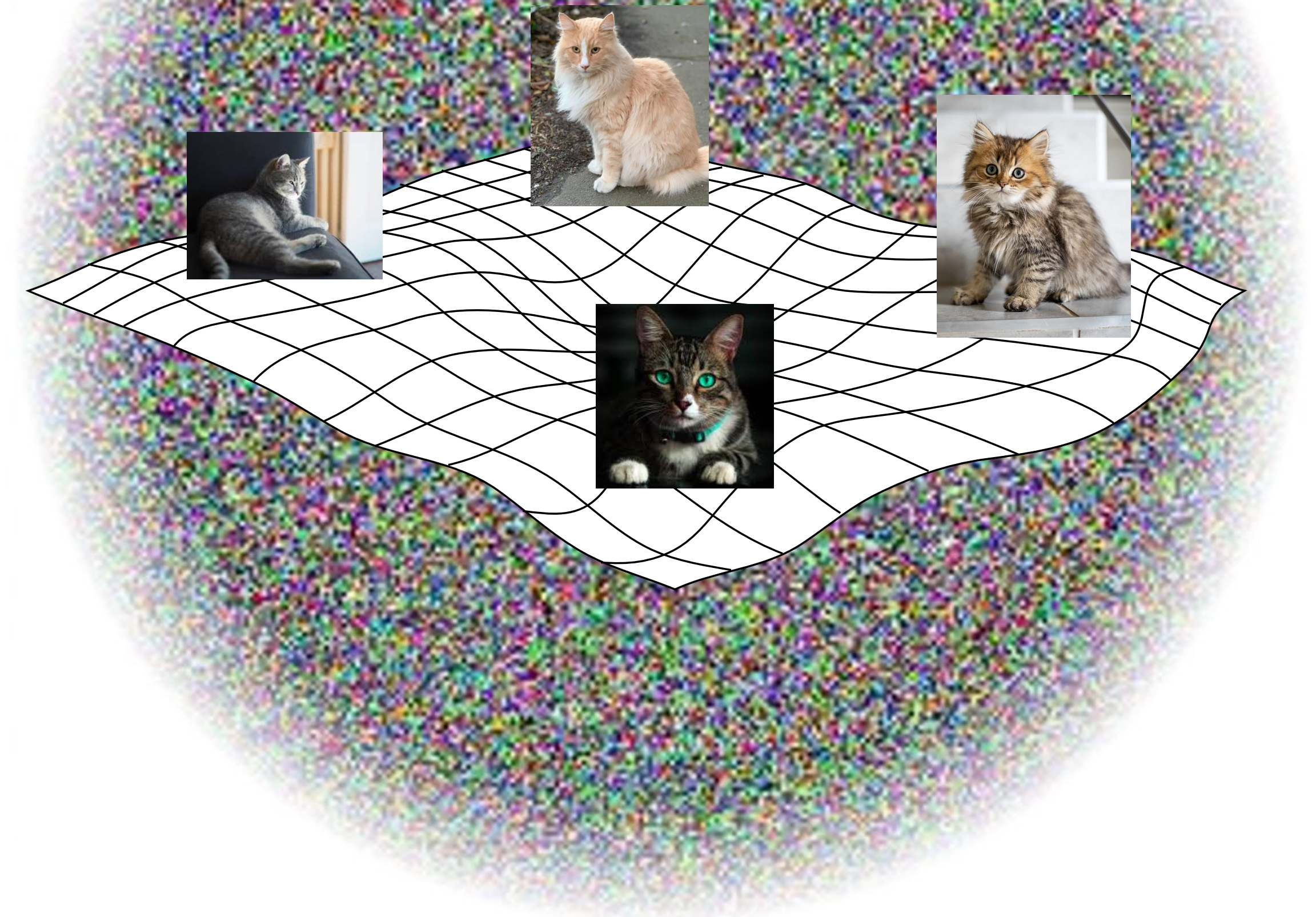


Visual Worlds



Natural Image Manifolds

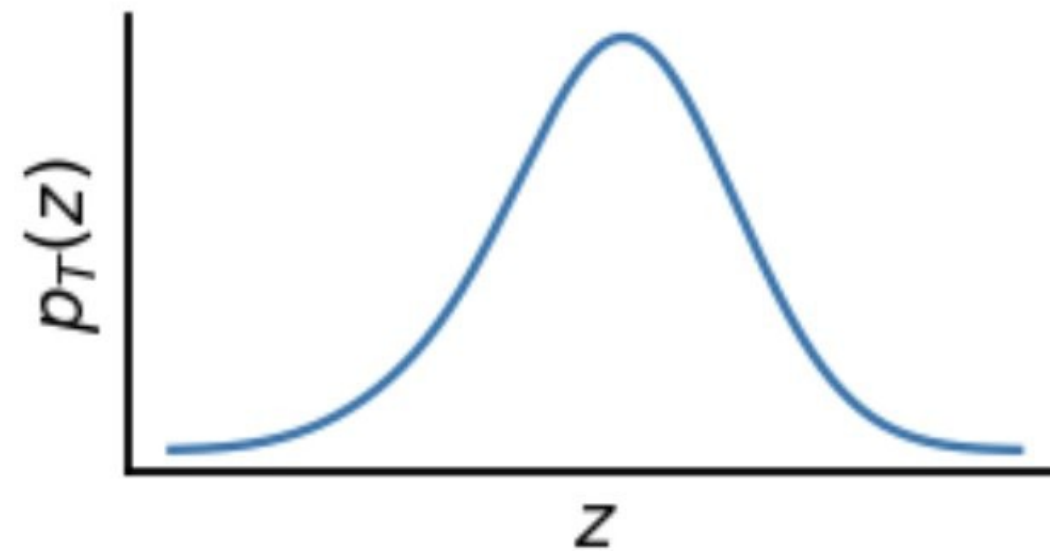
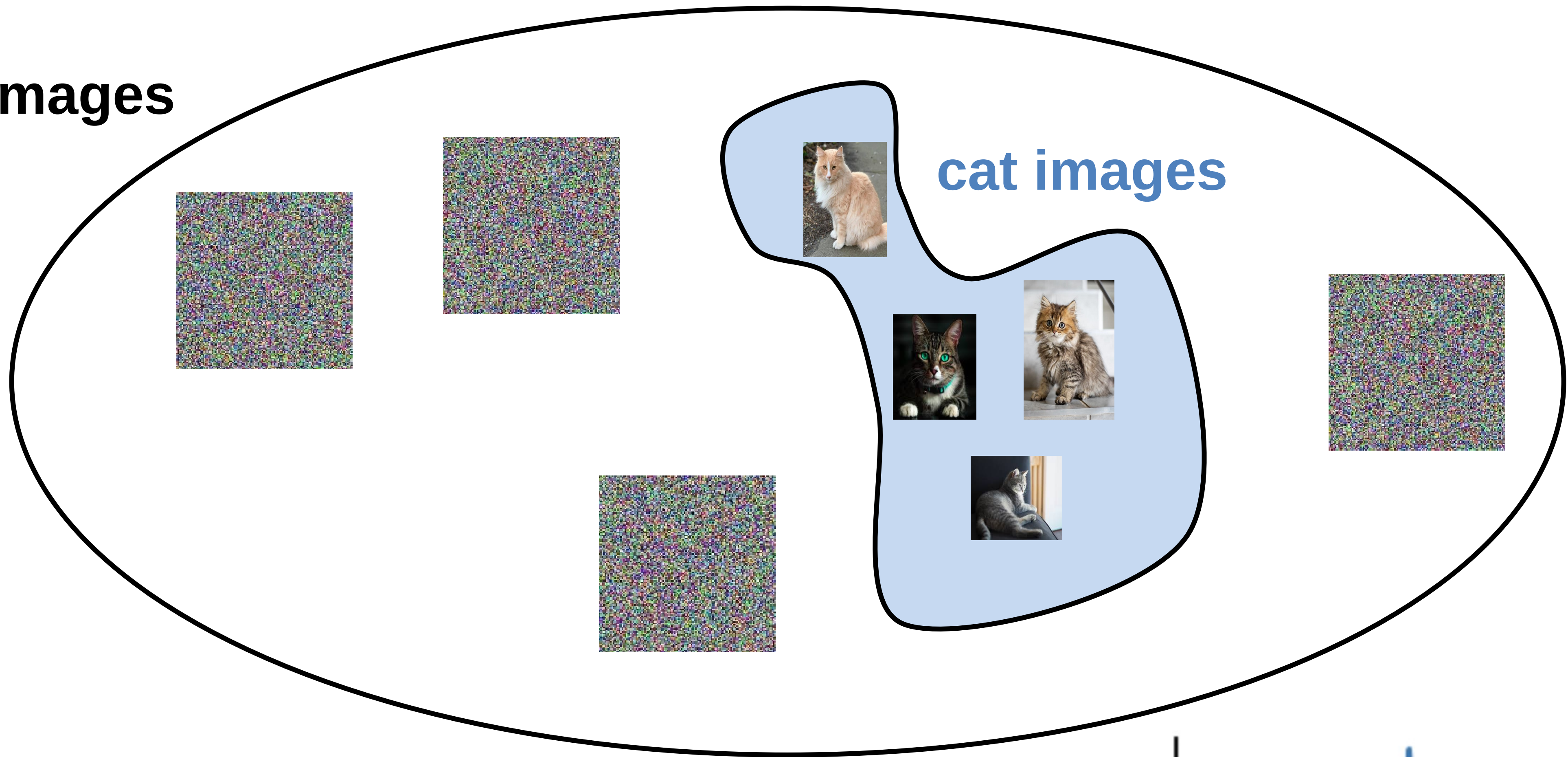
- Most images are “noise”
- “Meaningful” images tend to form some manifold within the space of all images



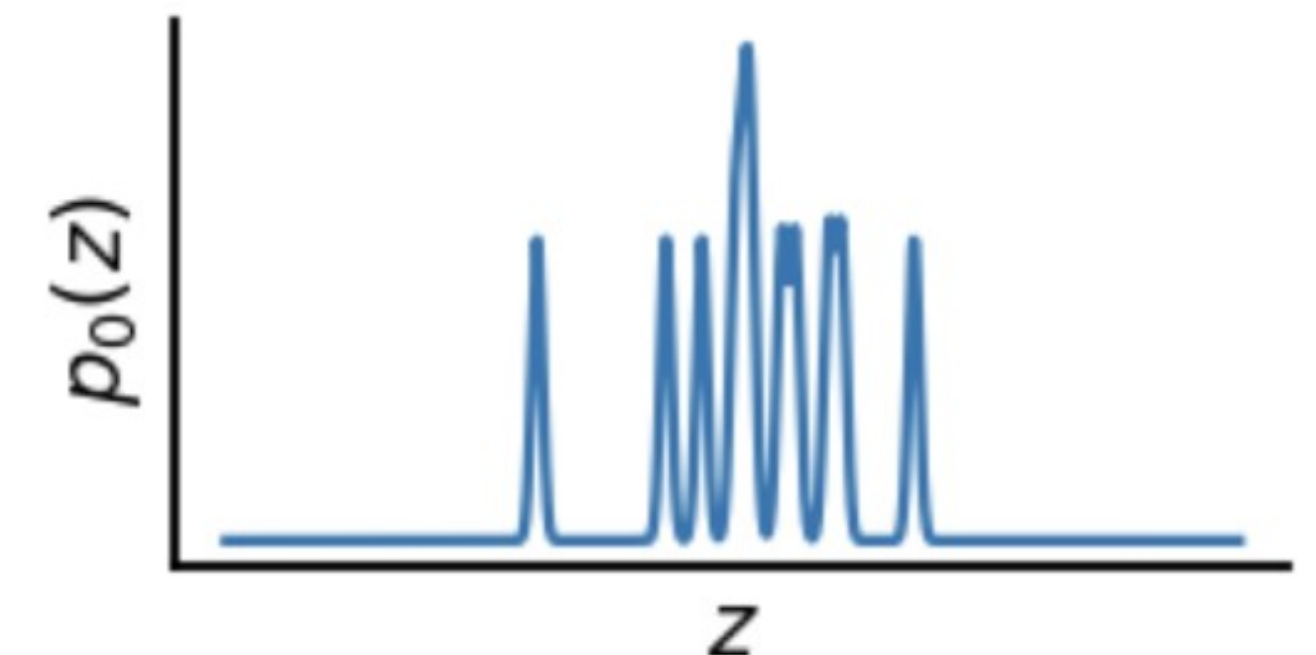
The Space of All Images

Generating Images from Noise

Random images



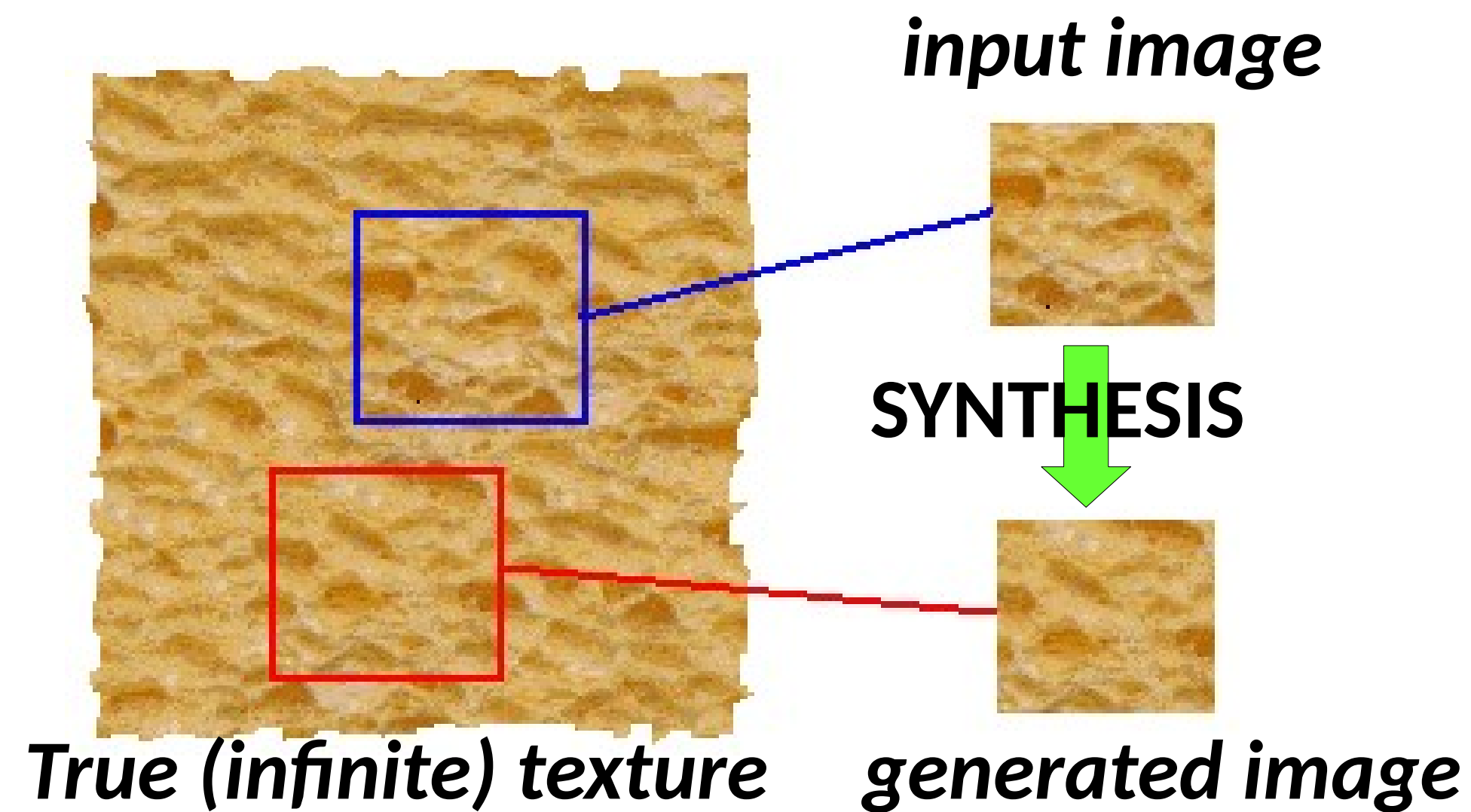
Noise to Images



Let's start with manifold of textures

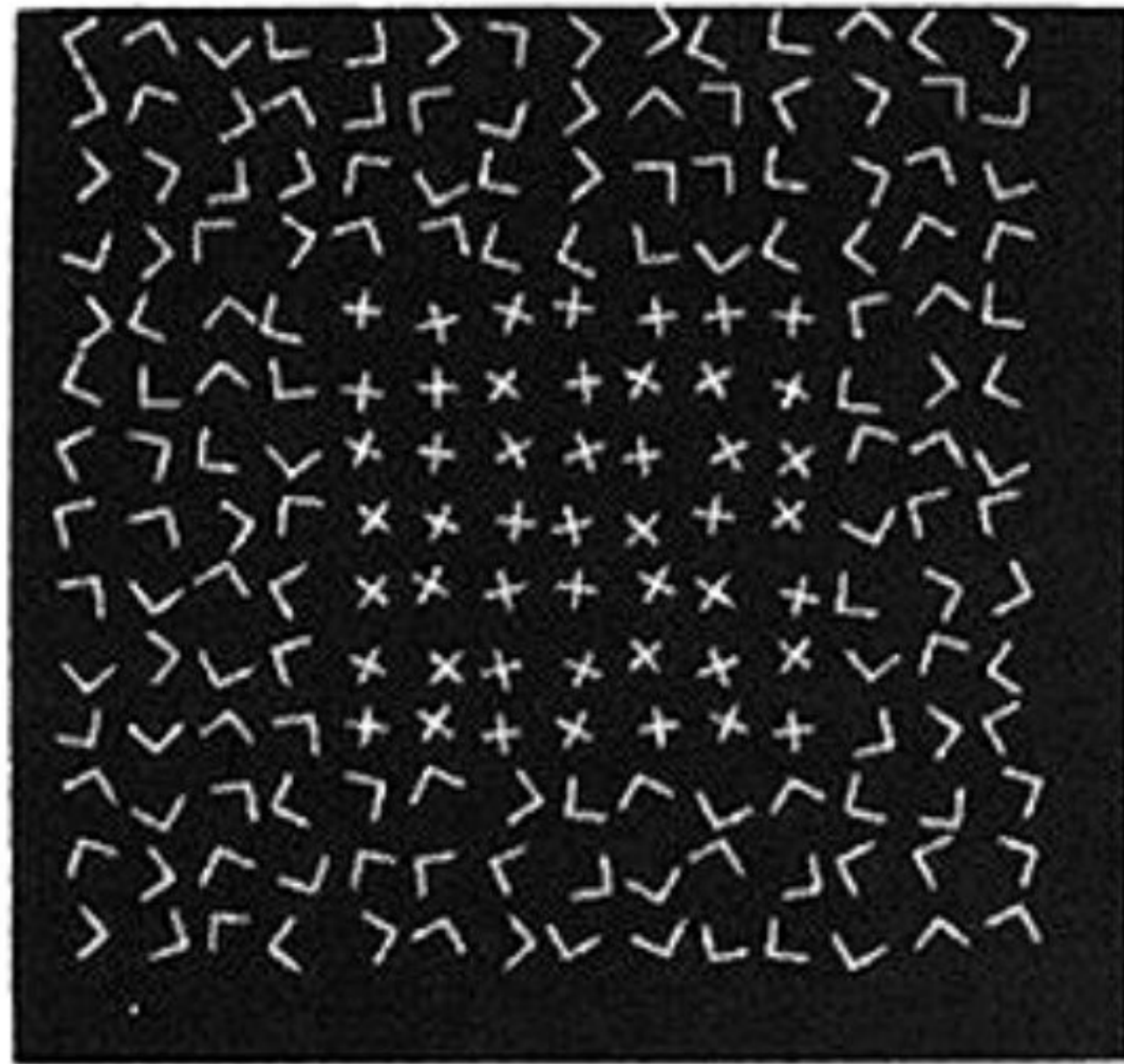


Parametric Texture Synthesis

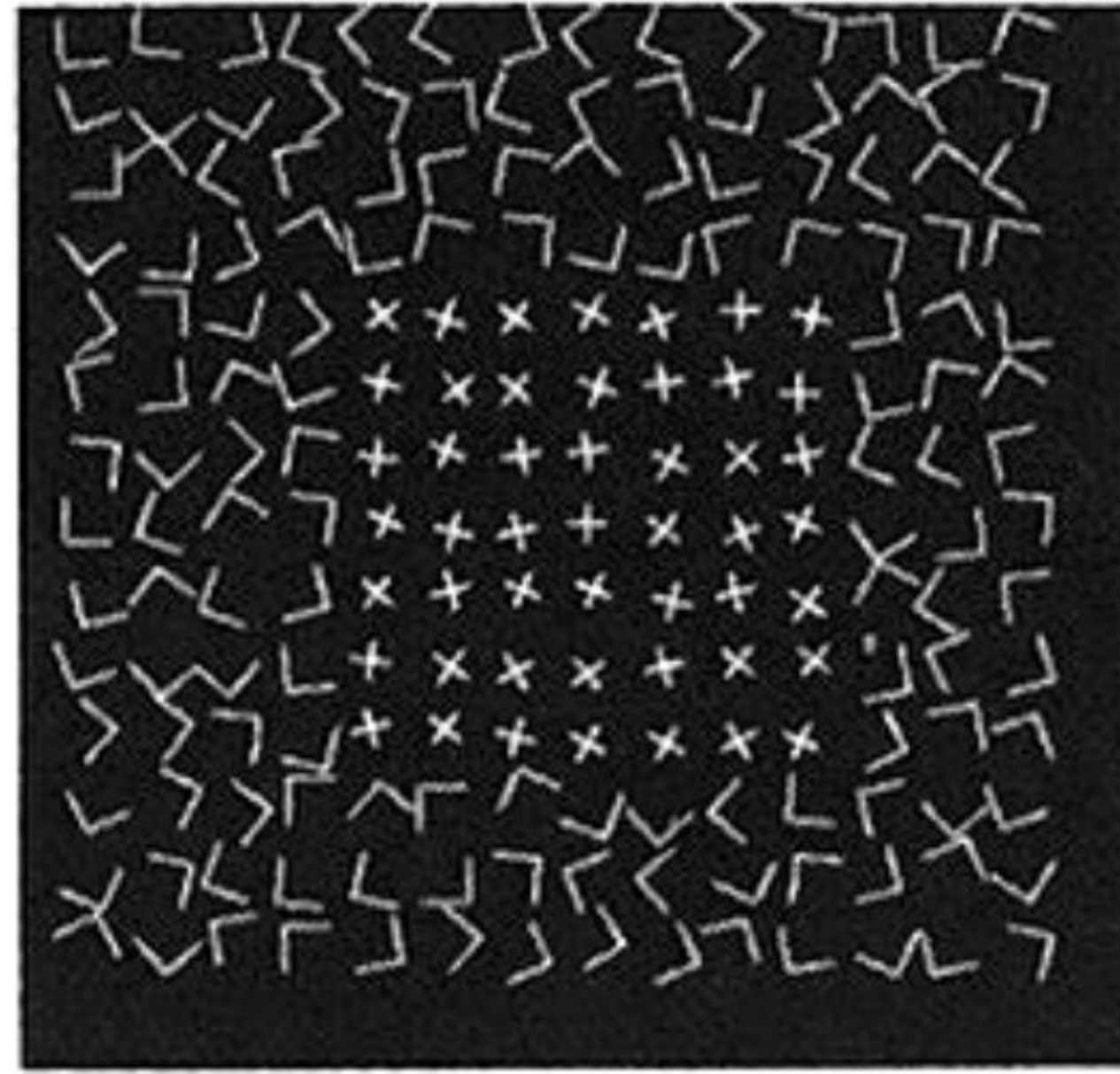


Goal: parametric **generative model** of the
“infinite texture”

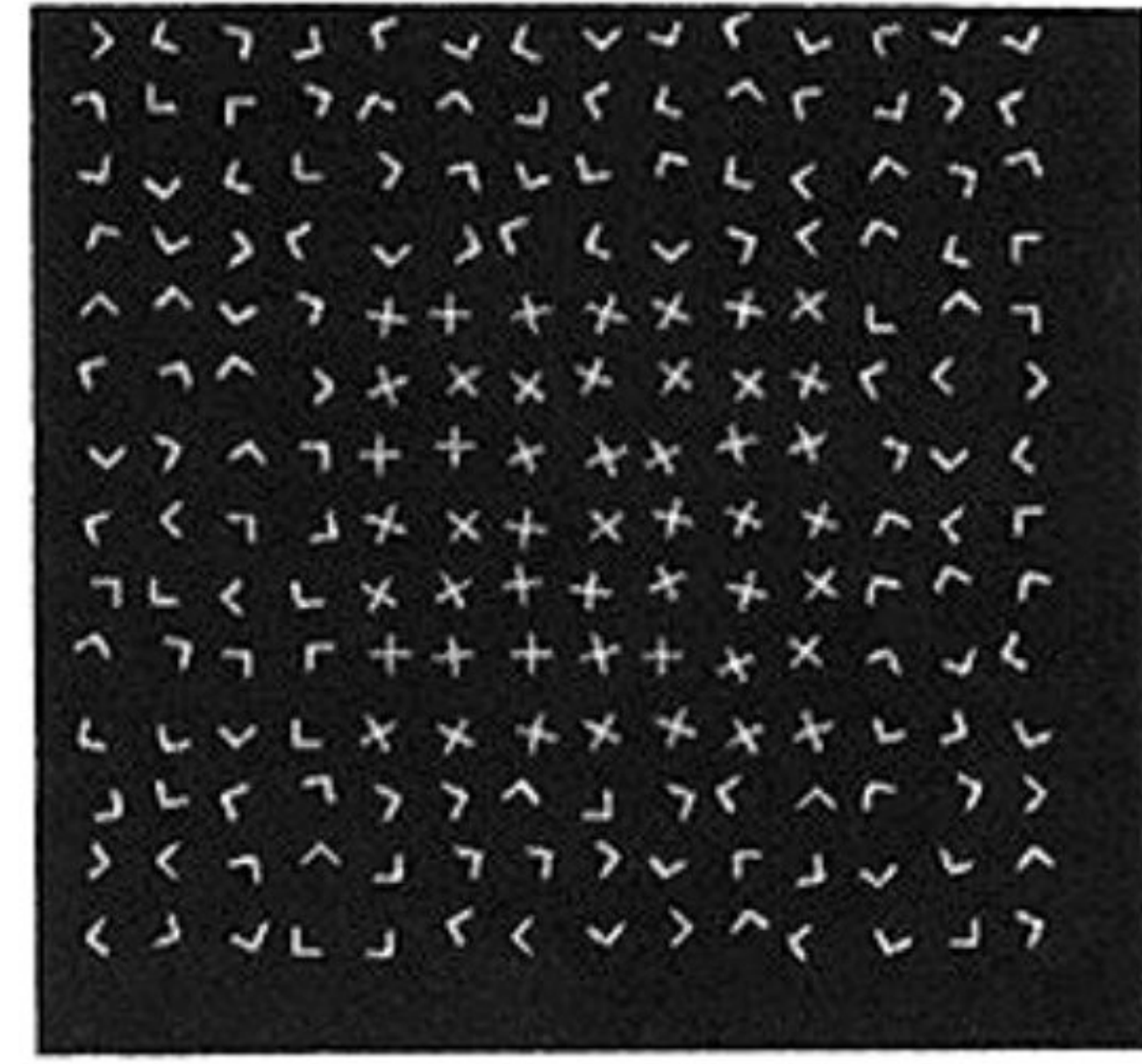
Making two textures more similar



a



b



c

Early vision and texture perception

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

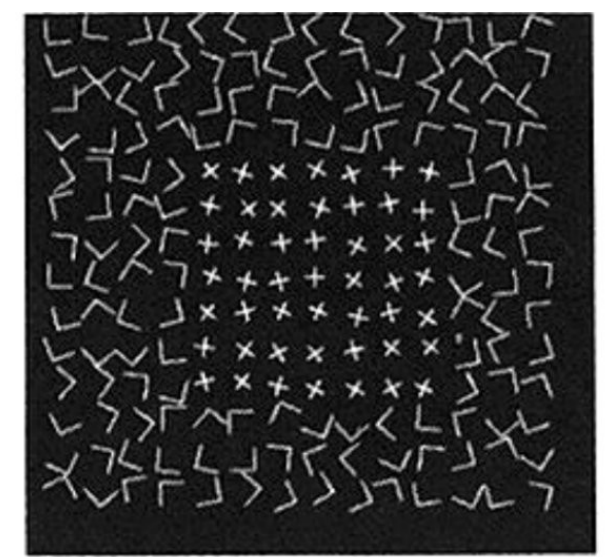
* SRI David Sarnoff Research Center, Princeton,
New Jersey 08540, USA

** Media Lab and Department of Brain and Cognitive Science,
Massachusetts Institute of Technology, Cambridge,
Massachusetts 02139, USA

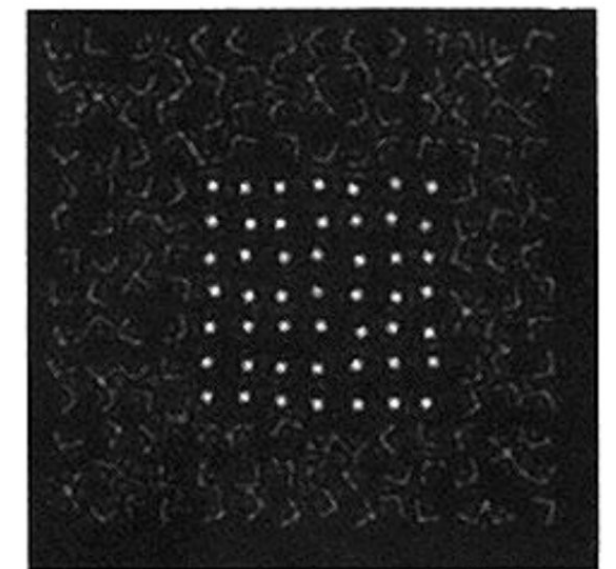
Reprinted from Nature, Vol. 333. No. 6171. pp. 363-364, 26 May 1988
© Macmillan Magazines Ltd., 1988

Jim Bergen's conjecture

"If matching the mean amplitude of a bandpass spatial filter's response goes a little way towards mimicking human texture perception, then maybe matching the histogram of the responses of filters (the marginal statistics) will do an even better job of capturing human texture perception."



b



e



Dave Heeger's response

"You mean if I took a pyramid of some random noise, and forced the histograms of each subband level to match those of some target texture, that the modified noise image would then look like that texture??? No way! I'll prove it to you; here, let me try it out...
hmm, gee, that worked pretty well..."

Heeger & Bergen (SIGGRAPH 1995)

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 wrap-

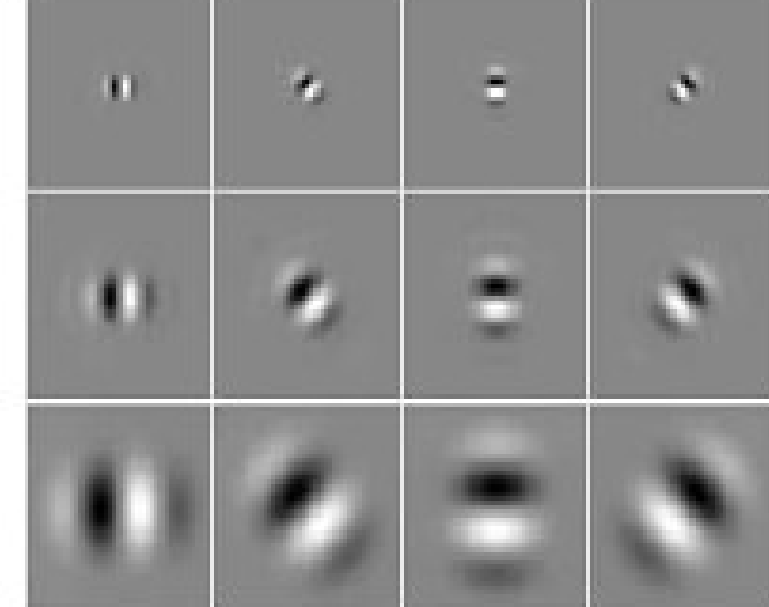
ping a texture image onto a surface. However, existing techniques for synthesizing solid textures can be quite cumbersome. One must learn how to tweak the parameters or procedures of the texture synthesizer to get a desired effect.

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.

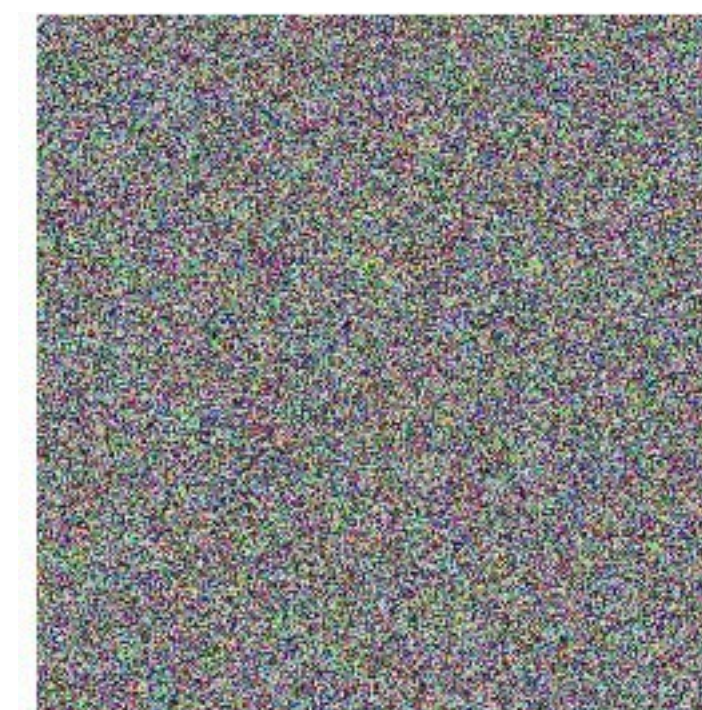


Figure 2: (Left) Input digitized sample texture: burl wood. (Middle) Input noise. (Right) Output synthetic texture that matches the appearance of the digitized sample. Note that the synthesized texture is larger than the digitized sample; our approach allows generation of as much texture as desired. In addition, the synthetic textures tile seamlessly.

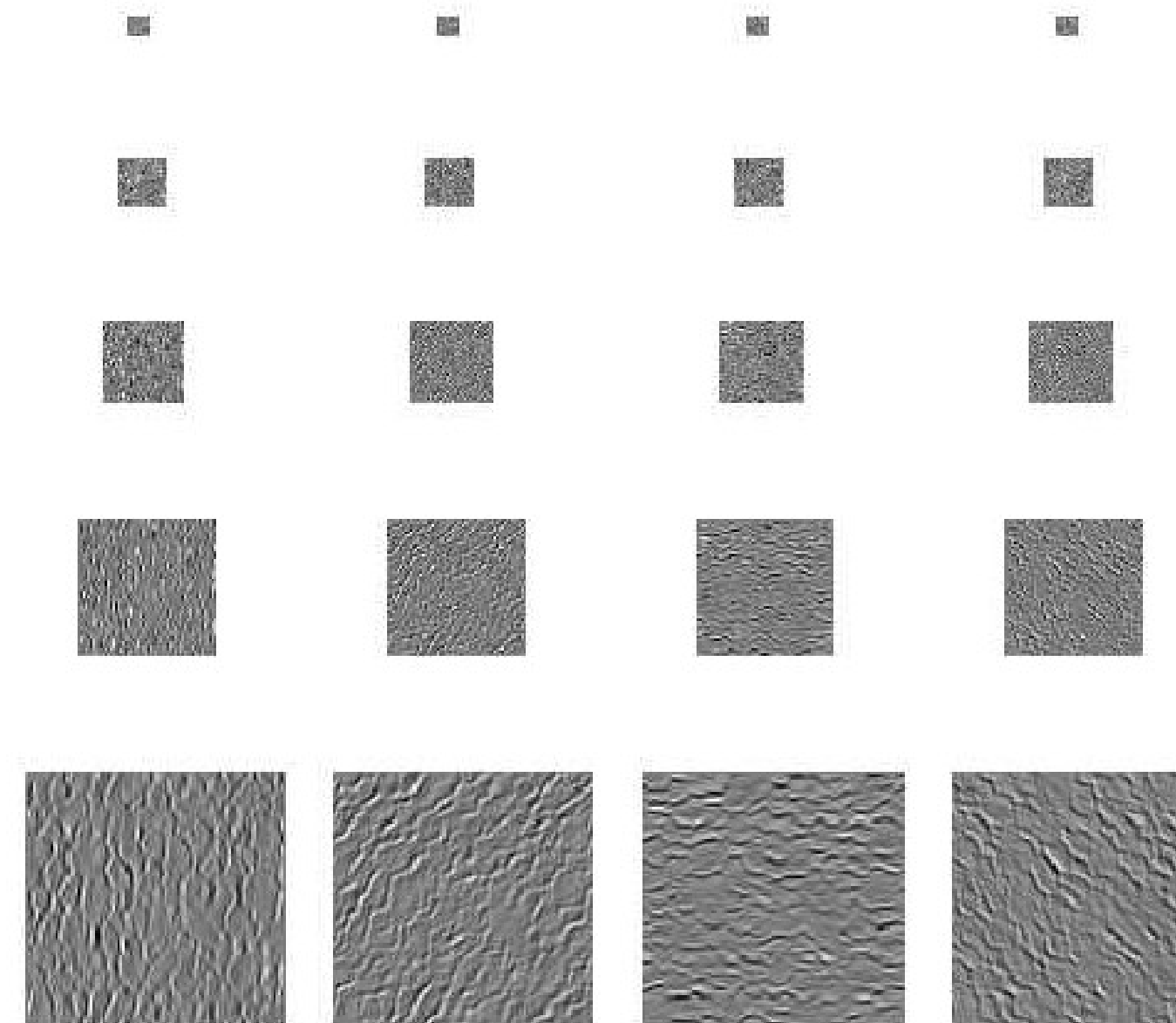
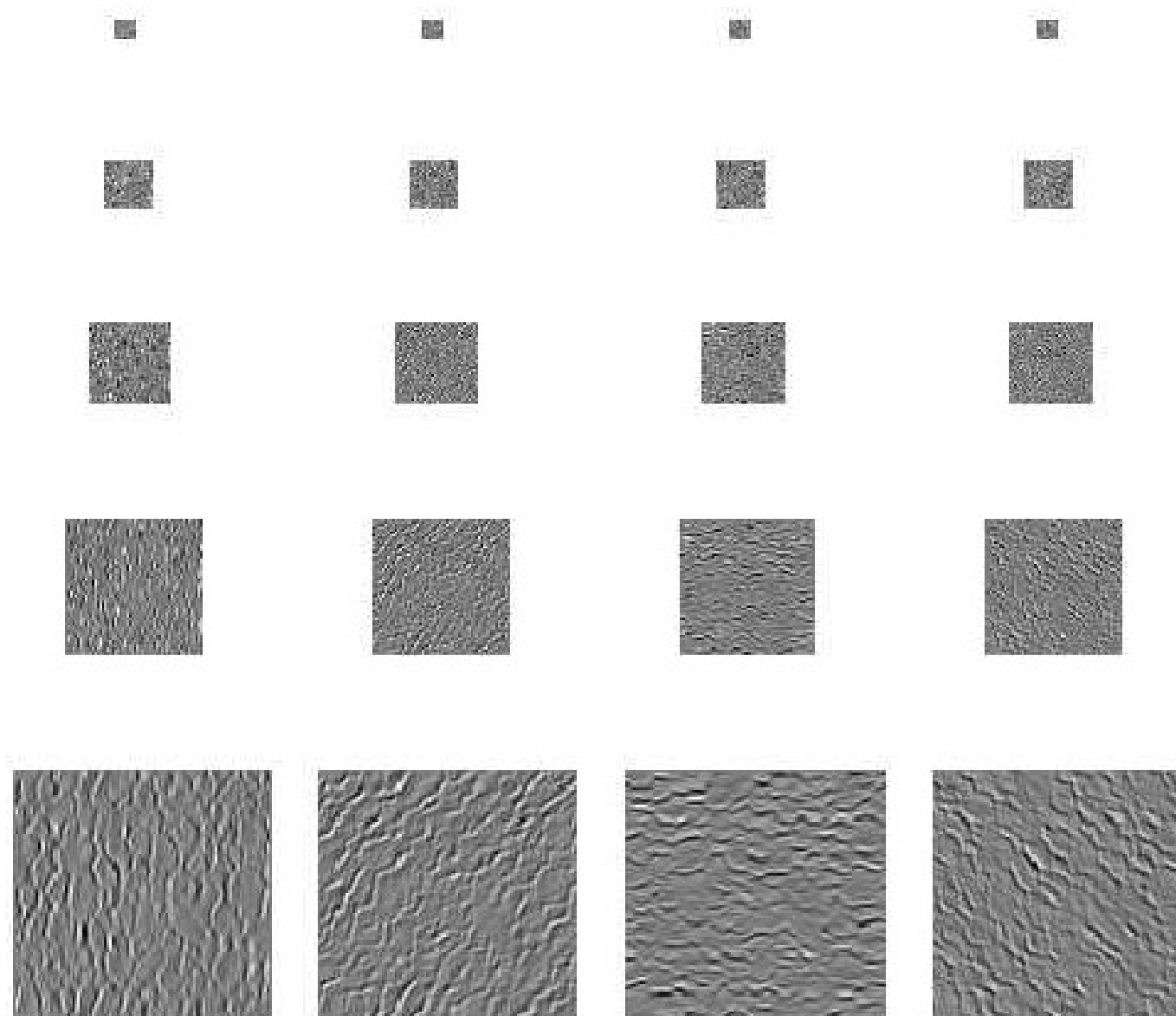
Step 1: Compute Pyramid Features



input



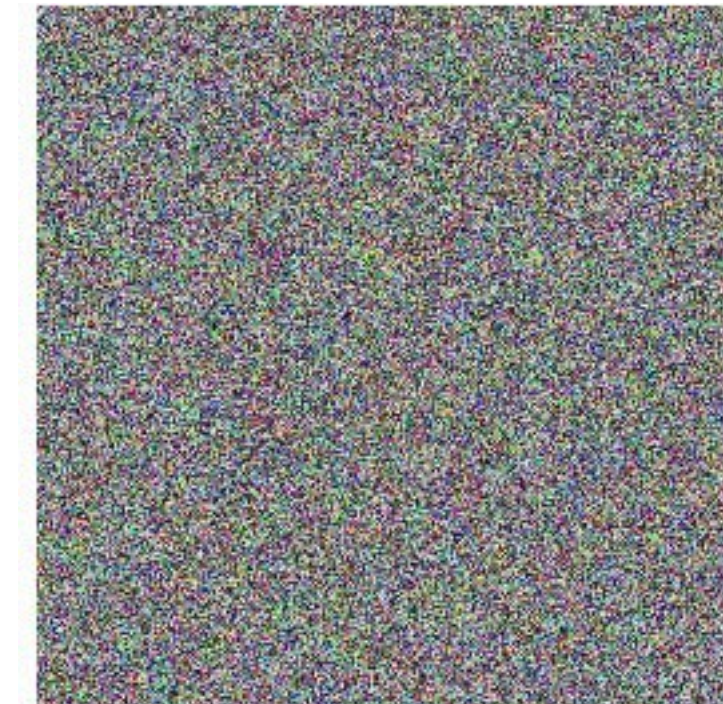
Noise image



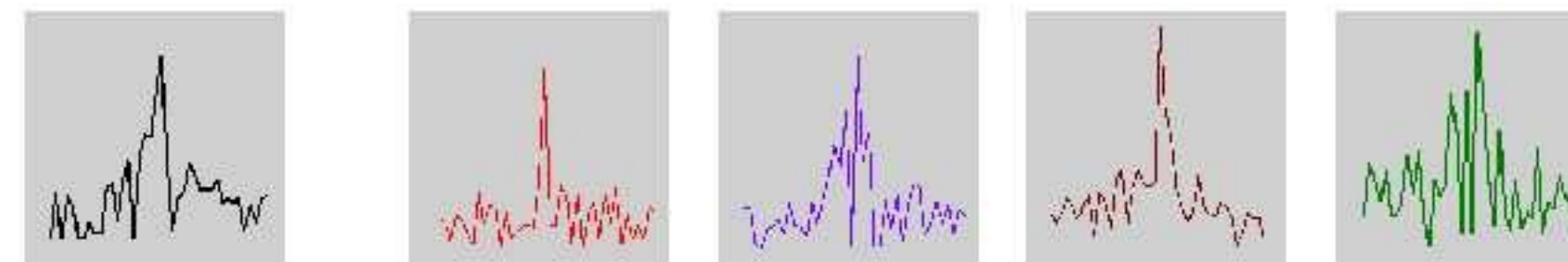
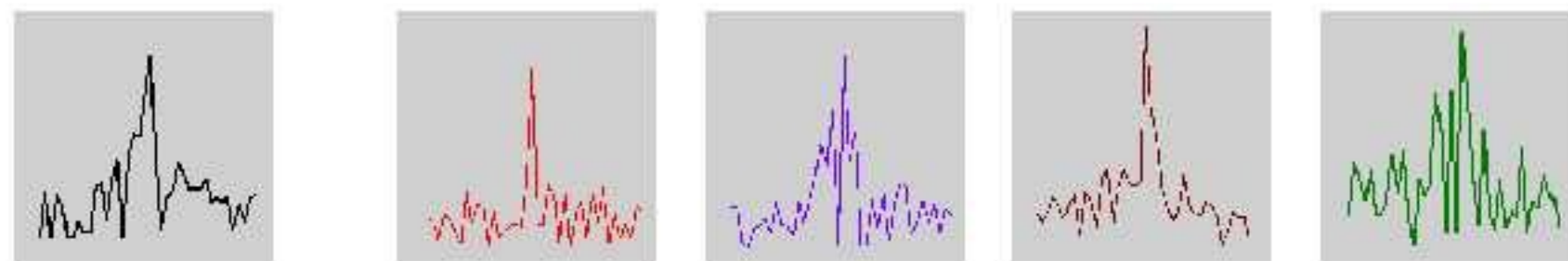
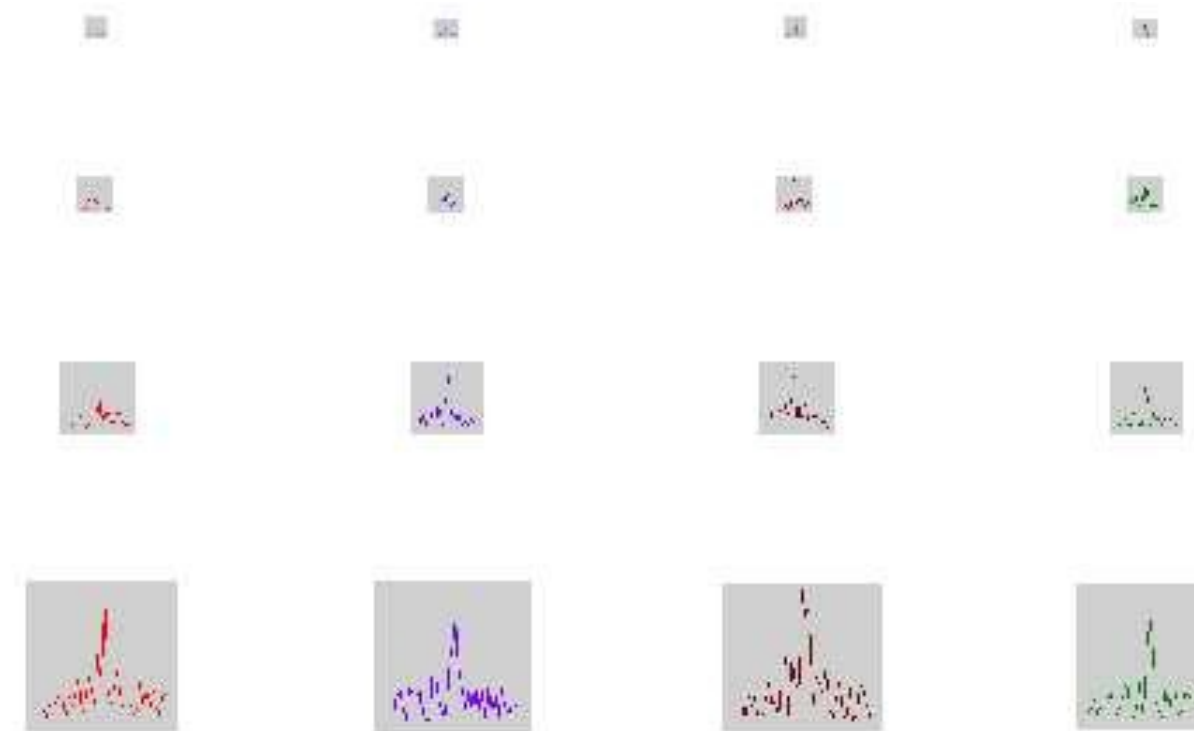
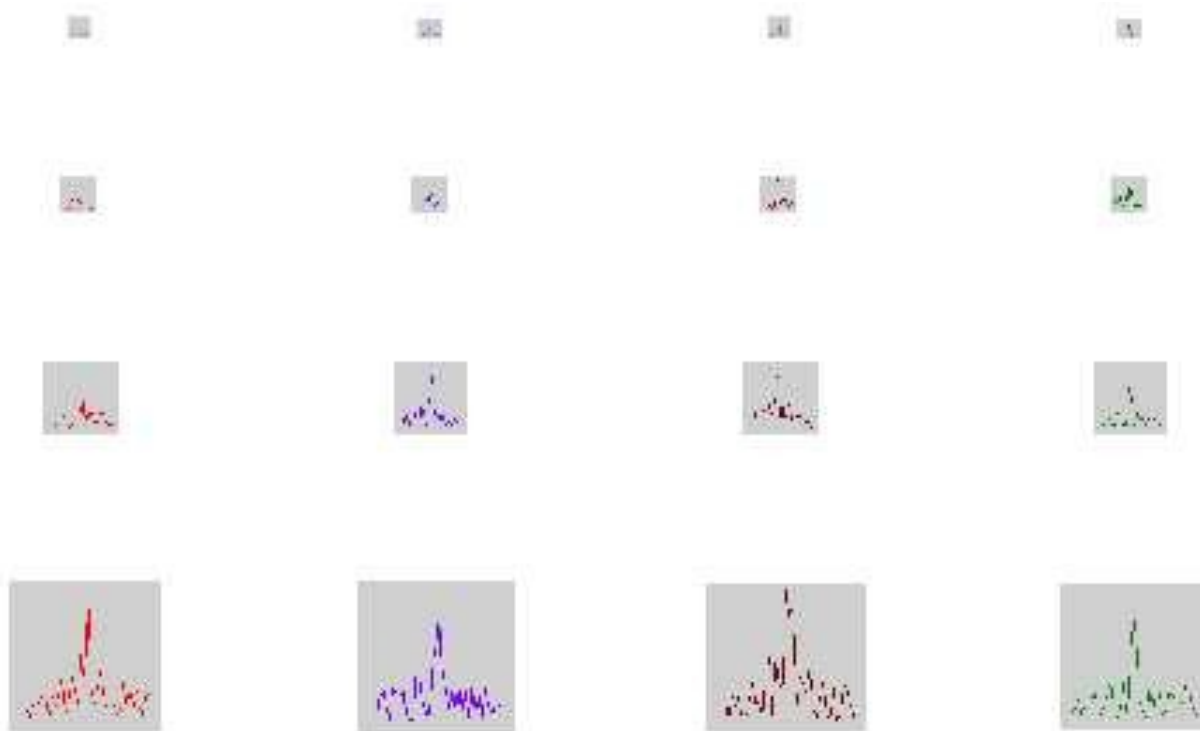
Step 2: match per-channel histograms



input



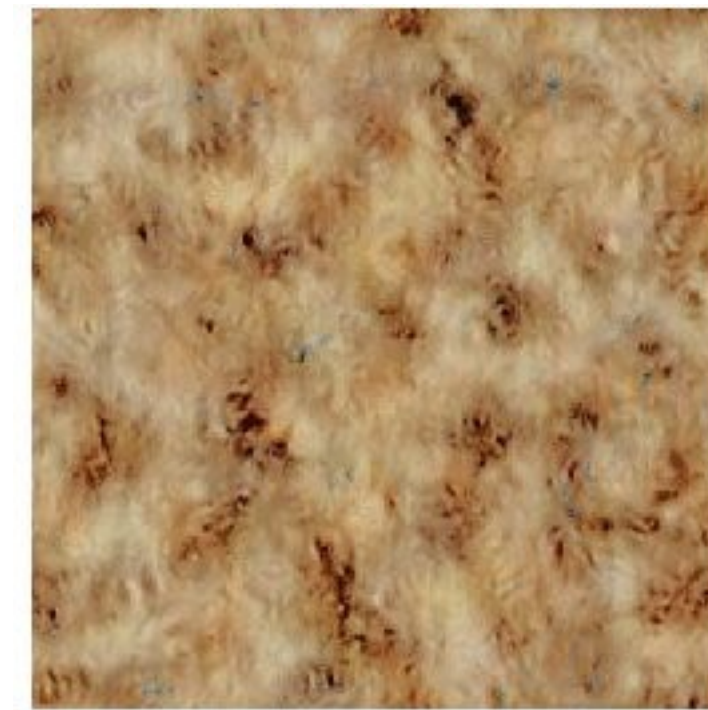
Noise image



Step 3: collapse pyramid and repeat!



input



Noise image

Texture-motivated Parametric Generative Image Models!

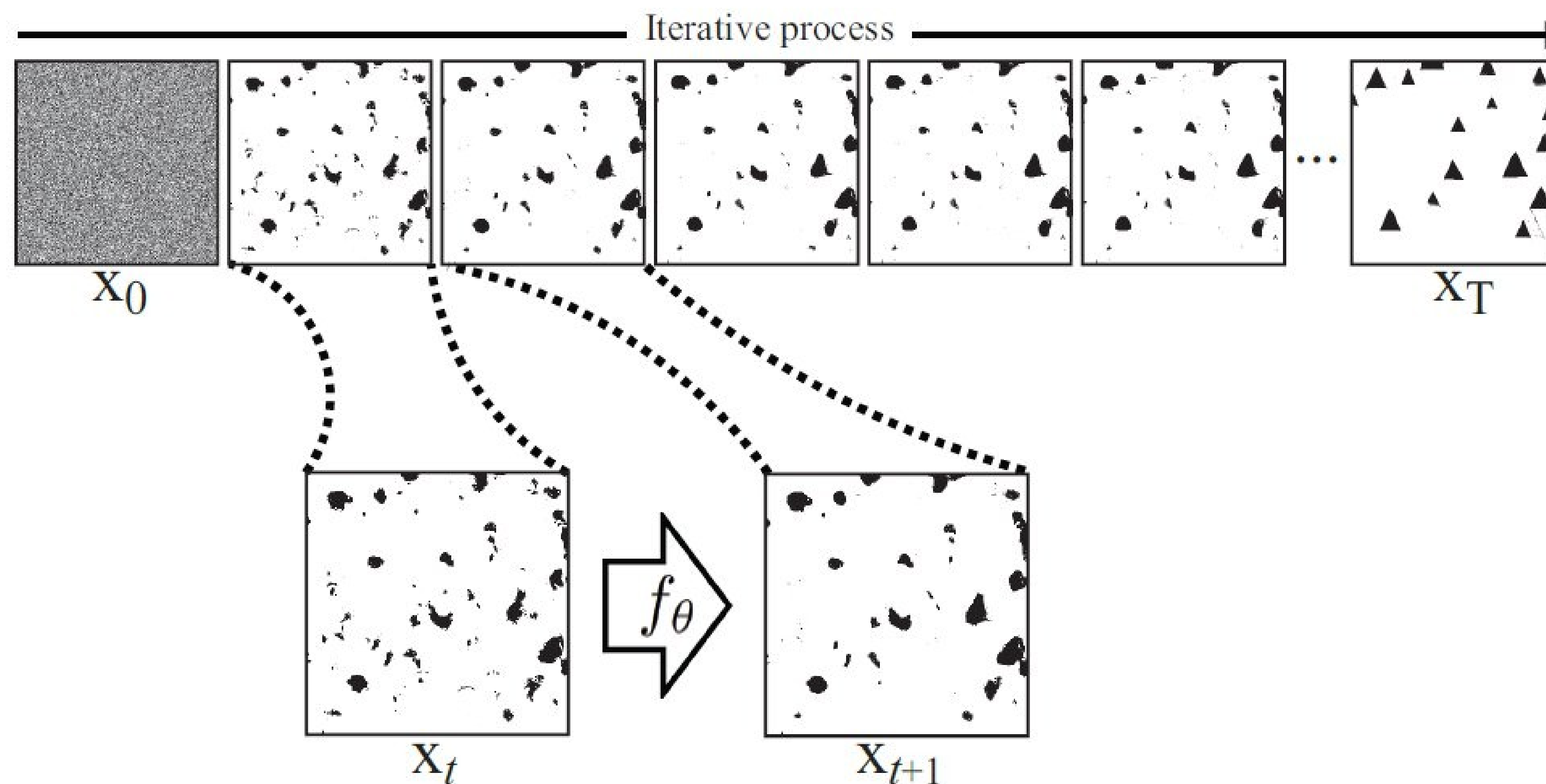
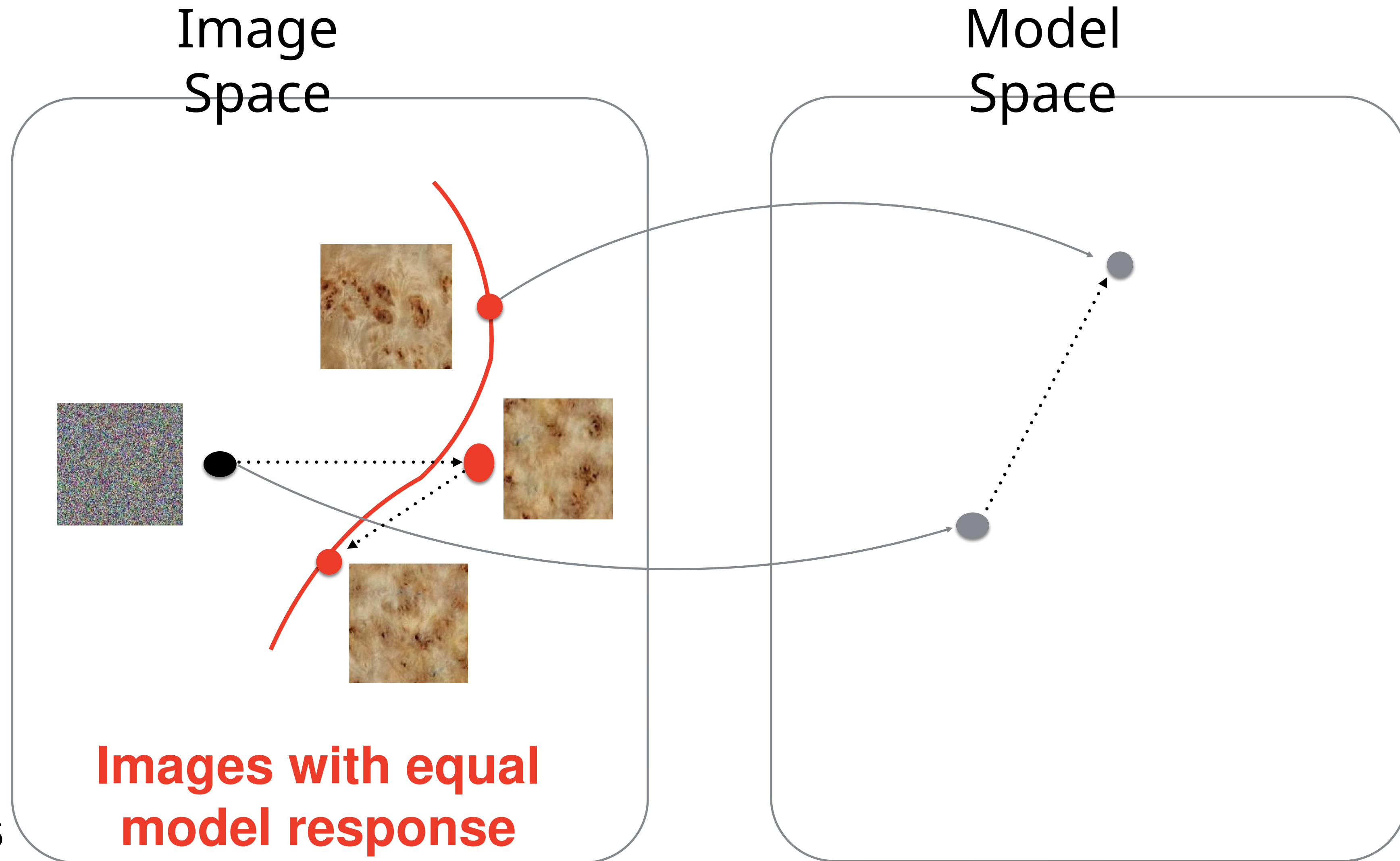


Figure 28.6: The steps of the Heeger-Bergen texture synthesis algorithm. The process starts with white noise input image. Each step takes as input the previous output, and it is modified by a function f_θ , where θ are the parameters describing a texture. At each step the output image x_t gets closer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

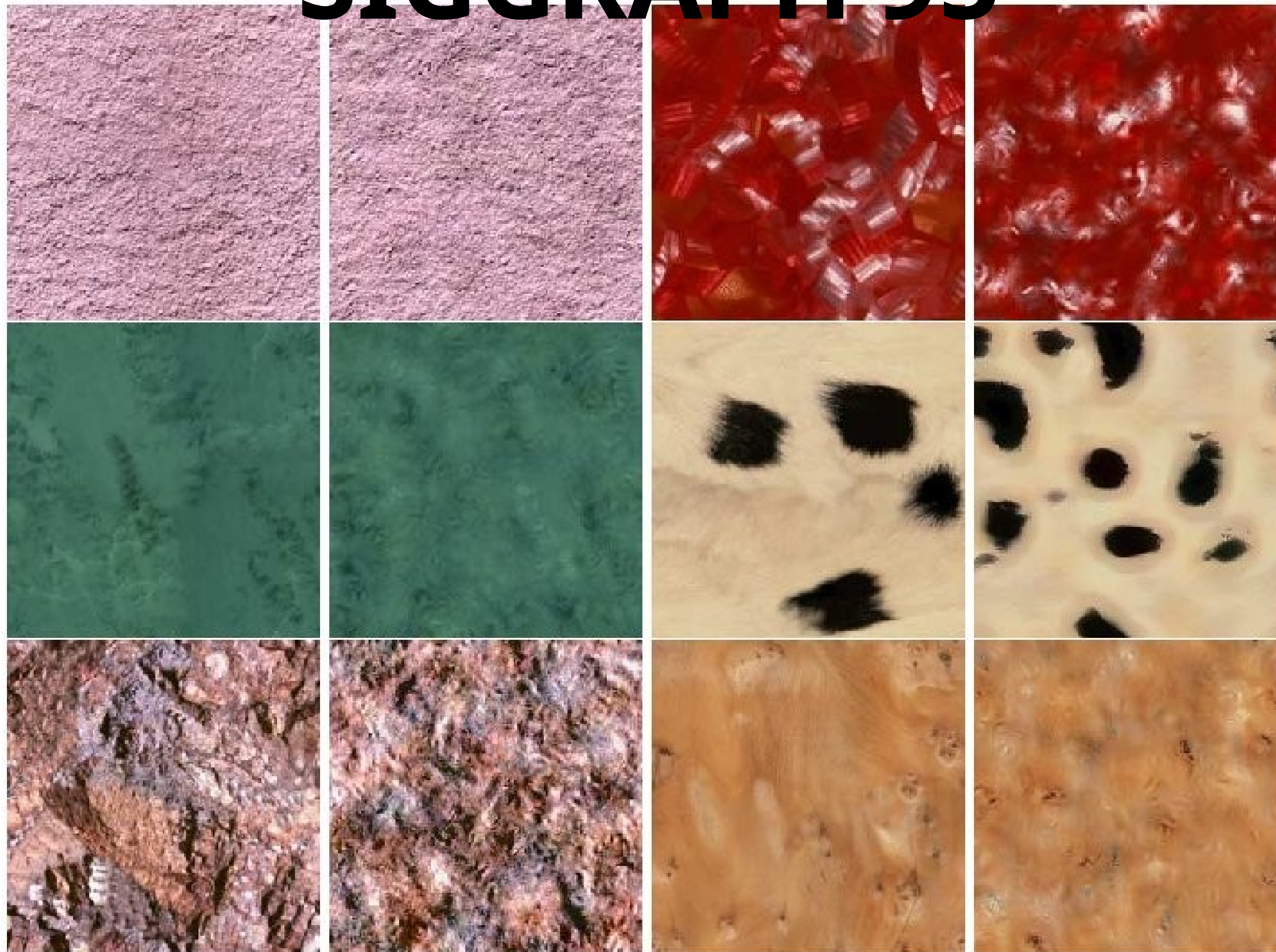
Parametric Texture Synthesis



Heeger & Bergen, 1995
Portilla & Simoncelli, 2000
Gatys et al, 2015

Portilla & Simoncelli
(2000)

Heeger & Bergen, SIGGRAPH'95





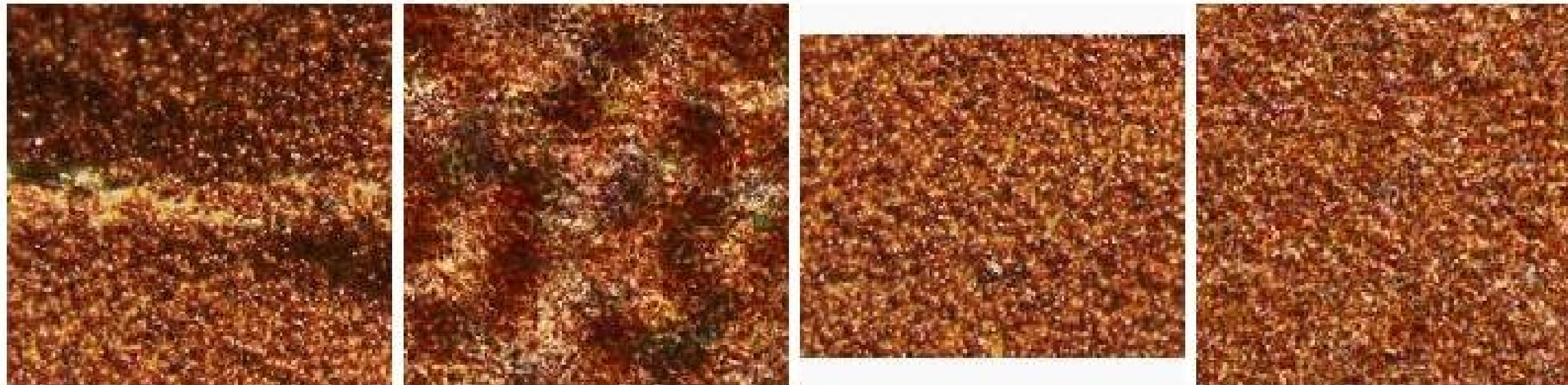


Figure 7: (Left pair) Inhomogeneous input texture produces blotchy synthetic texture. (Right pair) Homogenous input.

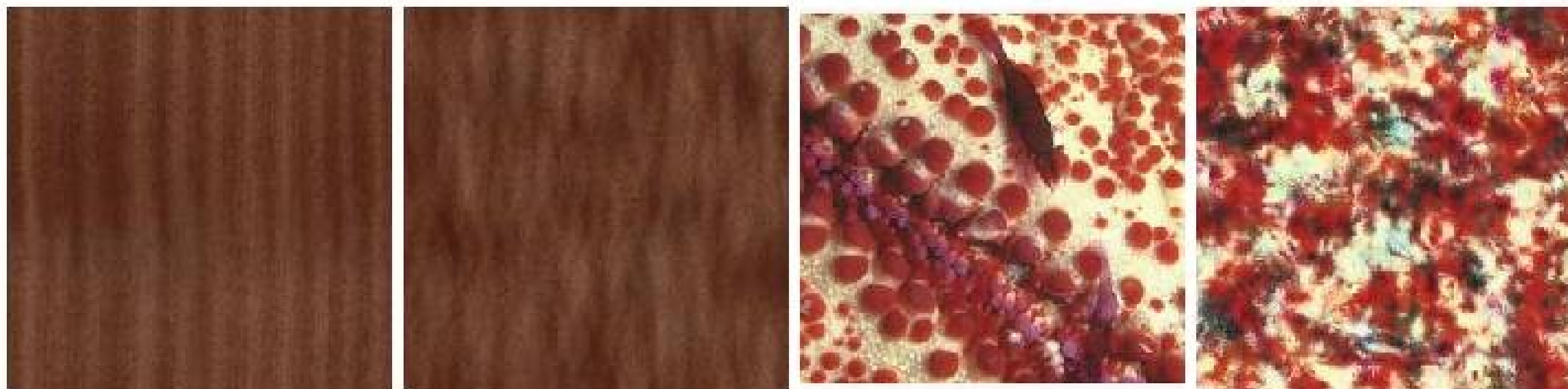


Figure 8: Examples of failures: wood grain and red coral.



Figure 9: More failures: hay and marble.

Simoncelli & Portilla '98+

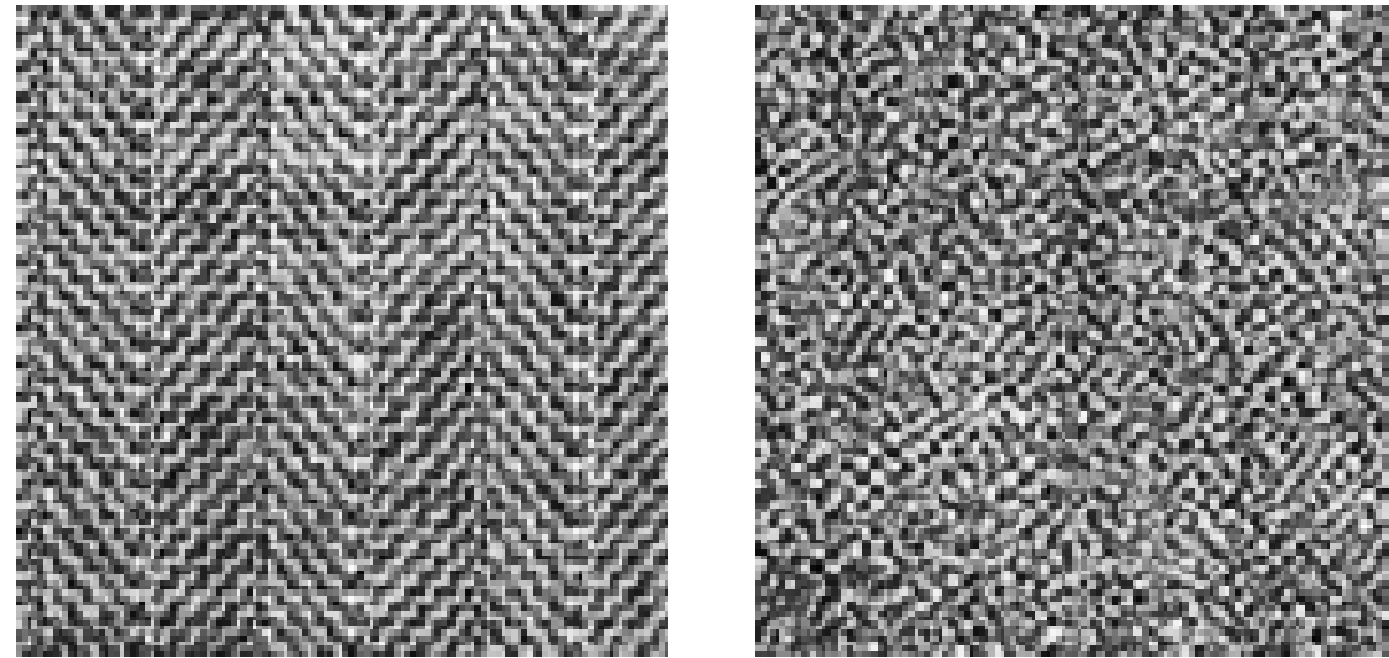
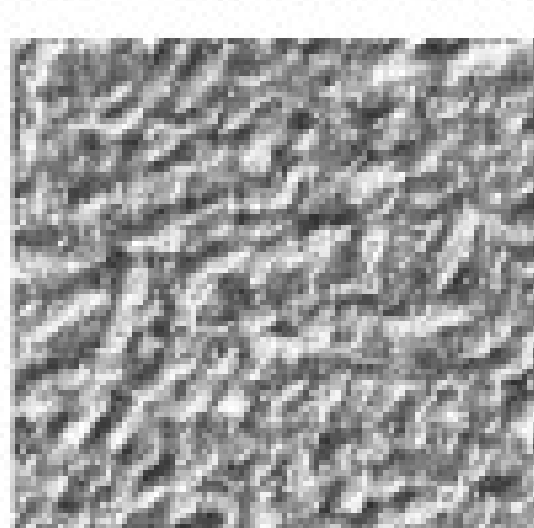
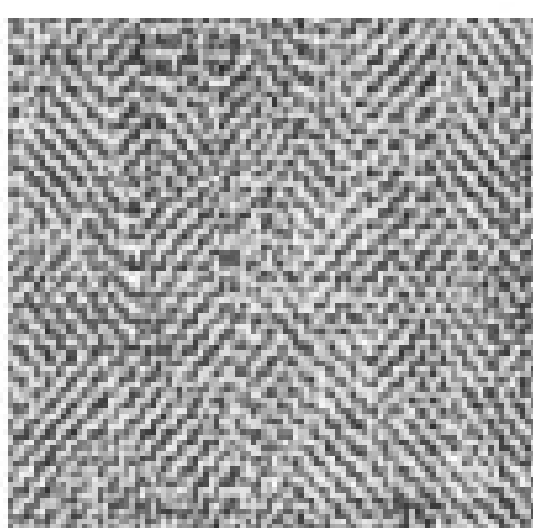
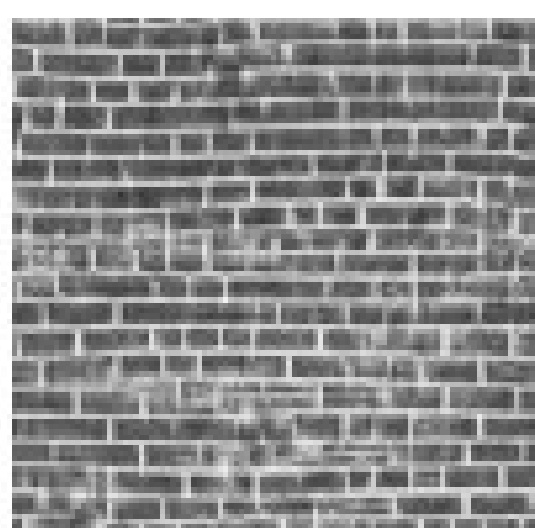
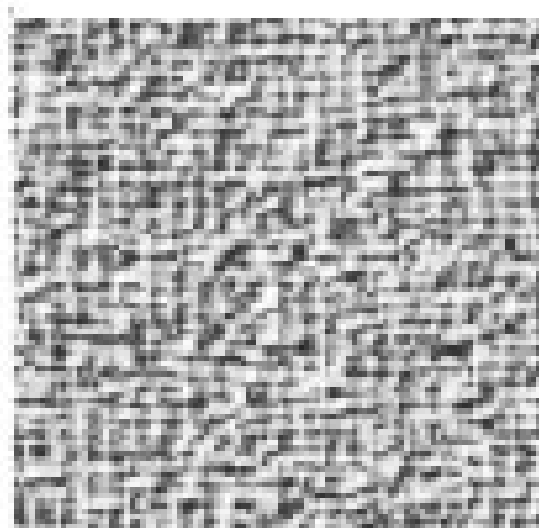
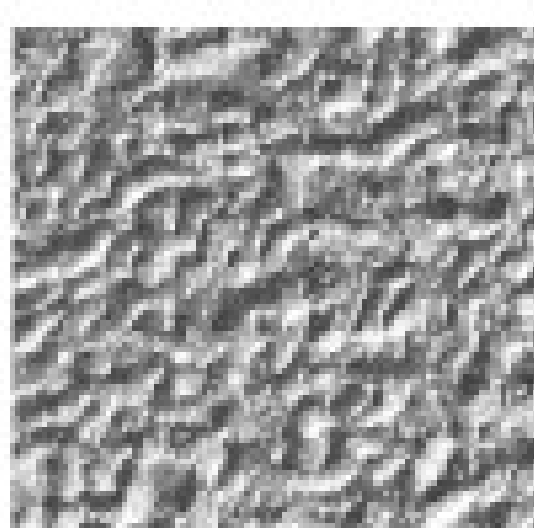
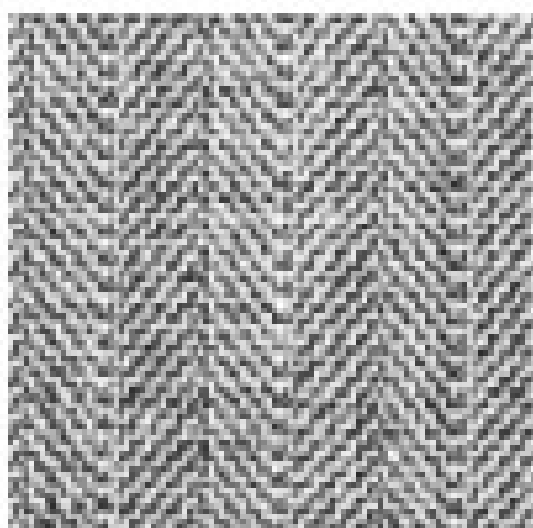
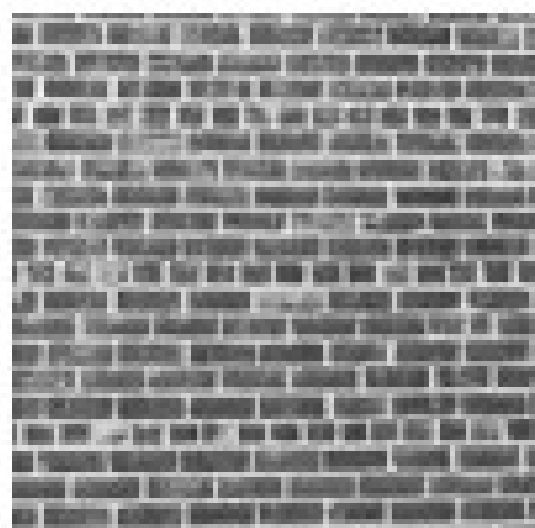
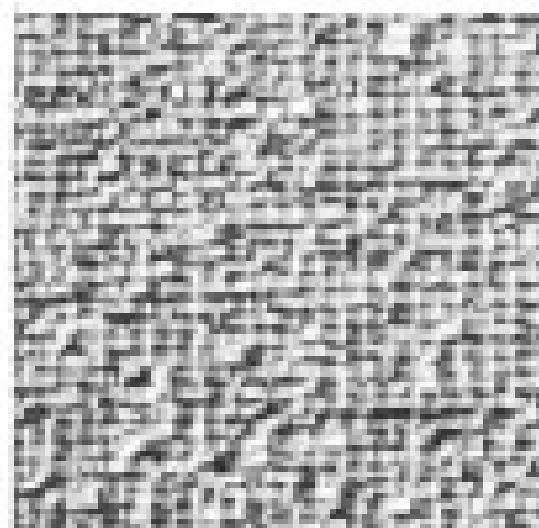
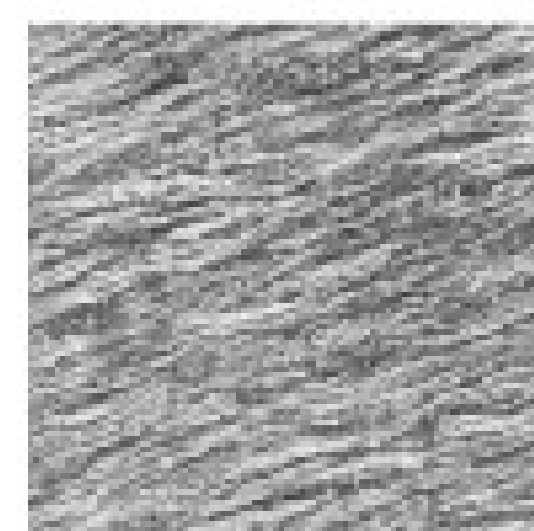
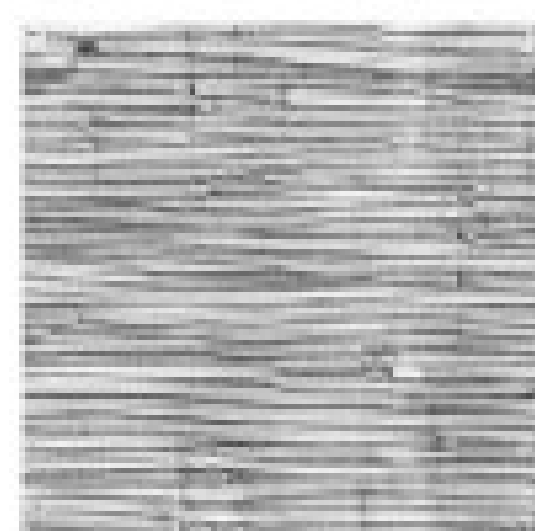
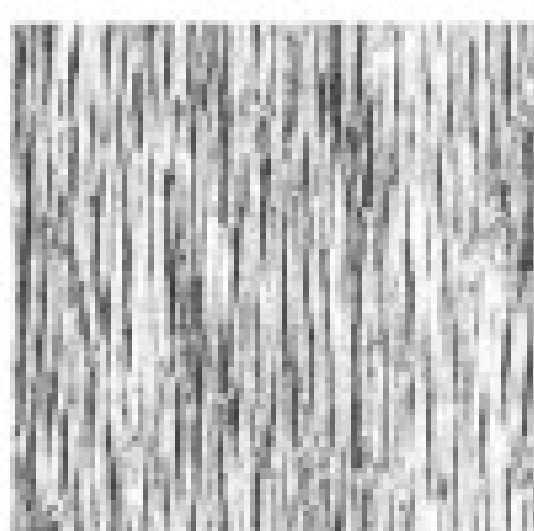
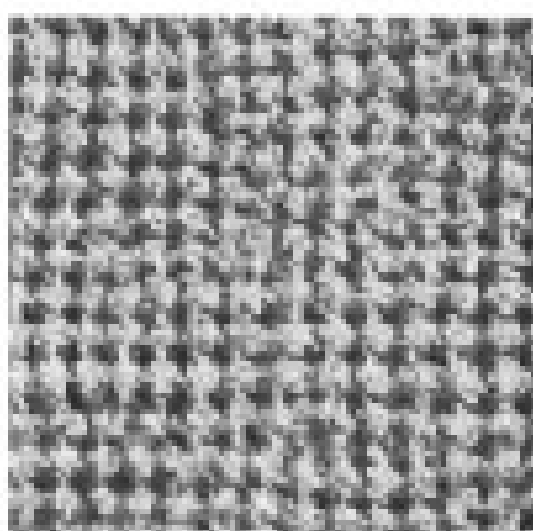
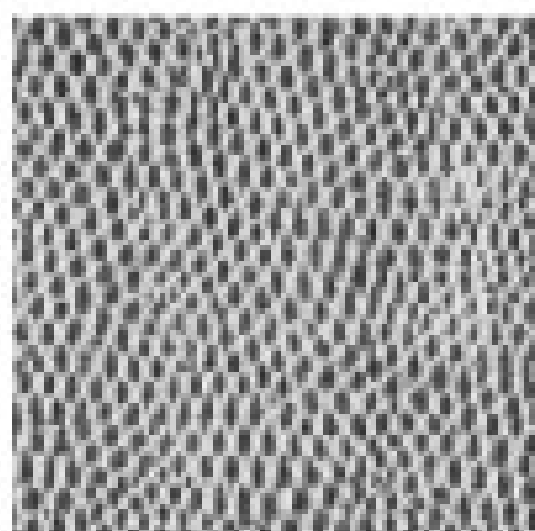
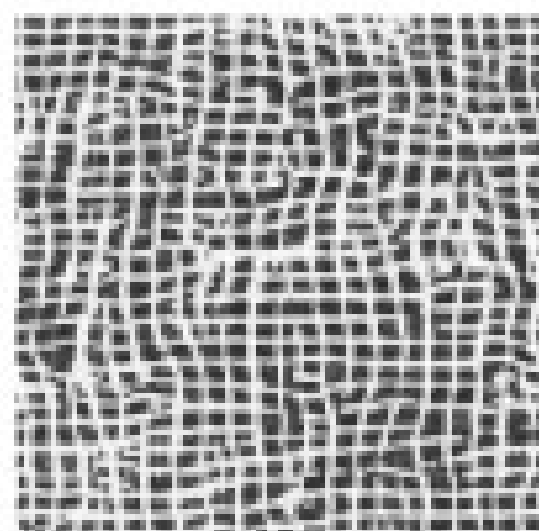
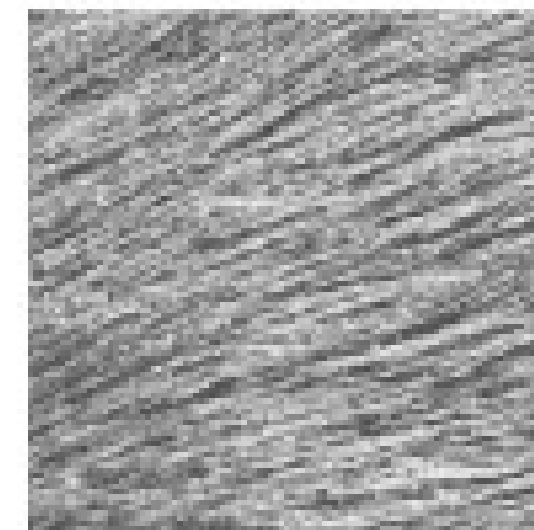
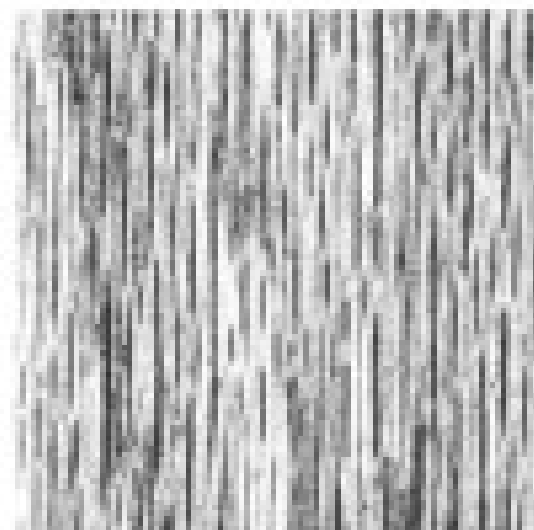
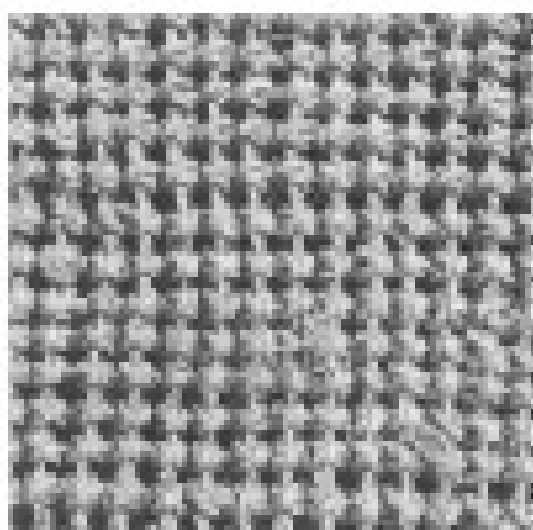
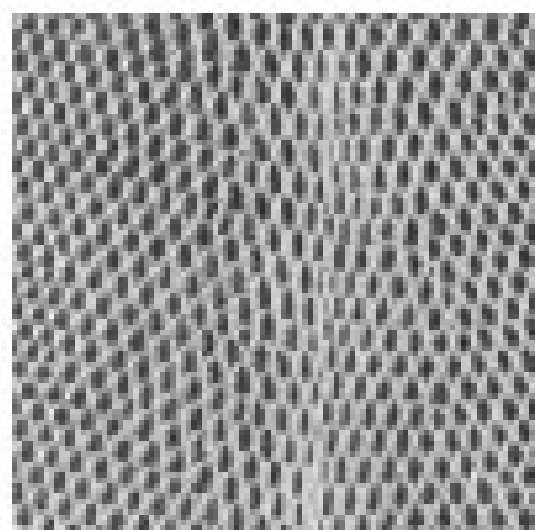
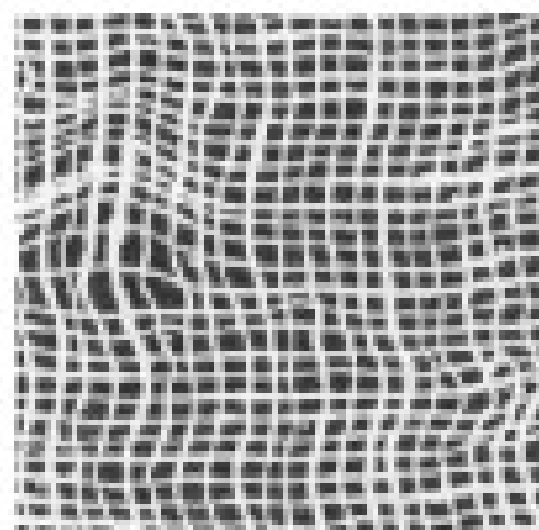
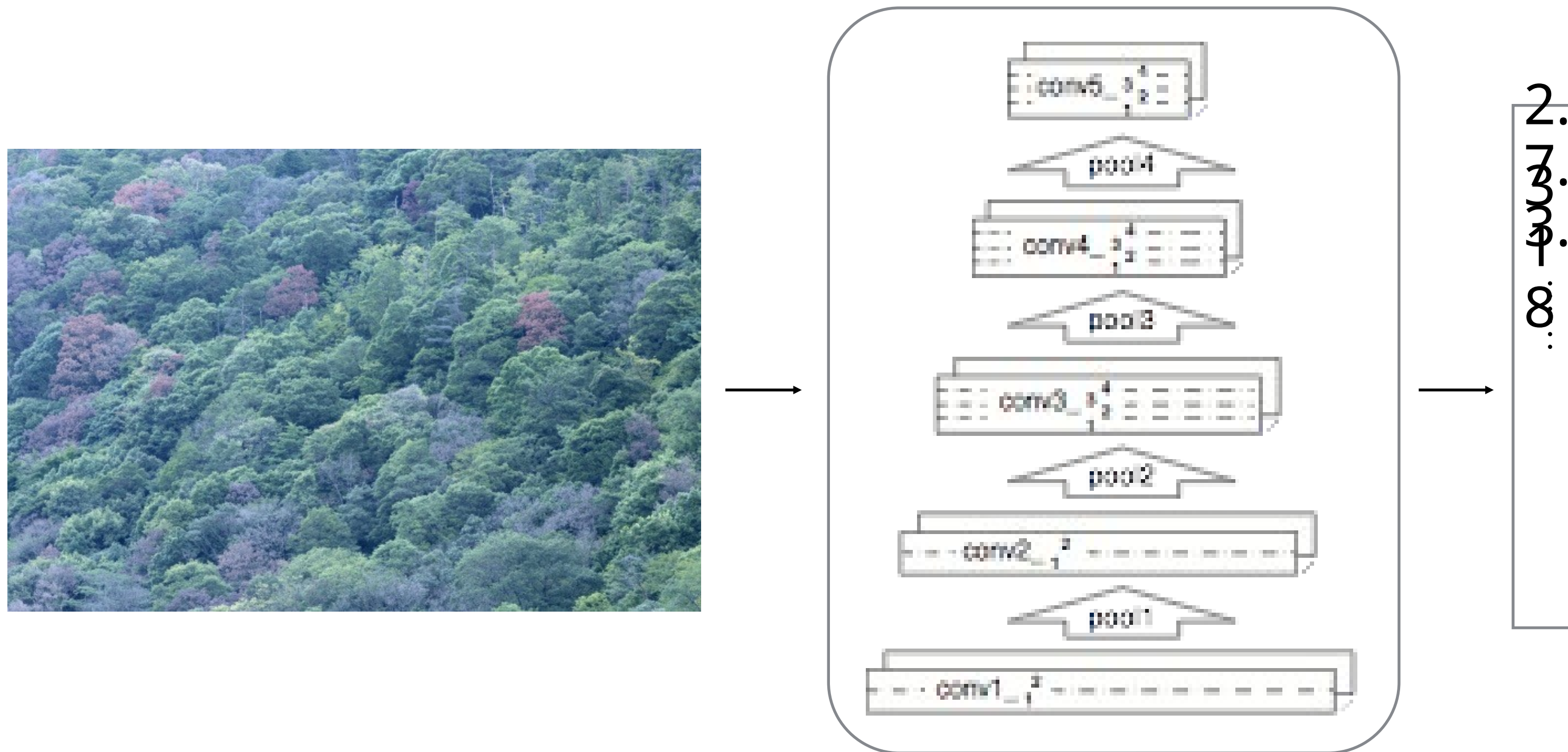


Figure 1. Textures with matching marginal statistics.

- 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.



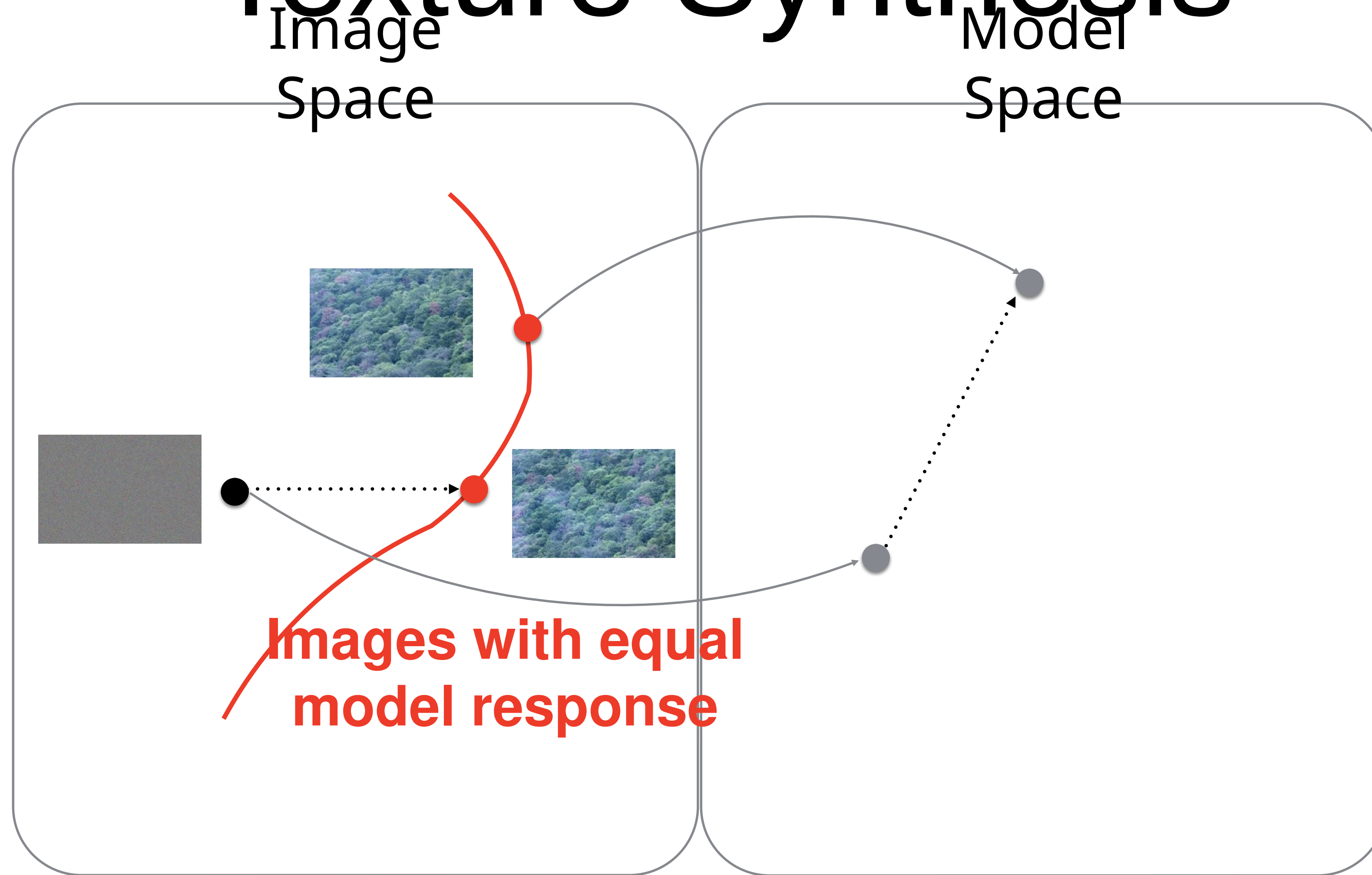
Convolutional Neural Network Texture Model



Convolutional Neural Network

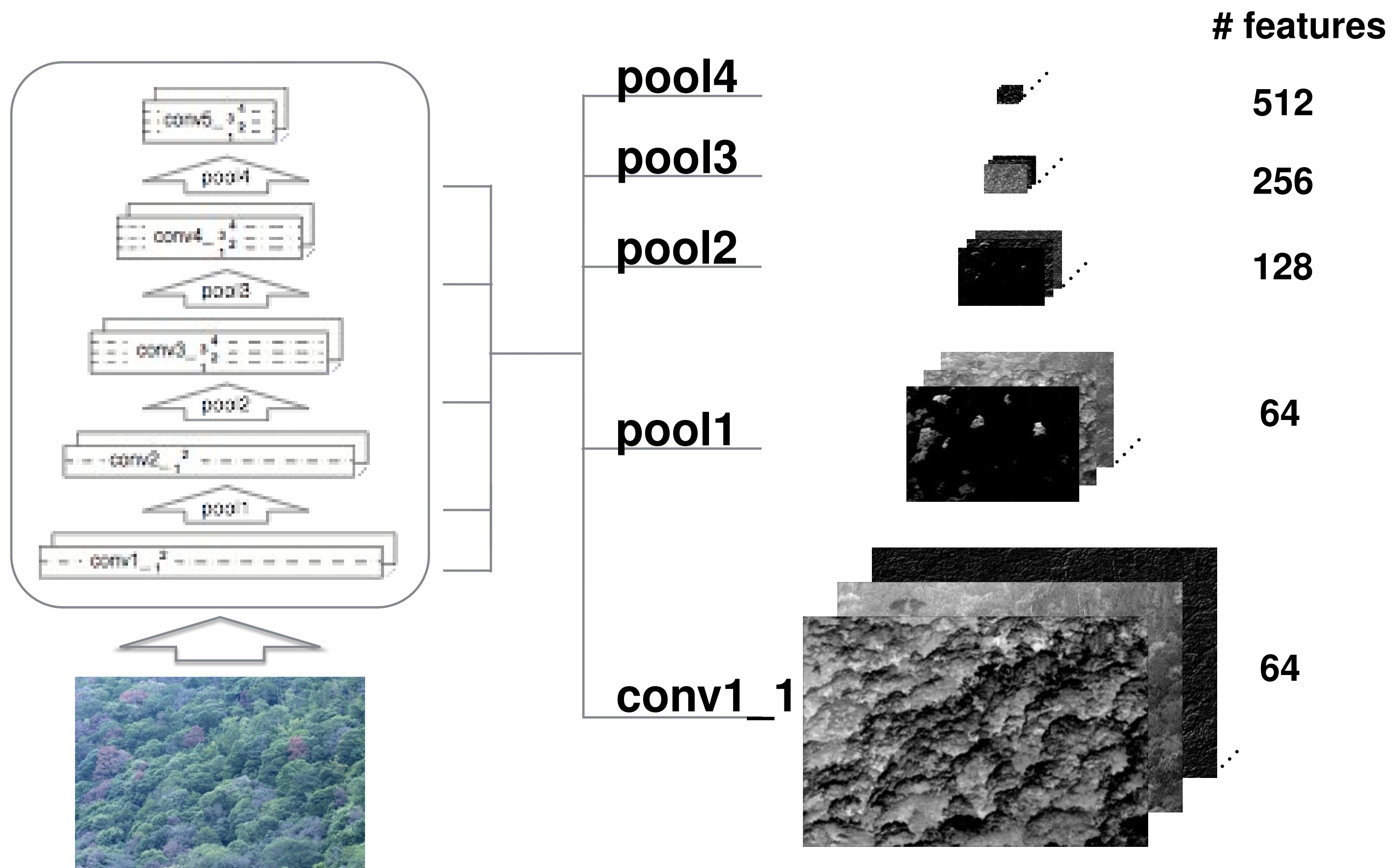
Gatys et al. (NIPS
2015)

Texture Synthesis

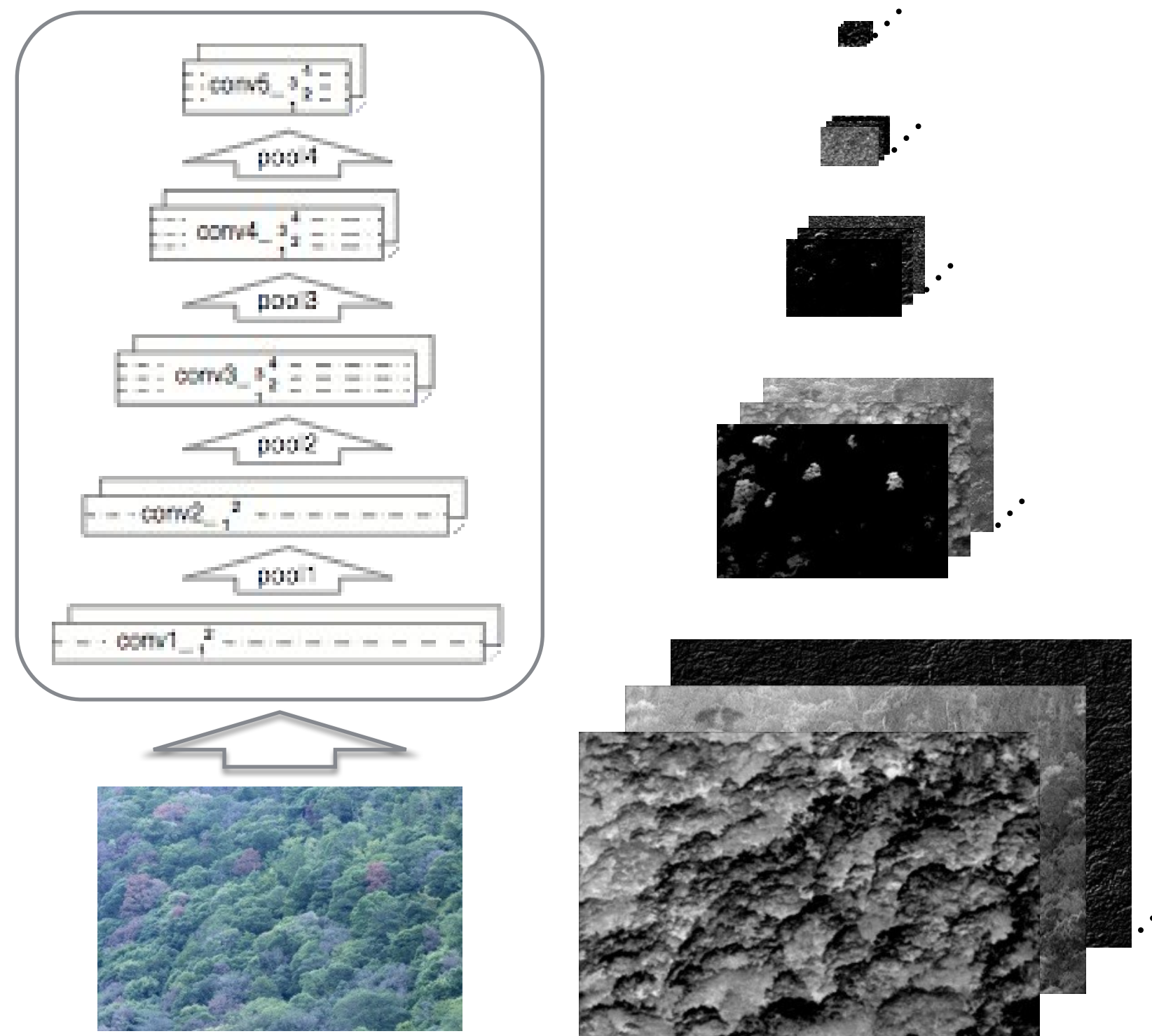


Portilla & Simoncelli
(2000)

CNN - Multiscale Filter Bank



CNN - Texture Features



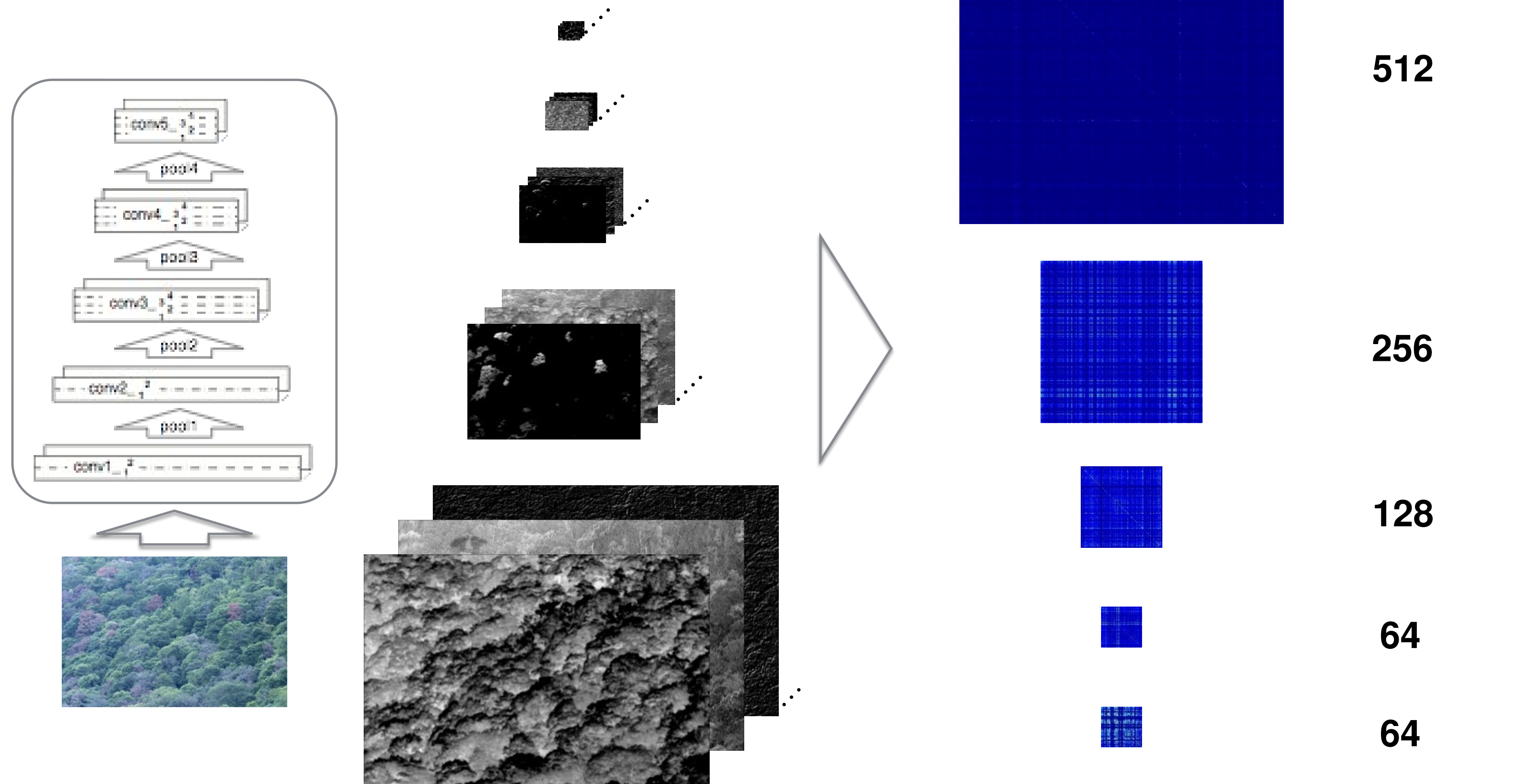
$$F = [\bar{f}_1, \bar{f}_2, \bar{f}_3, \dots, \bar{f}_N]^T$$

$$G = FF^T$$

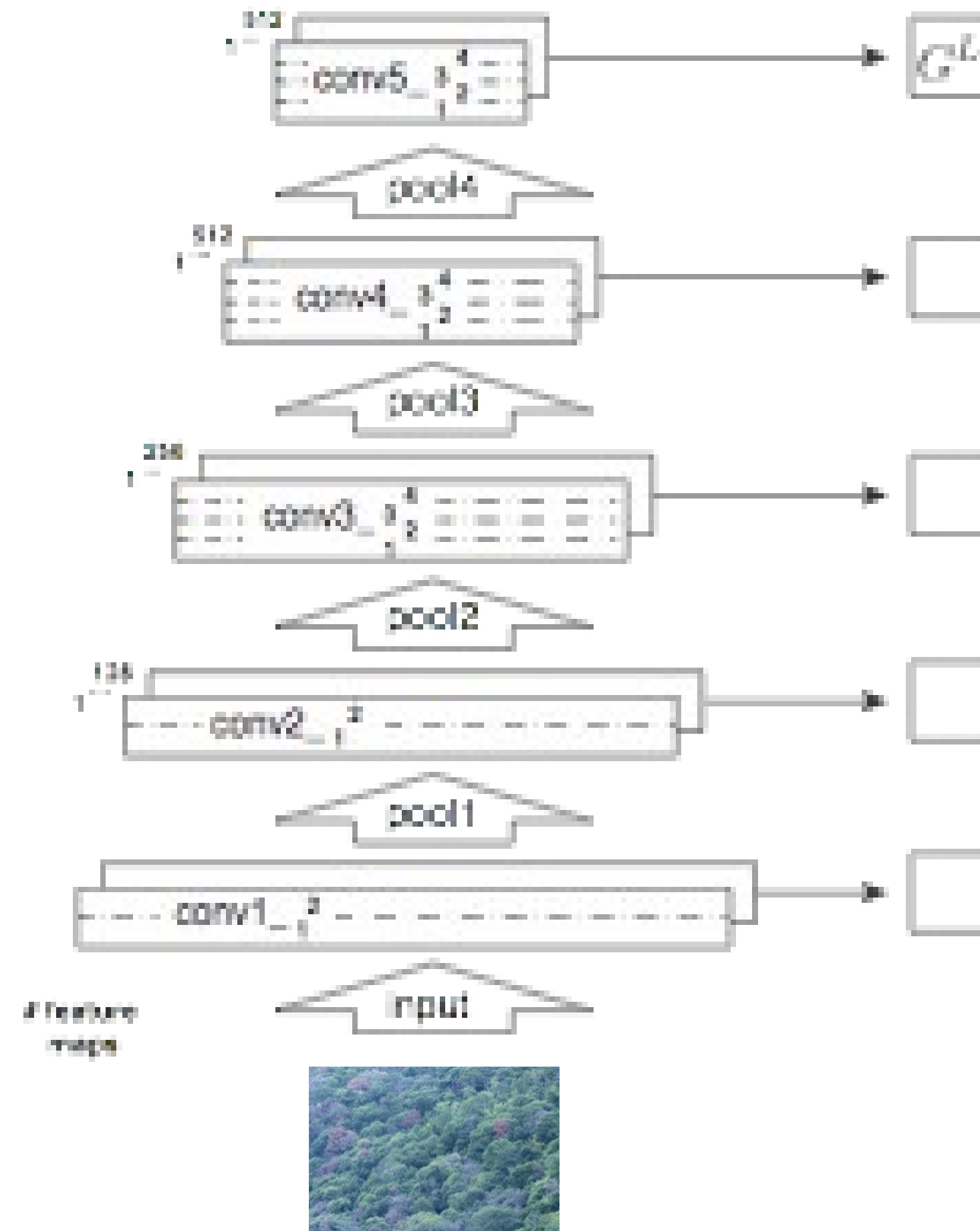
$$= \begin{pmatrix} \langle \bar{f}_1, \bar{f}_1 \rangle & \dots & \langle \bar{f}_1, \bar{f}_N \rangle \\ \langle \bar{f}_2, \bar{f}_1 \rangle & & \vdots \\ \vdots & \ddots & \vdots \\ \langle \bar{f}_N, \bar{f}_1 \rangle & \dots & \langle \bar{f}_N, \bar{f}_N \rangle \end{pmatrix}$$

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

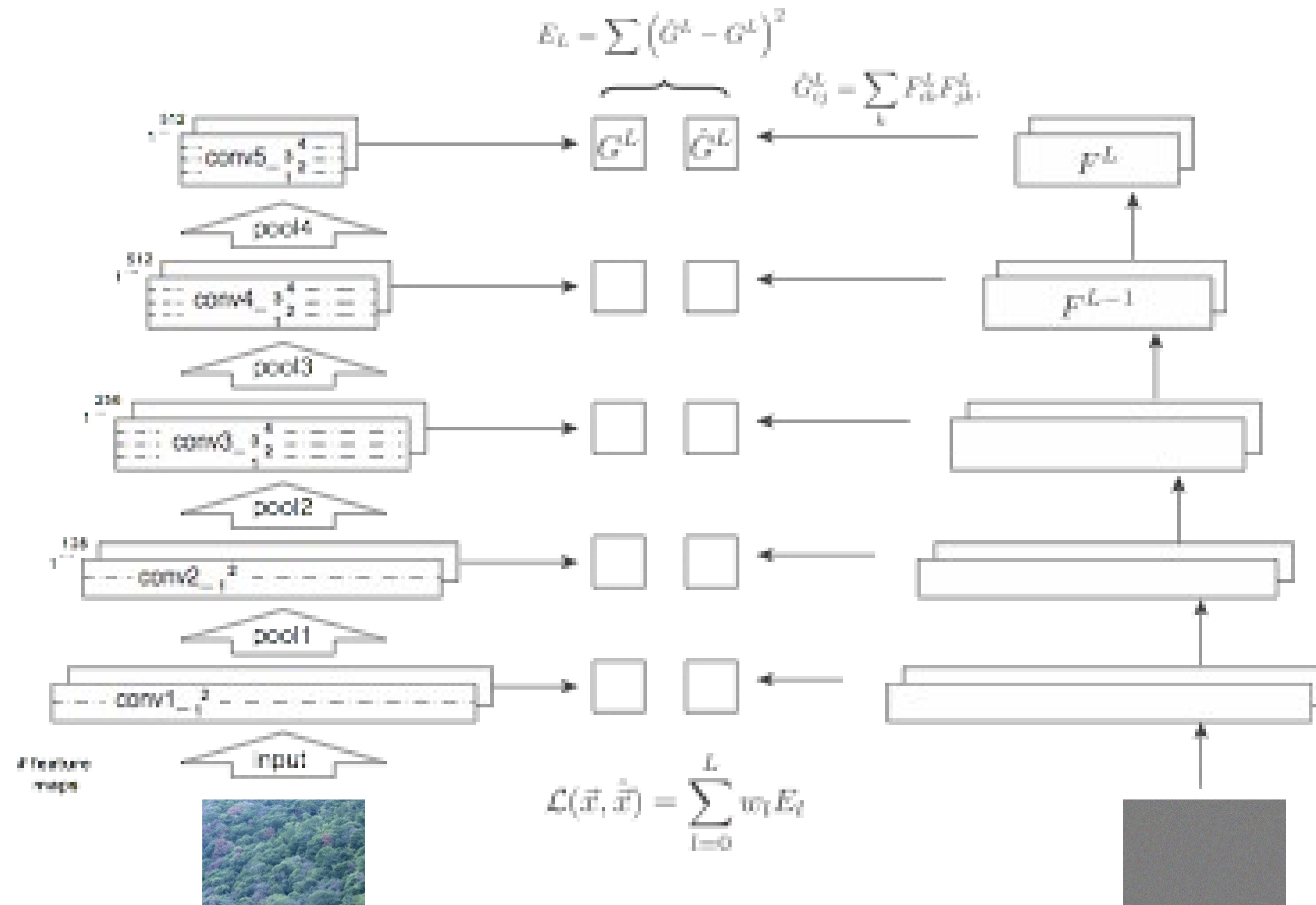
CNN-Texture Features



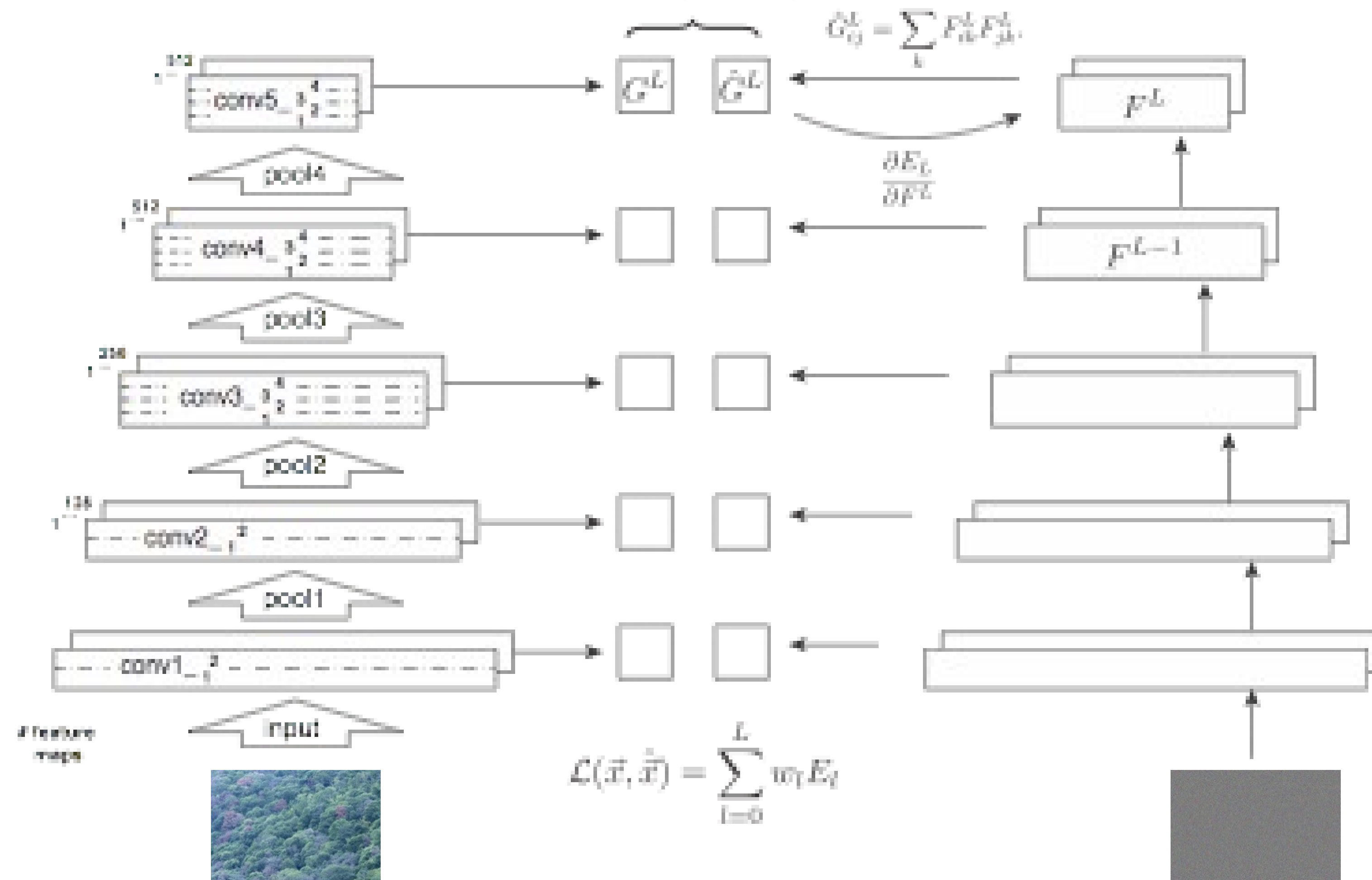
Texture Synthesis



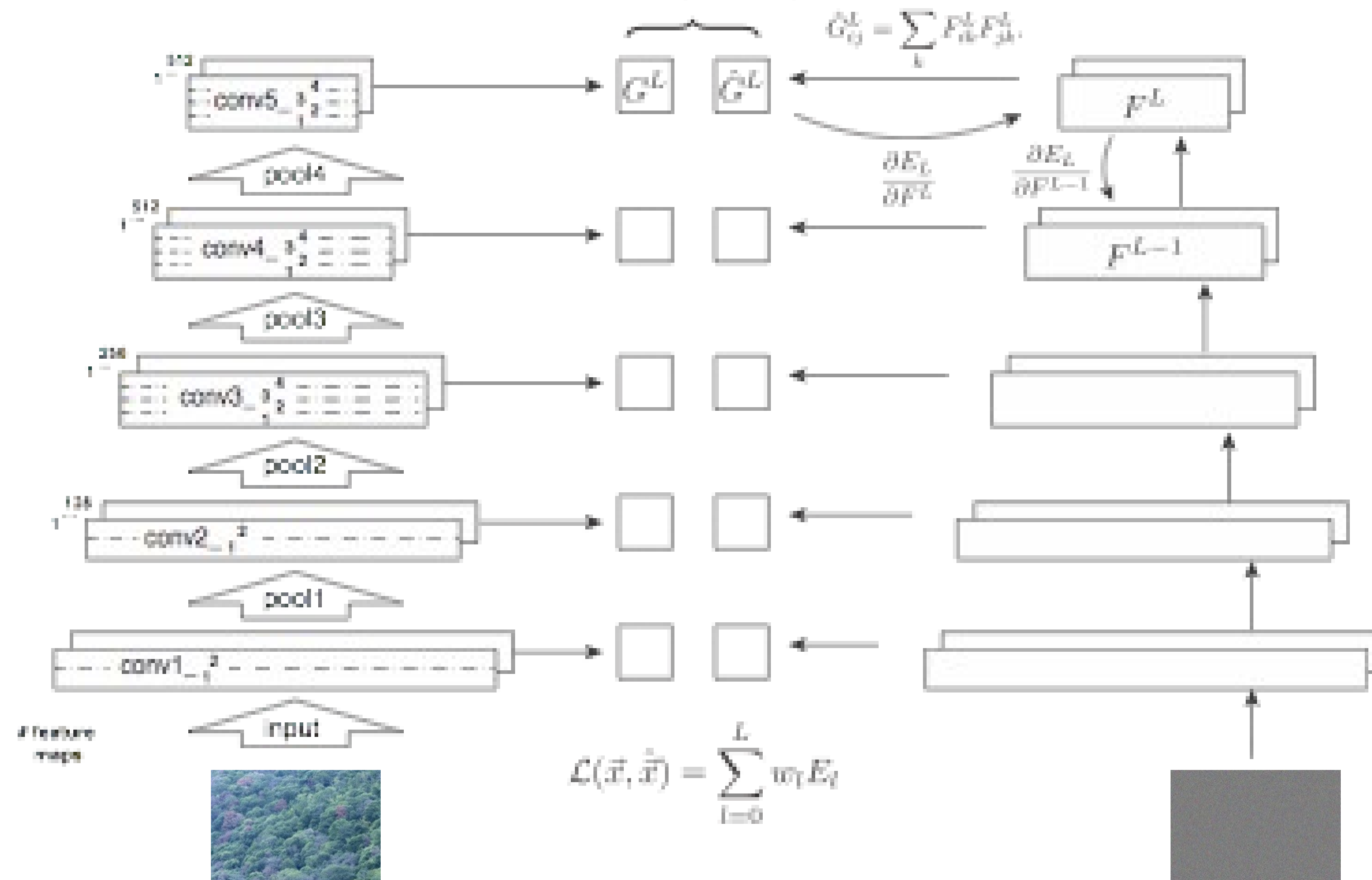
Texture Synthesis



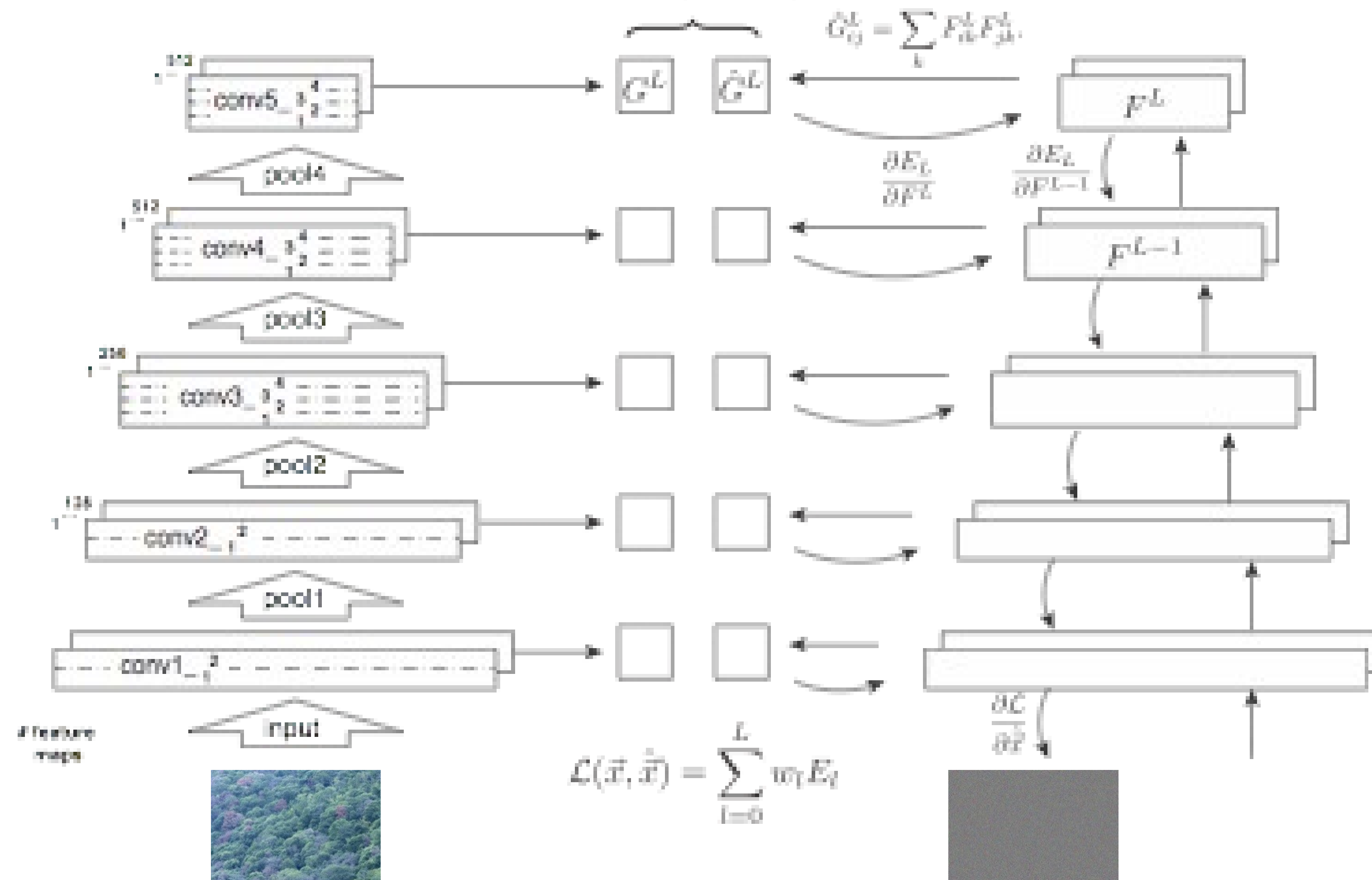
Texture Synthesis



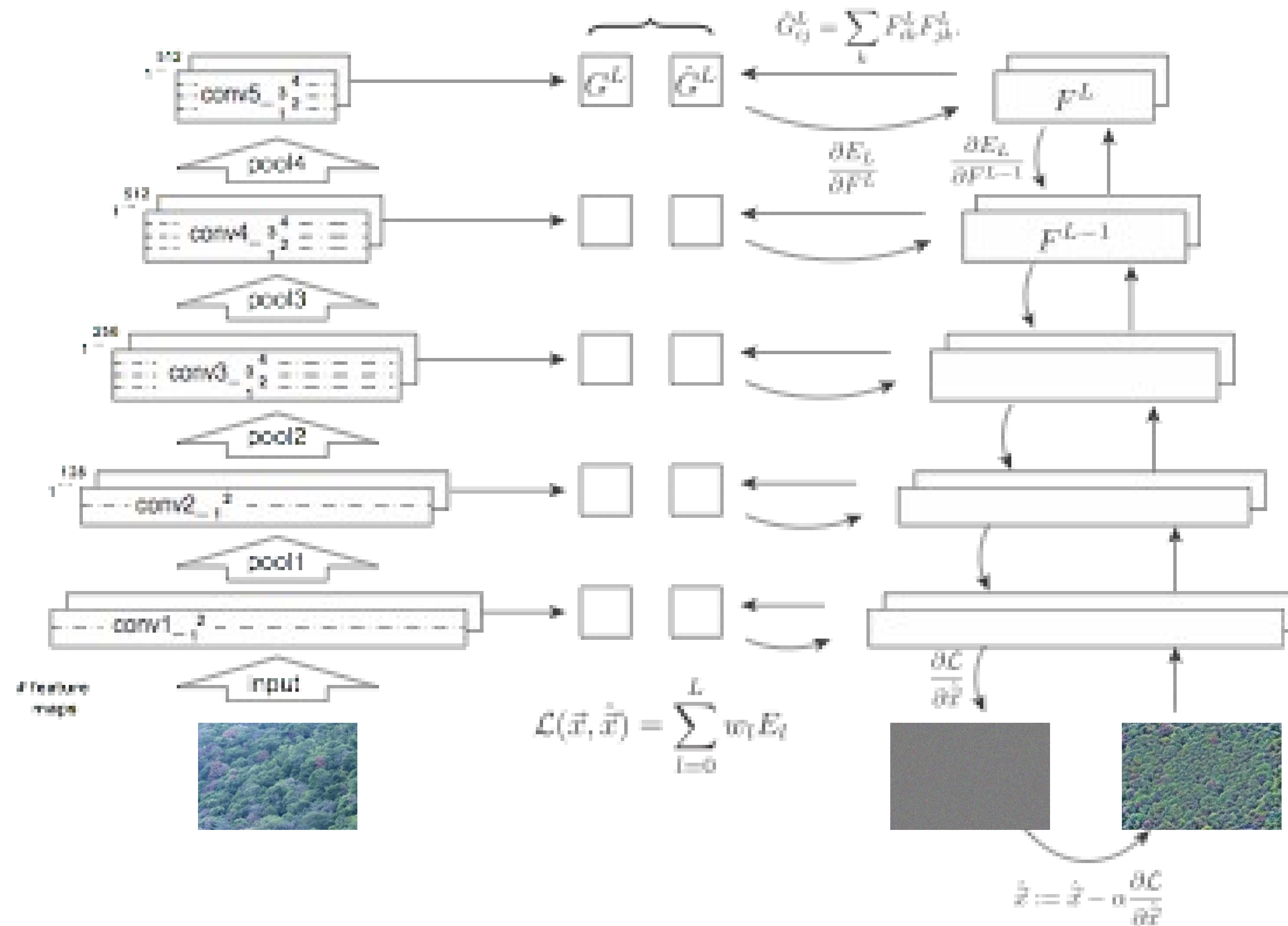
Texture Synthesis



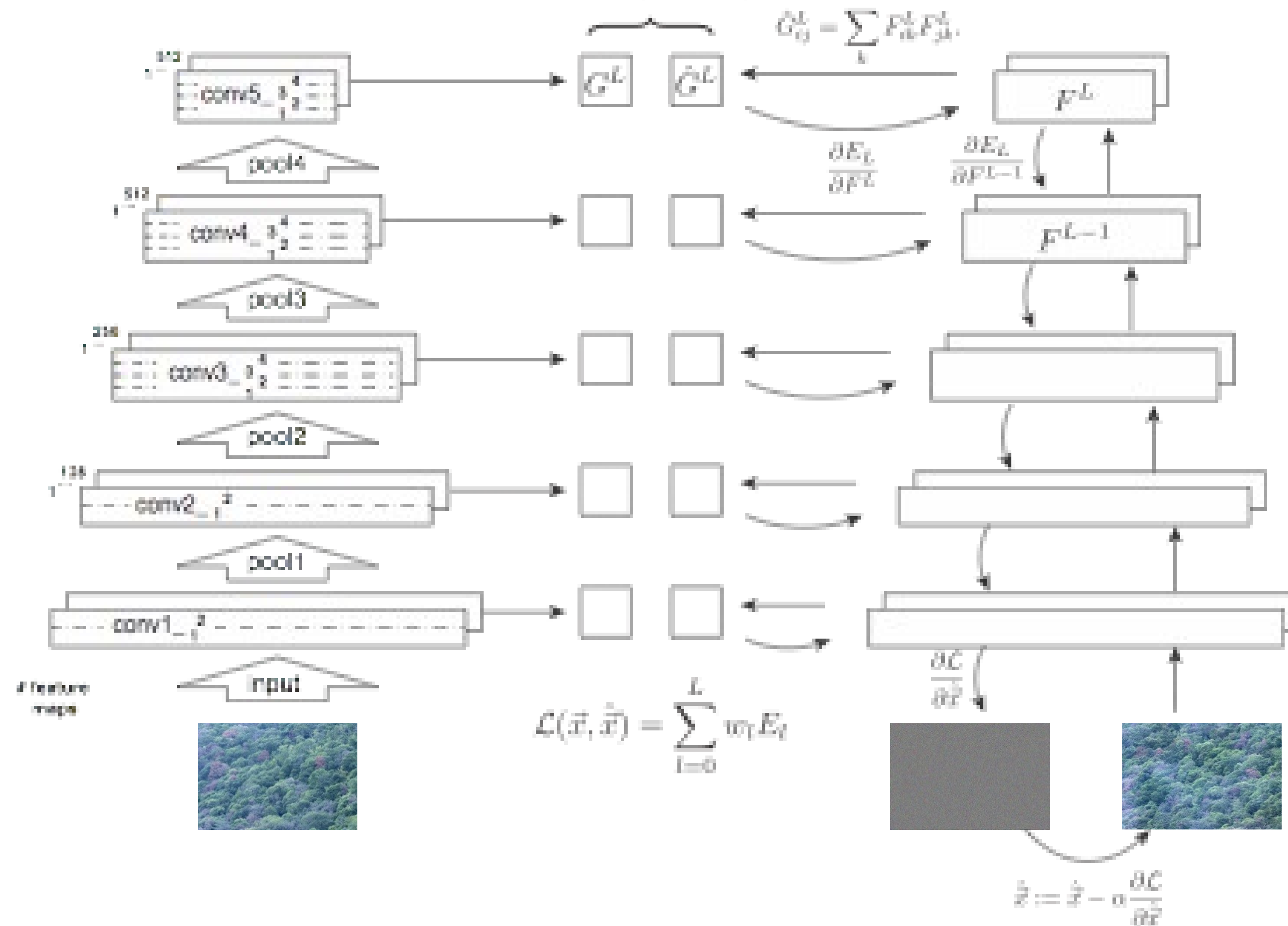
Texture Synthesis



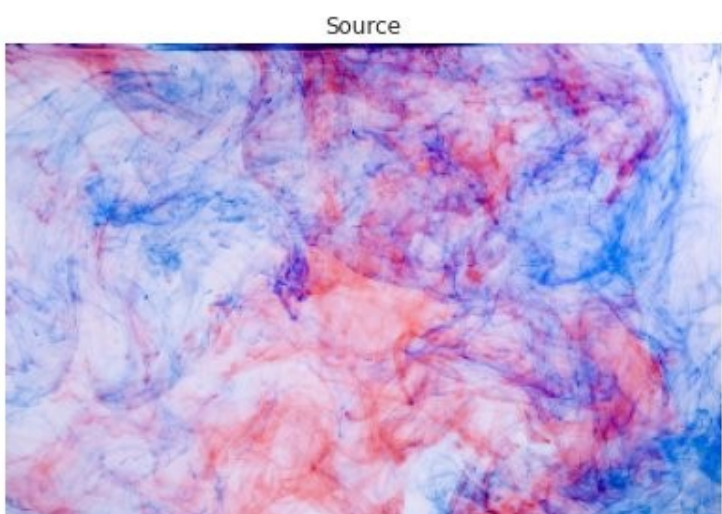
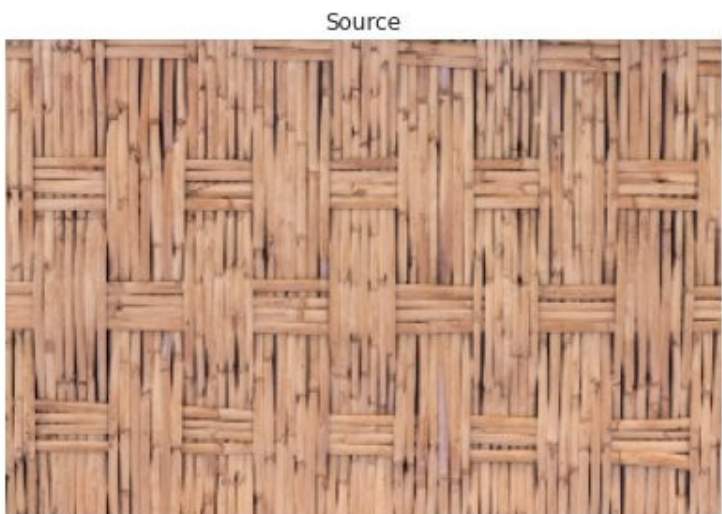
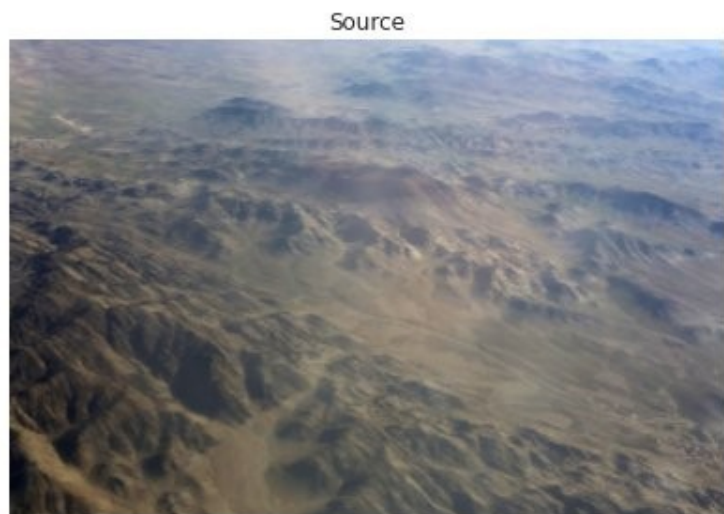
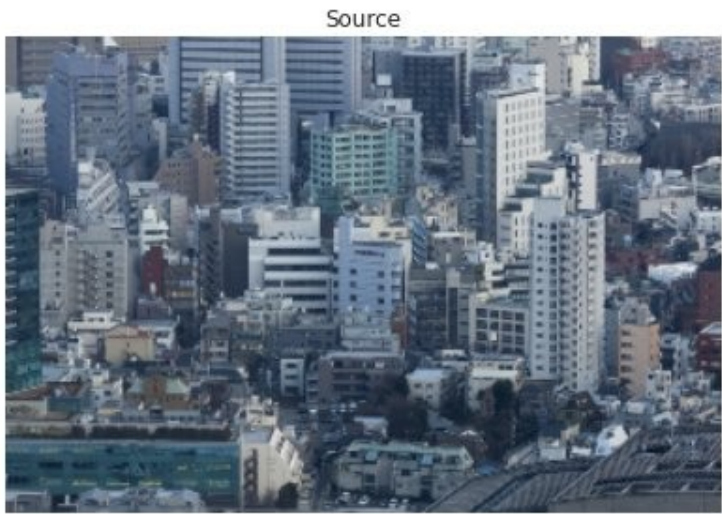
Texture Synthesis



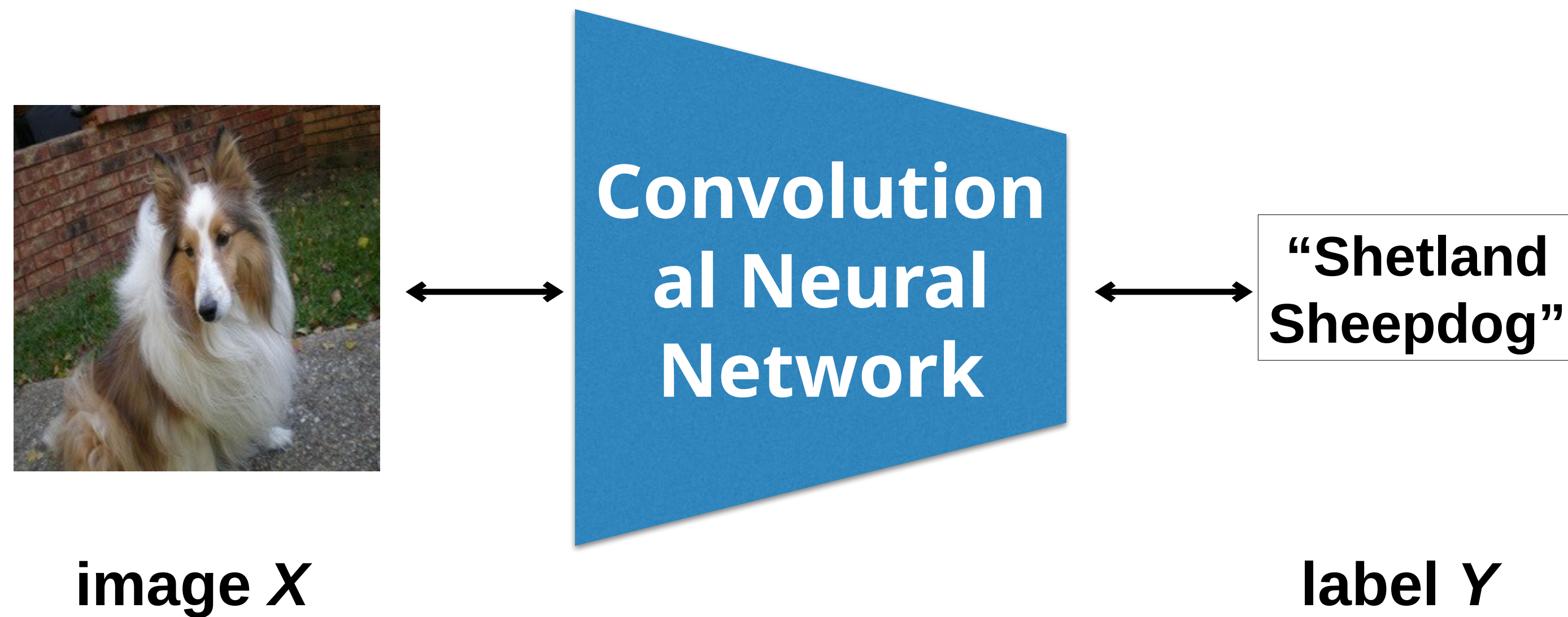
Texture Synthesis



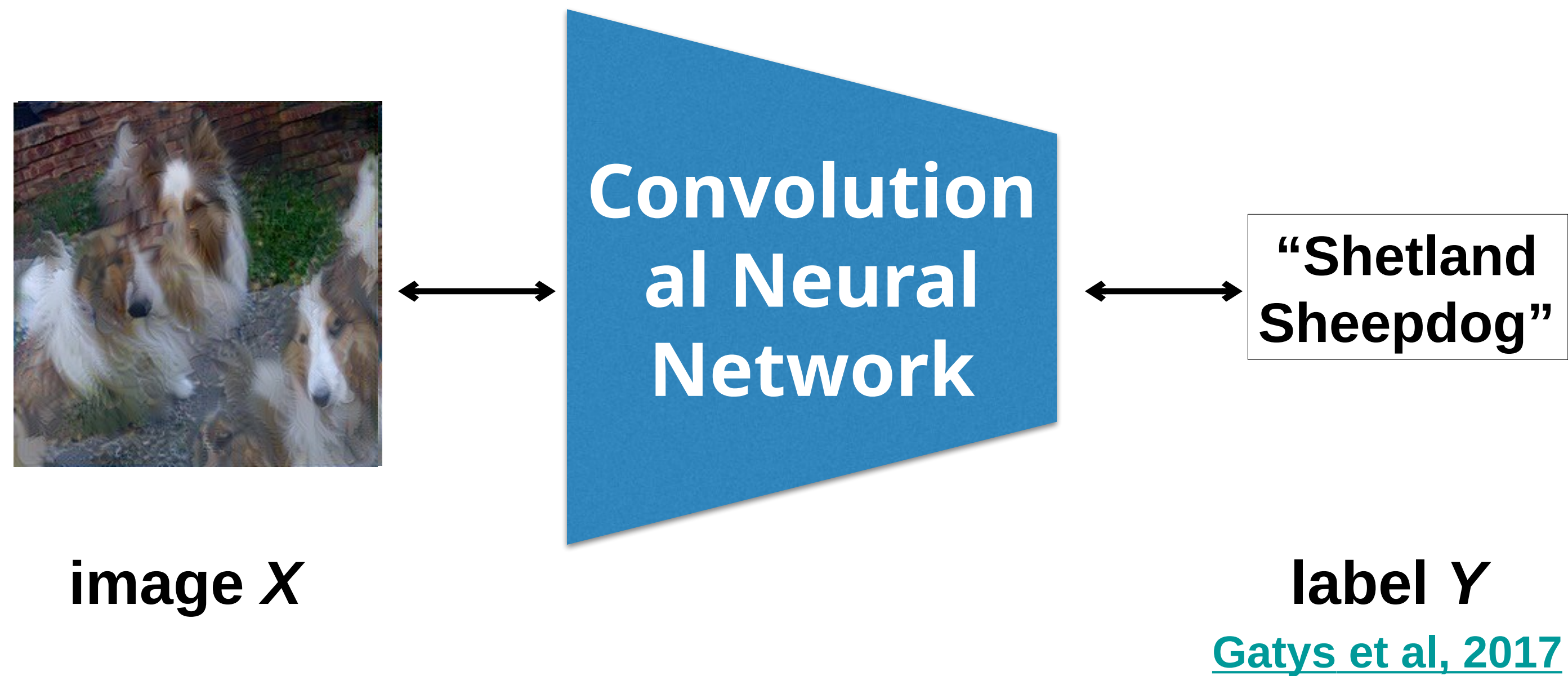
Gatys Texture Synthesis



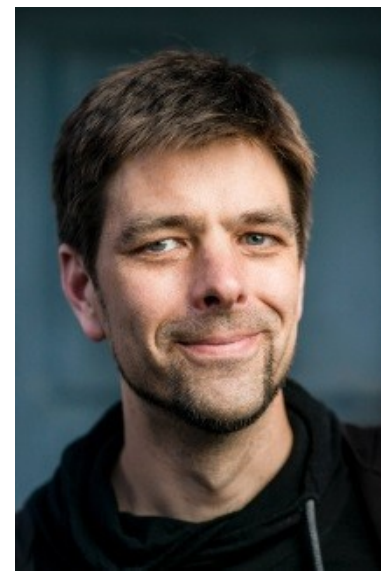
ImageNet Recognition is just Texture Recognition



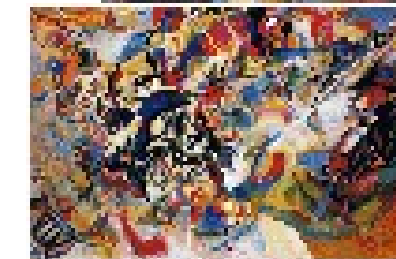
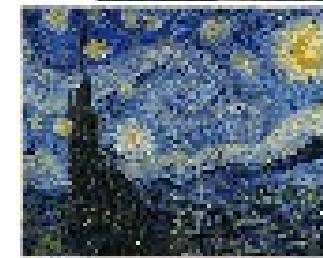
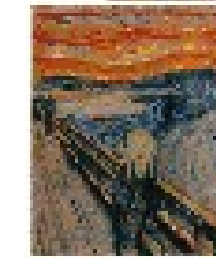
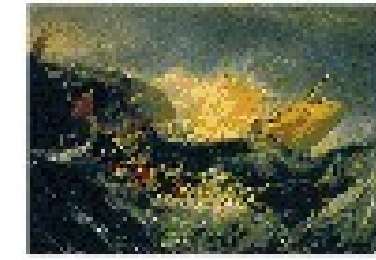
ImageNet Recognition is just Texture Recognition



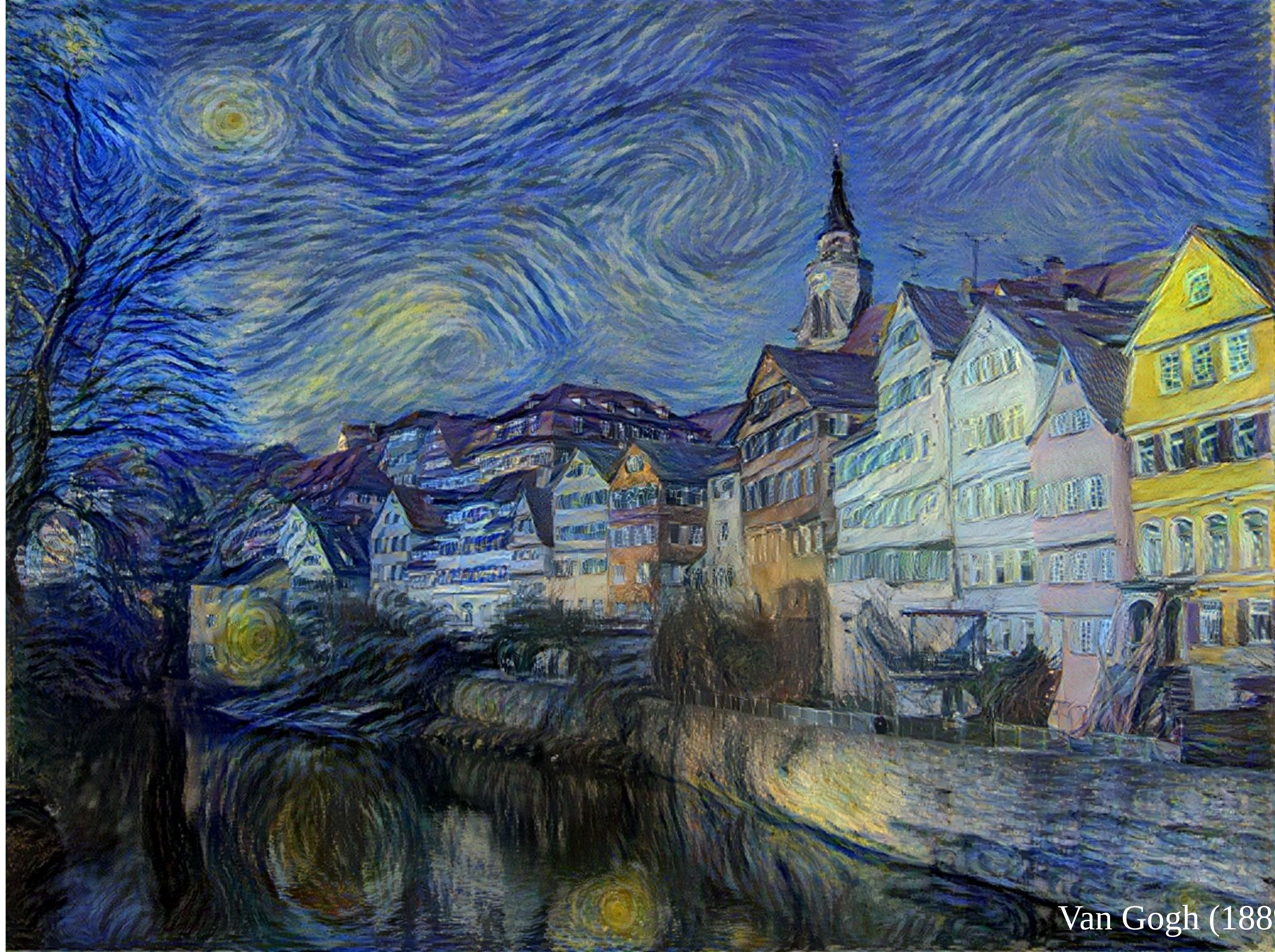
A Neural Algorithm of Artistic Style



Gatys, Ecker, Bethge (arXiv 2015)







Van Gogh (188



Picasso (1910)



Munch (1893)

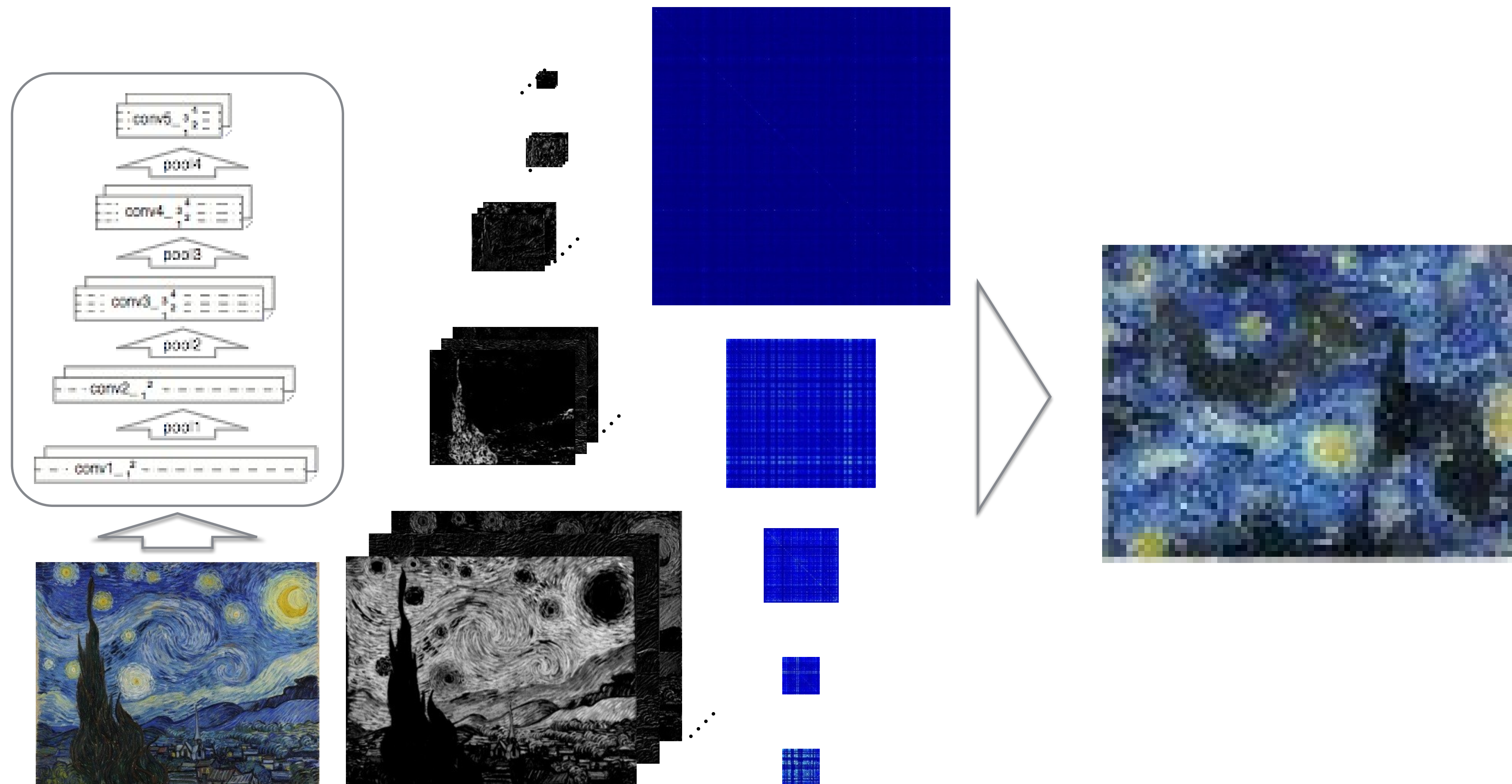


Turner (1805)



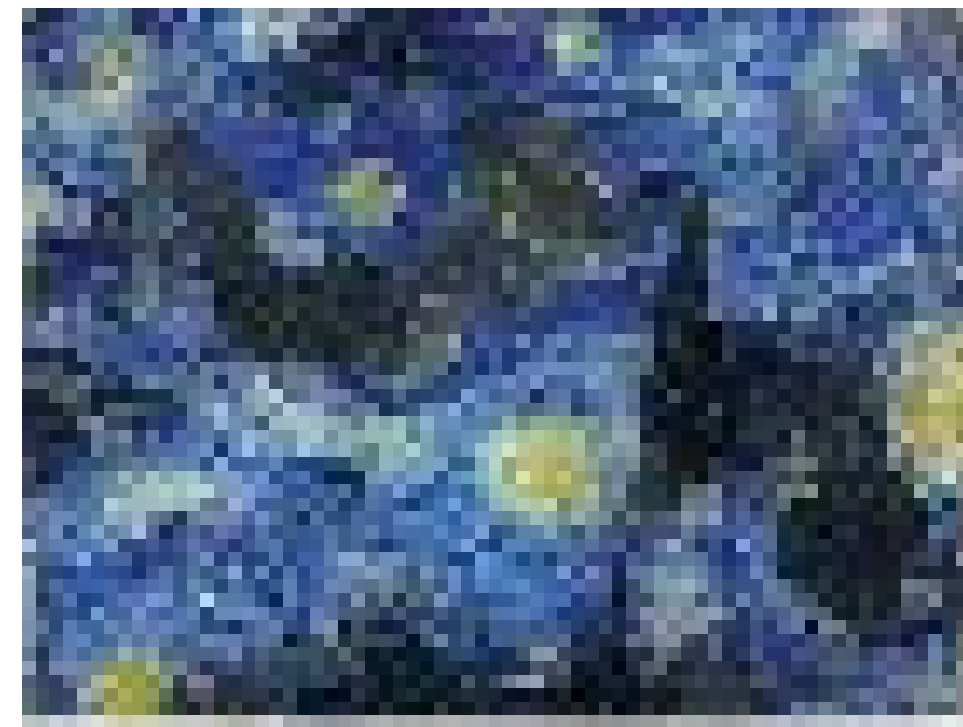
Kandinsky (1911)

CNN - Texture Synthesis

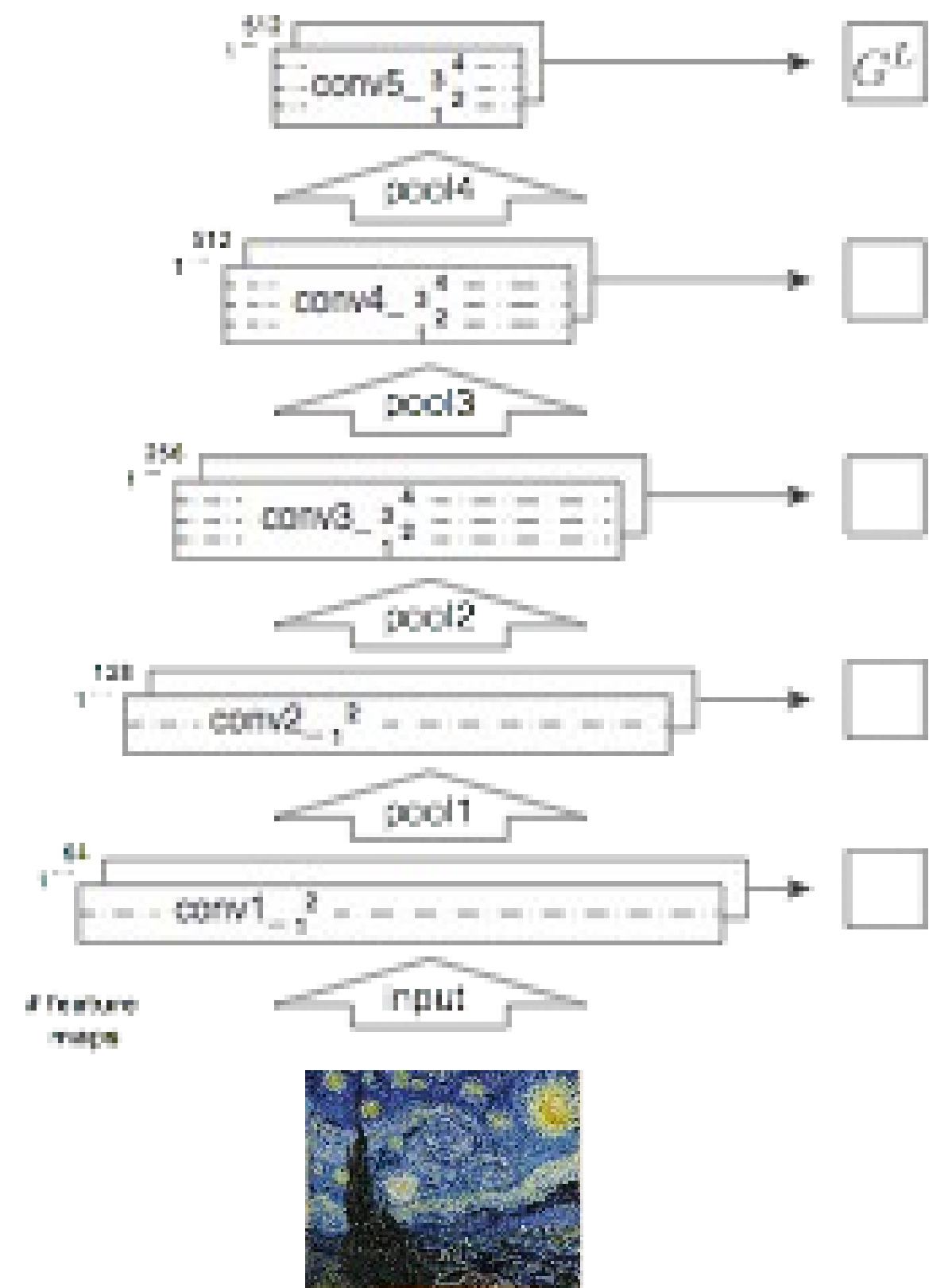


Gatys et al. (NIPS
2015)

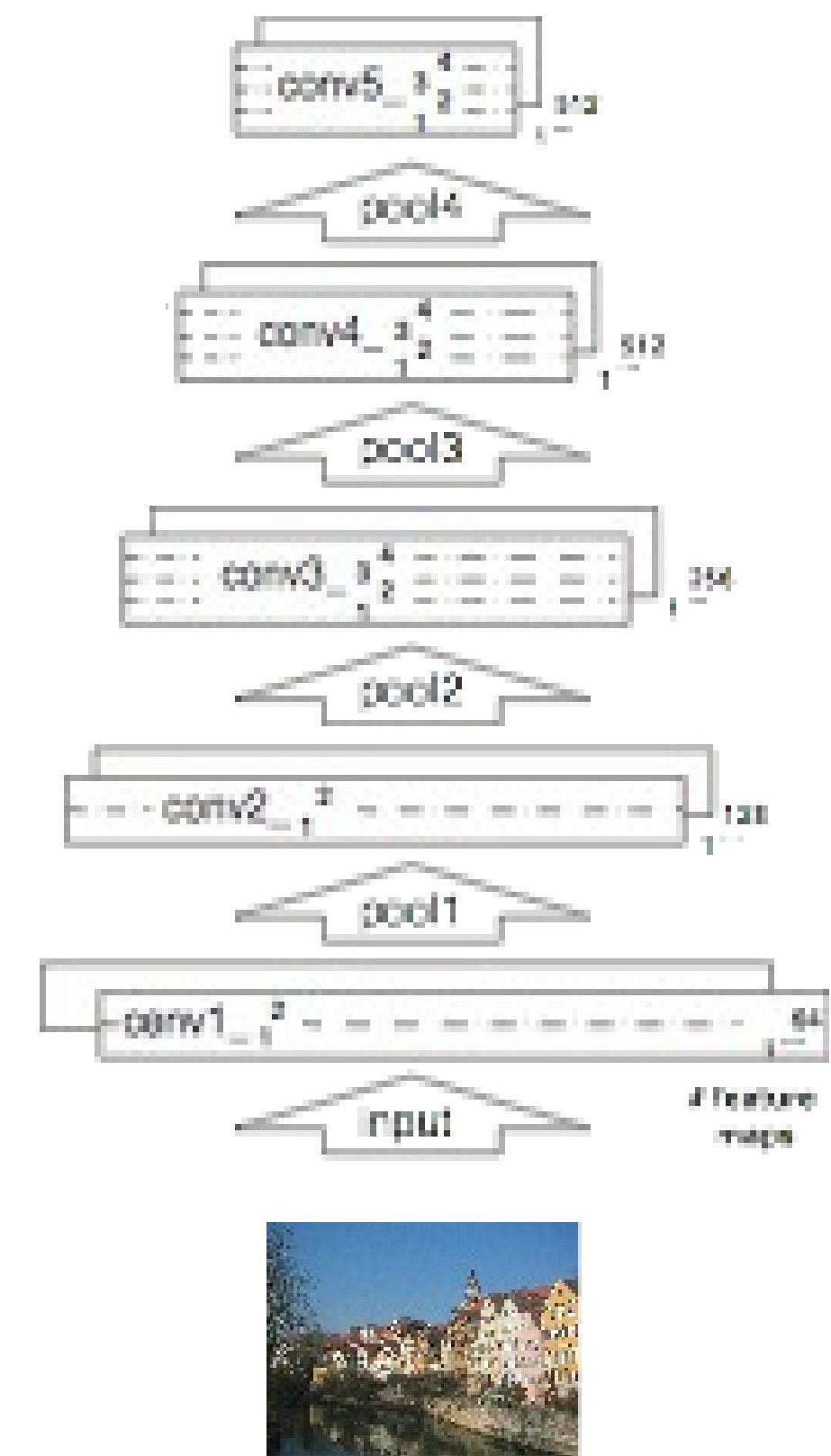
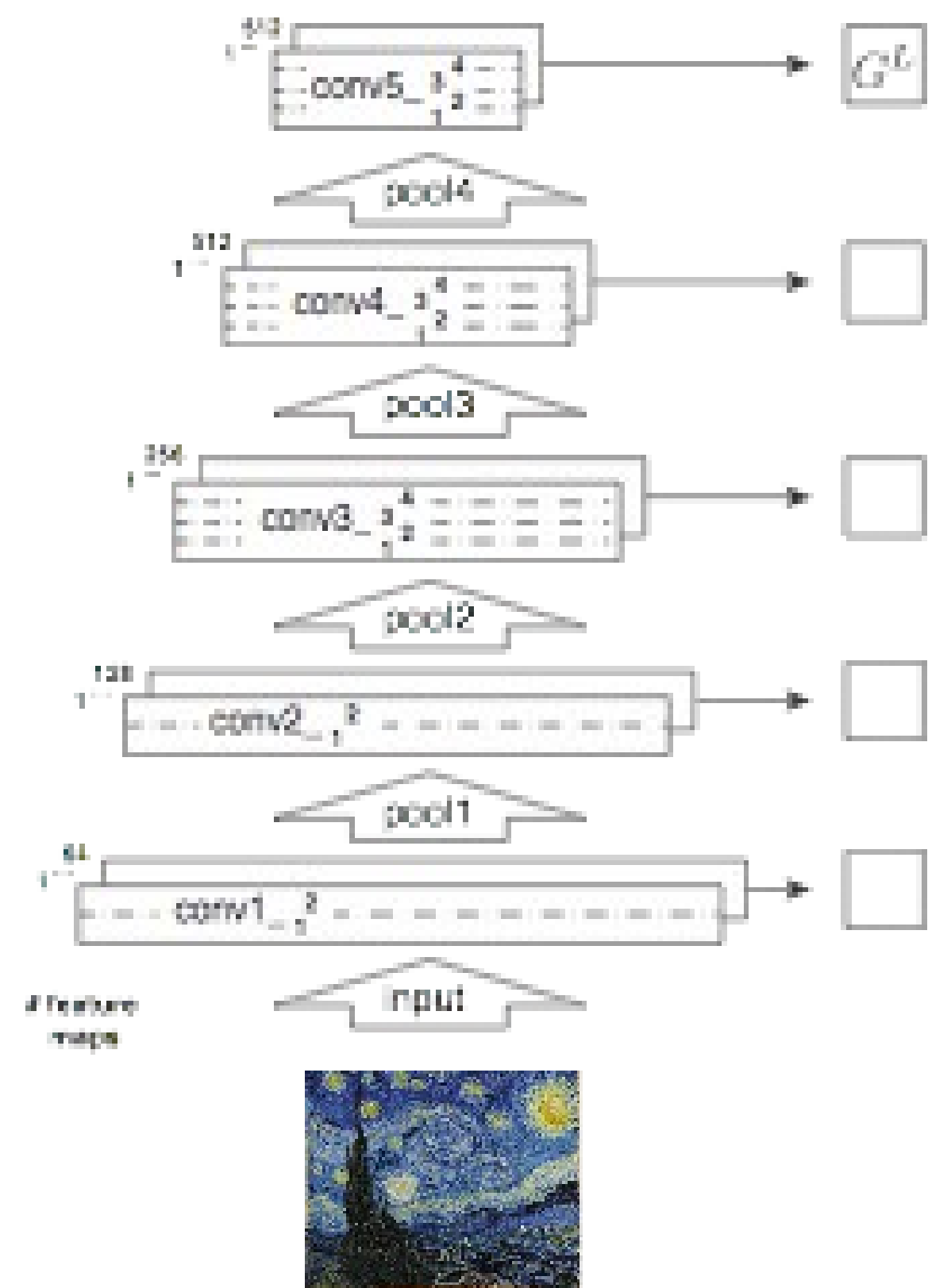
Artistic Style Transfer



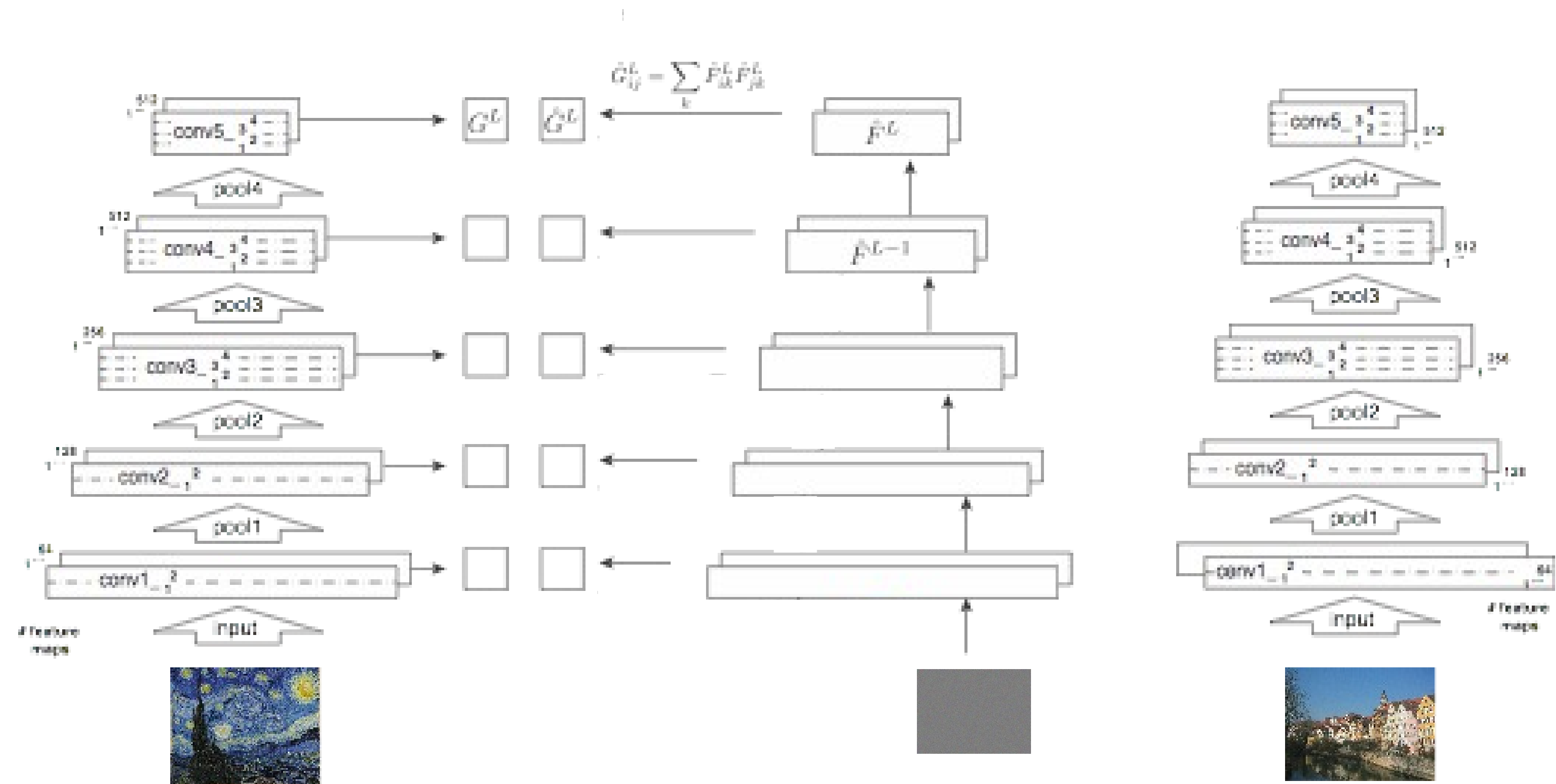
Artistic Style Transfer



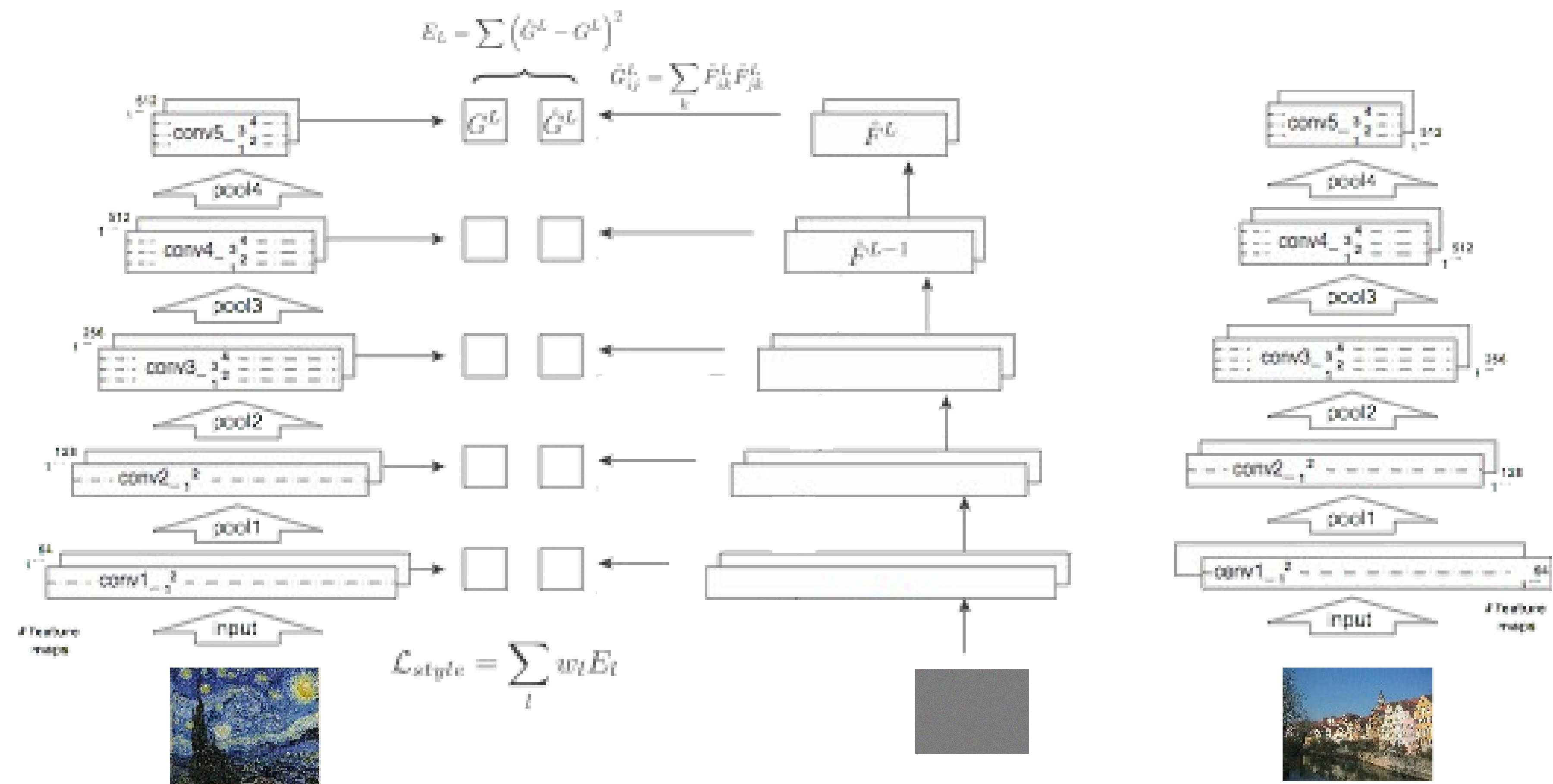
Artistic Style Transfer



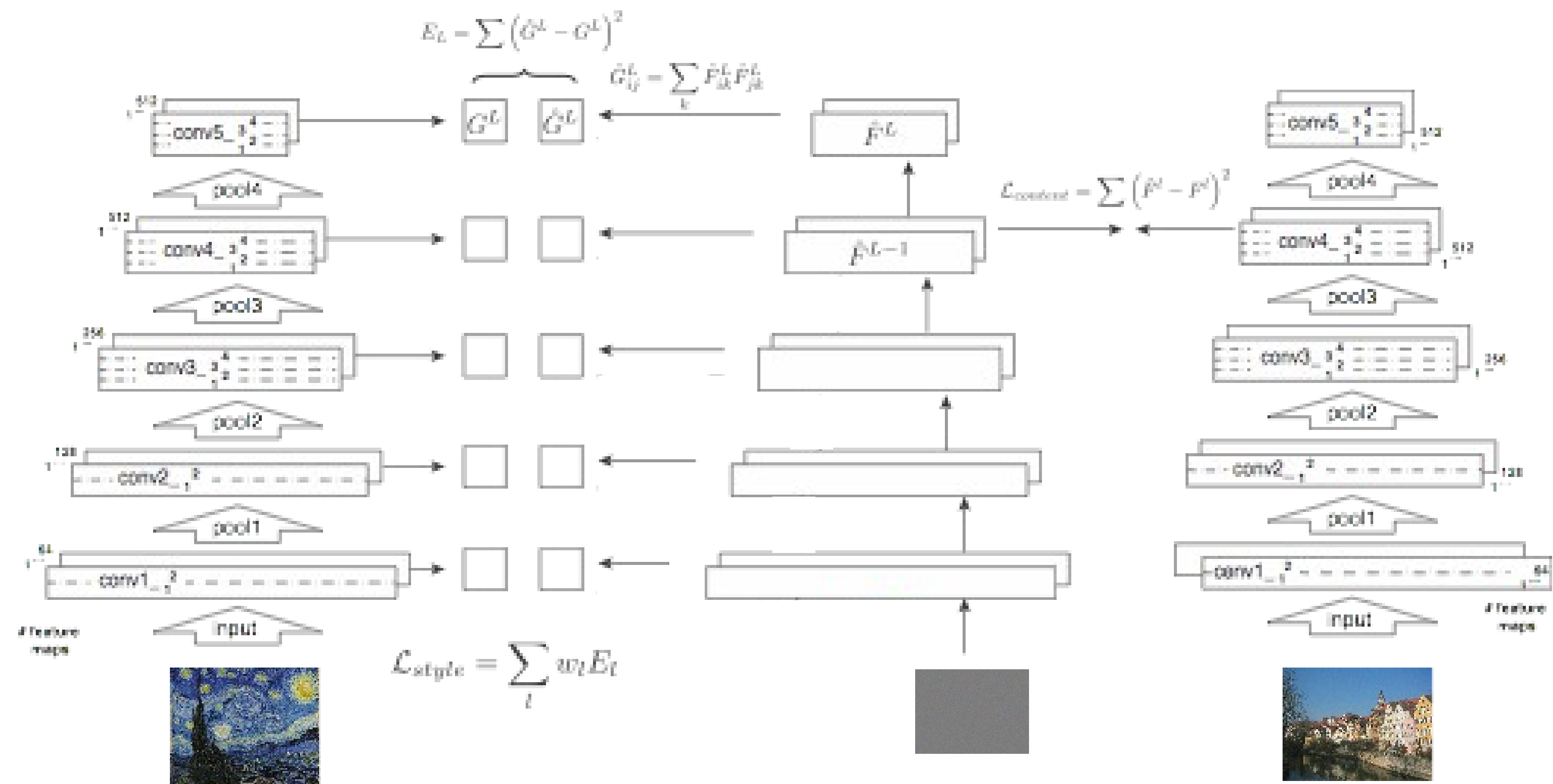
Artistic Style Transfer



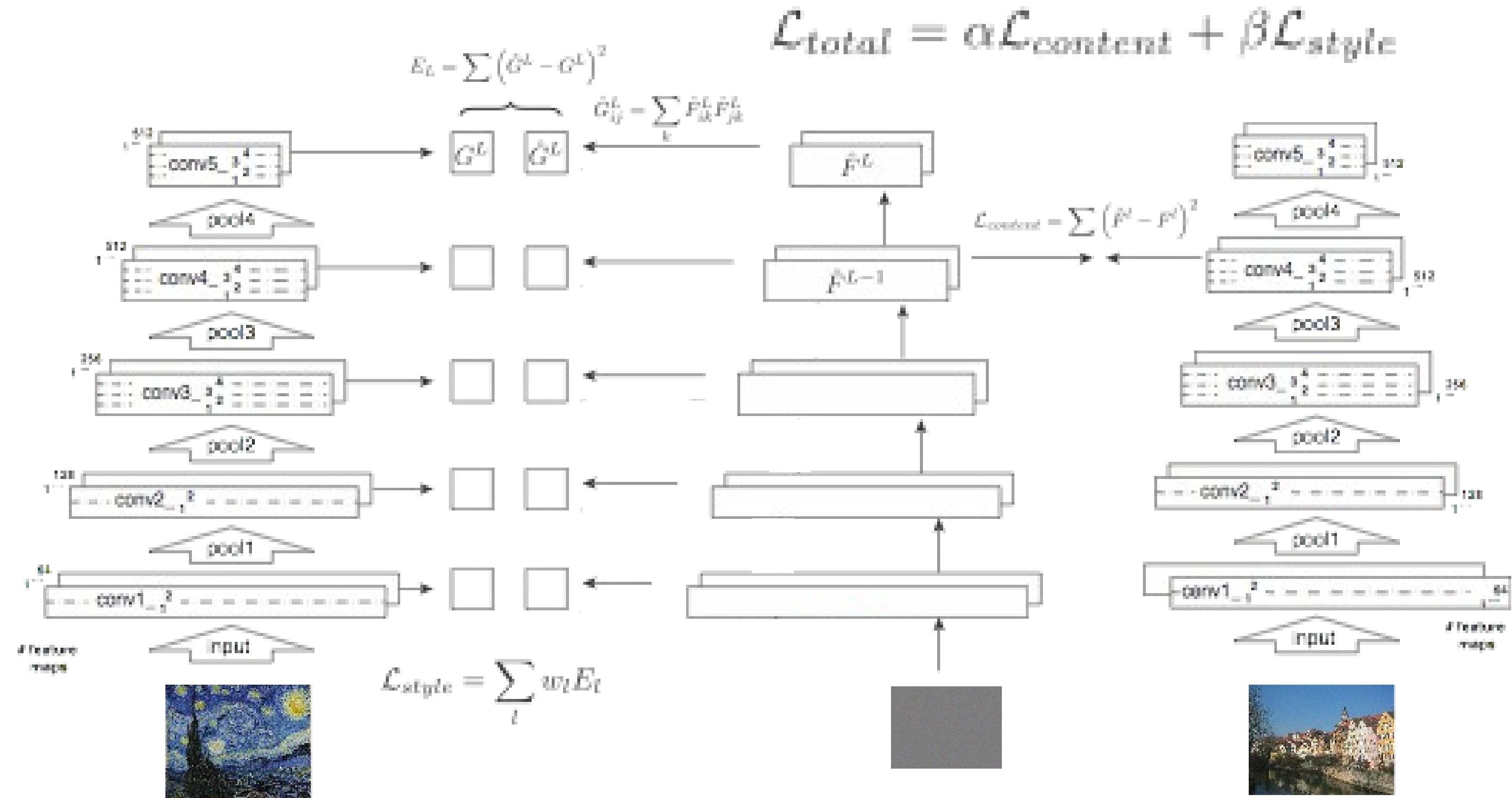
Artistic Style Transfer



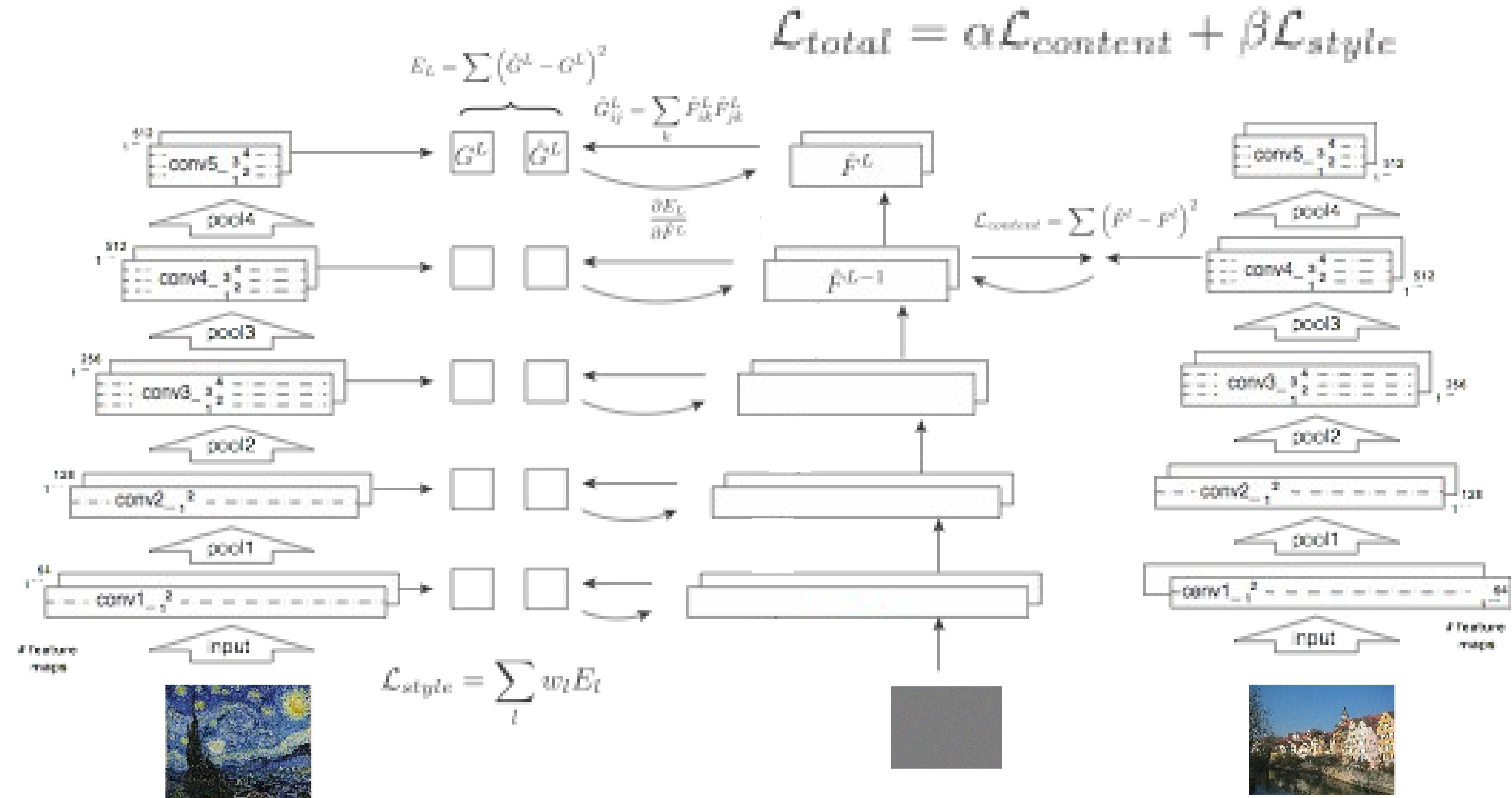
Artistic Style Transfer



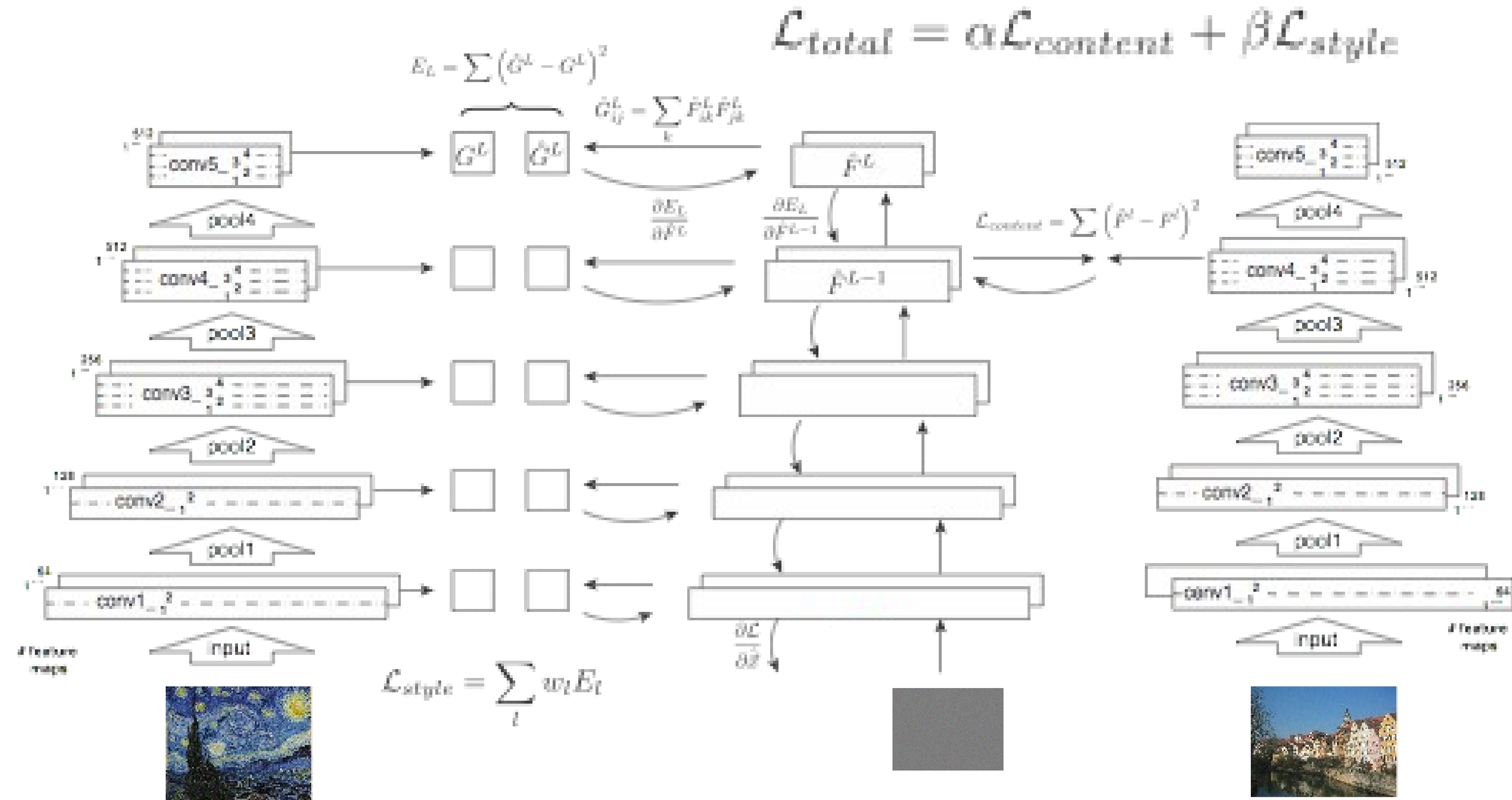
Artistic Style Transfer



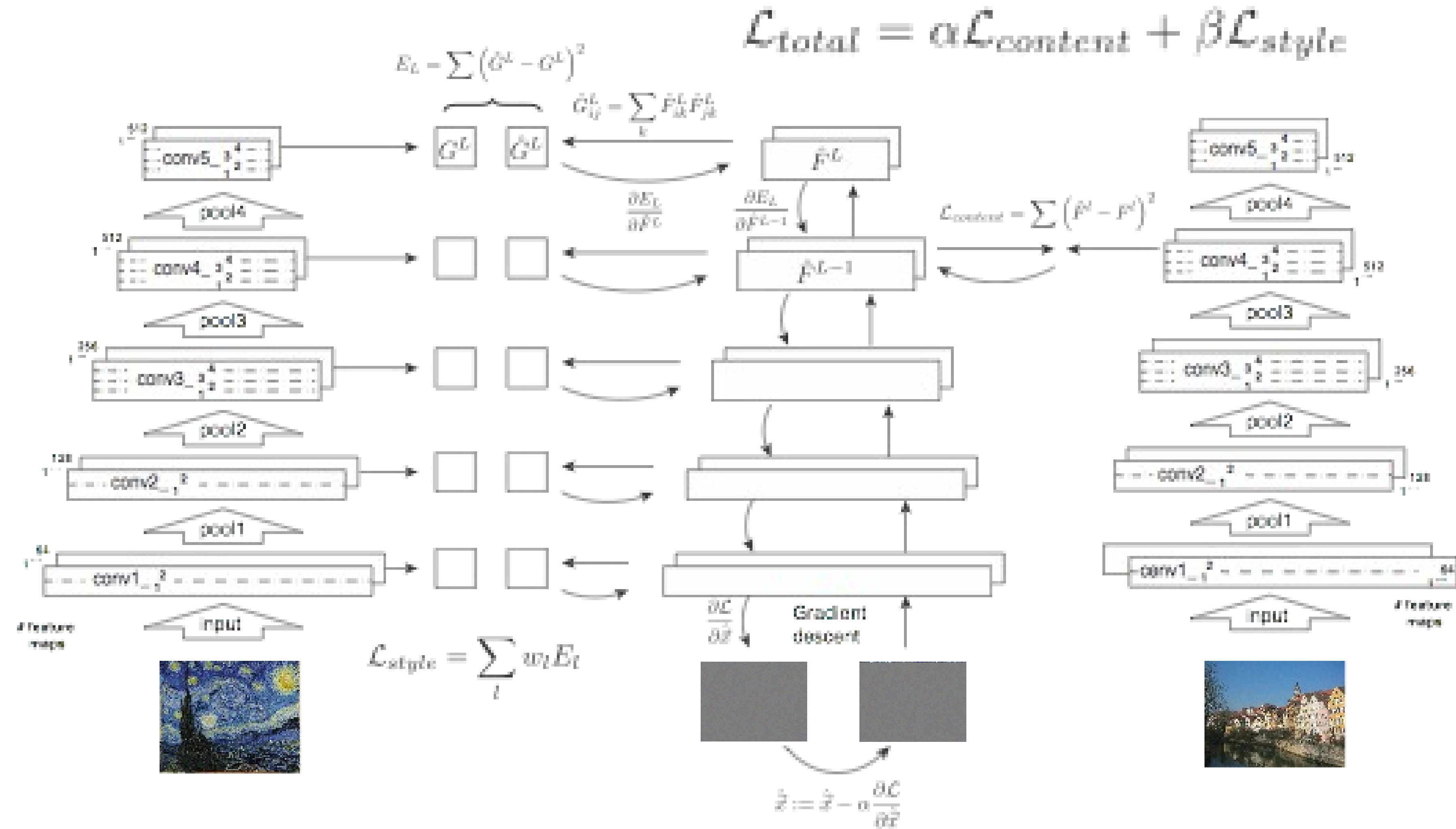
Artistic Style Transfer



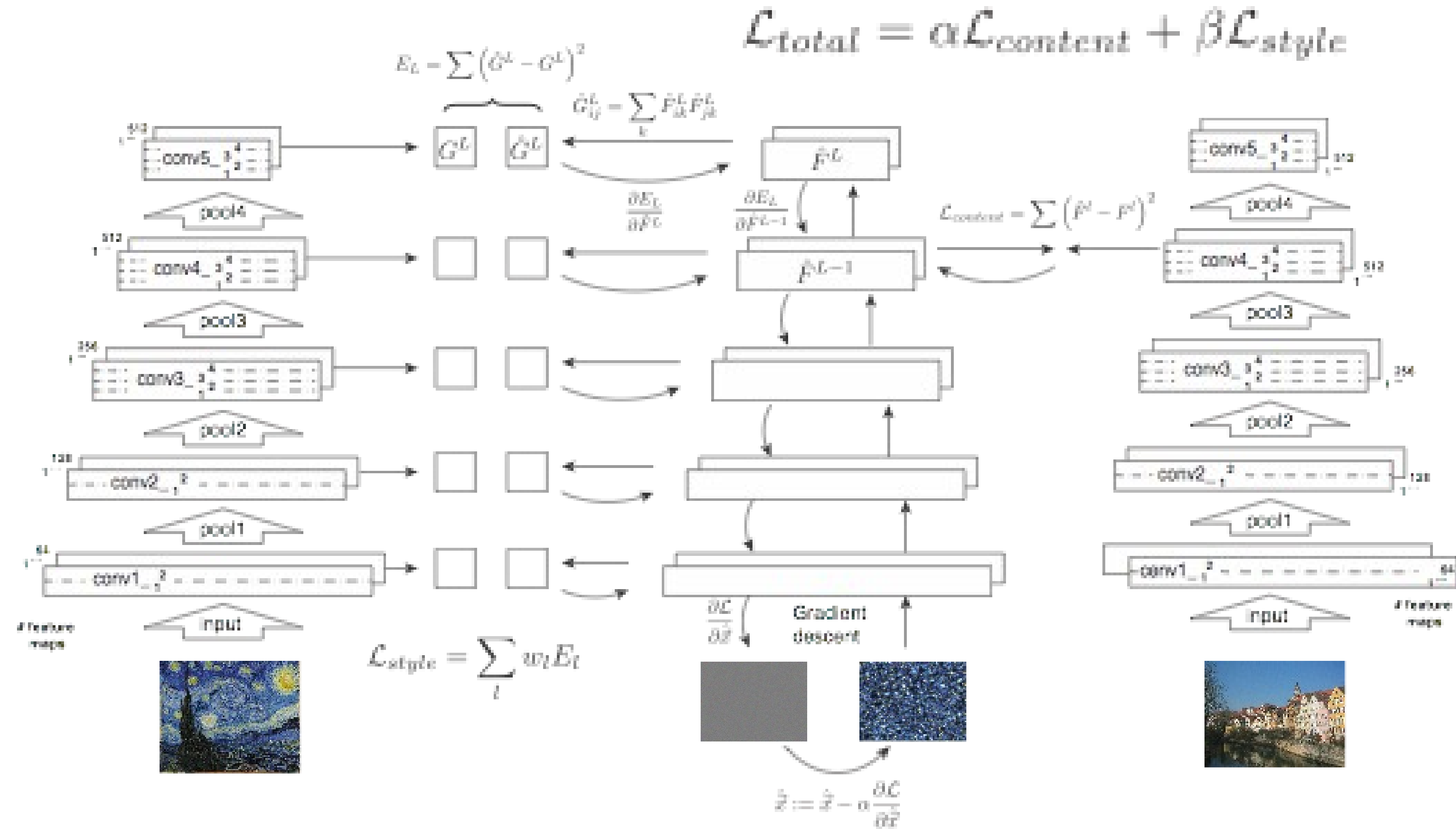
Artistic Style Transfer



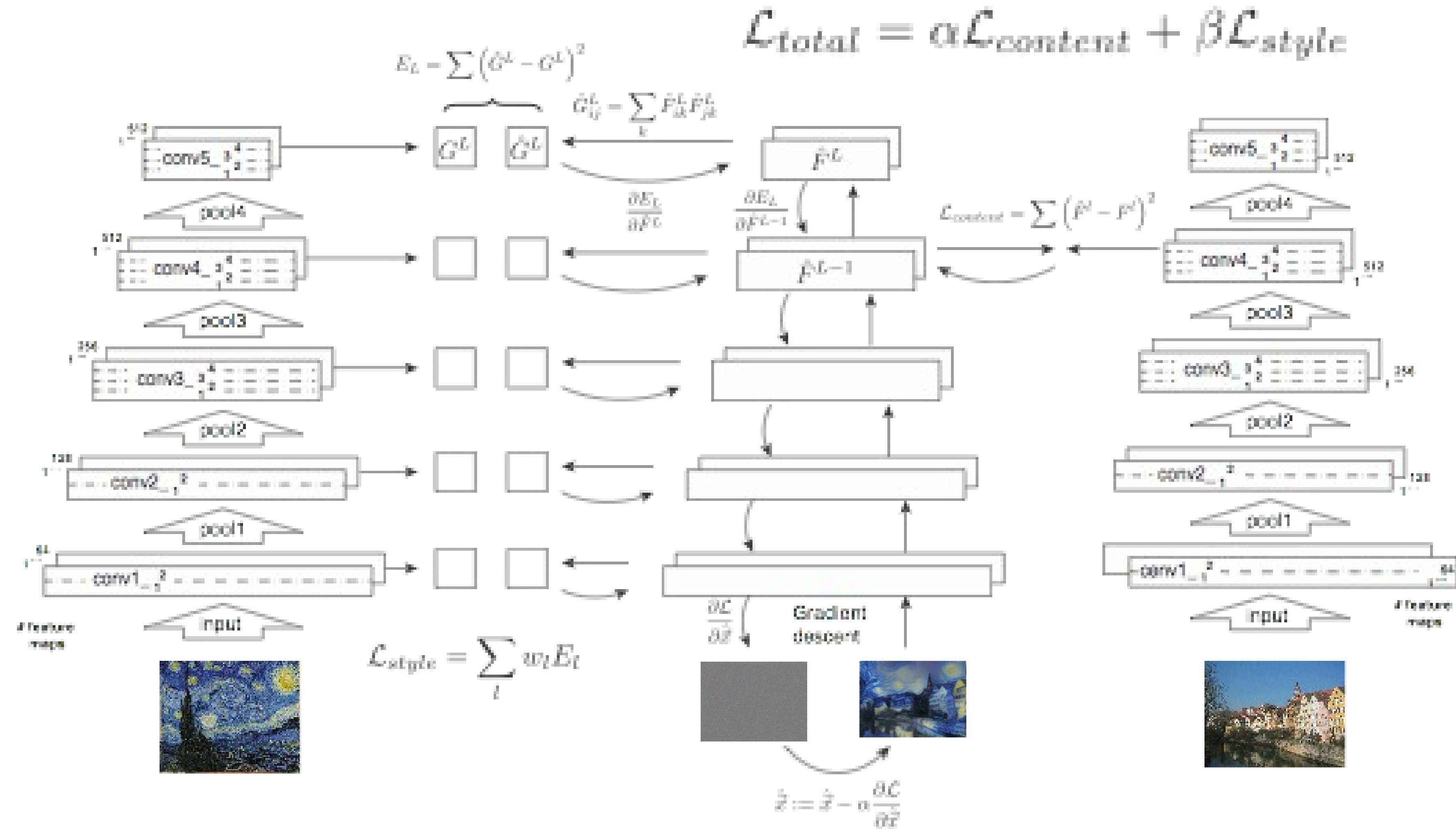
Artistic Style Transfer



Artistic Style Transfer



Artistic Style Transfer



General Style Transfer

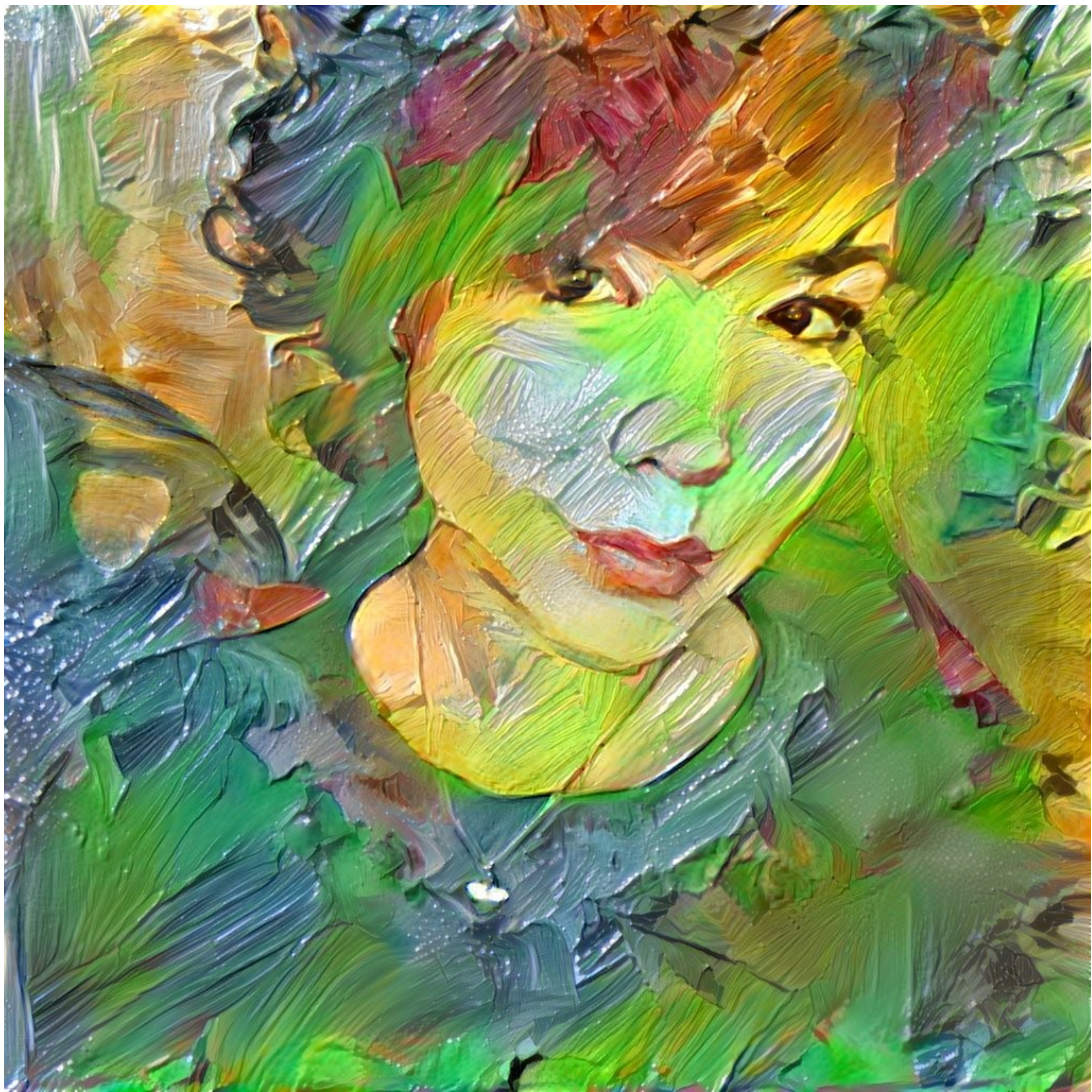


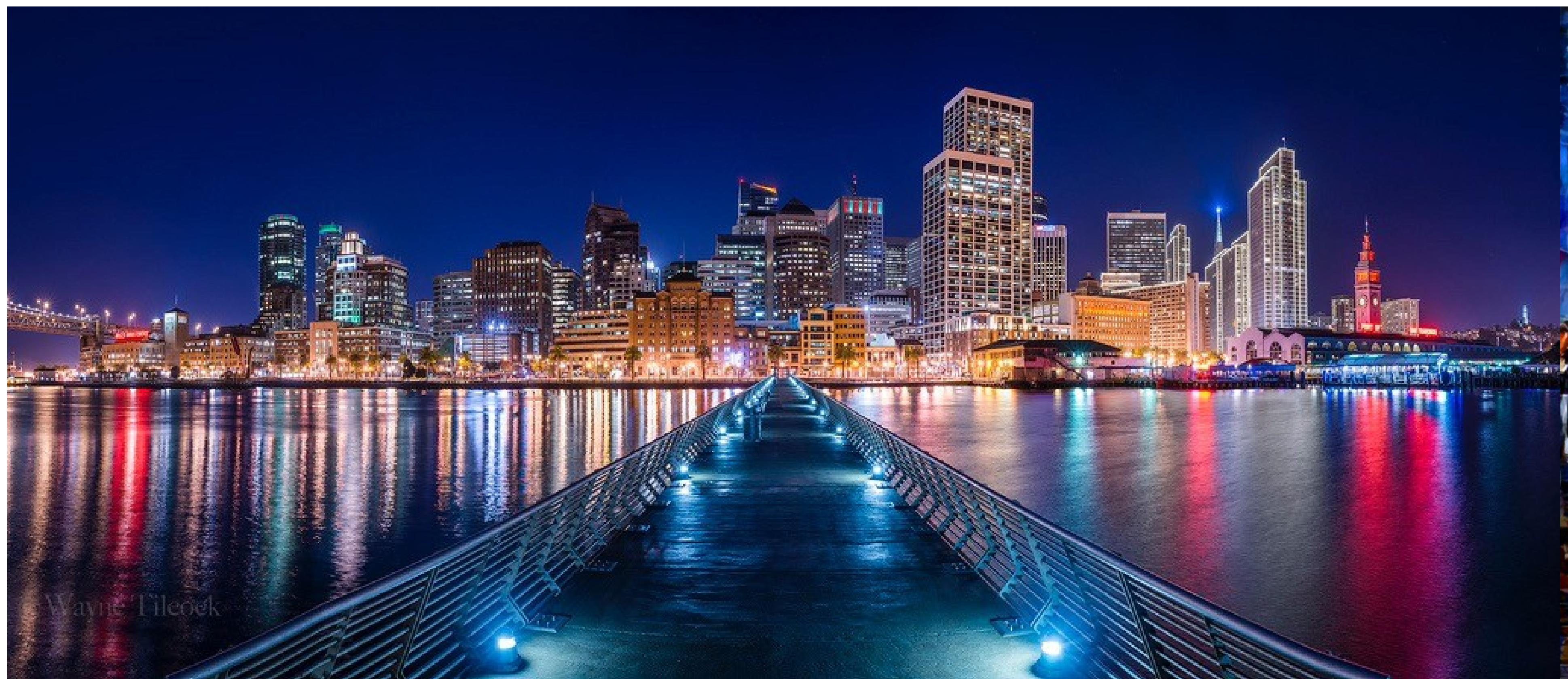
General Style Transfer



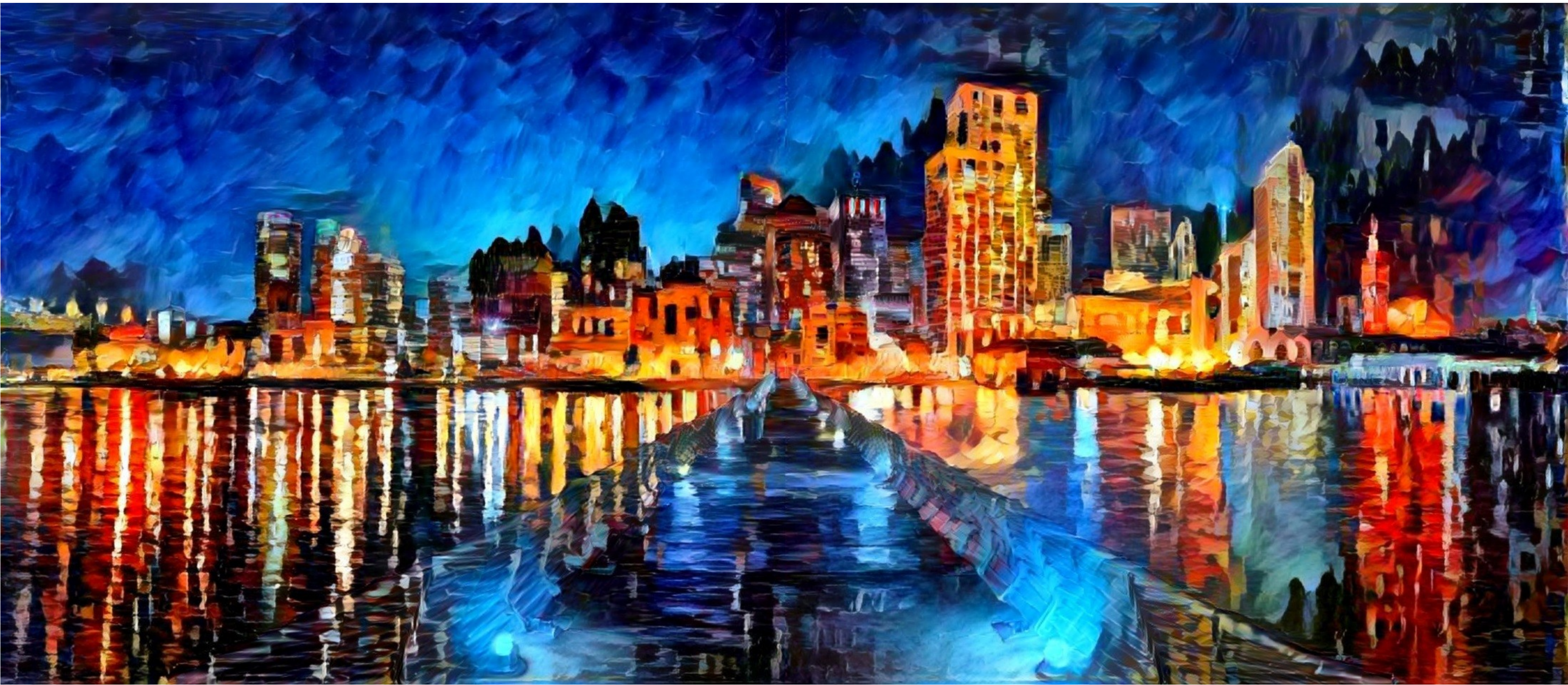






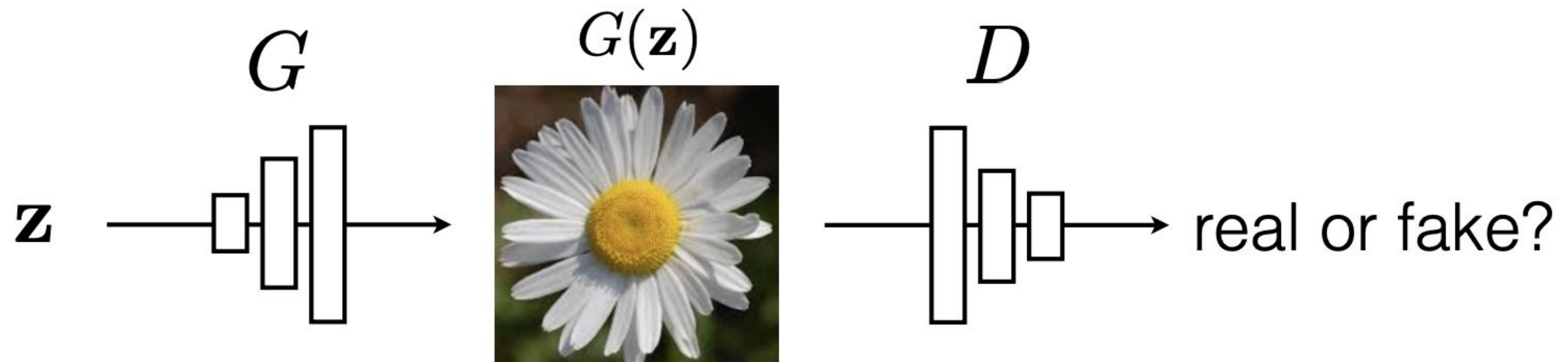


Wayne Filcock



GANs as generative models

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

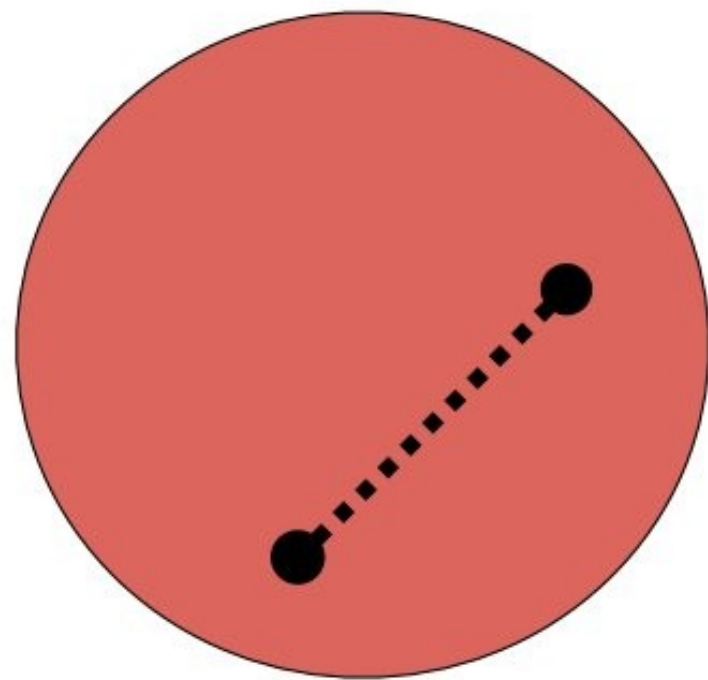


$$\arg \min_G \max_D \mathbb{E}_{\mathbf{z}, \mathbf{x}} [\log D(G(\mathbf{z})) + \log (1 - D(\mathbf{x}))]$$

GANs can “walk” on the manifold

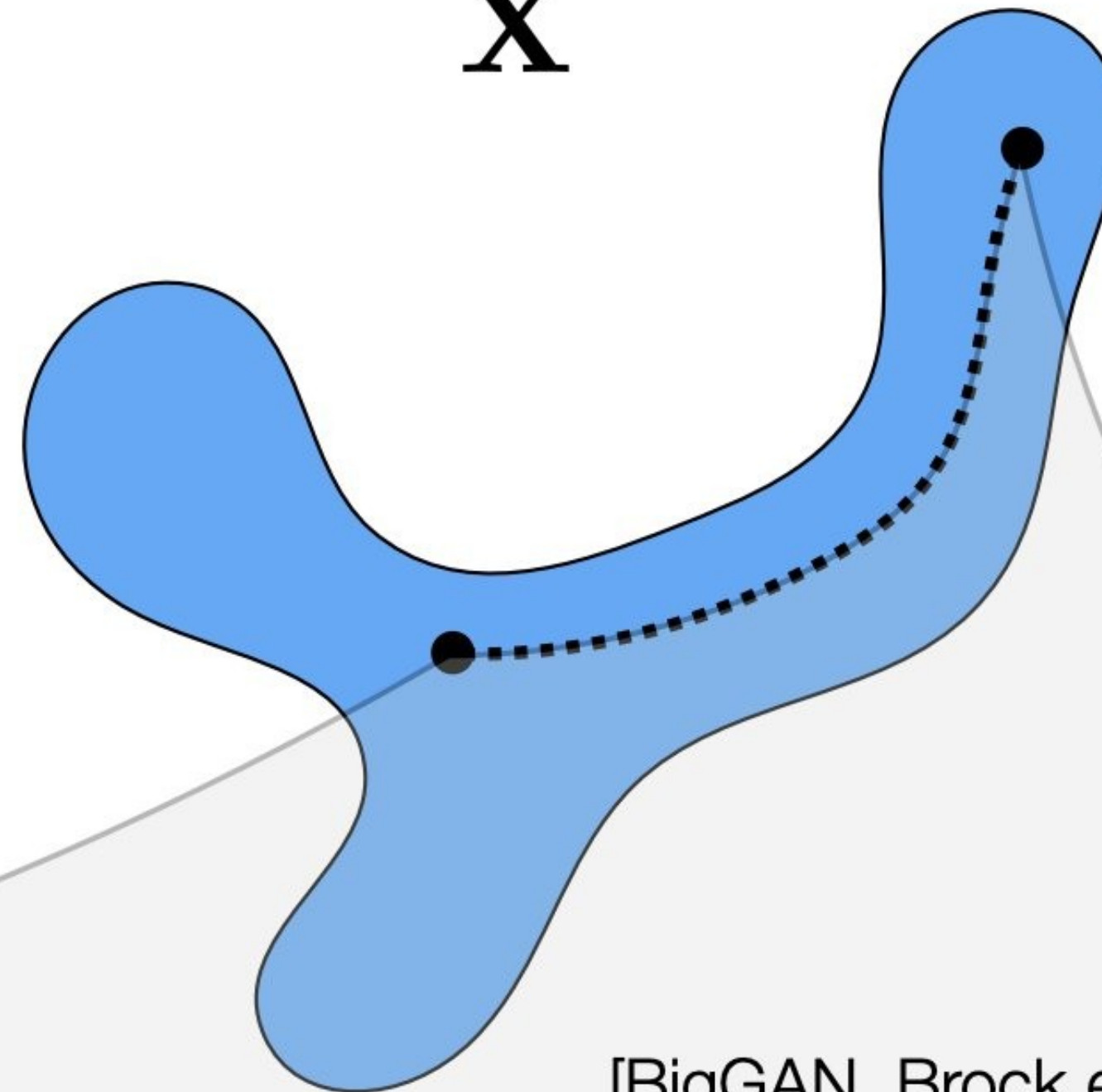
Latent space
(Gaussian)

Z



Data space
(Natural image manifold)

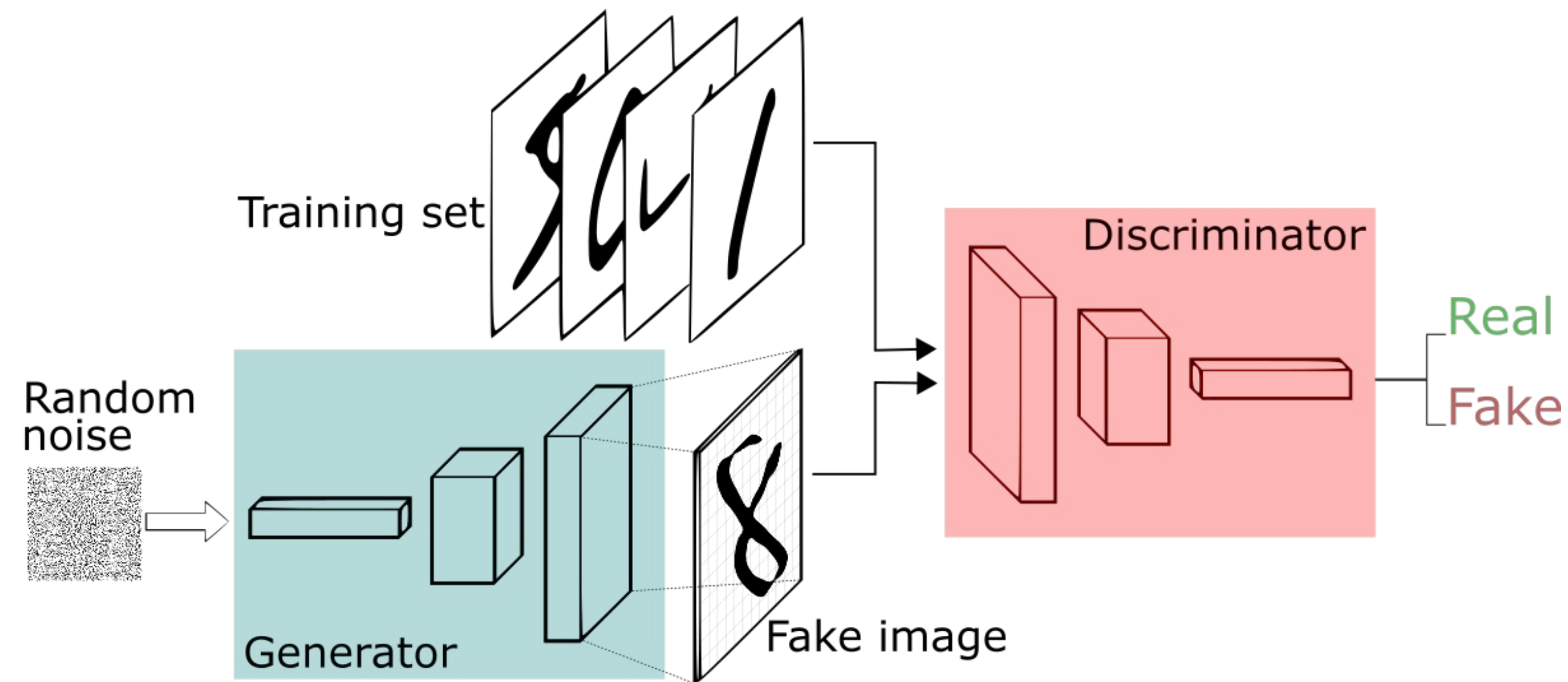
X



[BigGAN, Brock et al. 2018]



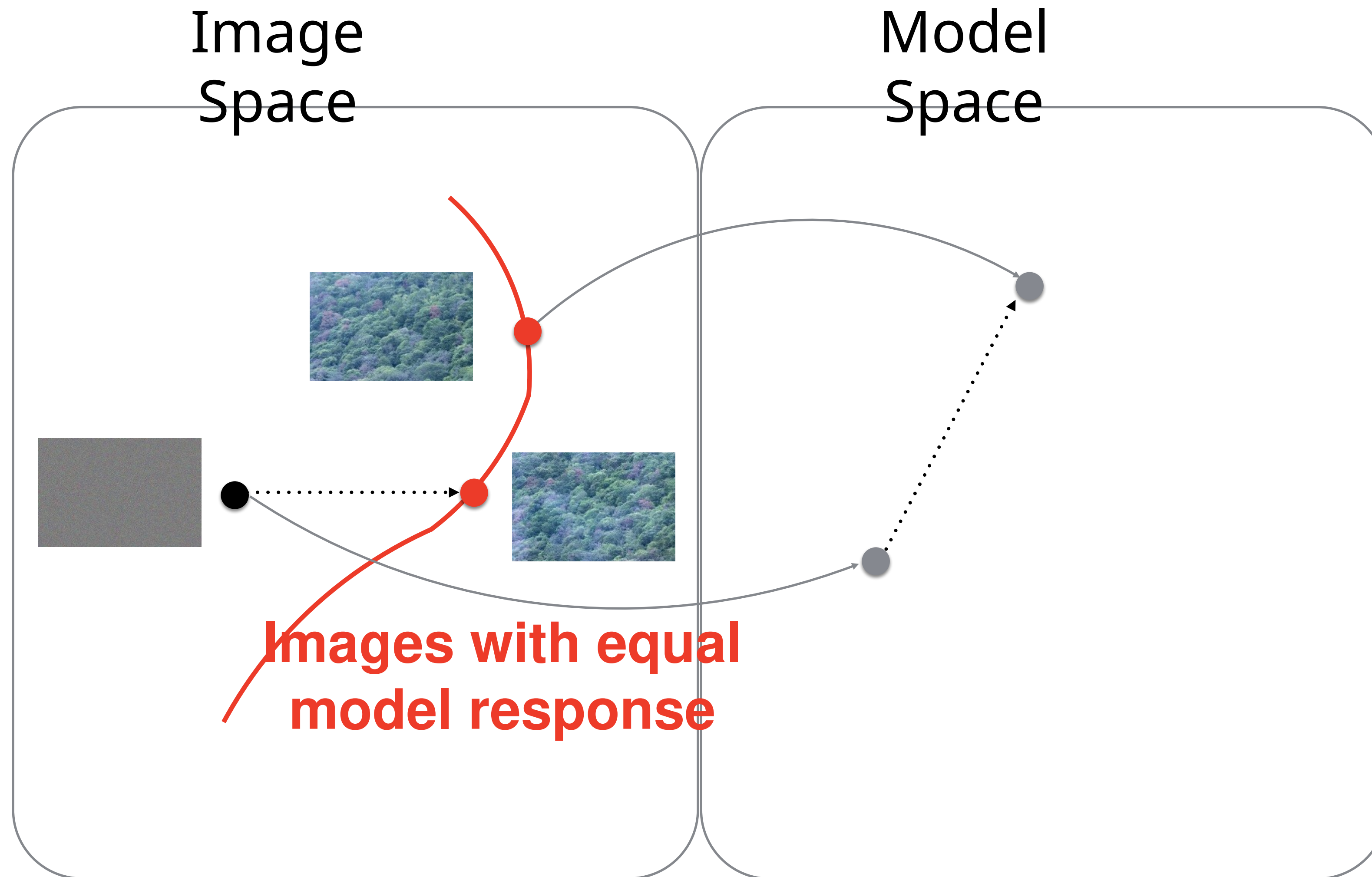
GANs as Texture Synthesis?



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



Projecting onto the “Image Manifold”





Diffusion / Flow Matching

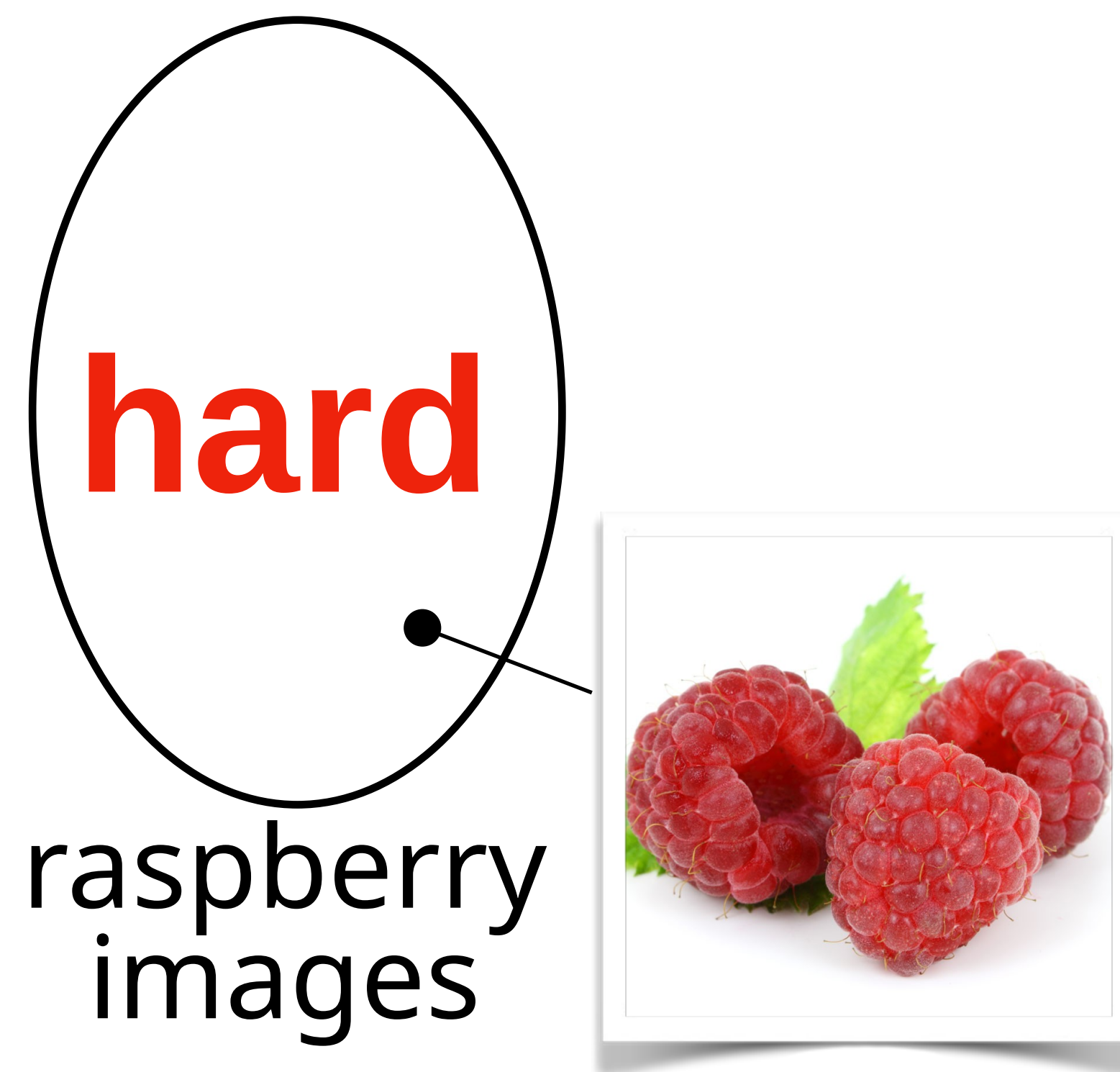
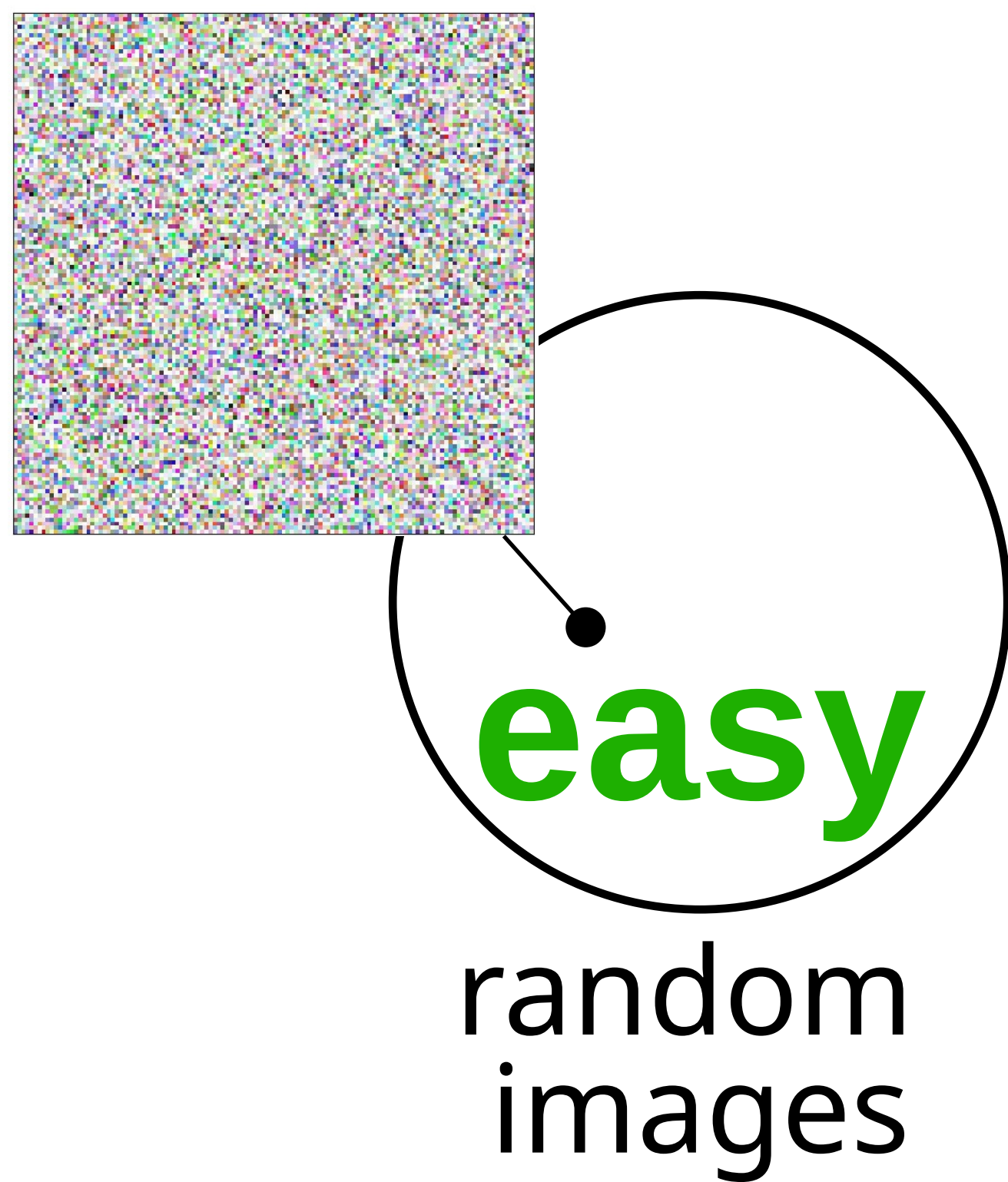
Generate 100 images

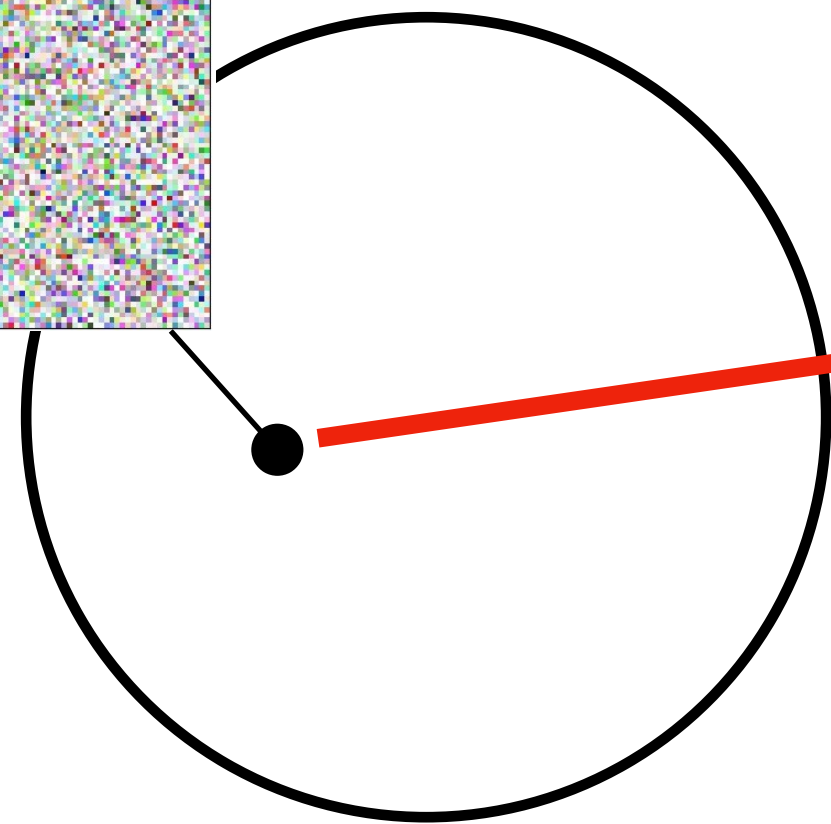
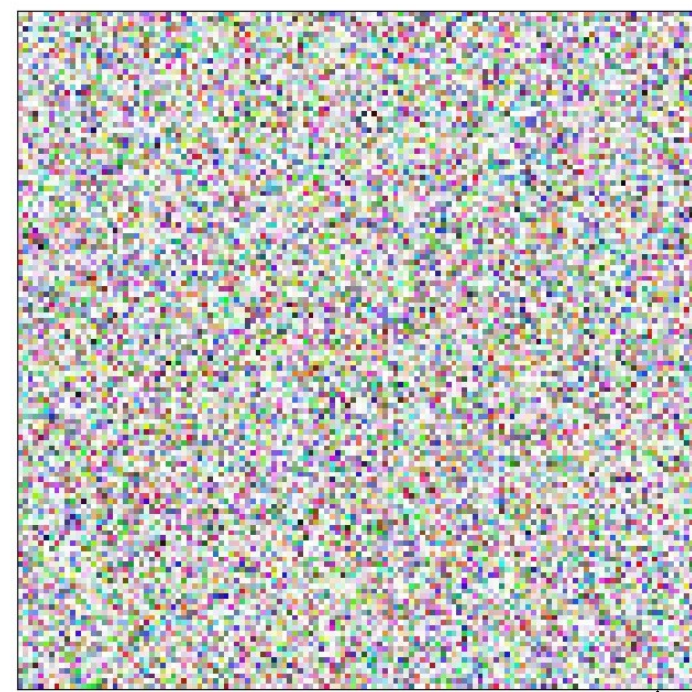


```
pixels = np.random.rand(100,100,3)
```

Generate 100 images

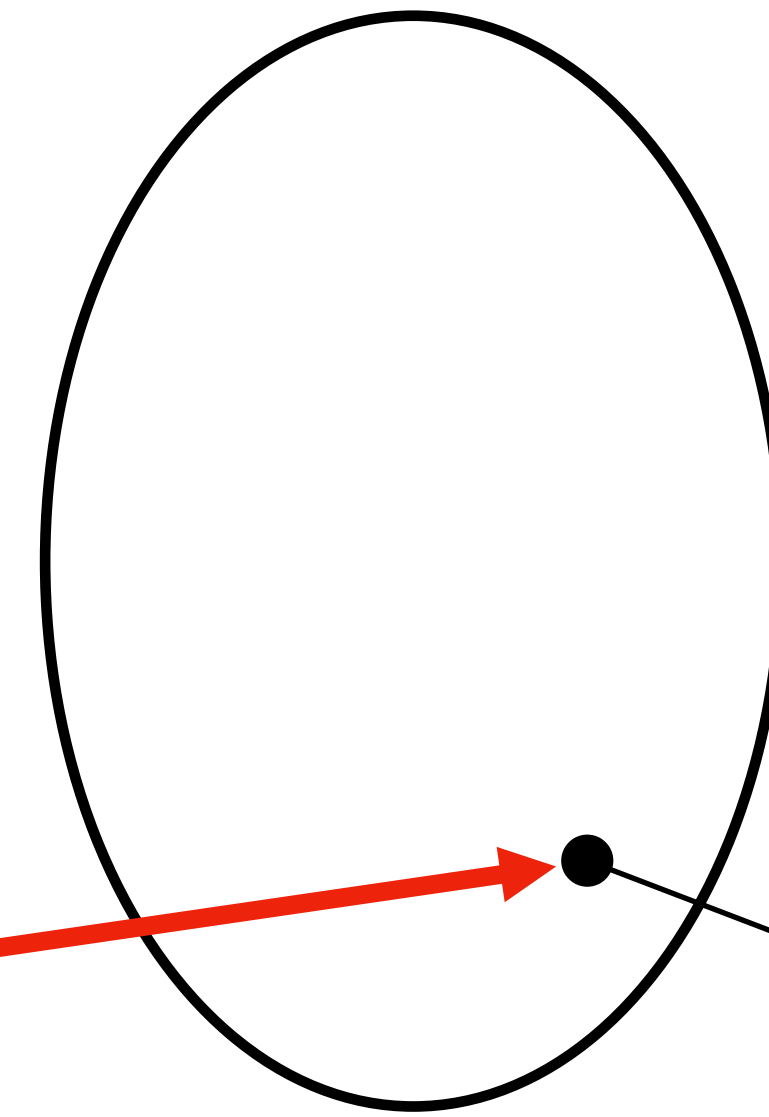
Generate 100 images of raspberries





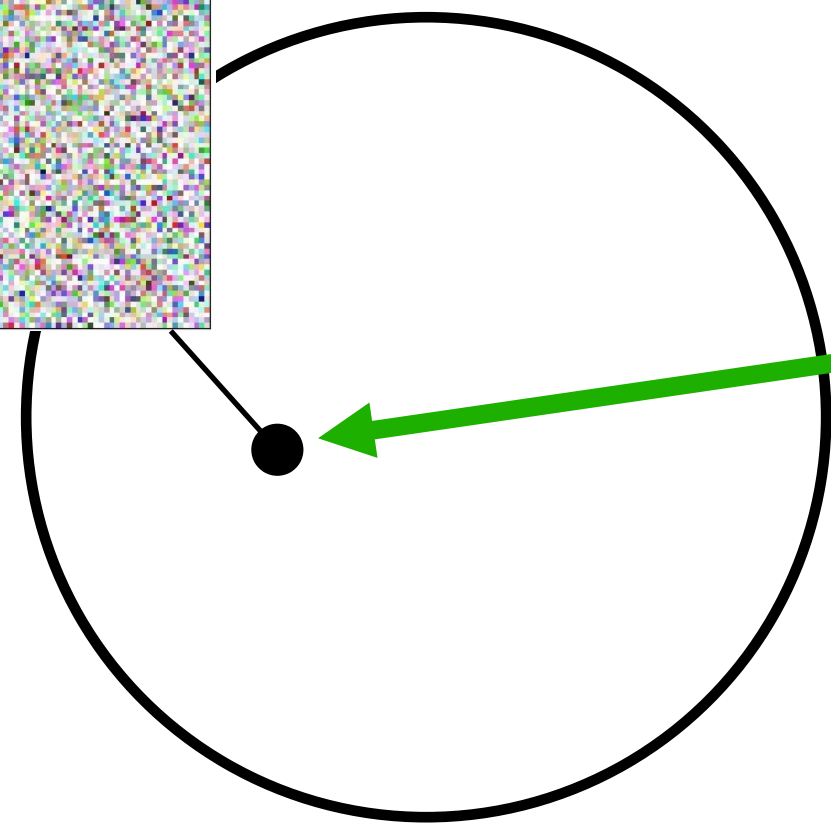
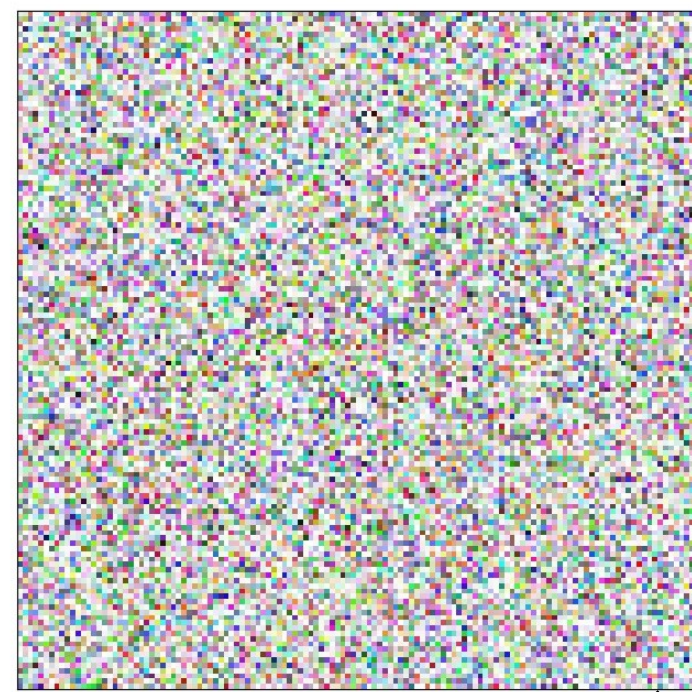
random
images

hard



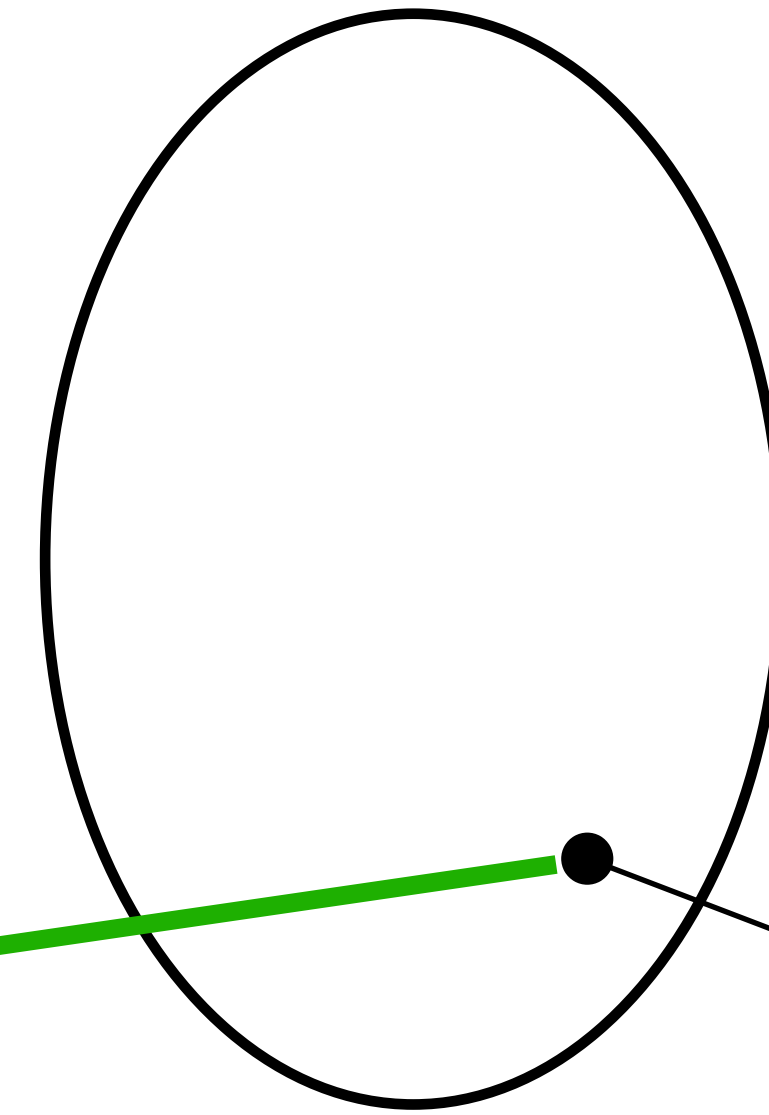
raspberry
images





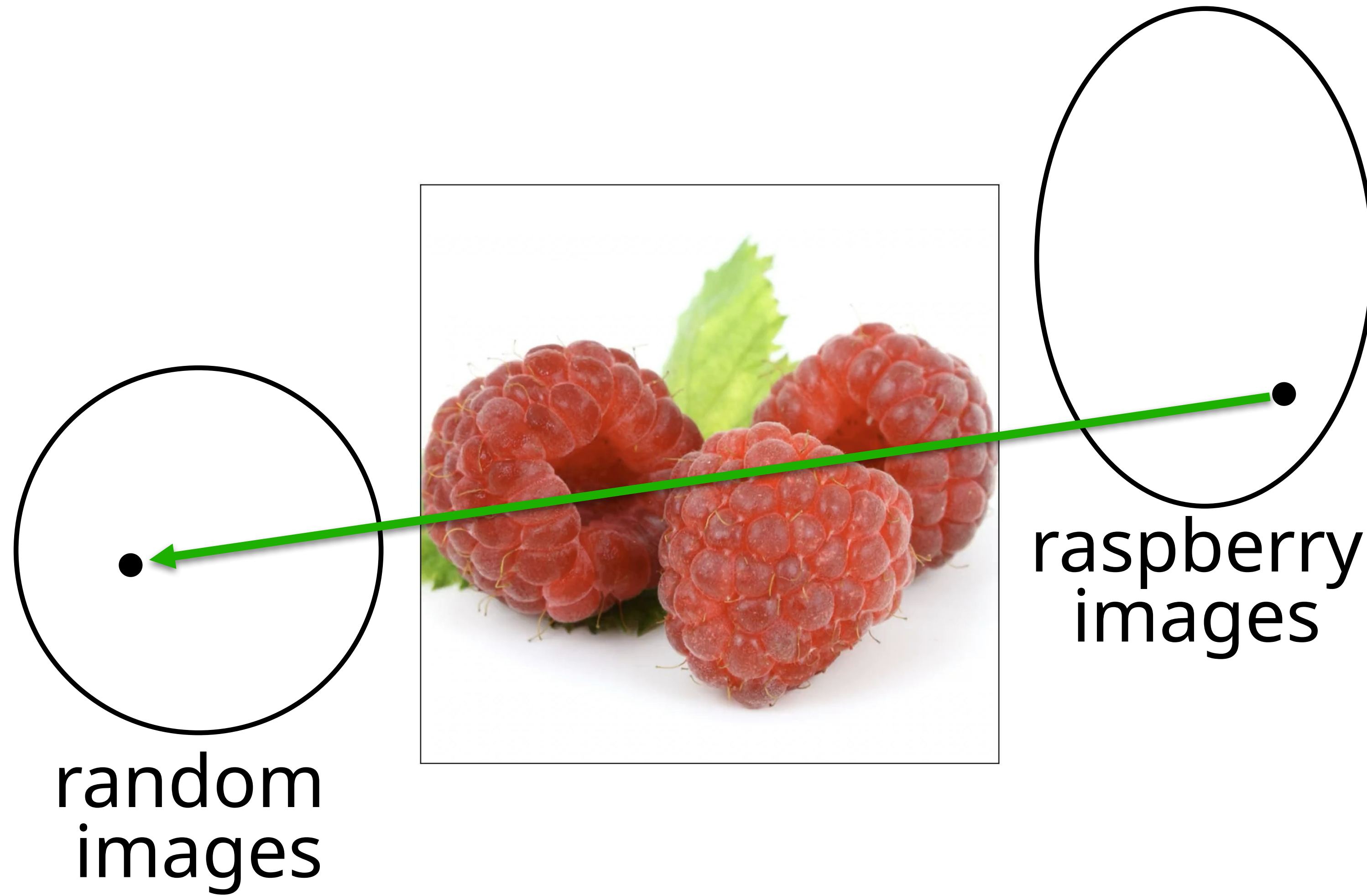
random
images

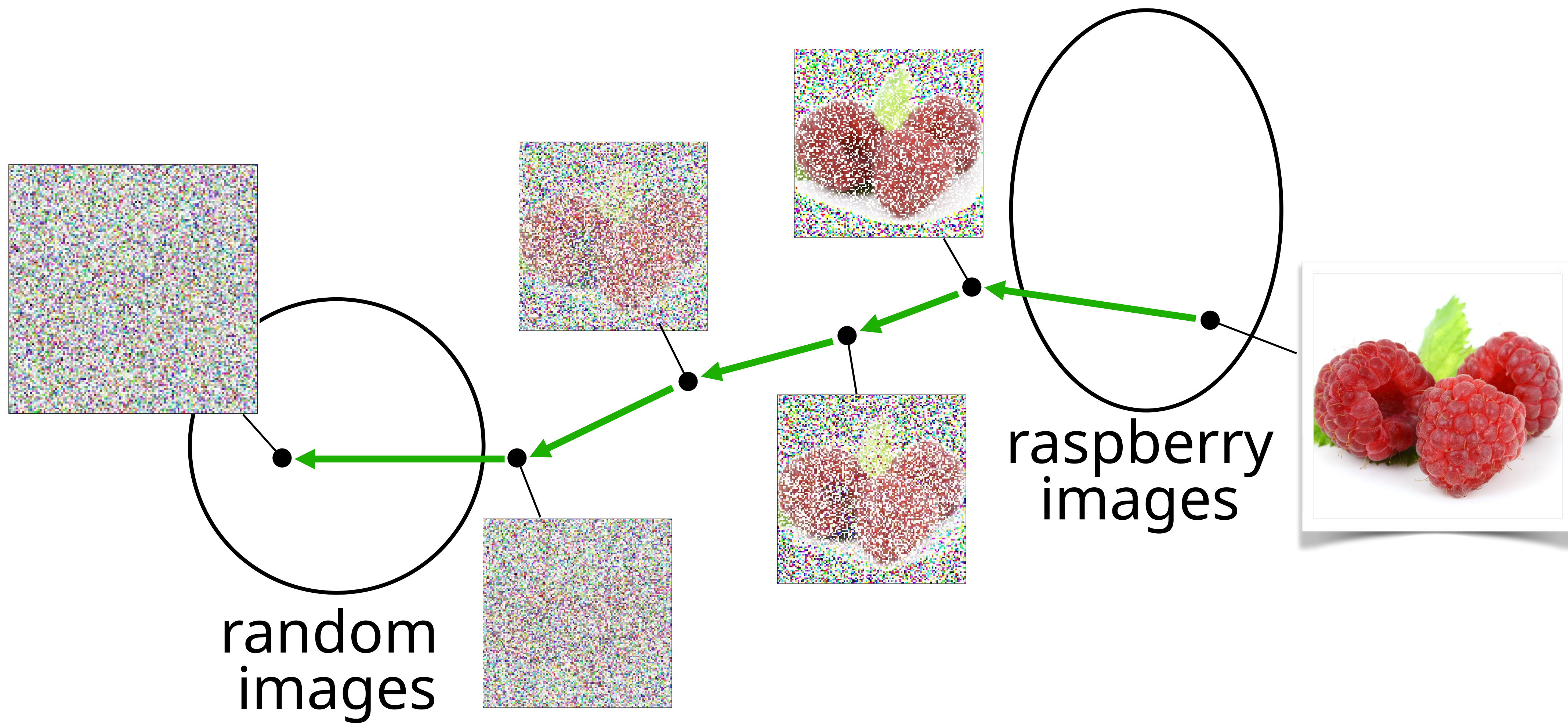
easy

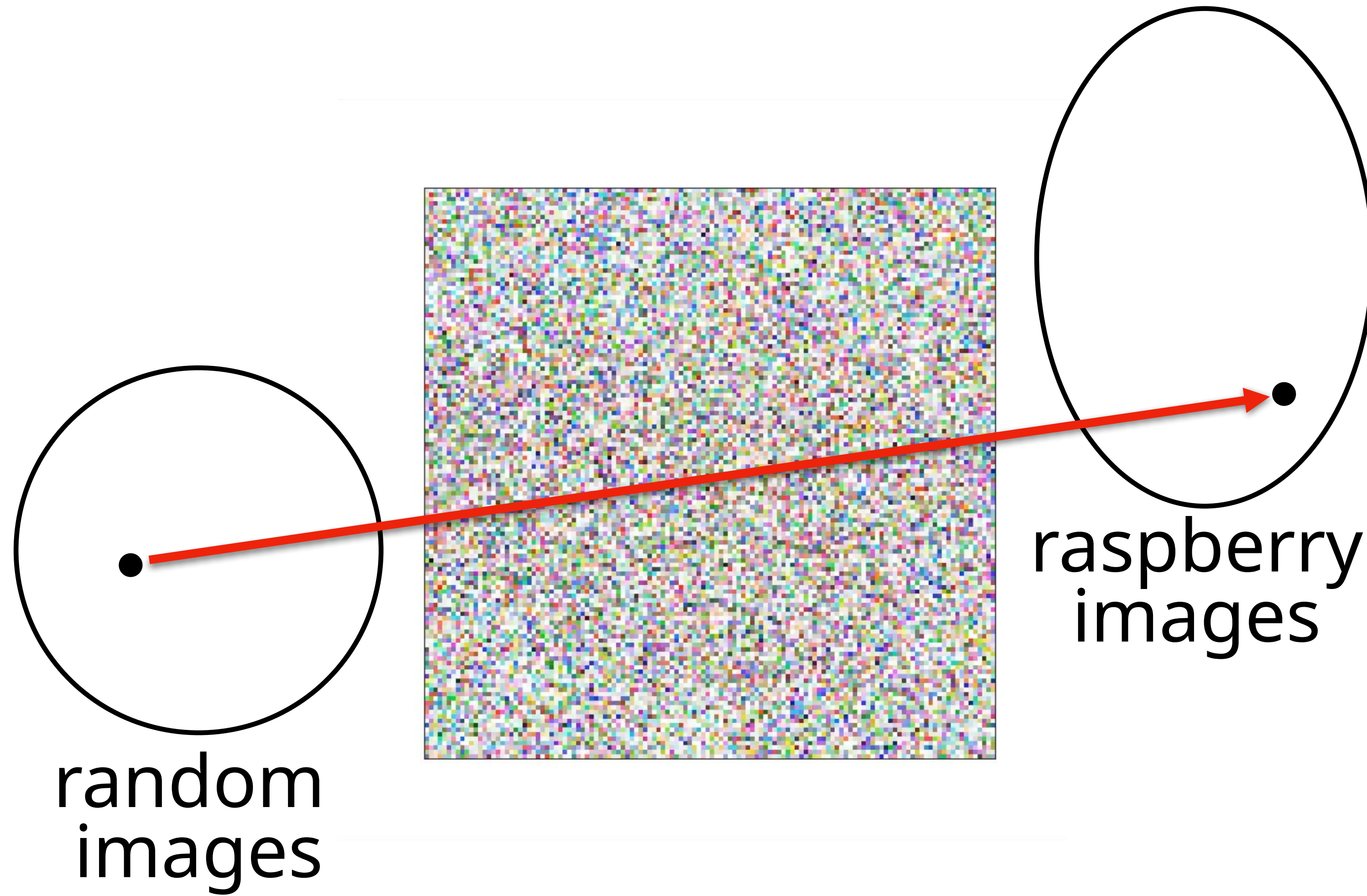


raspberry
images



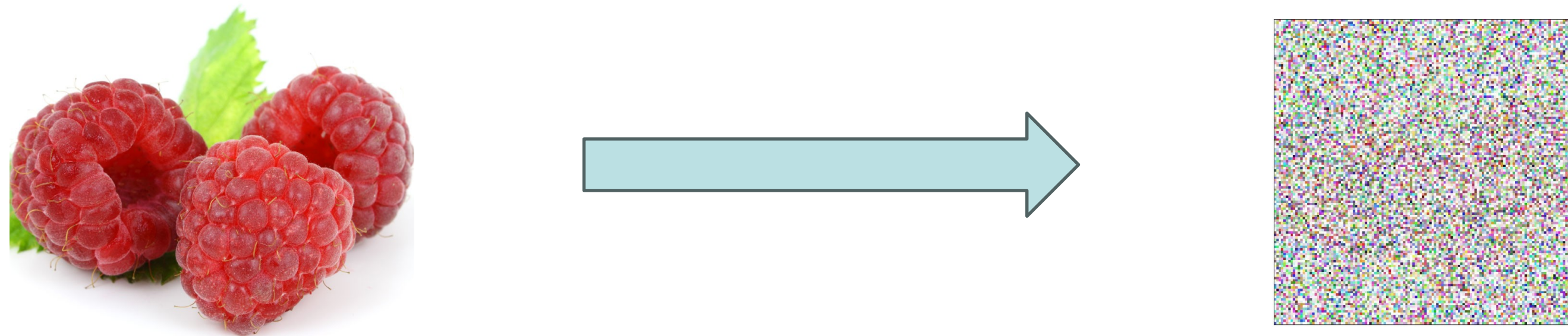




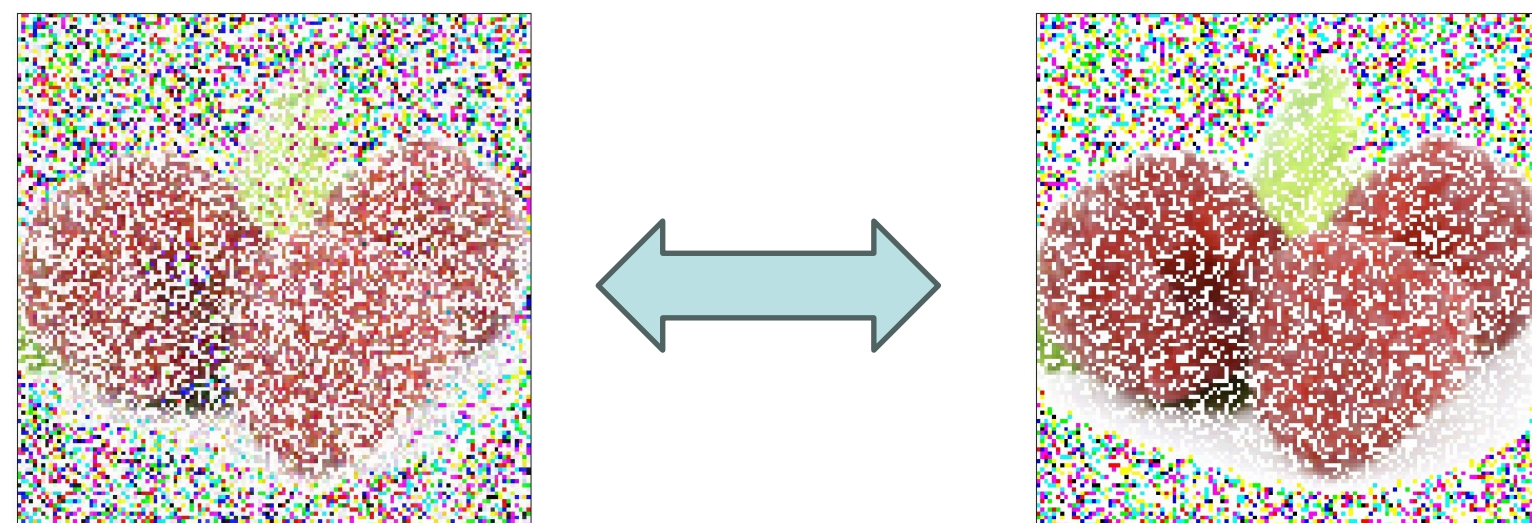


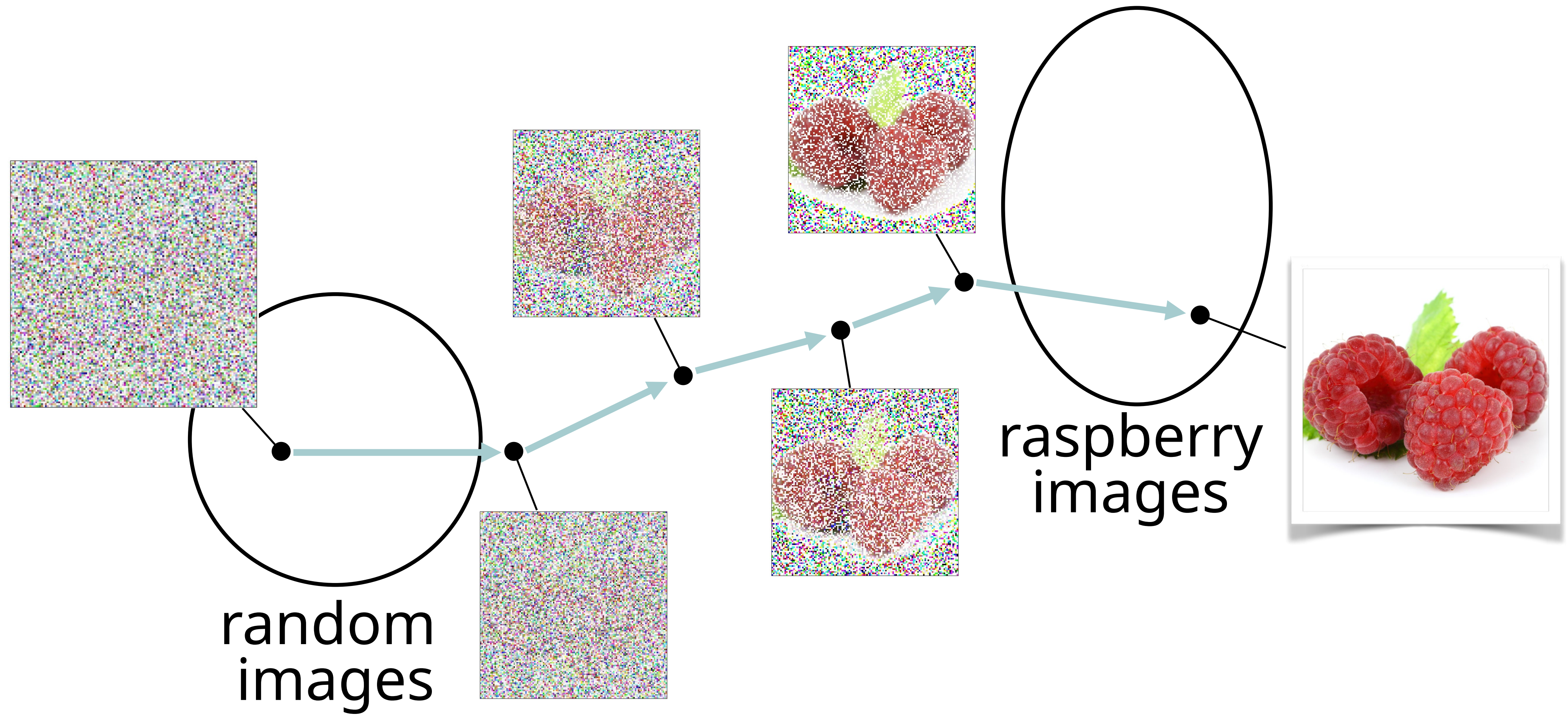
Key insight

- Globally, creation is much harder than destruction

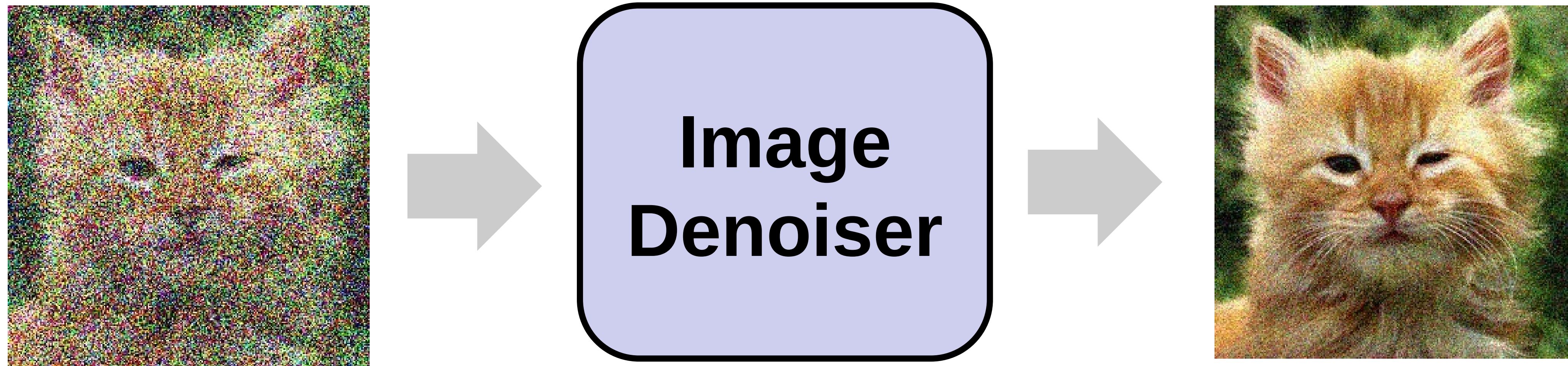


- But locally, they are almost reversible!





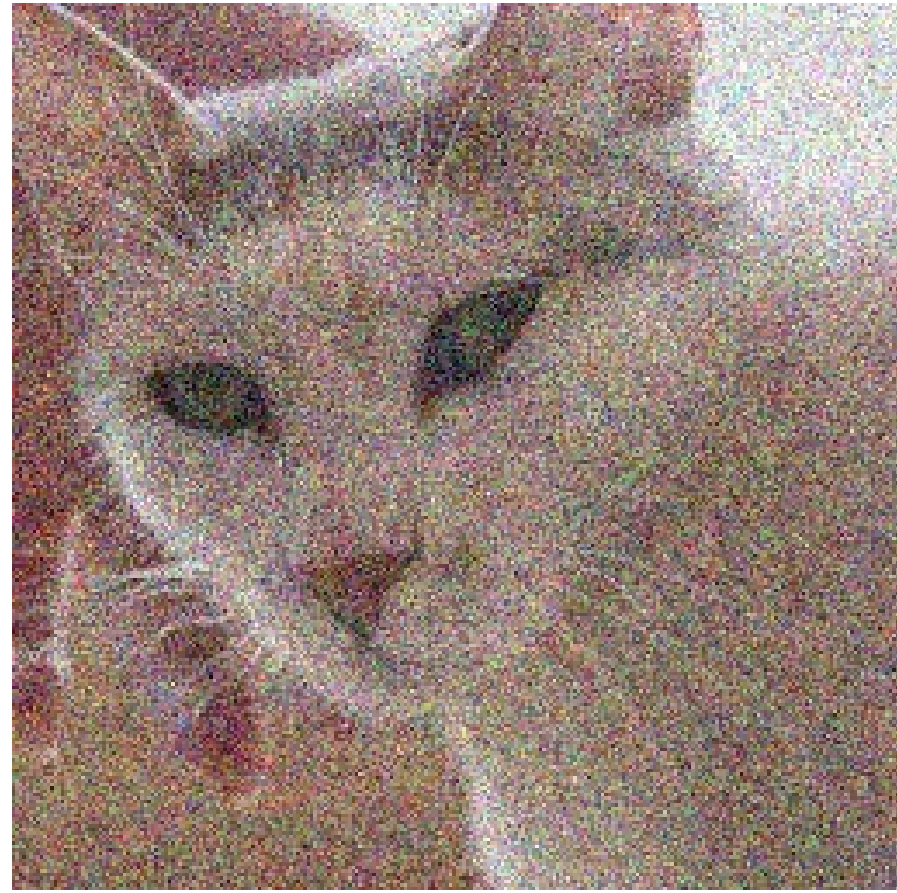
Denoising neural network



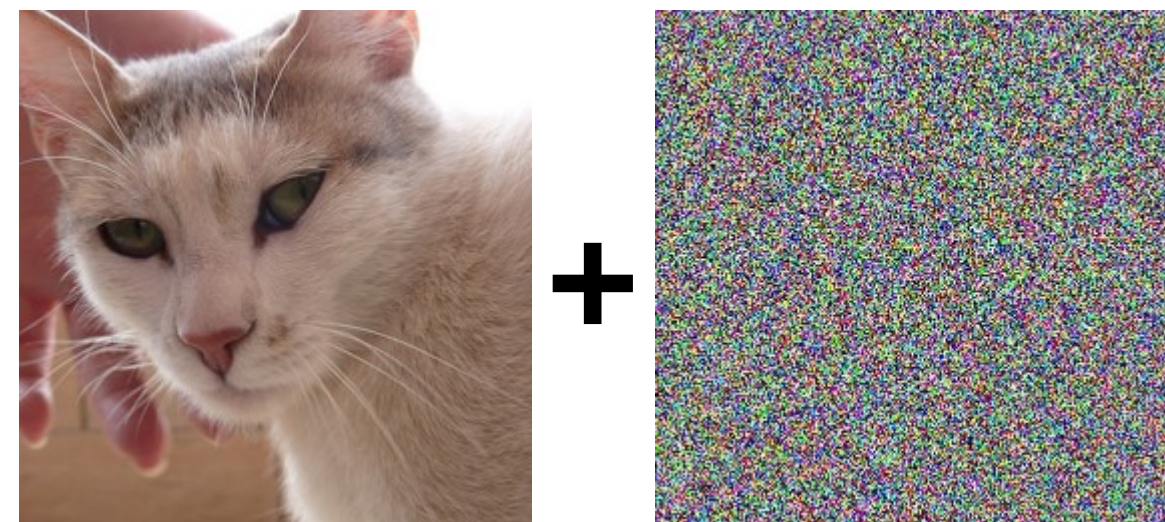
This network can be a U-Net or other
suitable image-to-image network

Training U-Net Denoiser

x_t

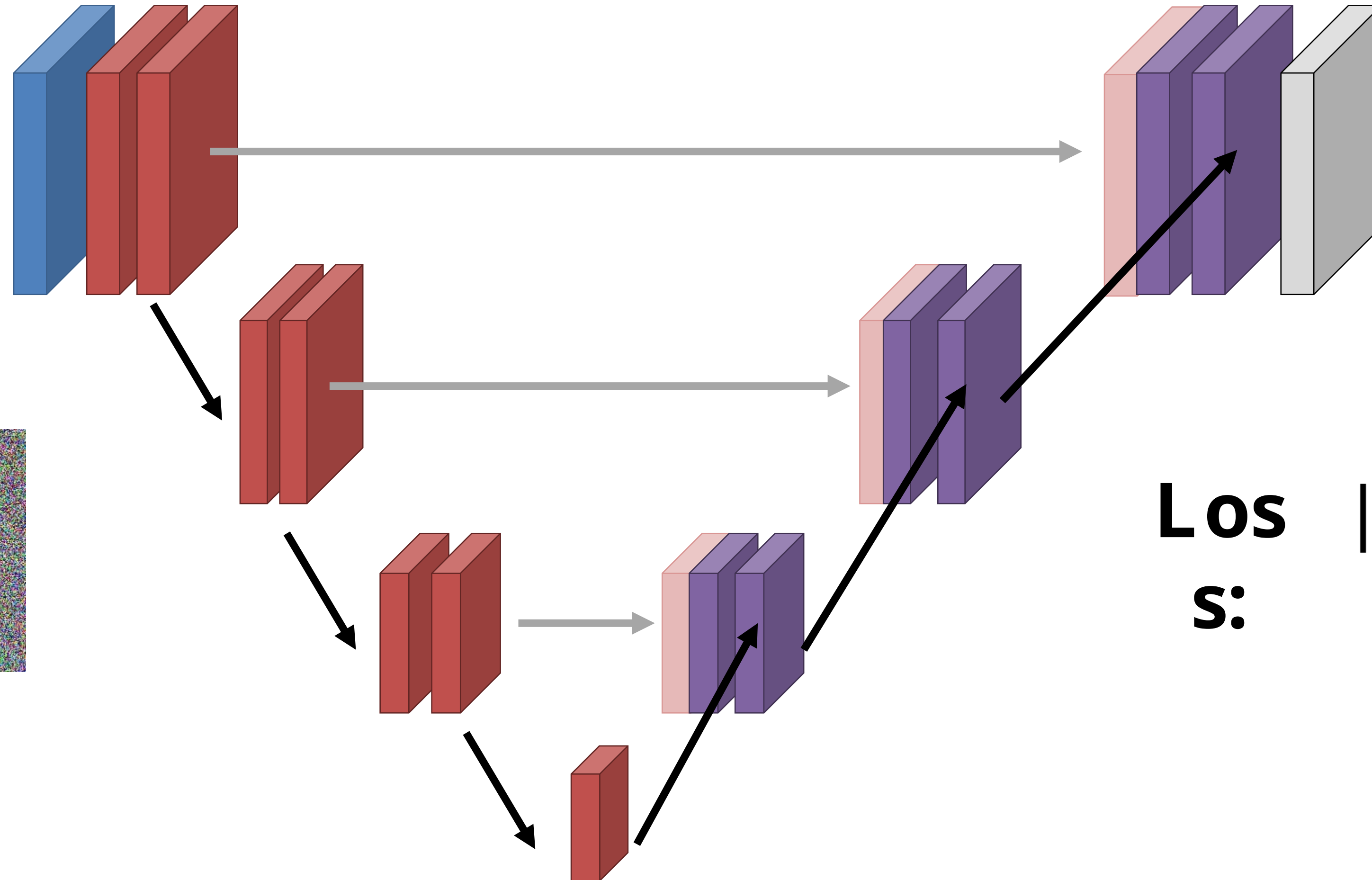


$\hat{x}$



x_{clean}

ϵ

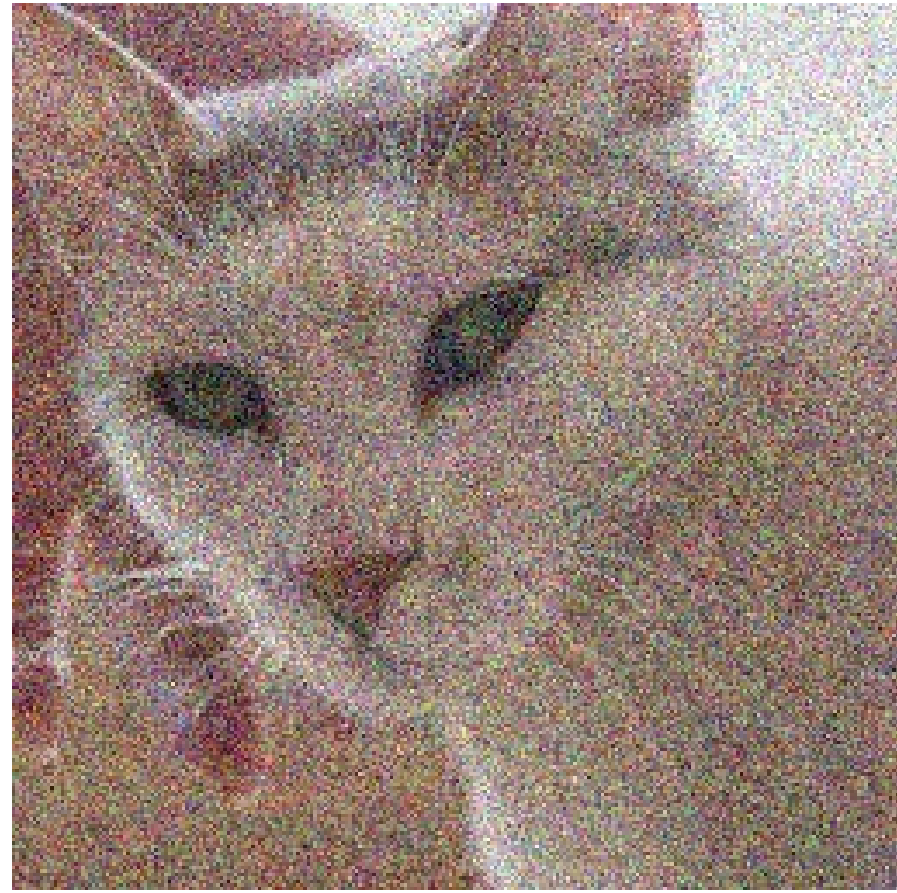


Loss: $\|x_{clean} - \hat{x}\|^2$

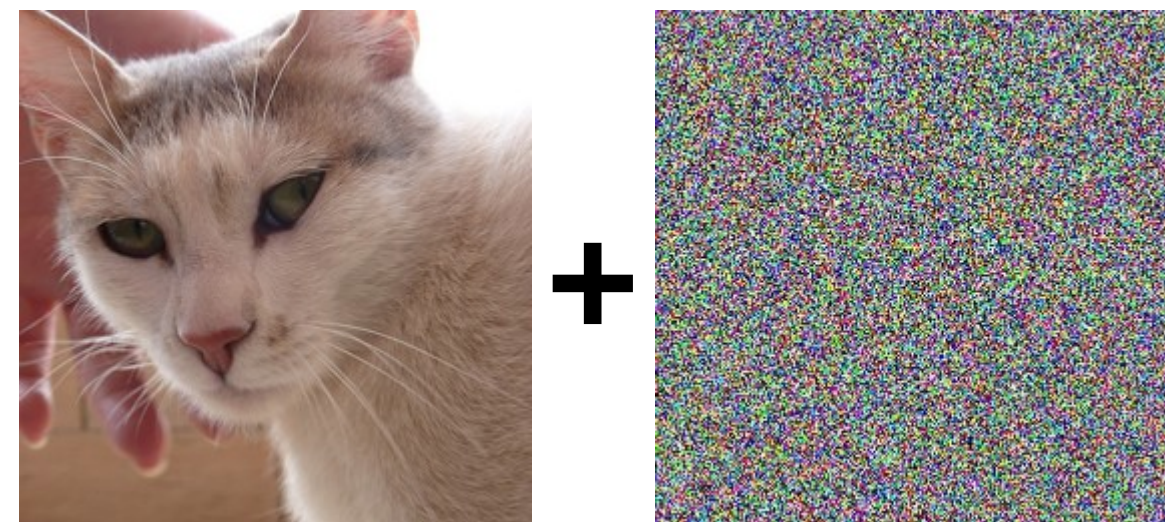
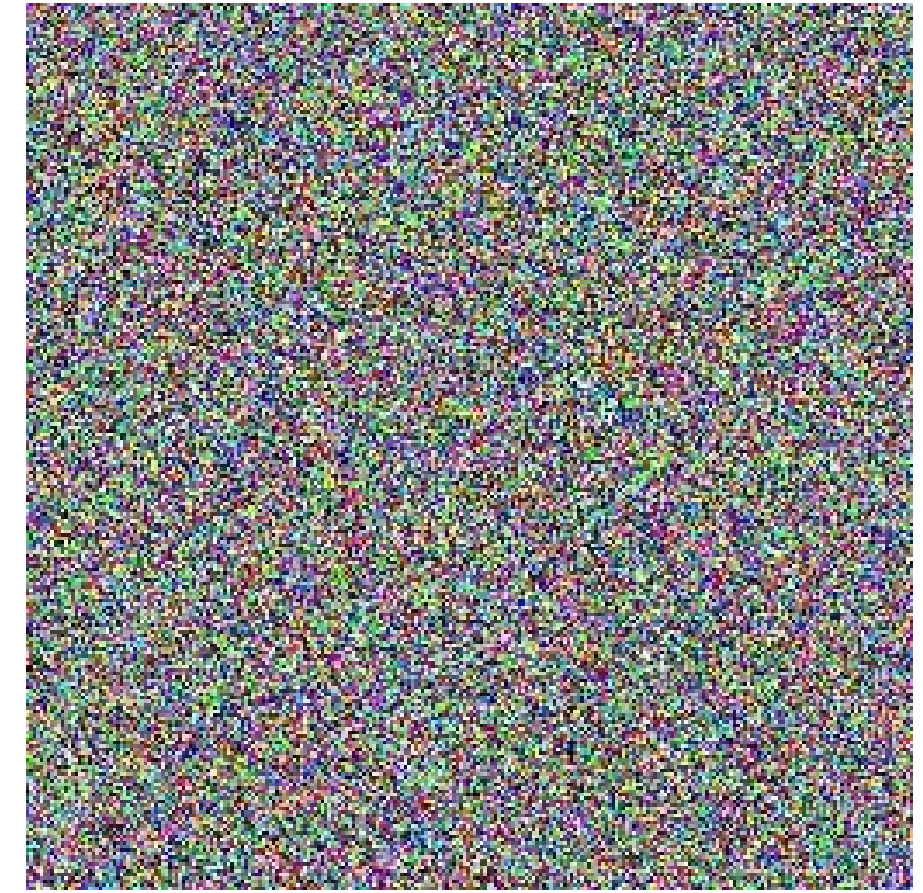
Image Denoising

Training U-Net Denoiser

x_t

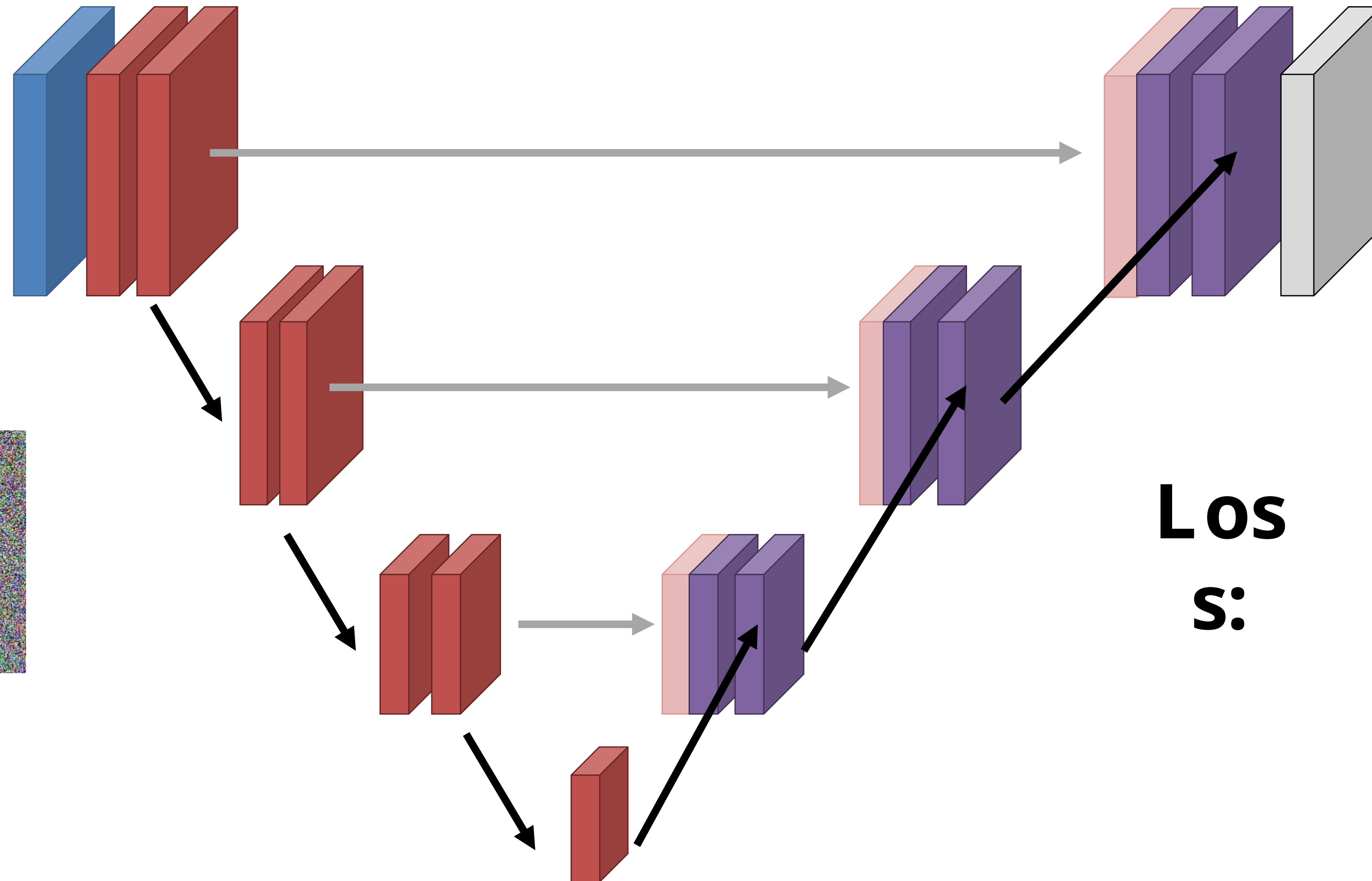


$\hat{\varepsilon}$



x_{clean}

ε



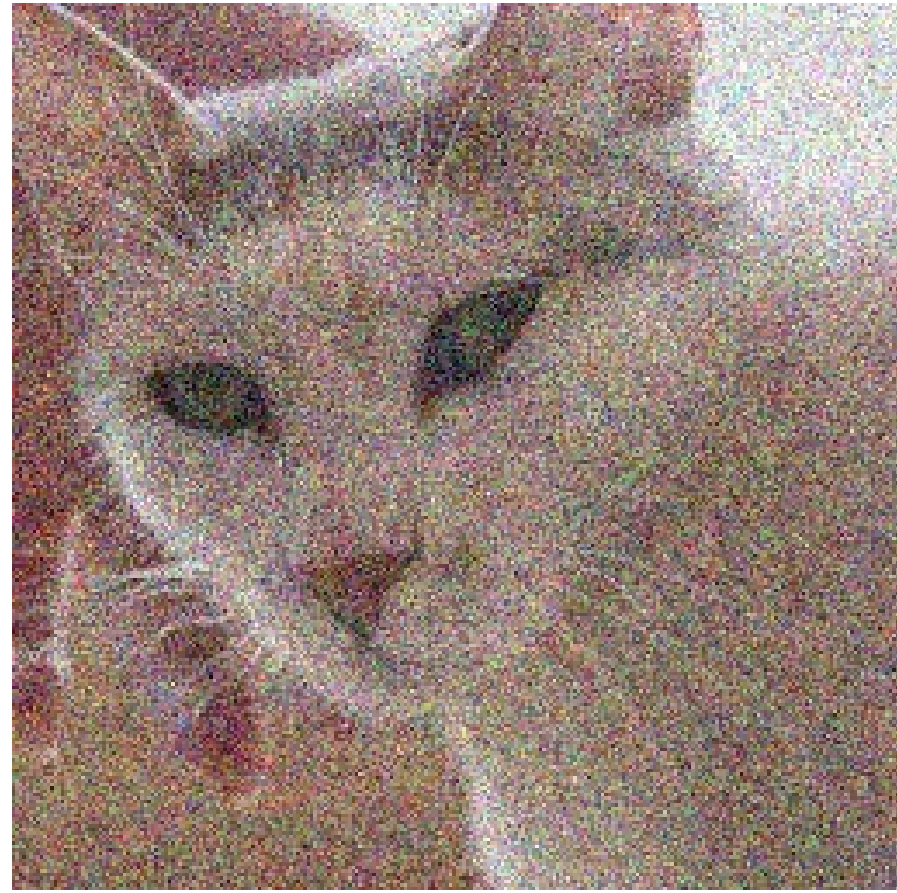
Loss:

$$\|\varepsilon - \hat{\varepsilon}\|^2$$

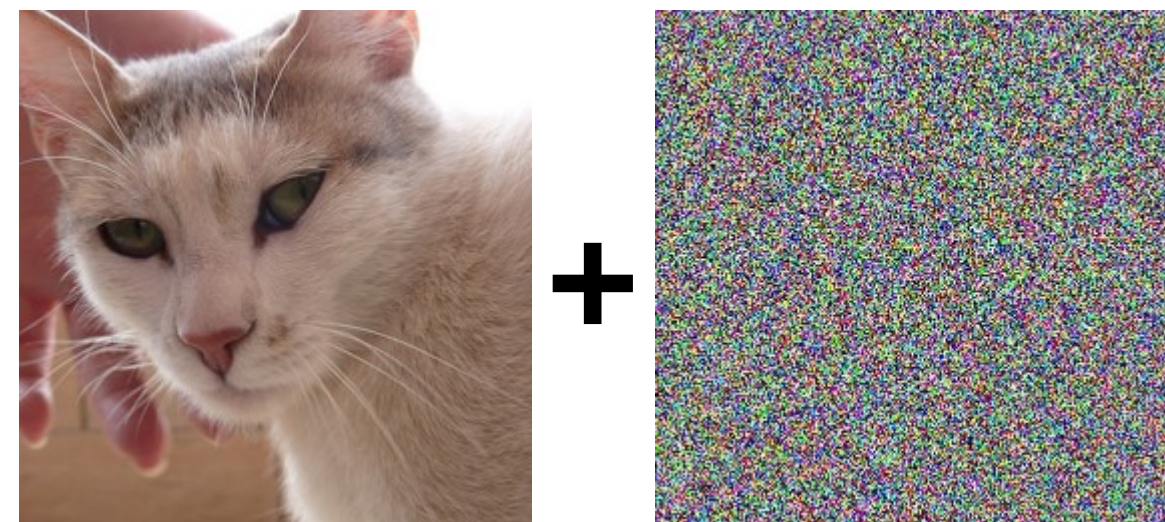
Image Diffusion

Training U-Net Denoiser

x_t

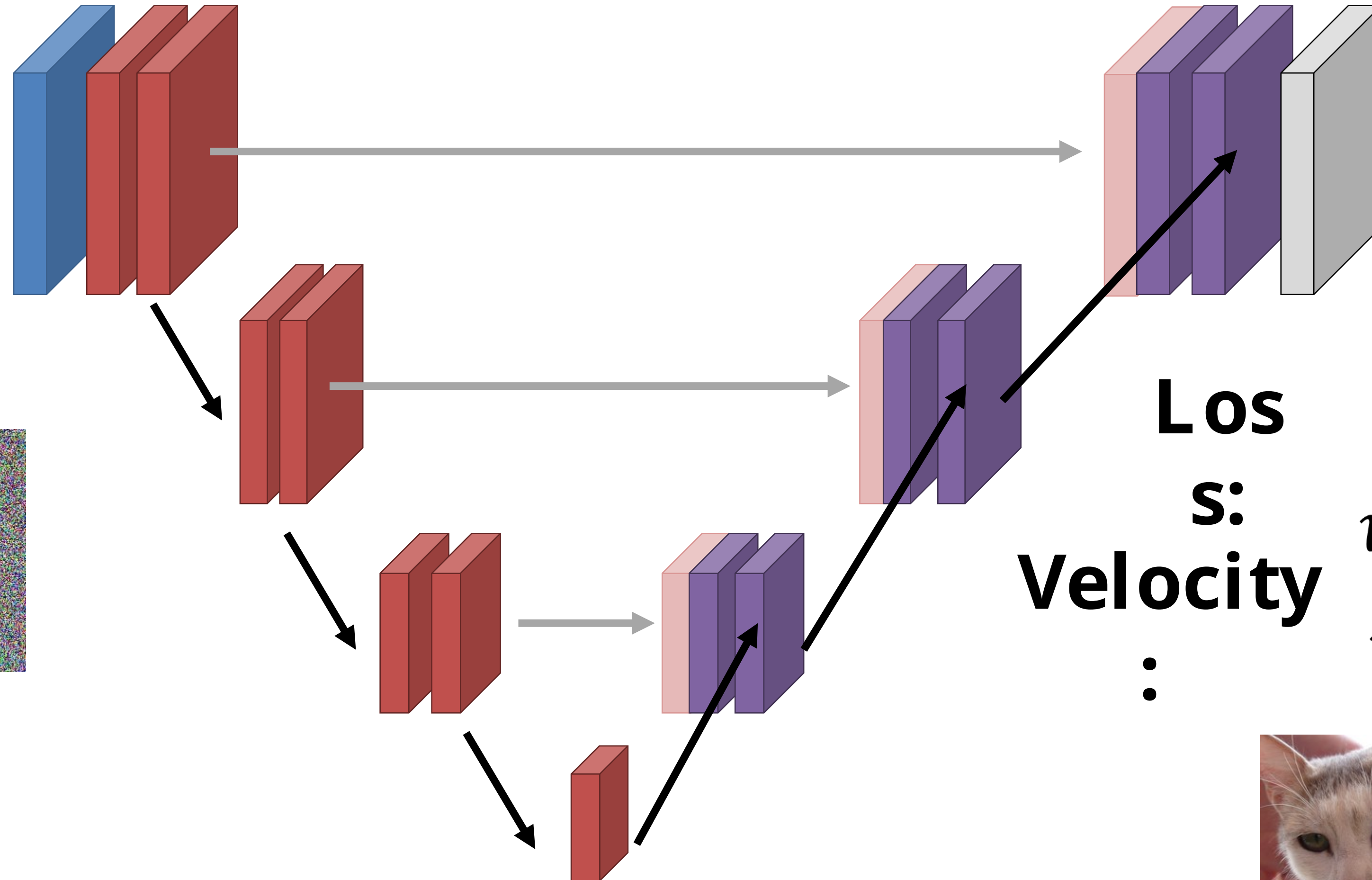


$\hat{v}$



x_{clean}

ϵ



Loss

s:

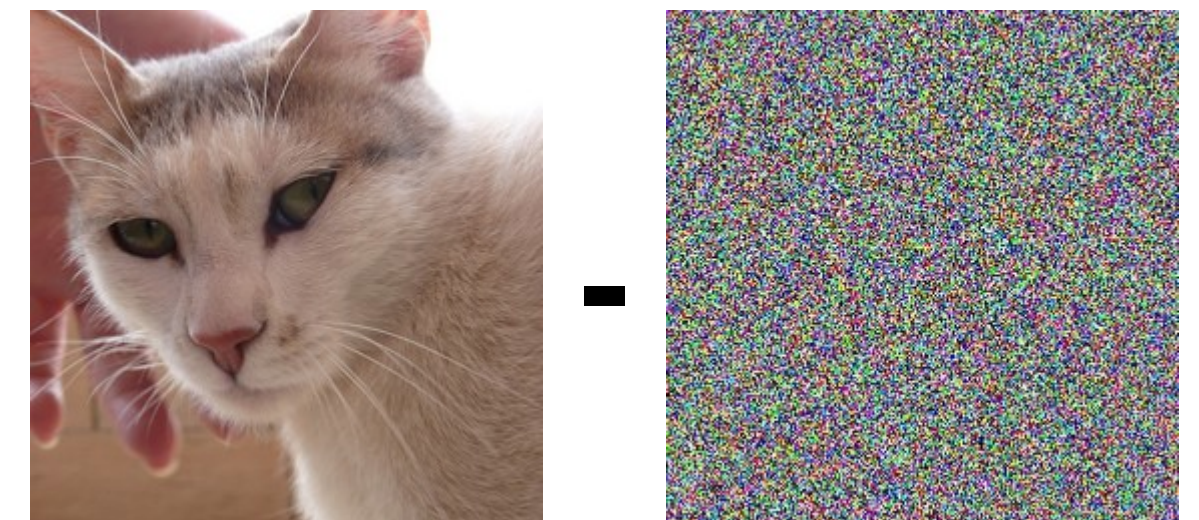
Velocity

:

$$\|v - \hat{v}\|^2$$

$$v = \frac{x_{clean} - x_t}{(1 - t)}$$

$$v = x_{clean} - \epsilon$$



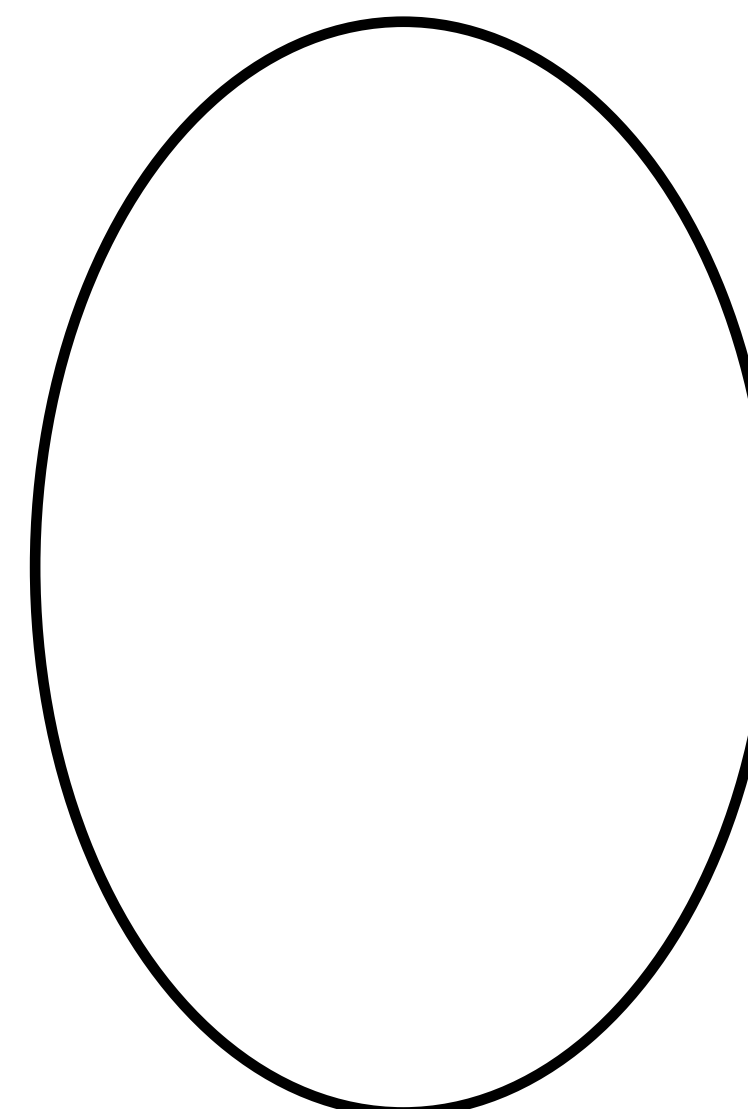
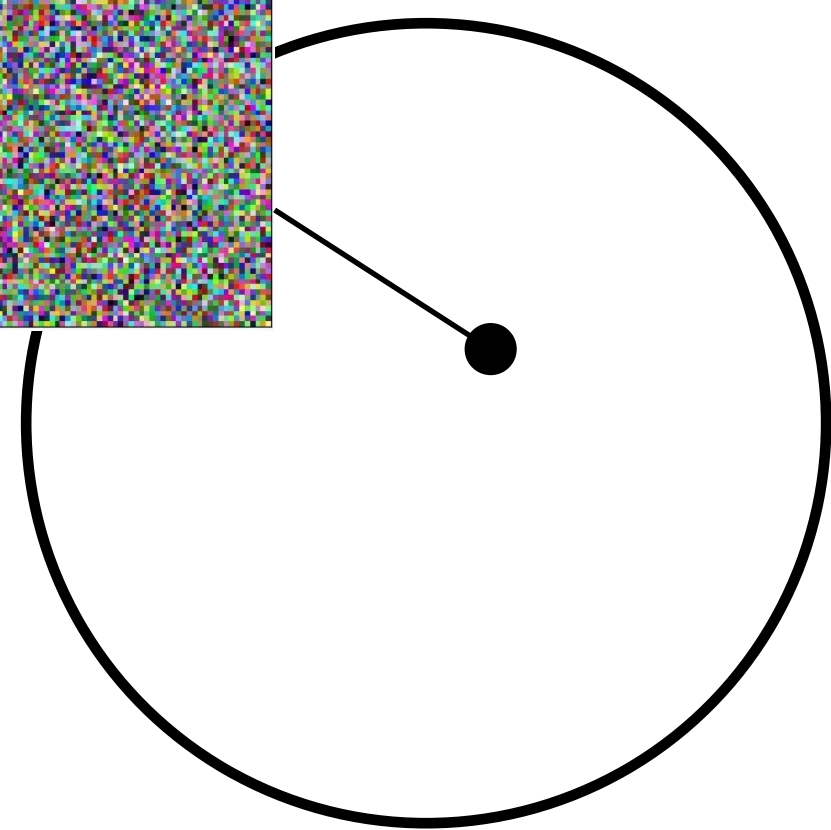
x_{clean}

ϵ

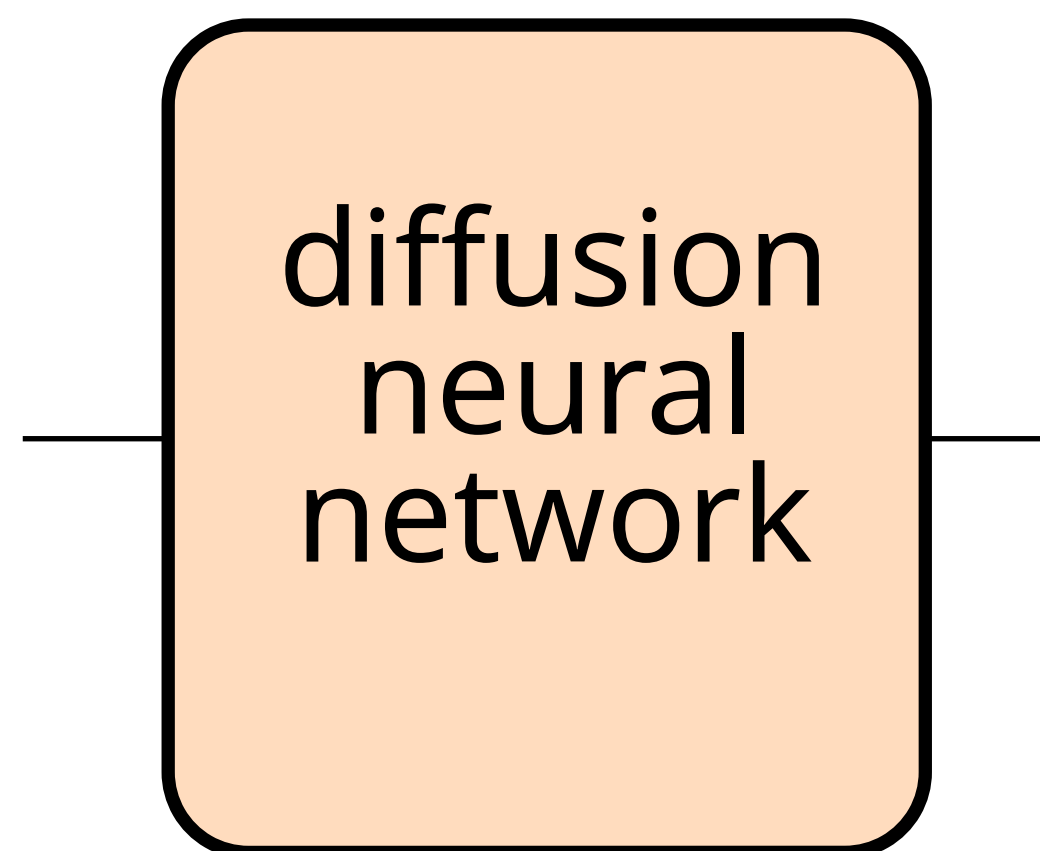
Flow Matching



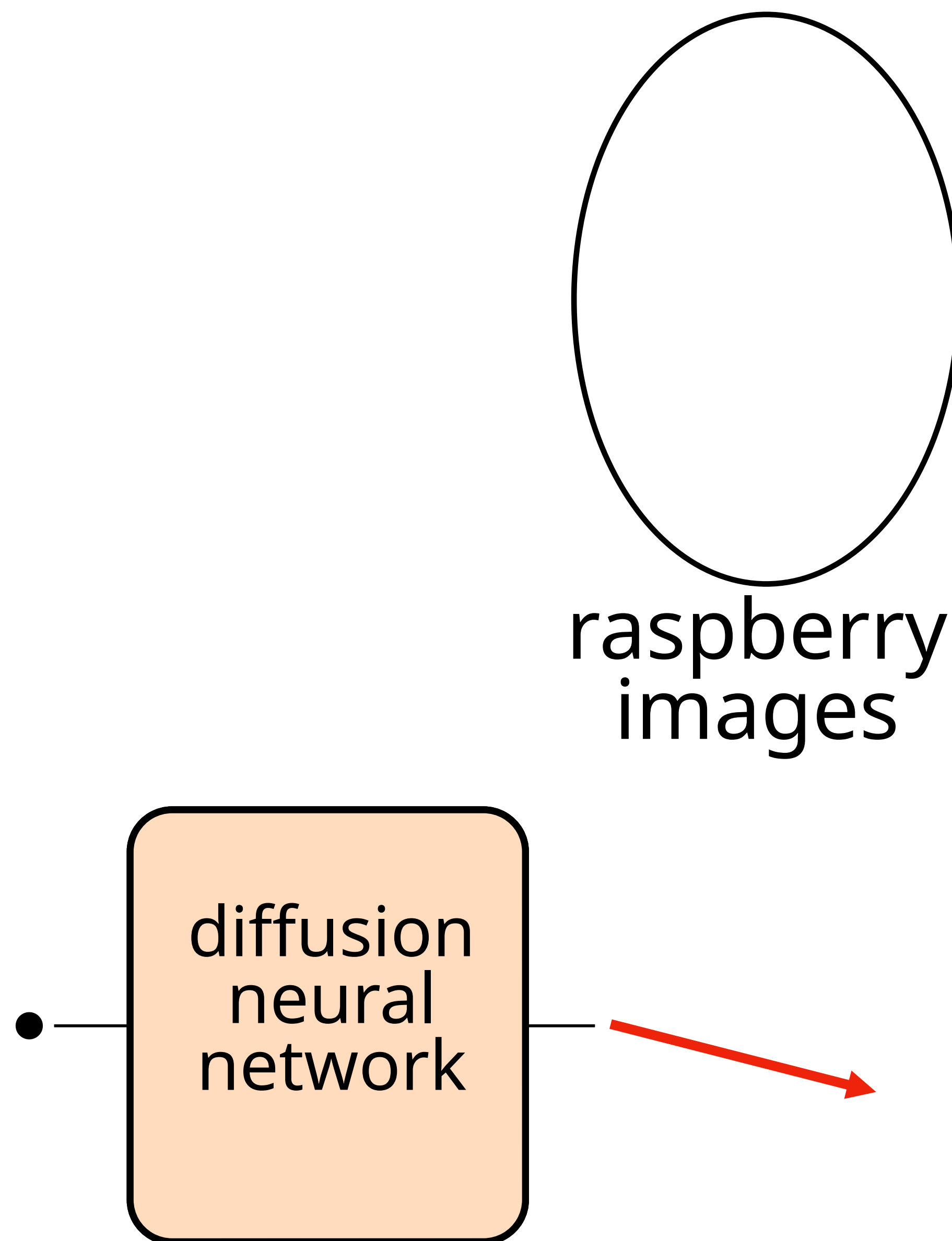
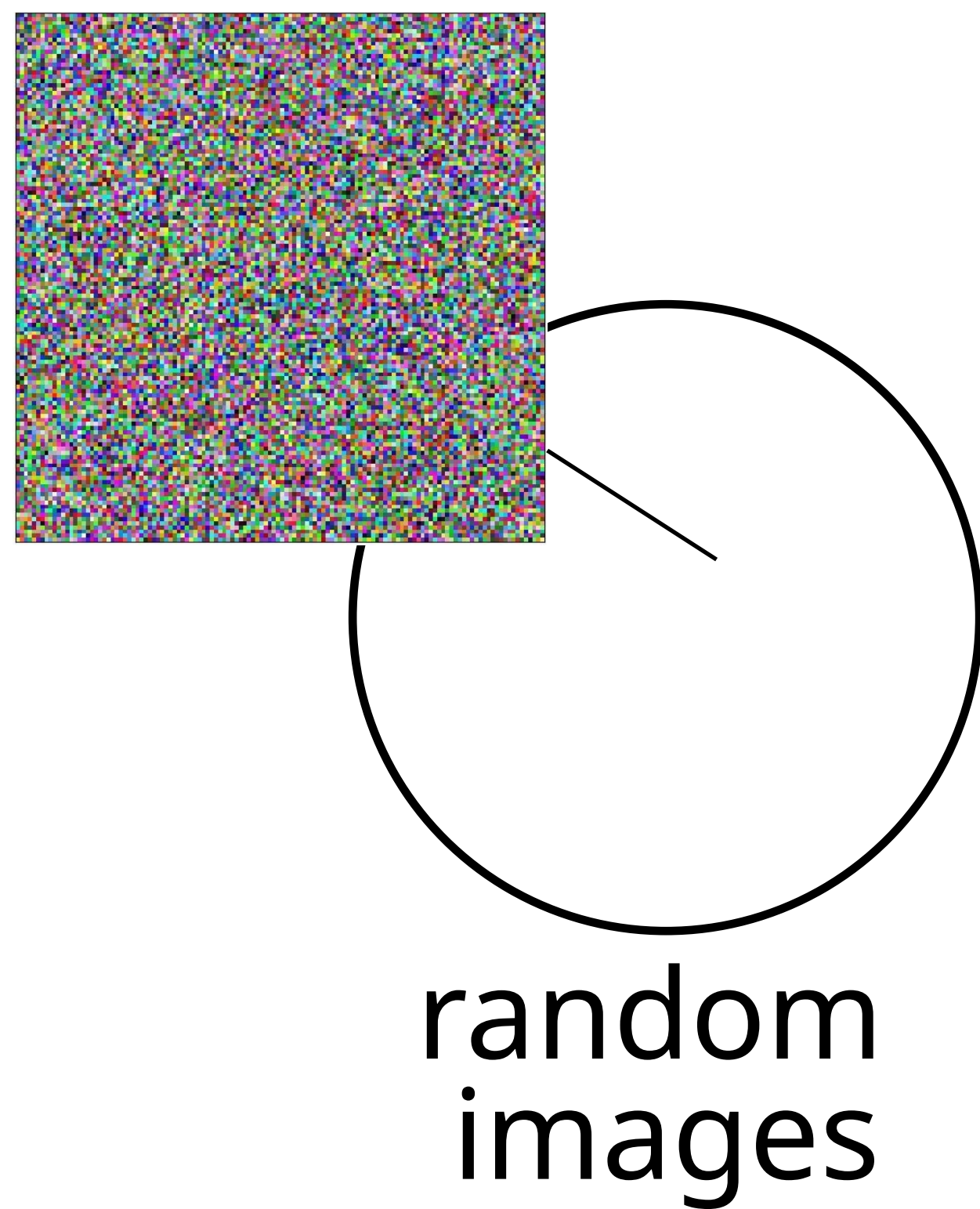
random
images

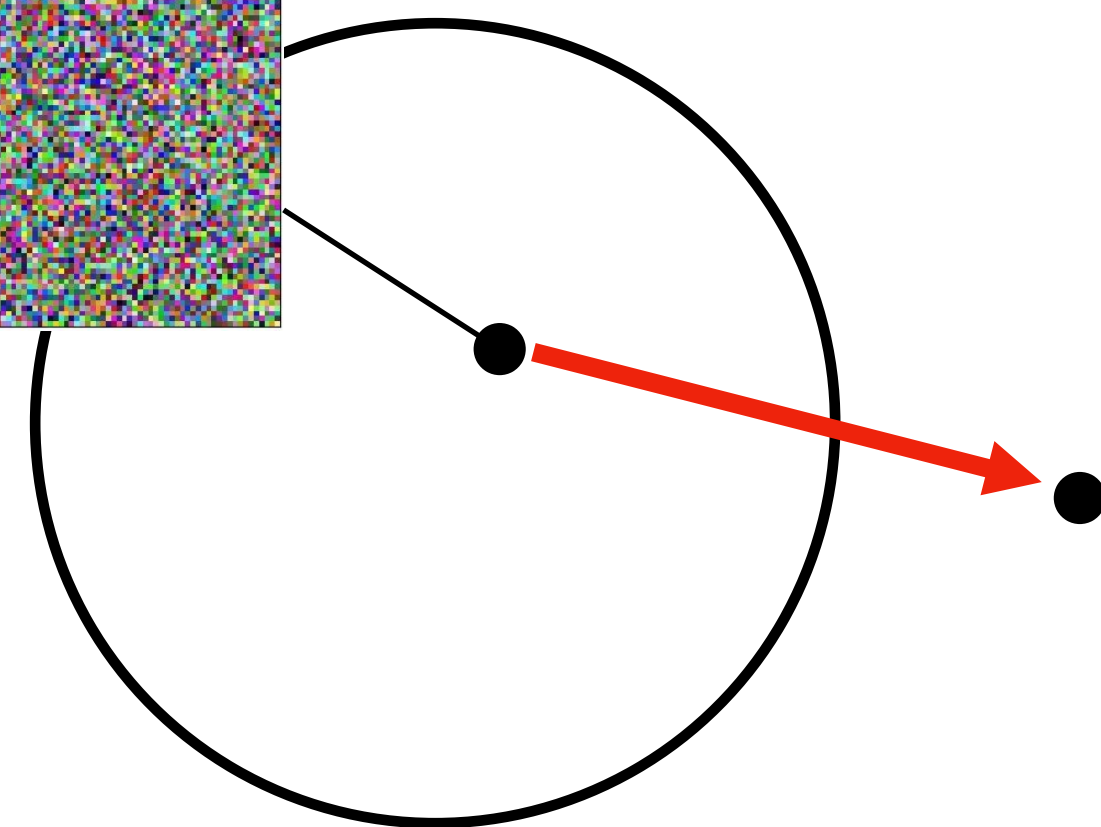


raspberry
images

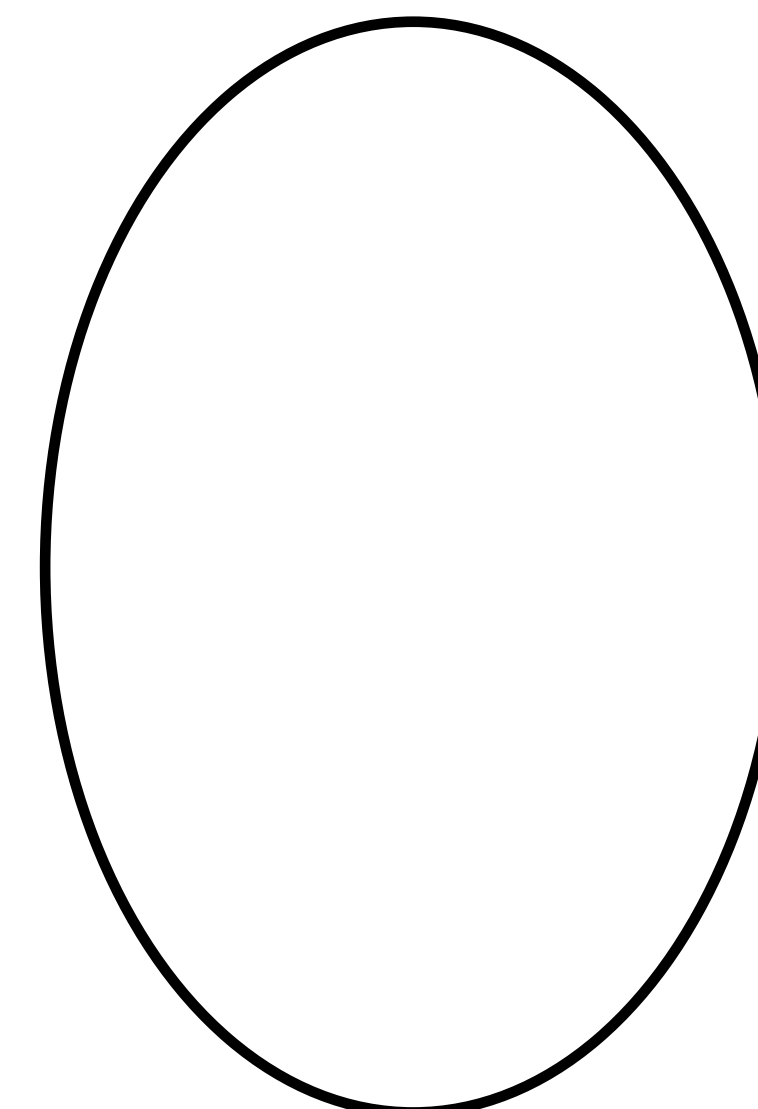
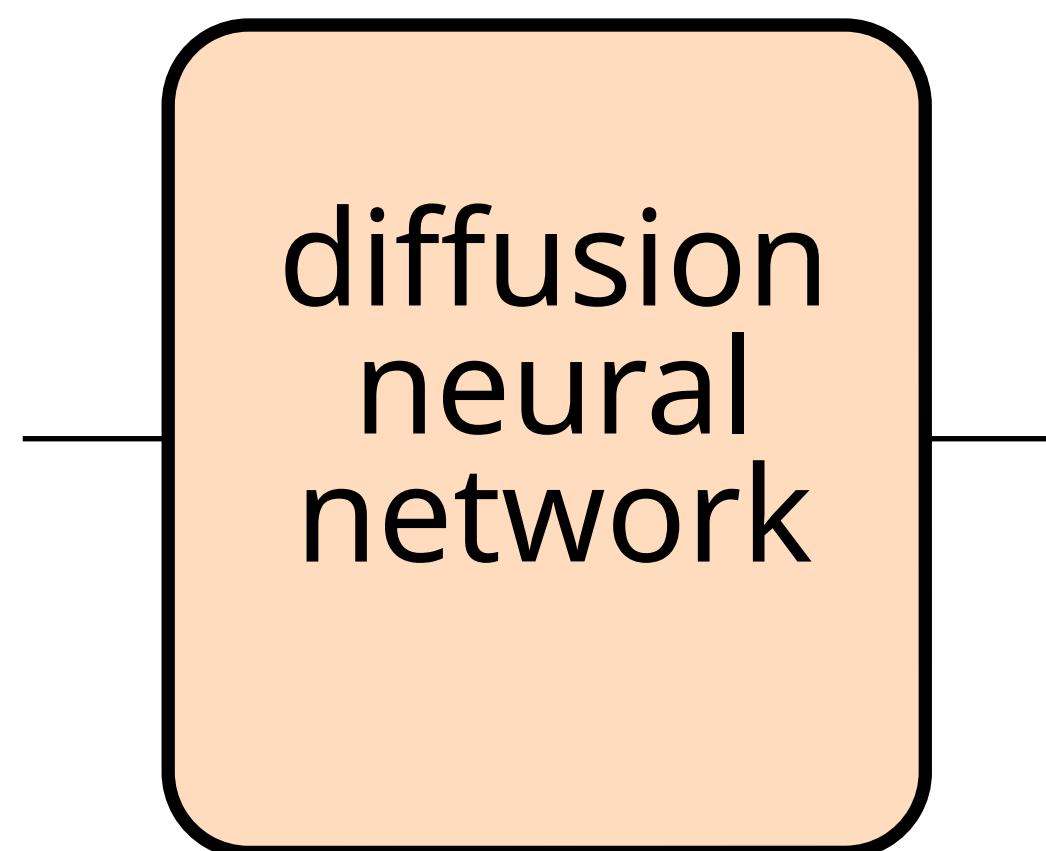


diffusion
neural
network

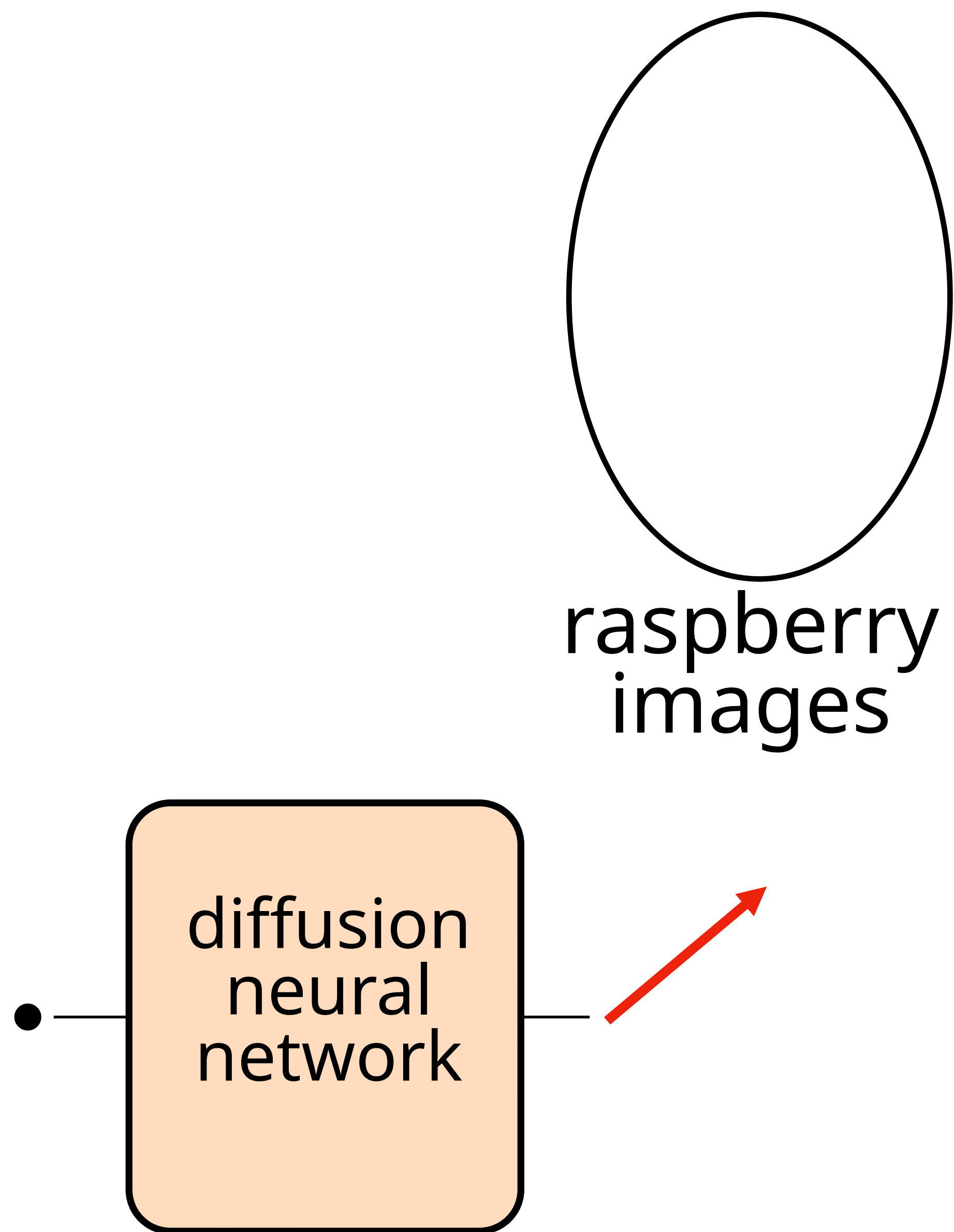
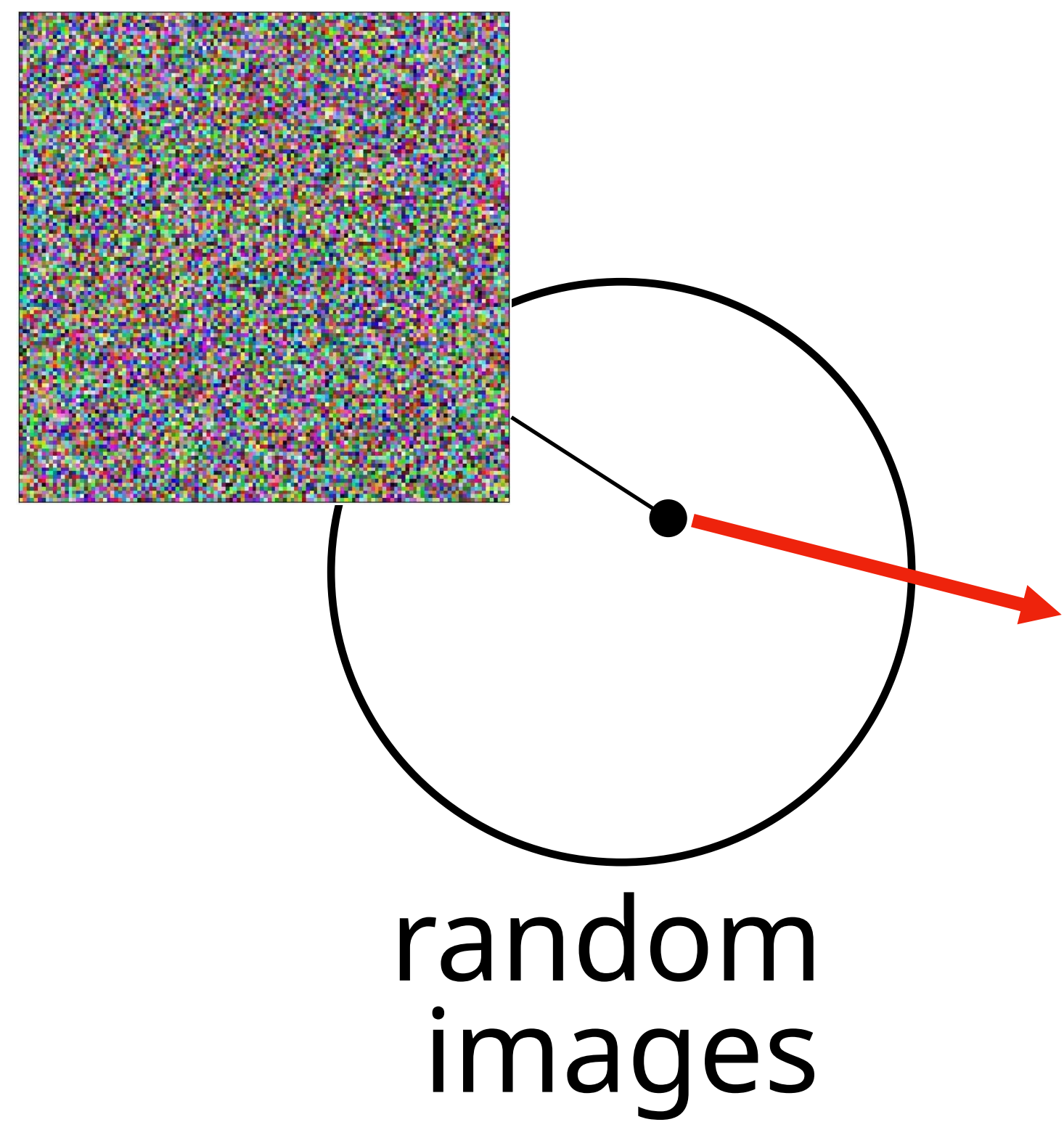


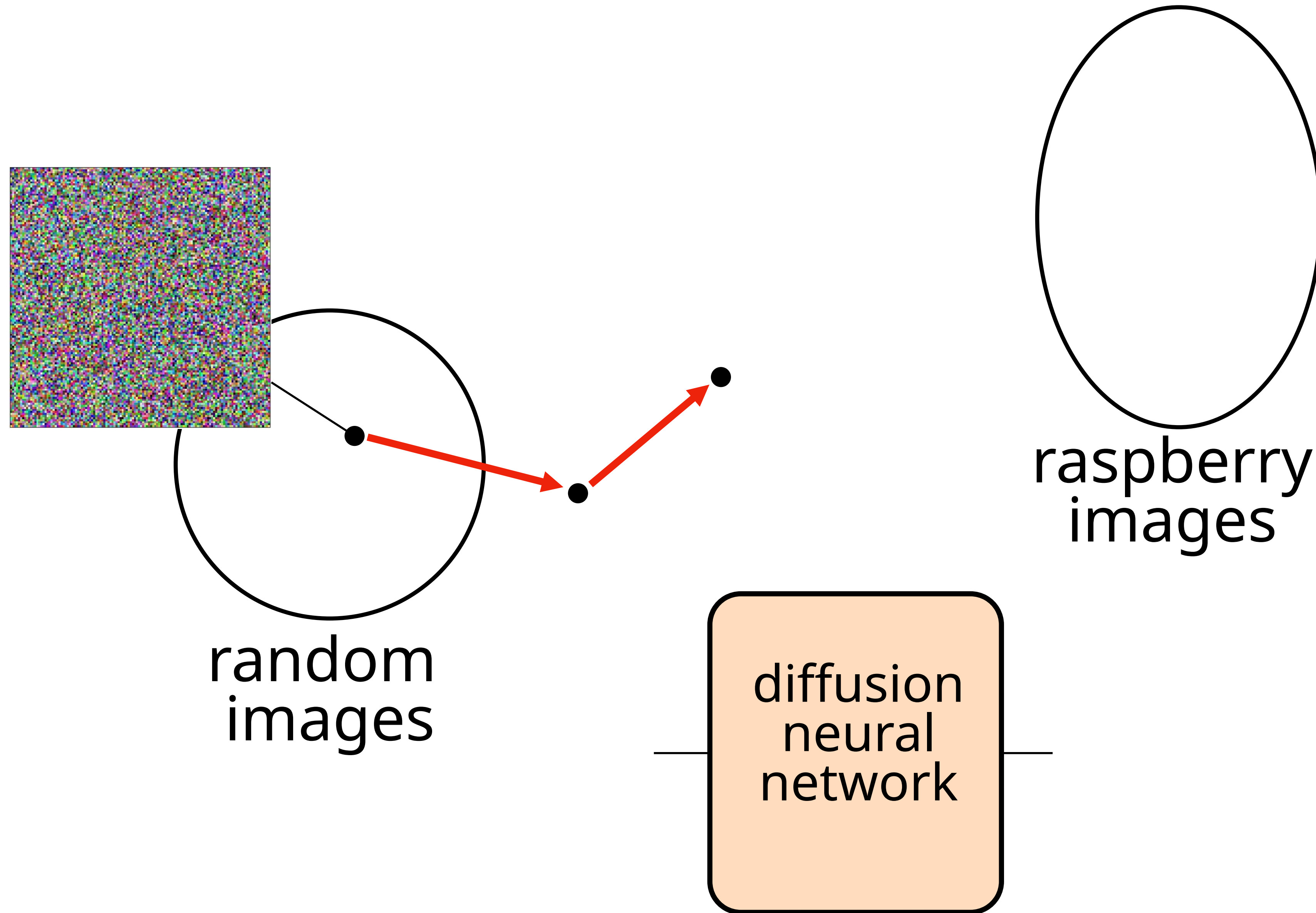


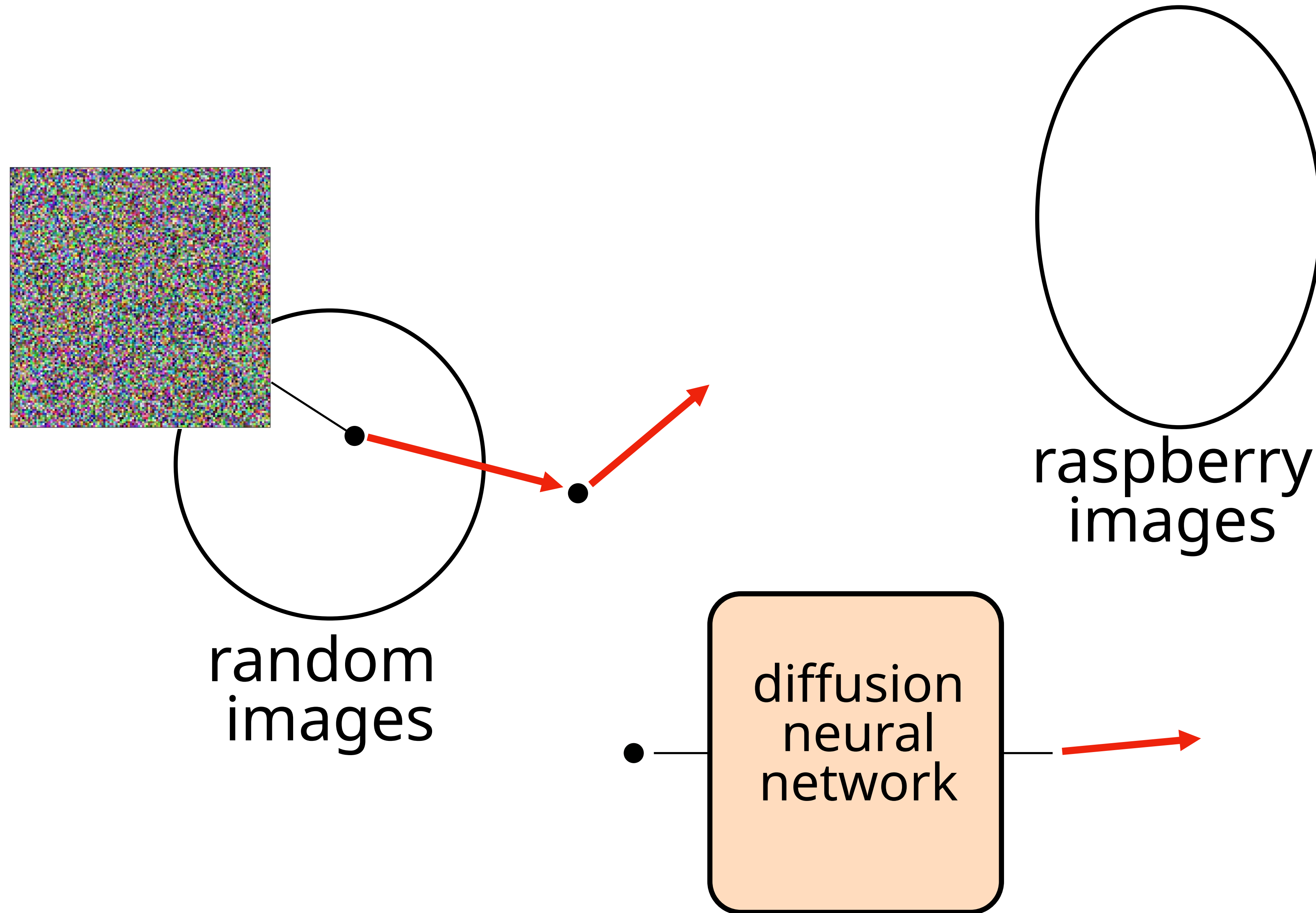
random
images

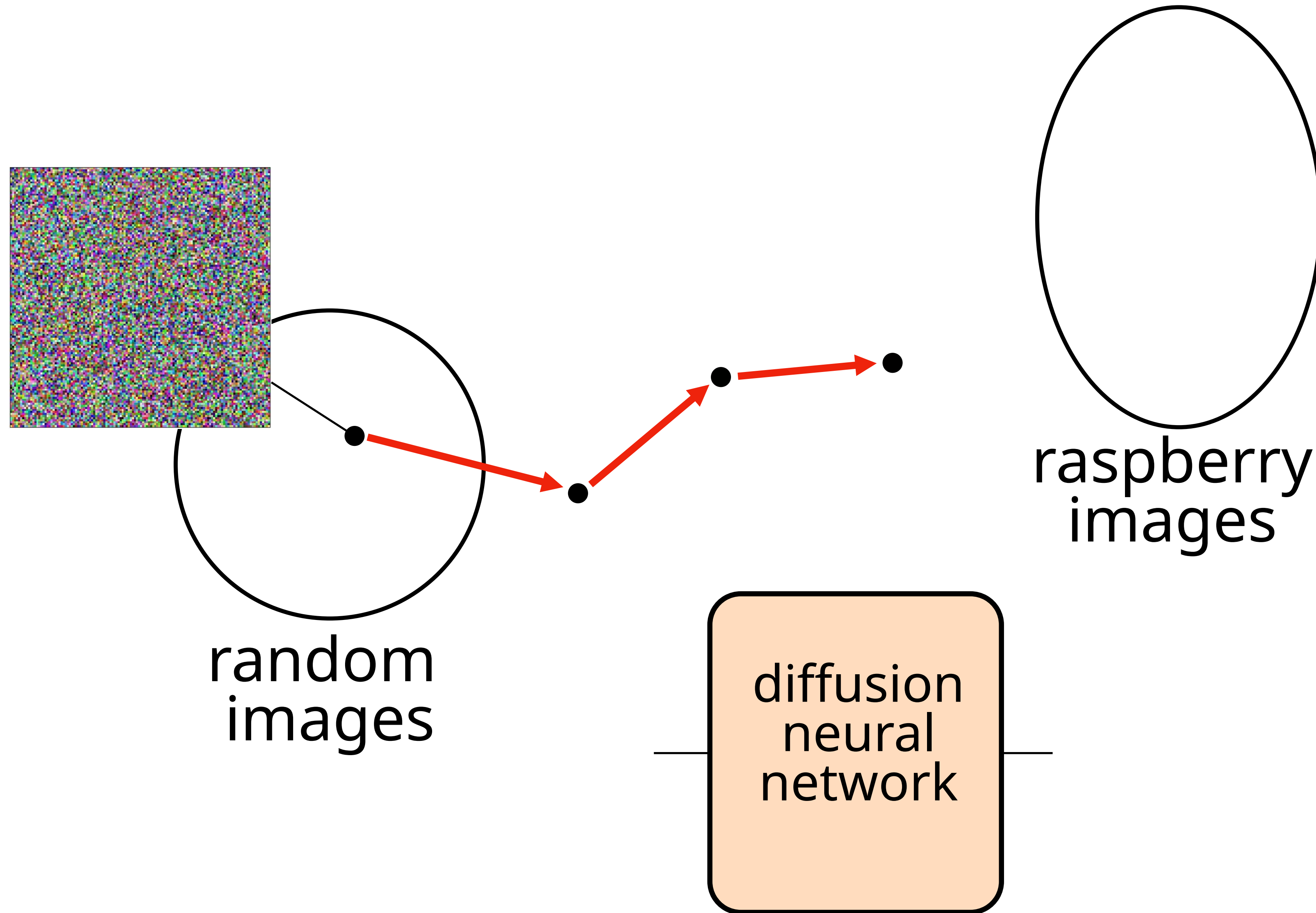


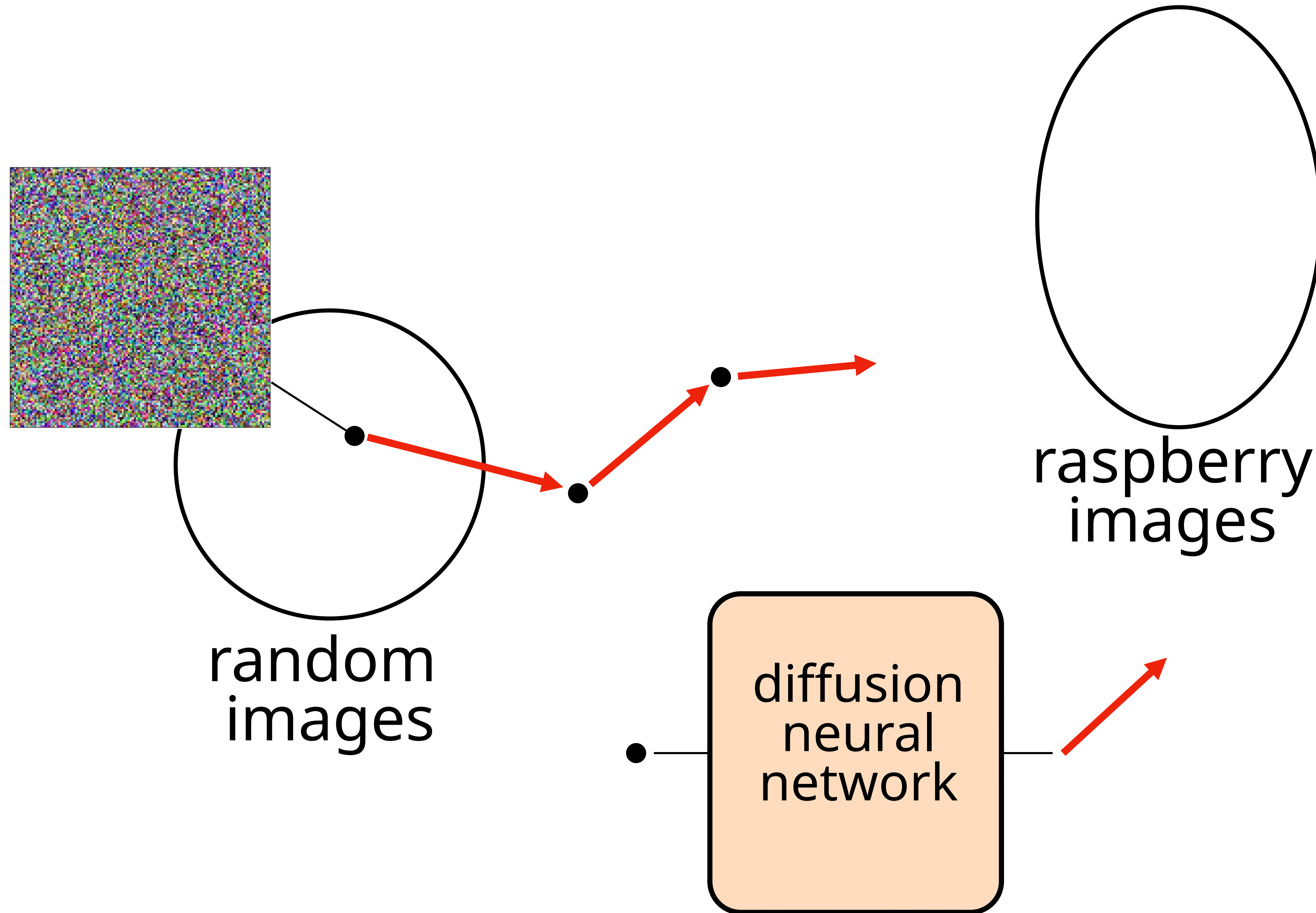
raspberry
images

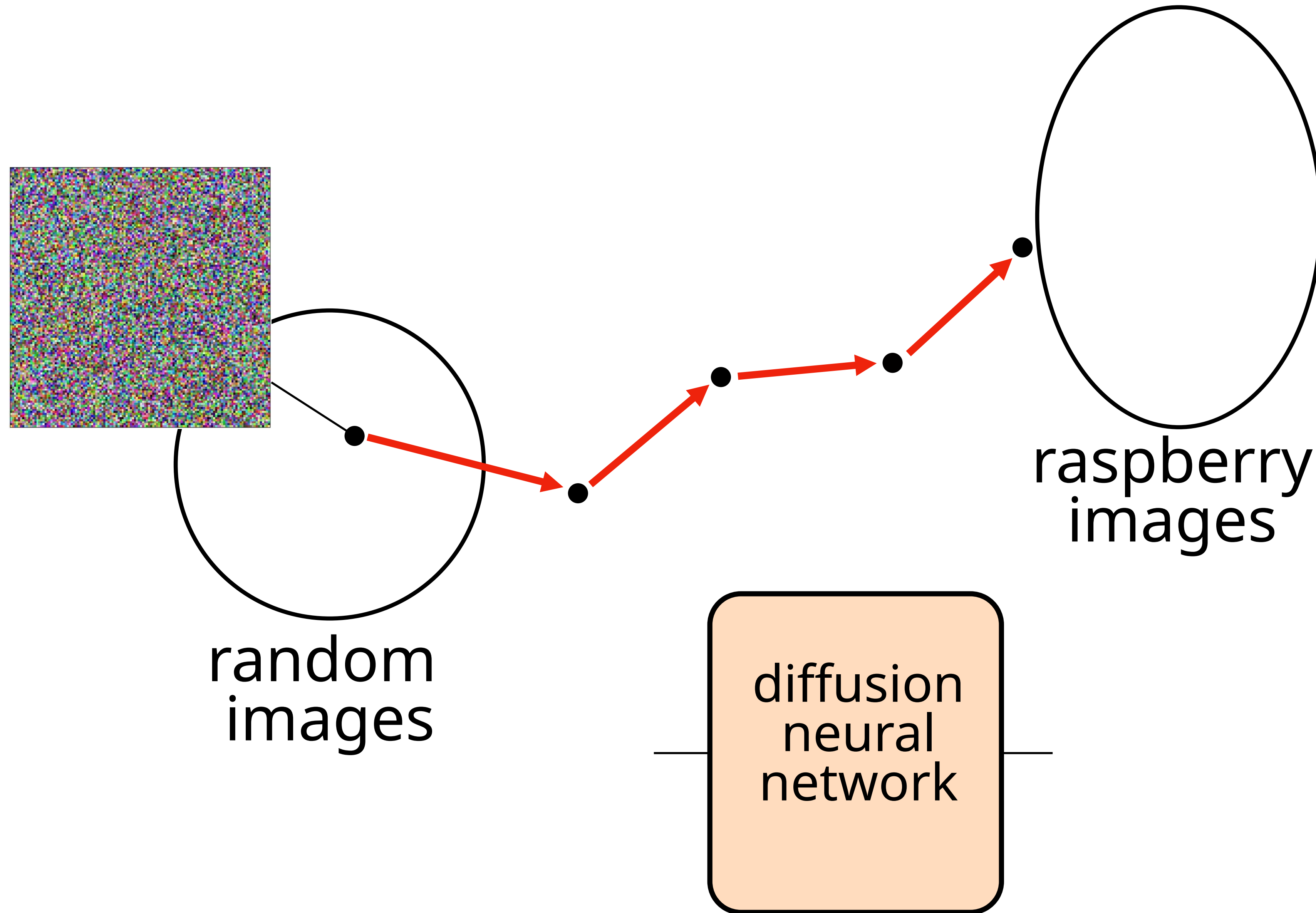


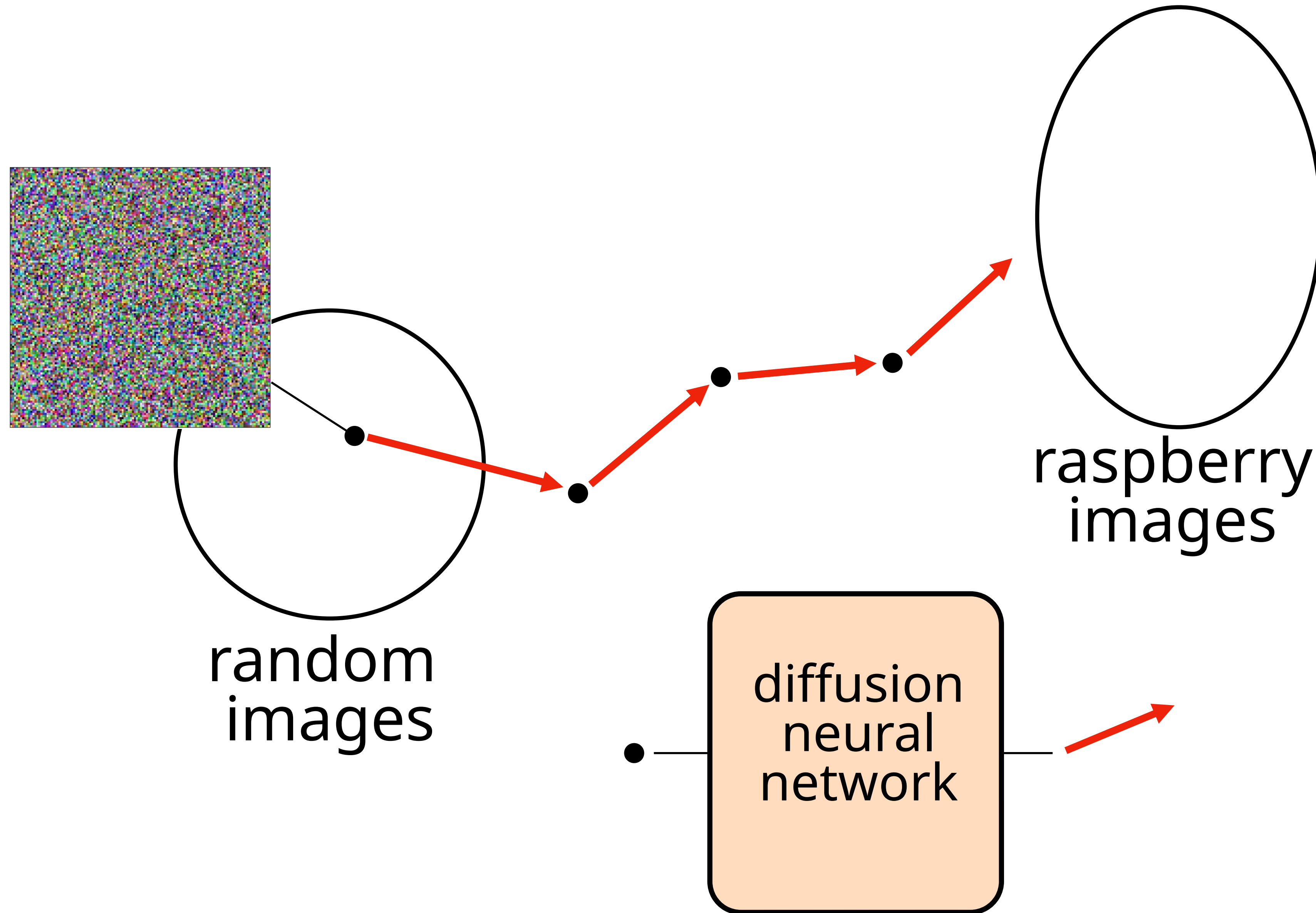












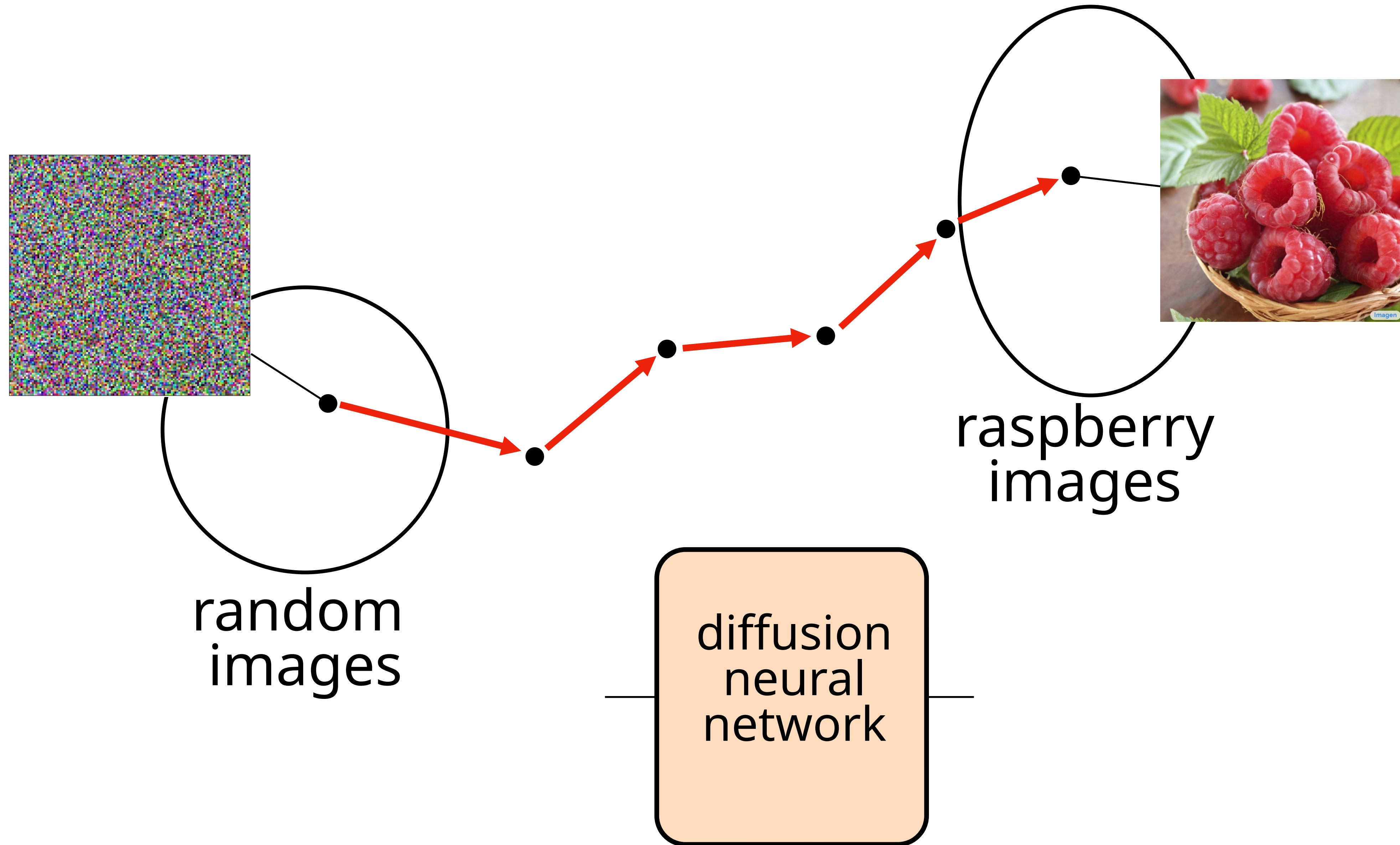
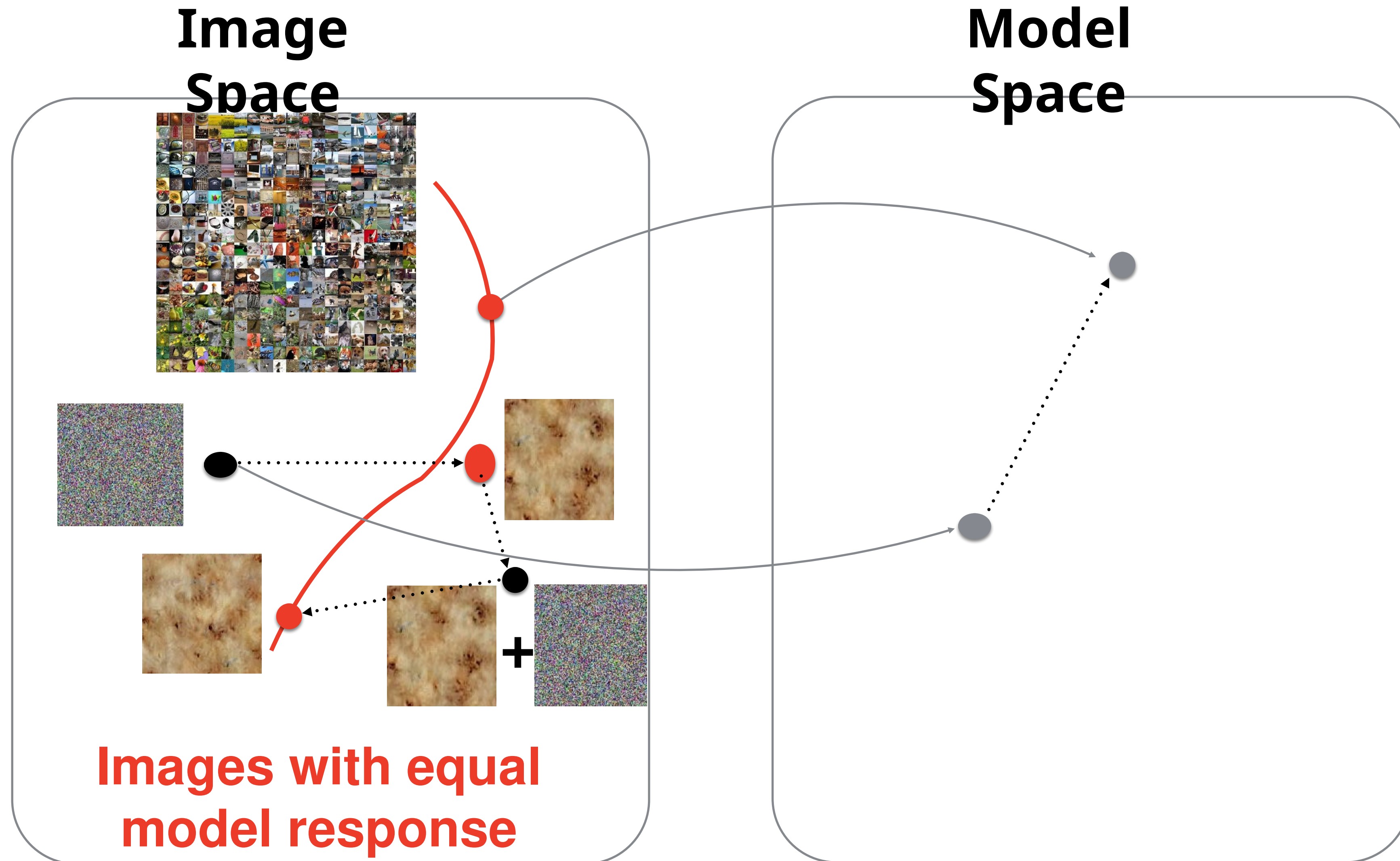
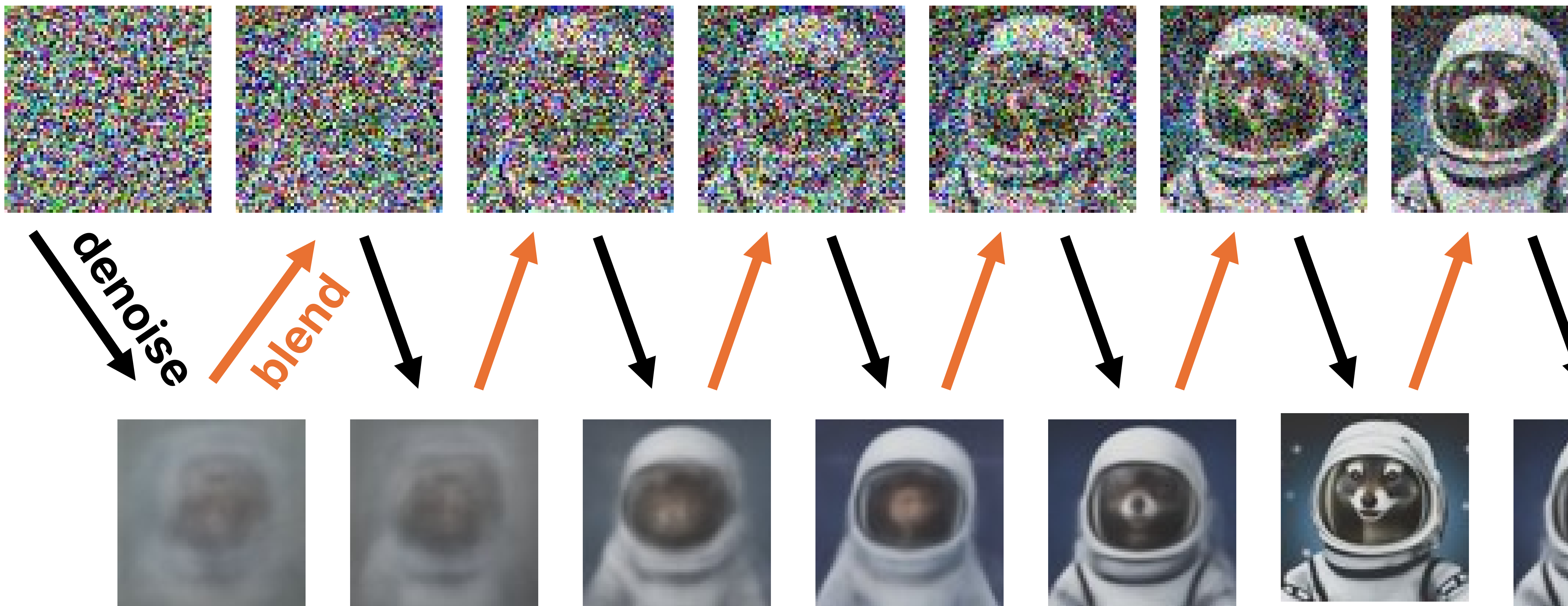
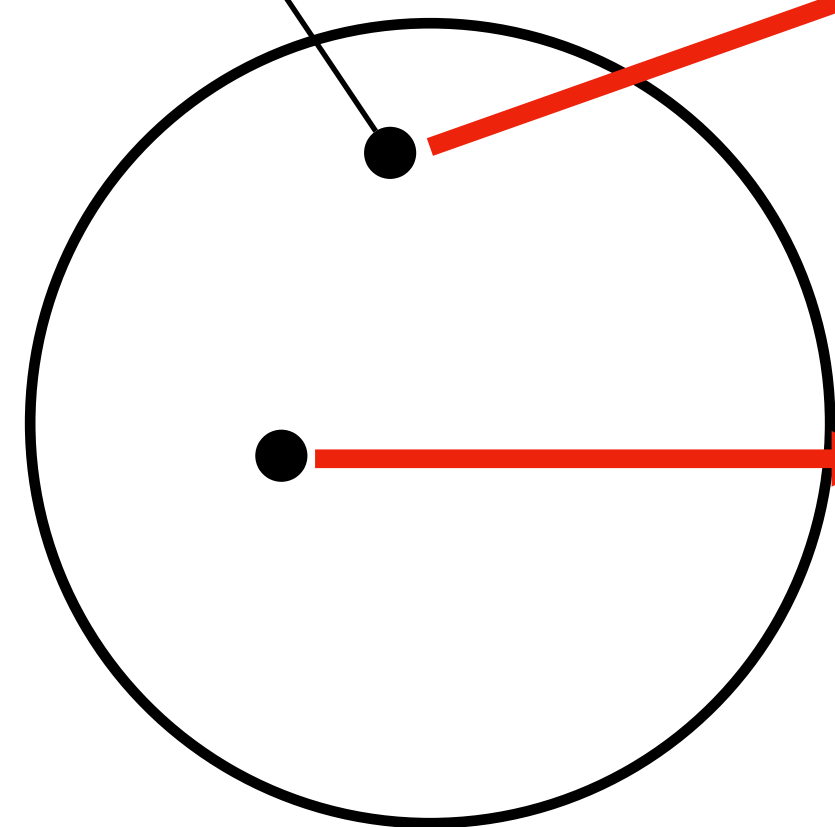
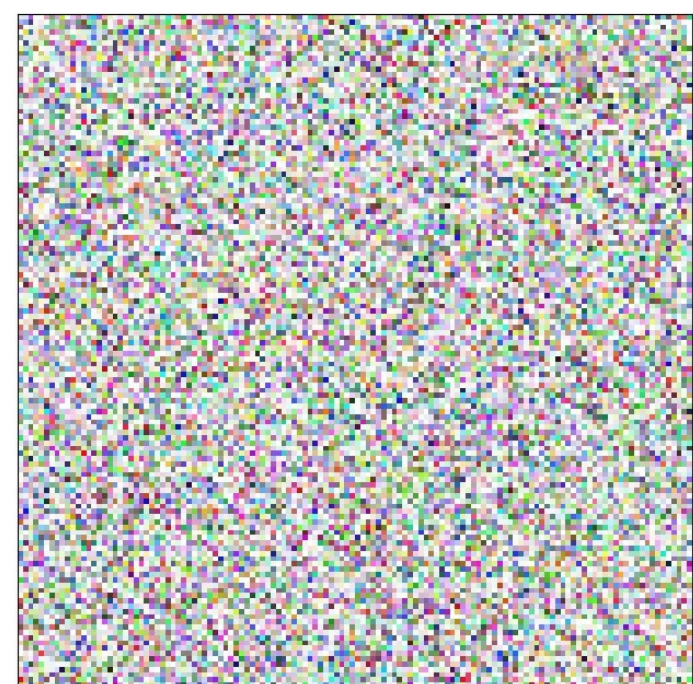


Image Diffusion

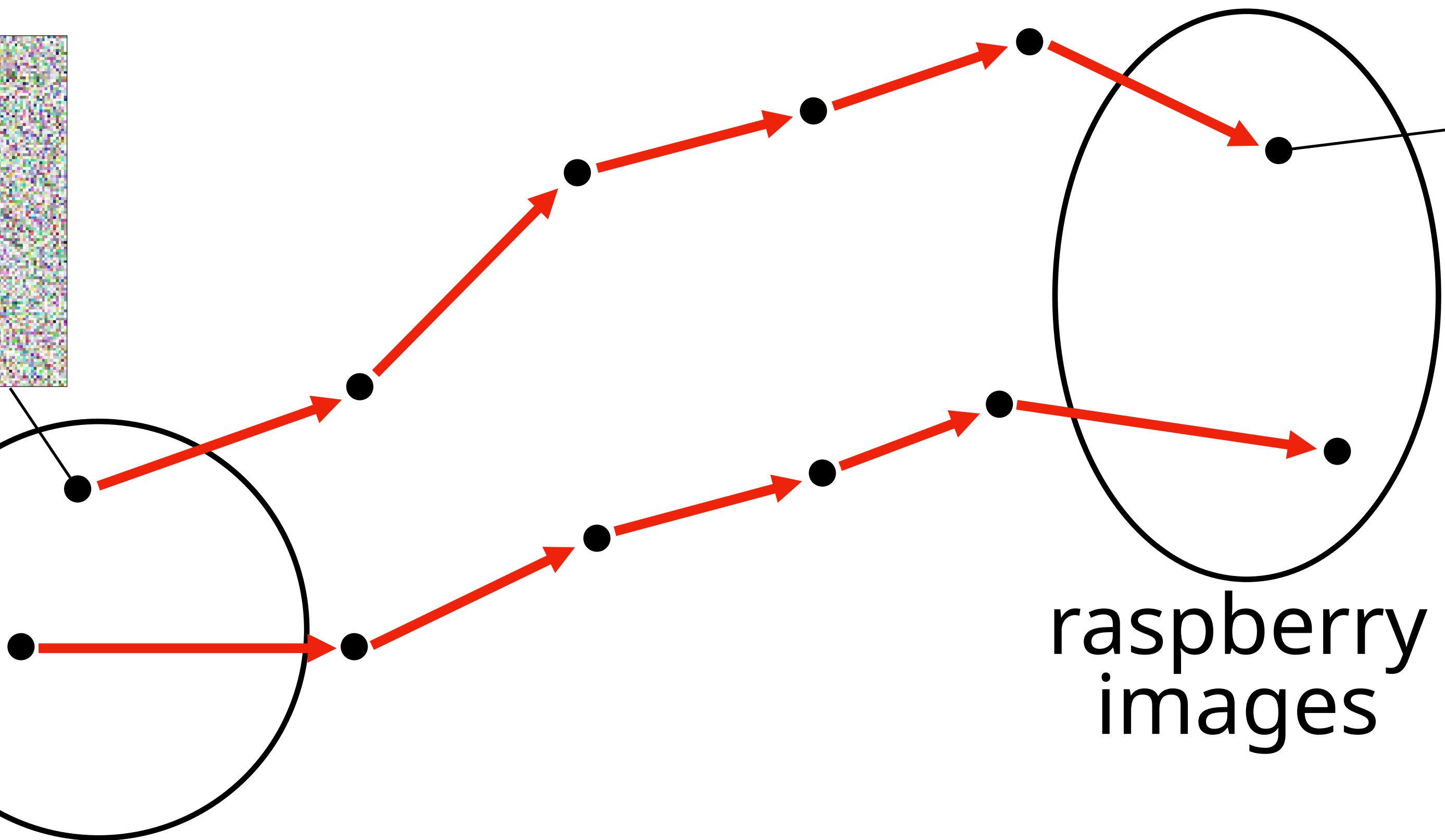


Flow Matching

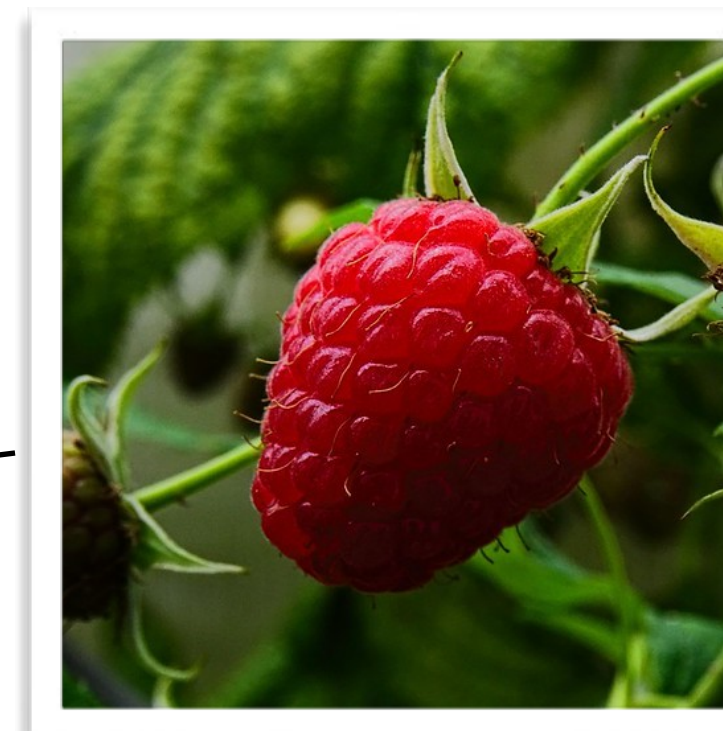


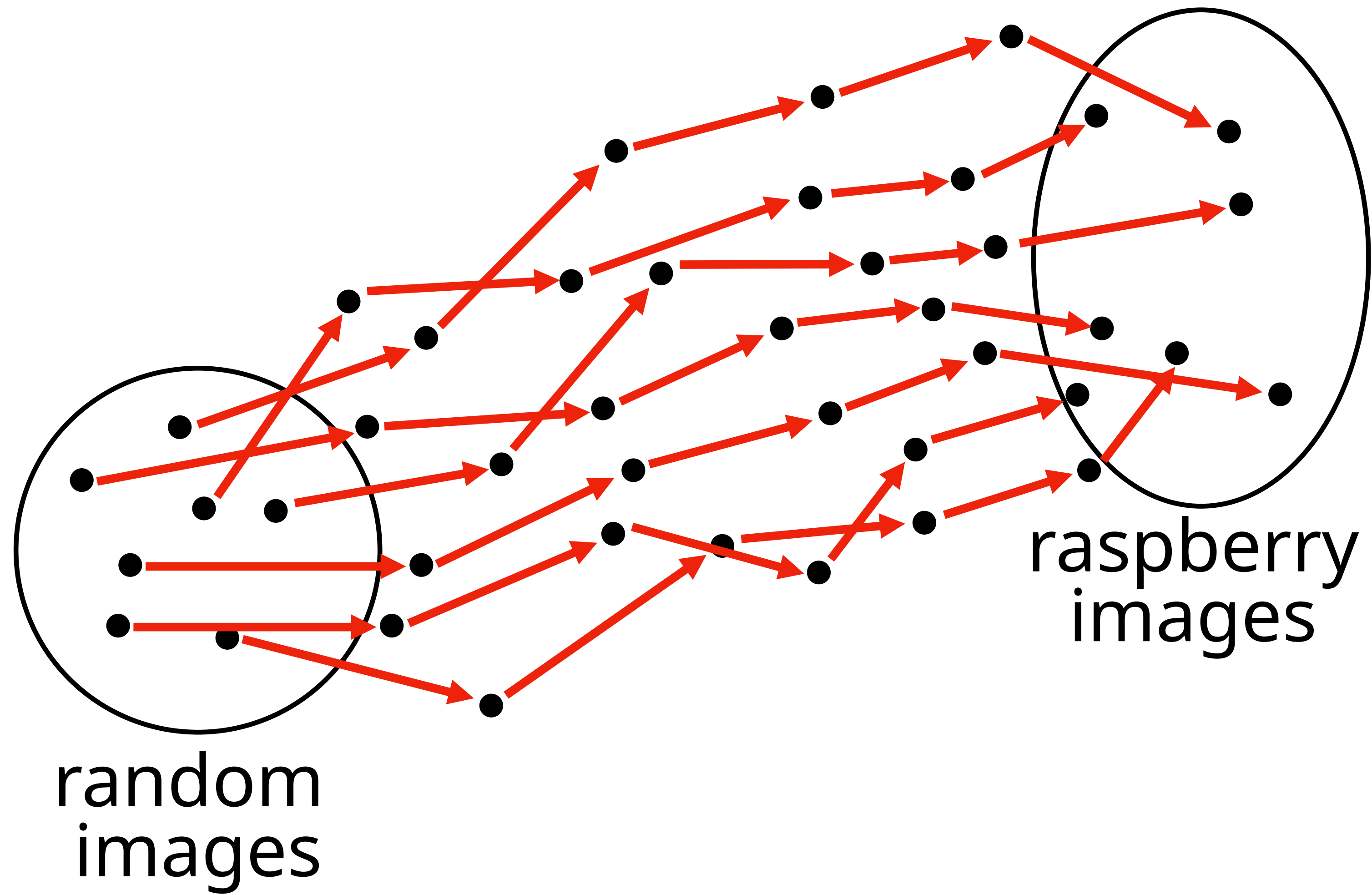


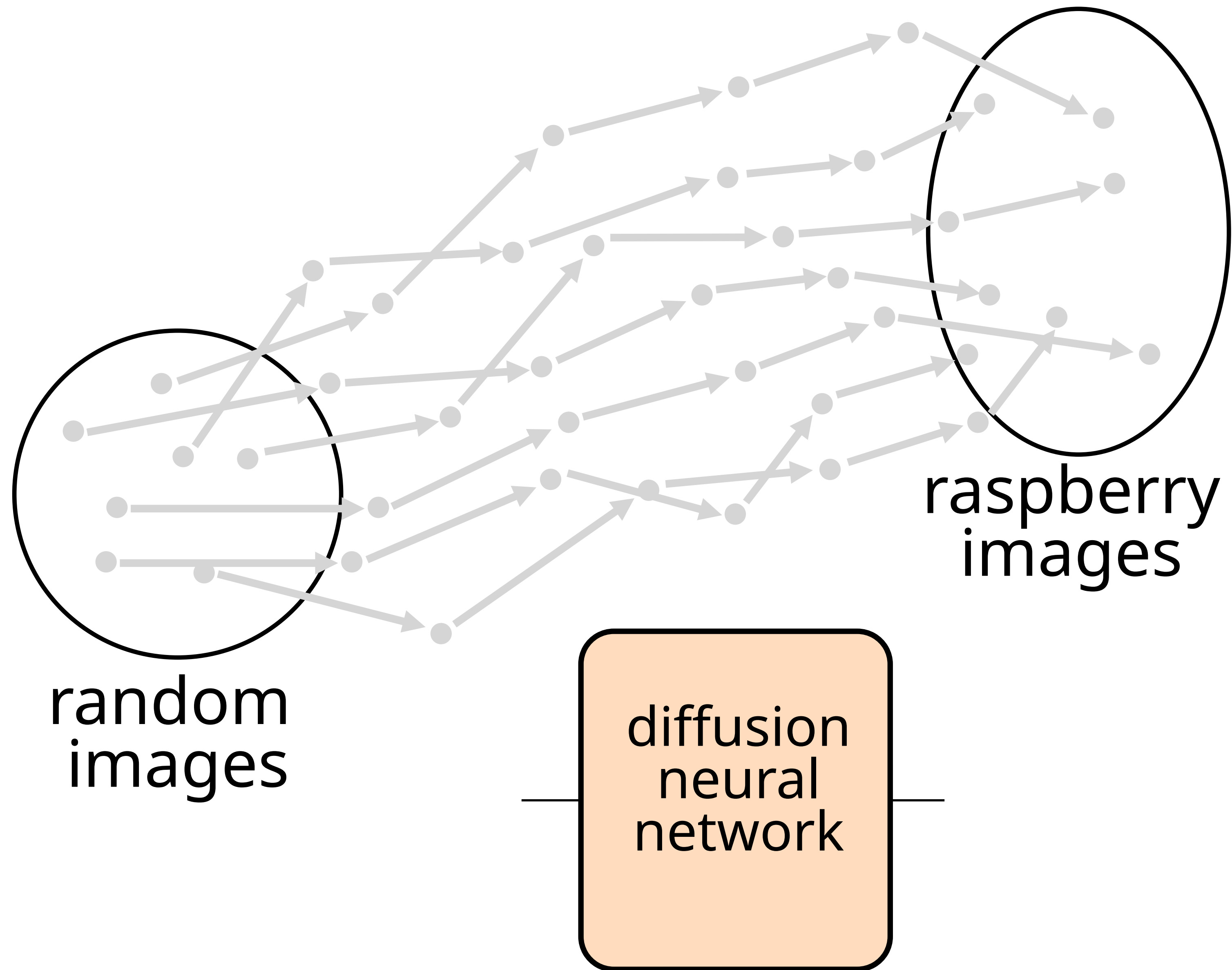
random
images

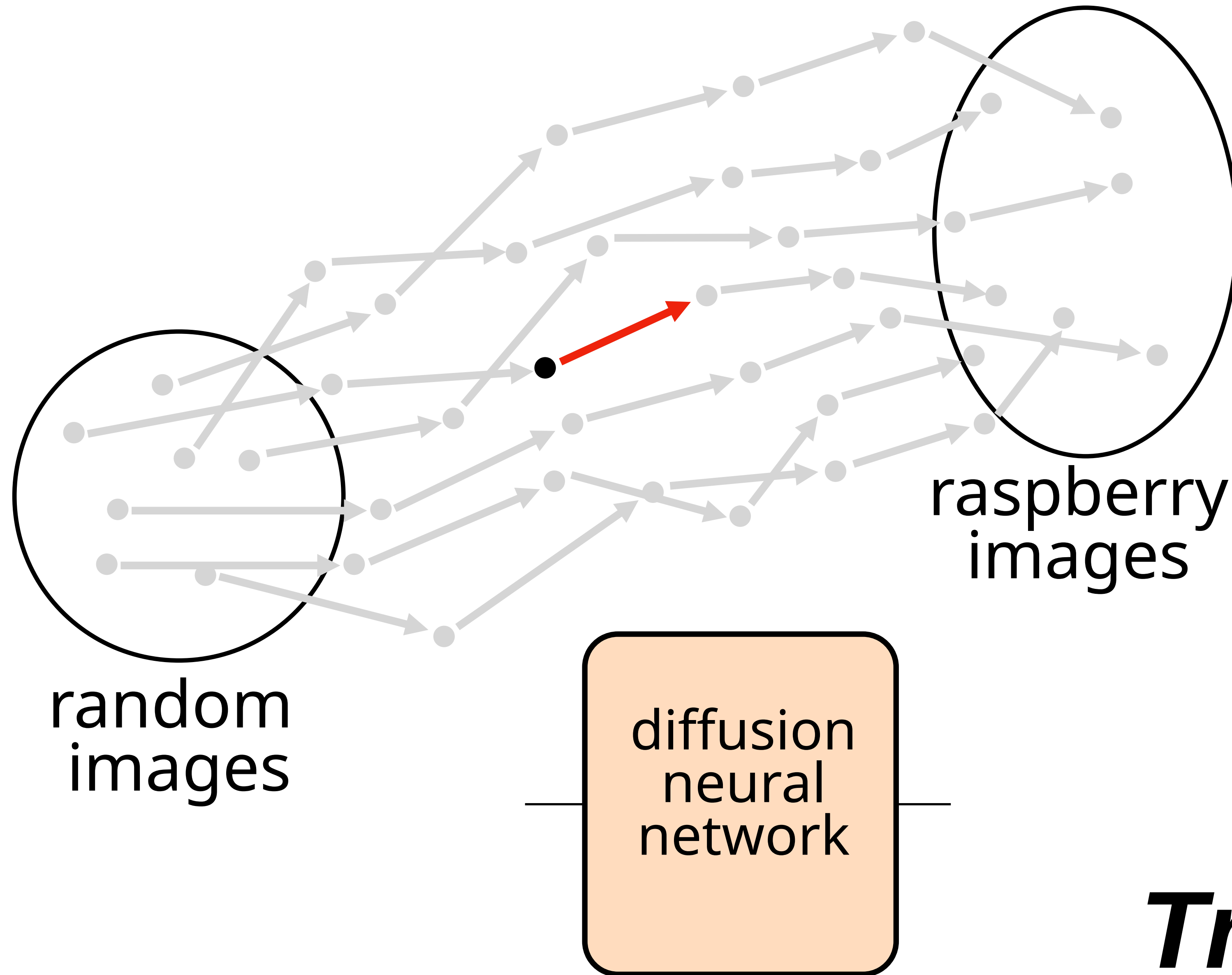


raspberry
images



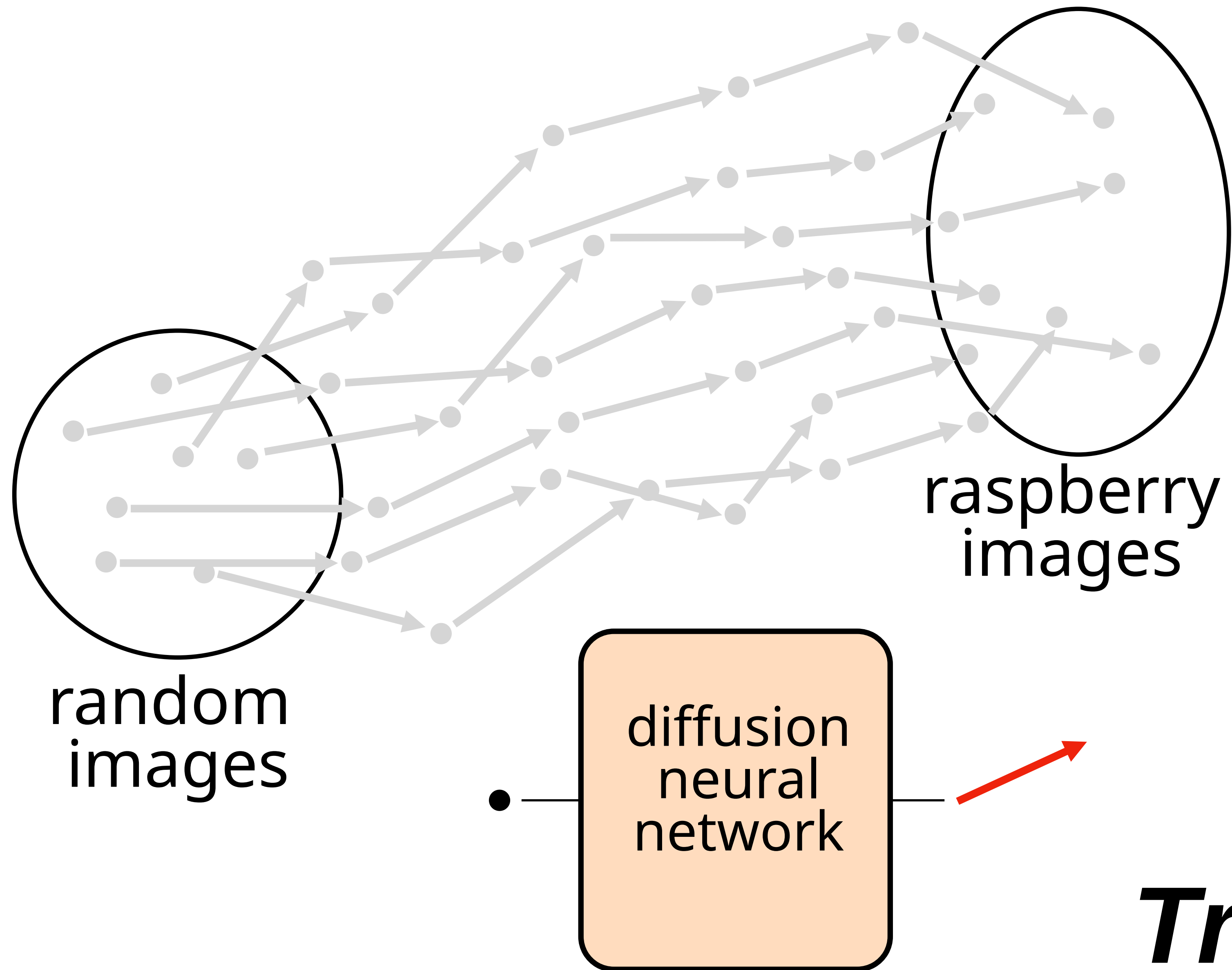






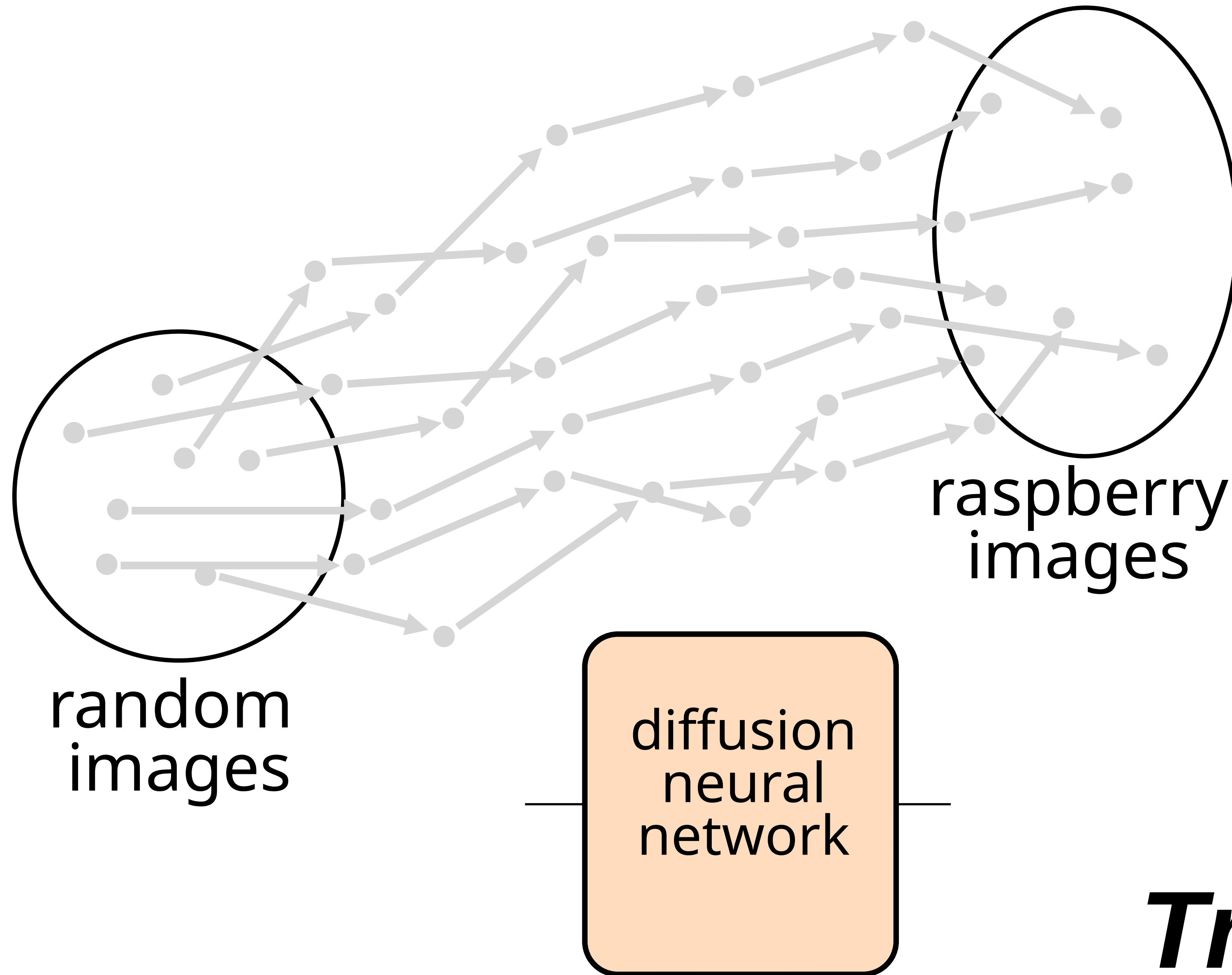
Training

slide from Steve Seitz's [video](#)



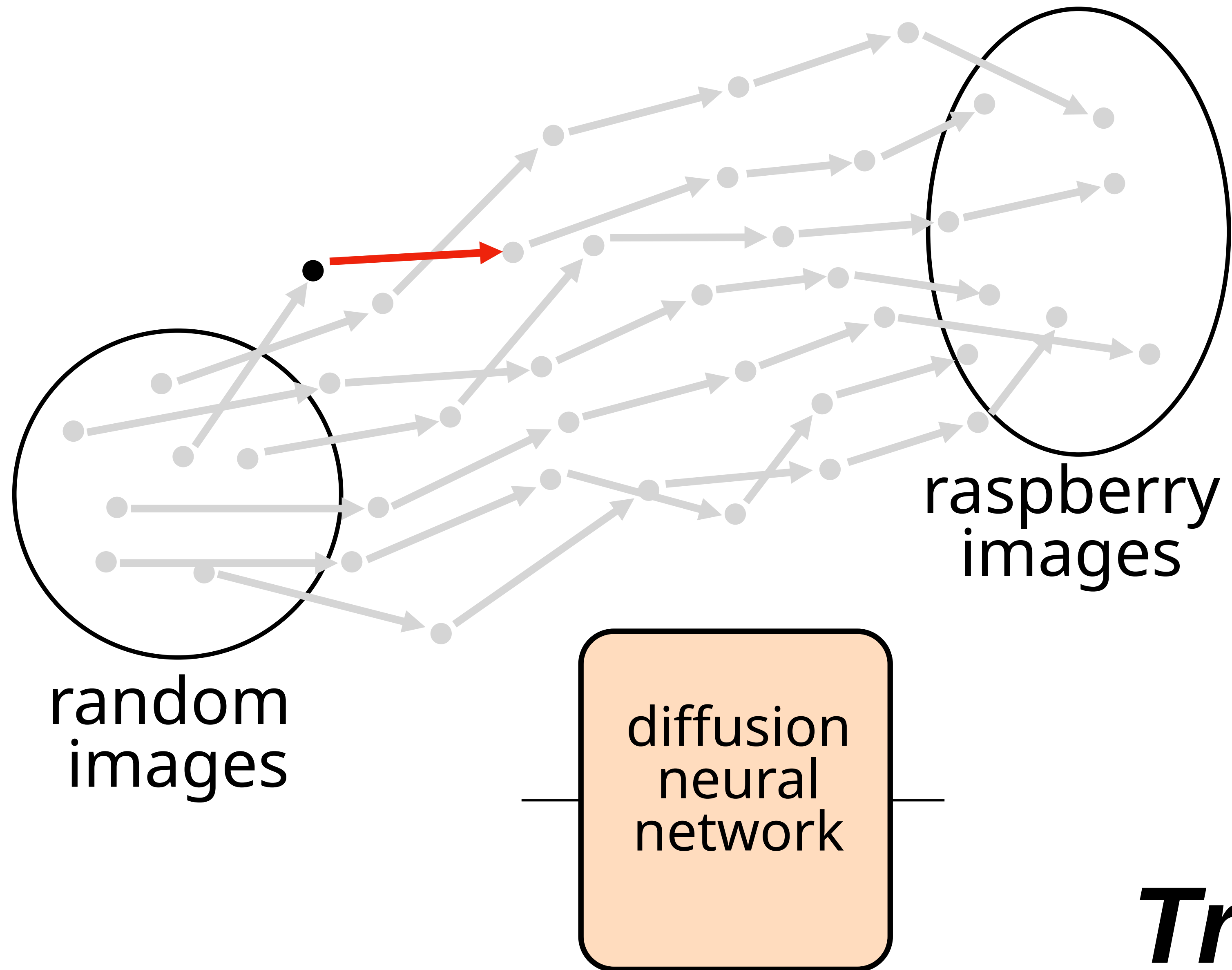
Training

slide from Steve Seitz's [video](#)



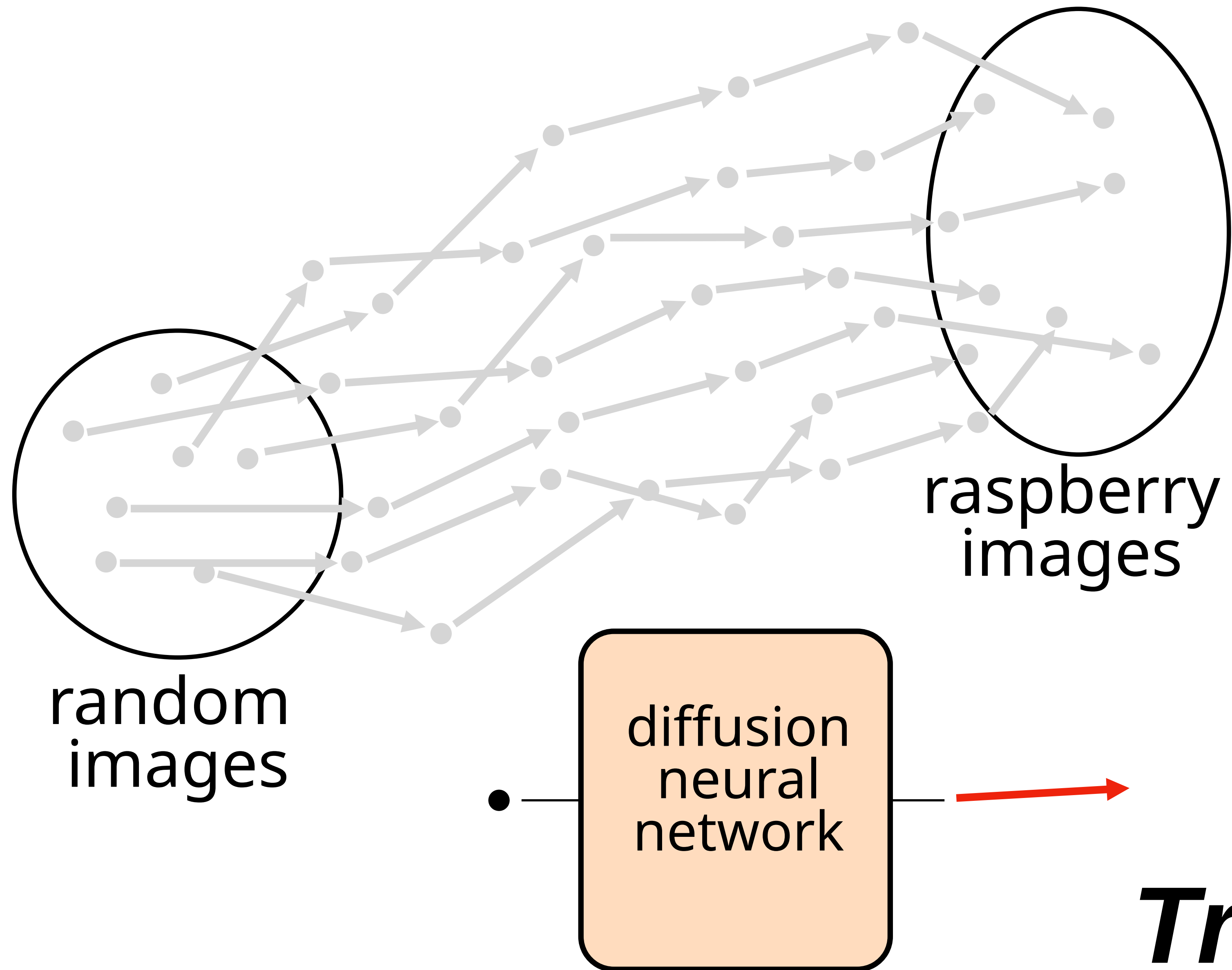
Training

slide from Steve Seitz's [video](#)



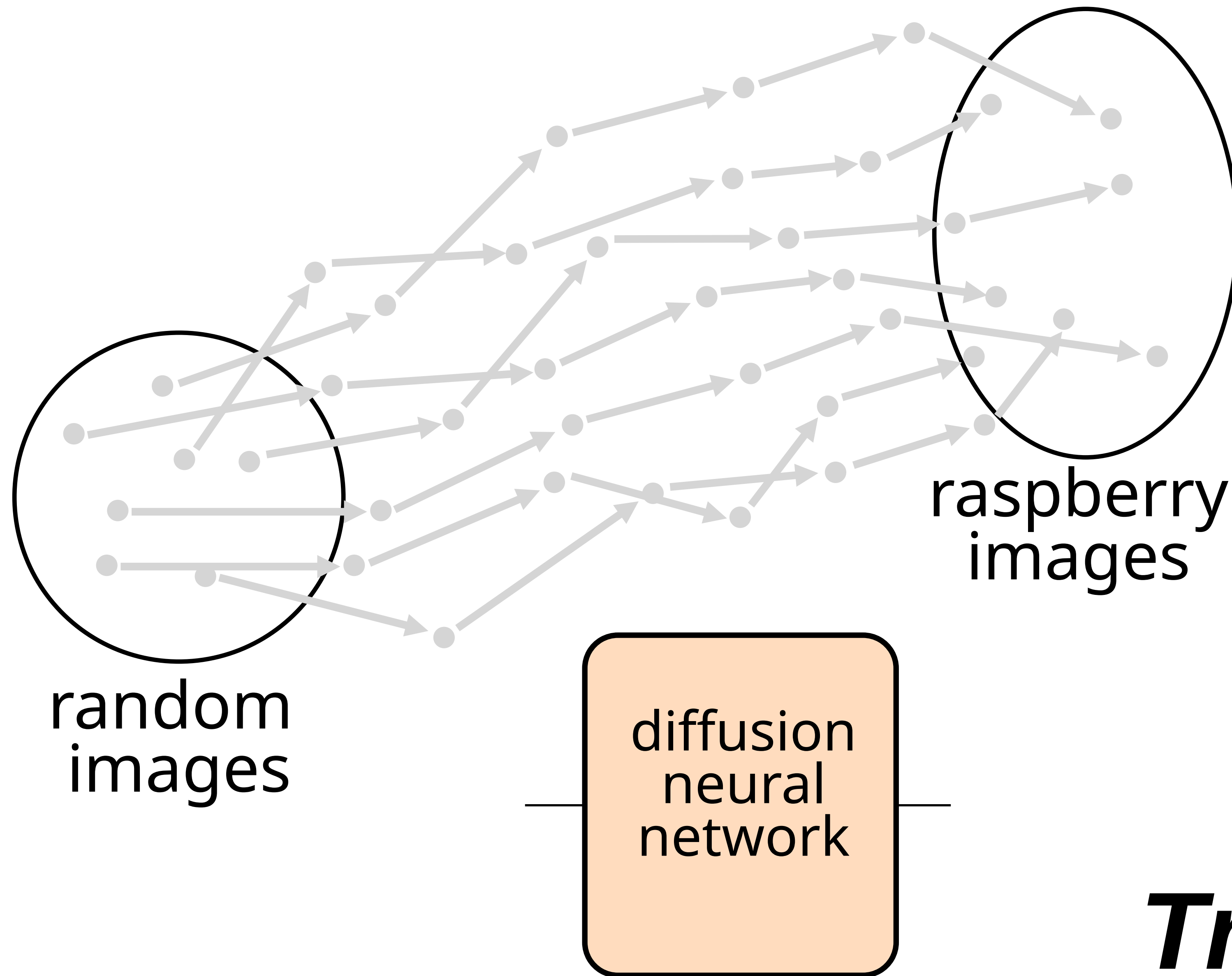
Training

slide from Steve Seitz's [video](#)



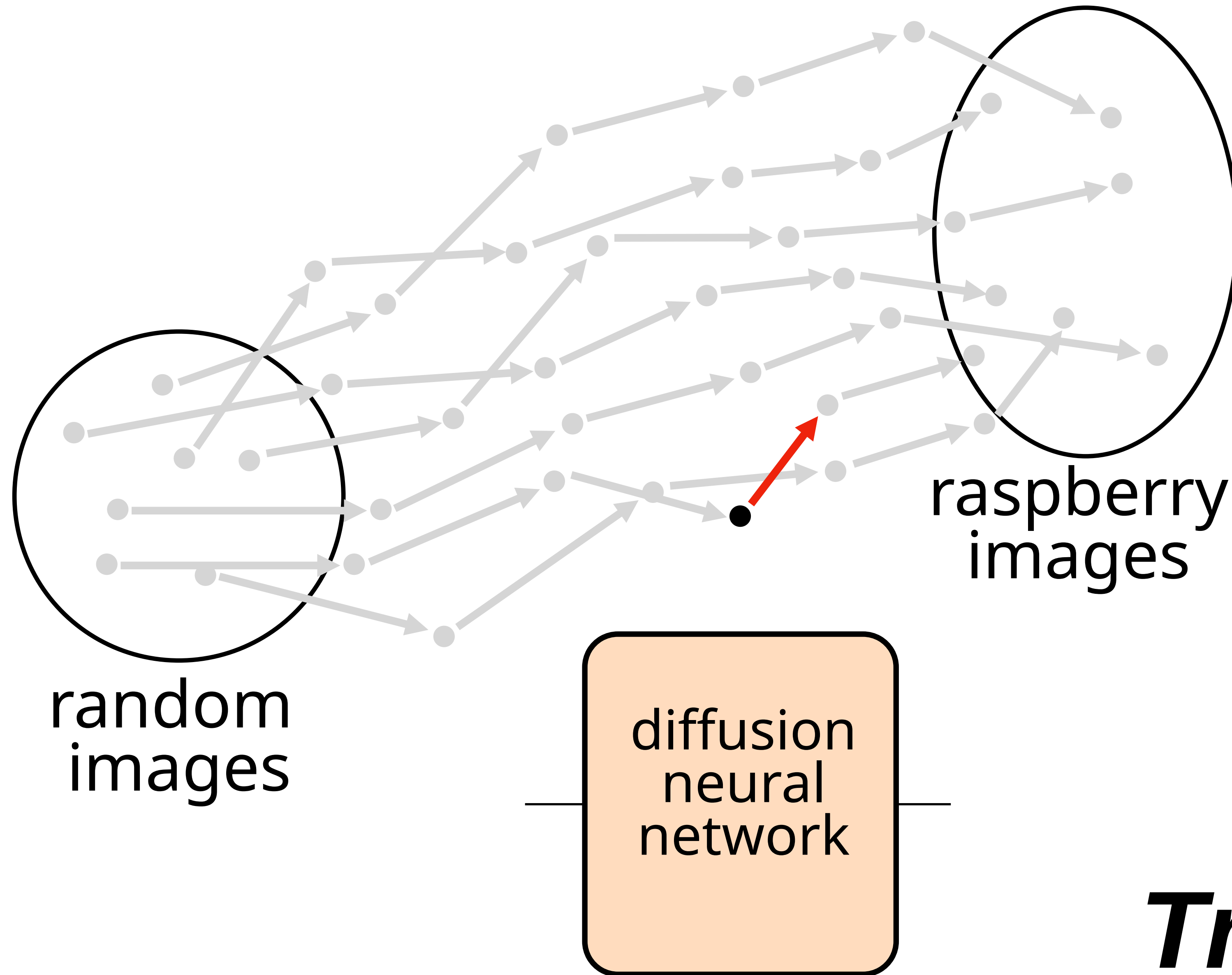
Training

slide from Steve Seitz's [video](#)



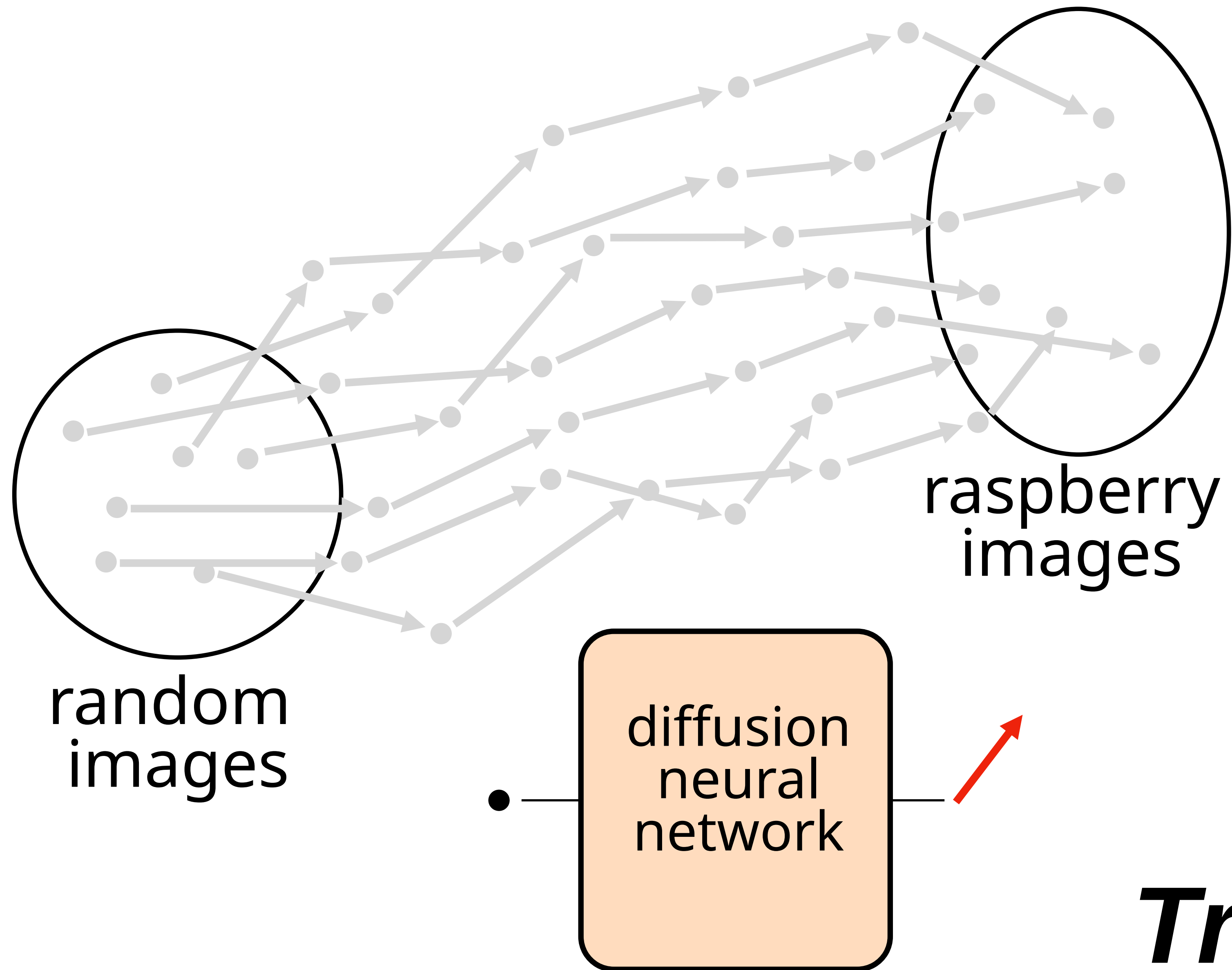
Training

slide from Steve Seitz's [video](#)

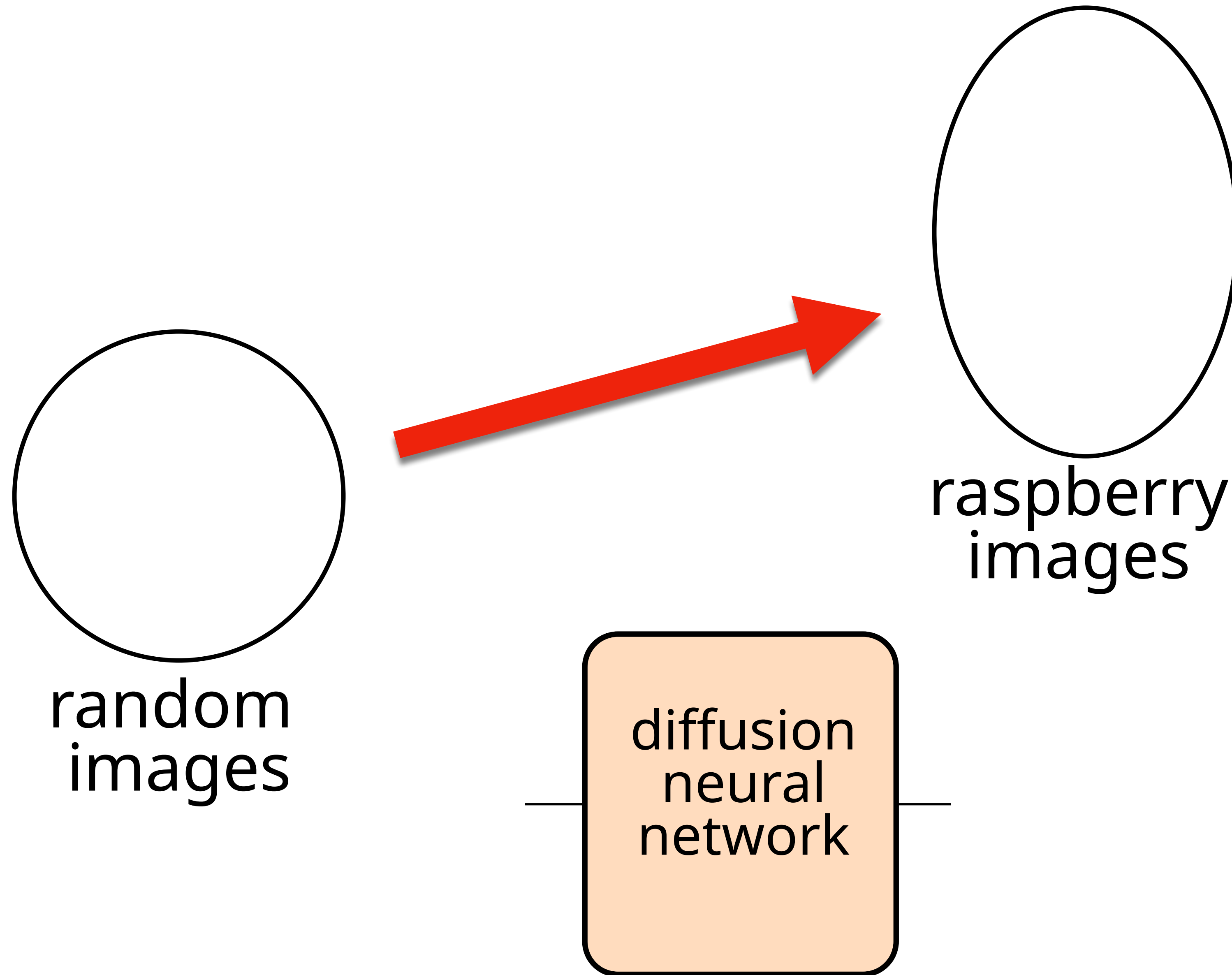


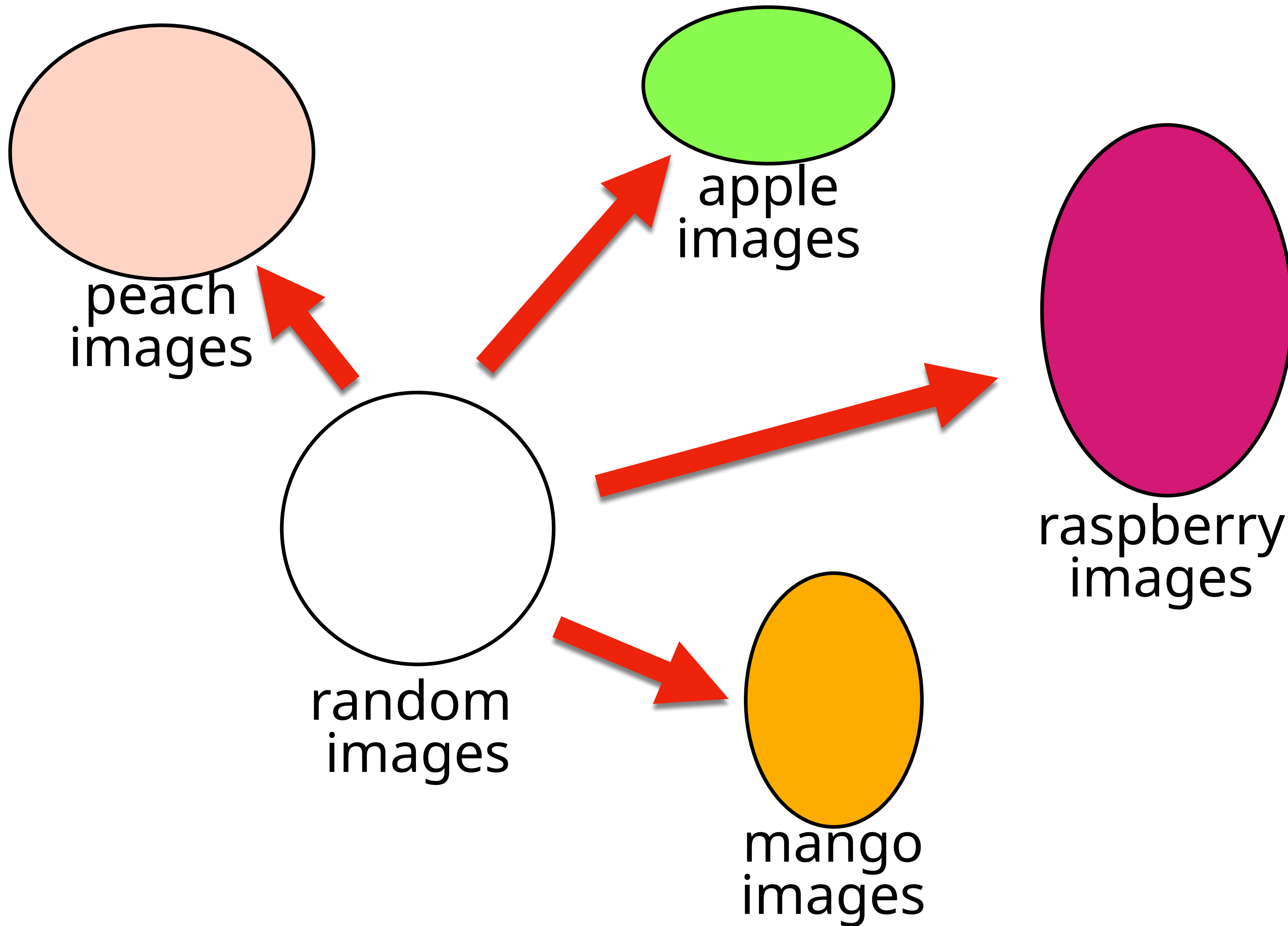
Training

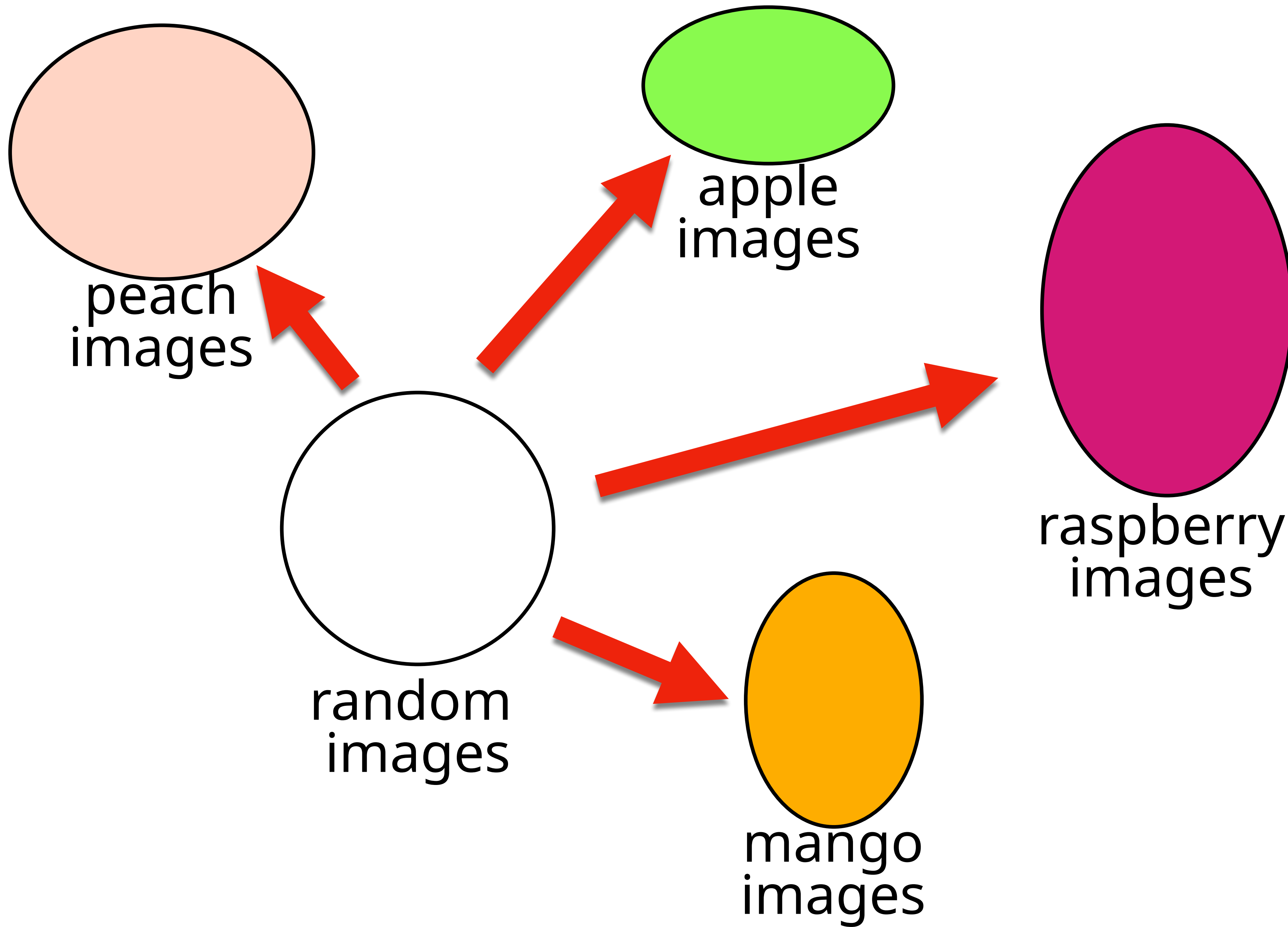
slide from Steve Seitz's [video](#)

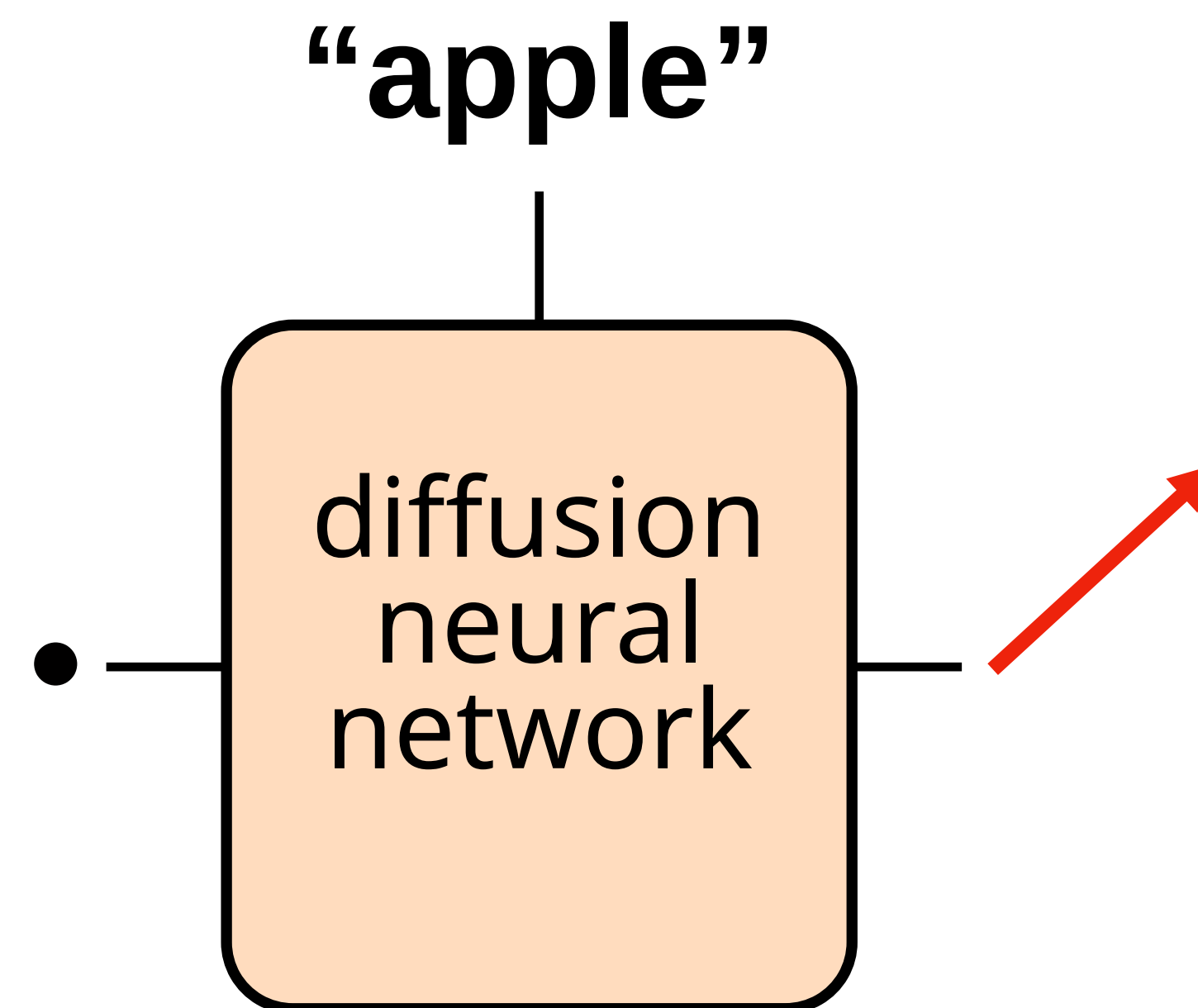
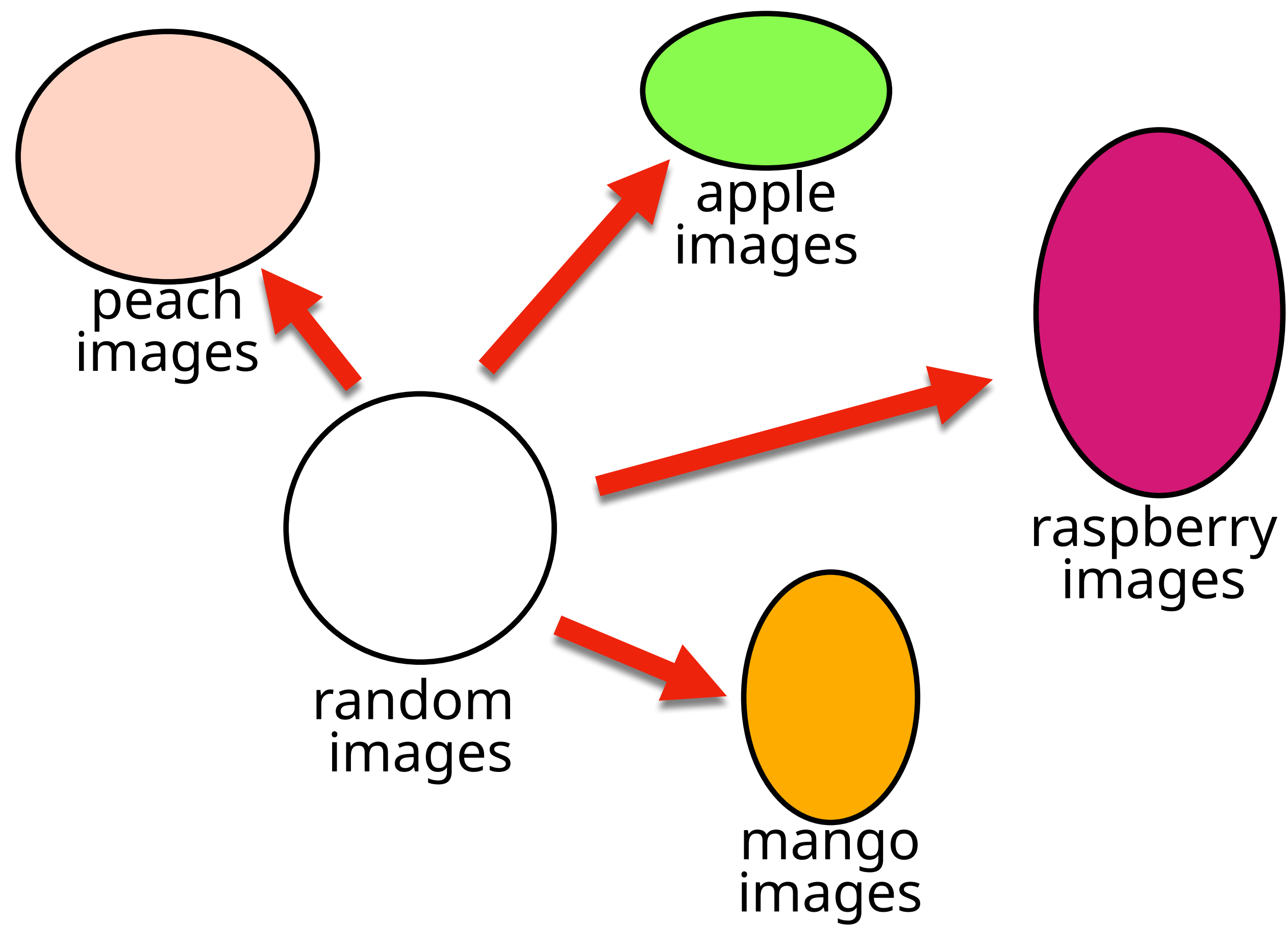


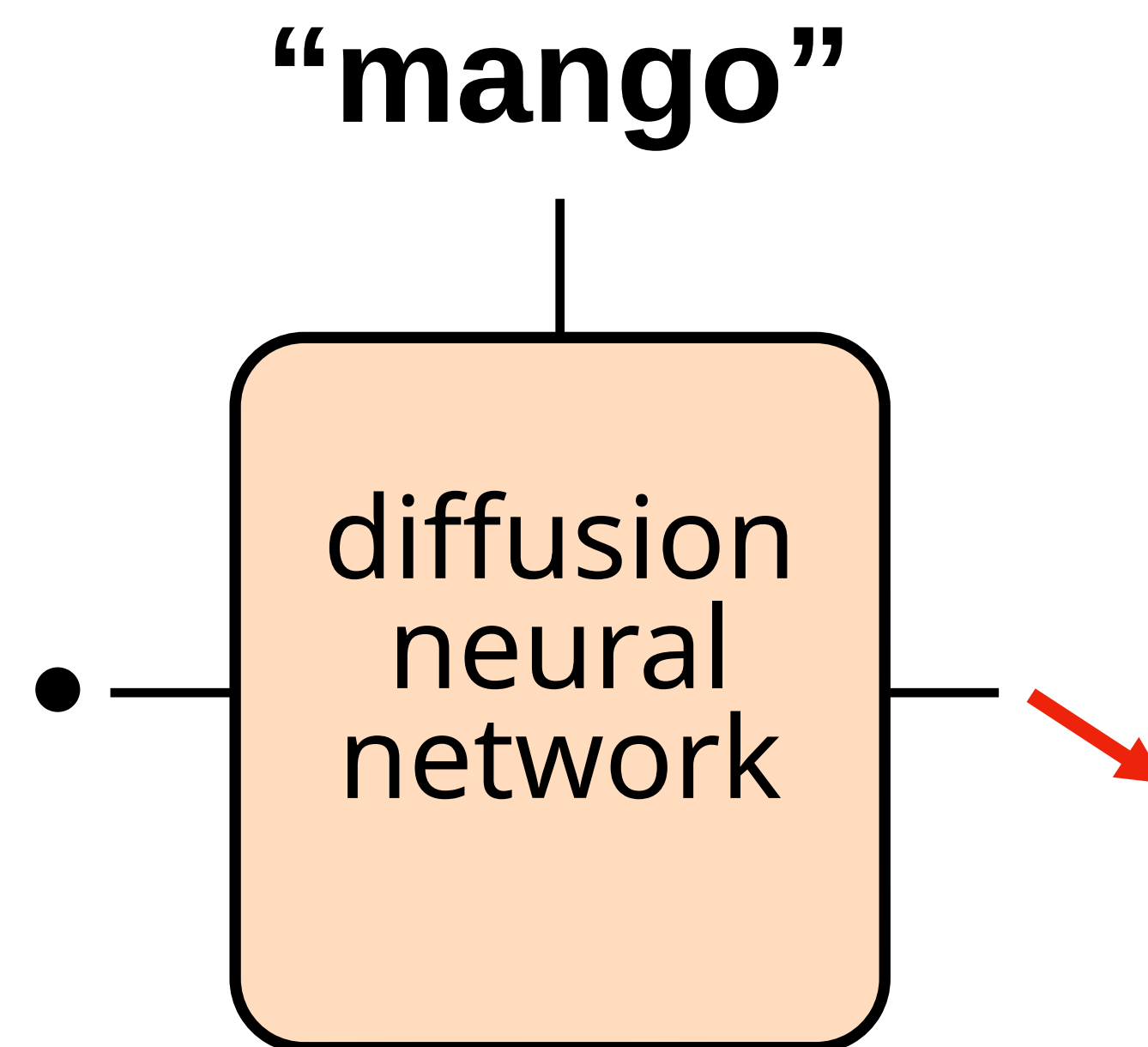
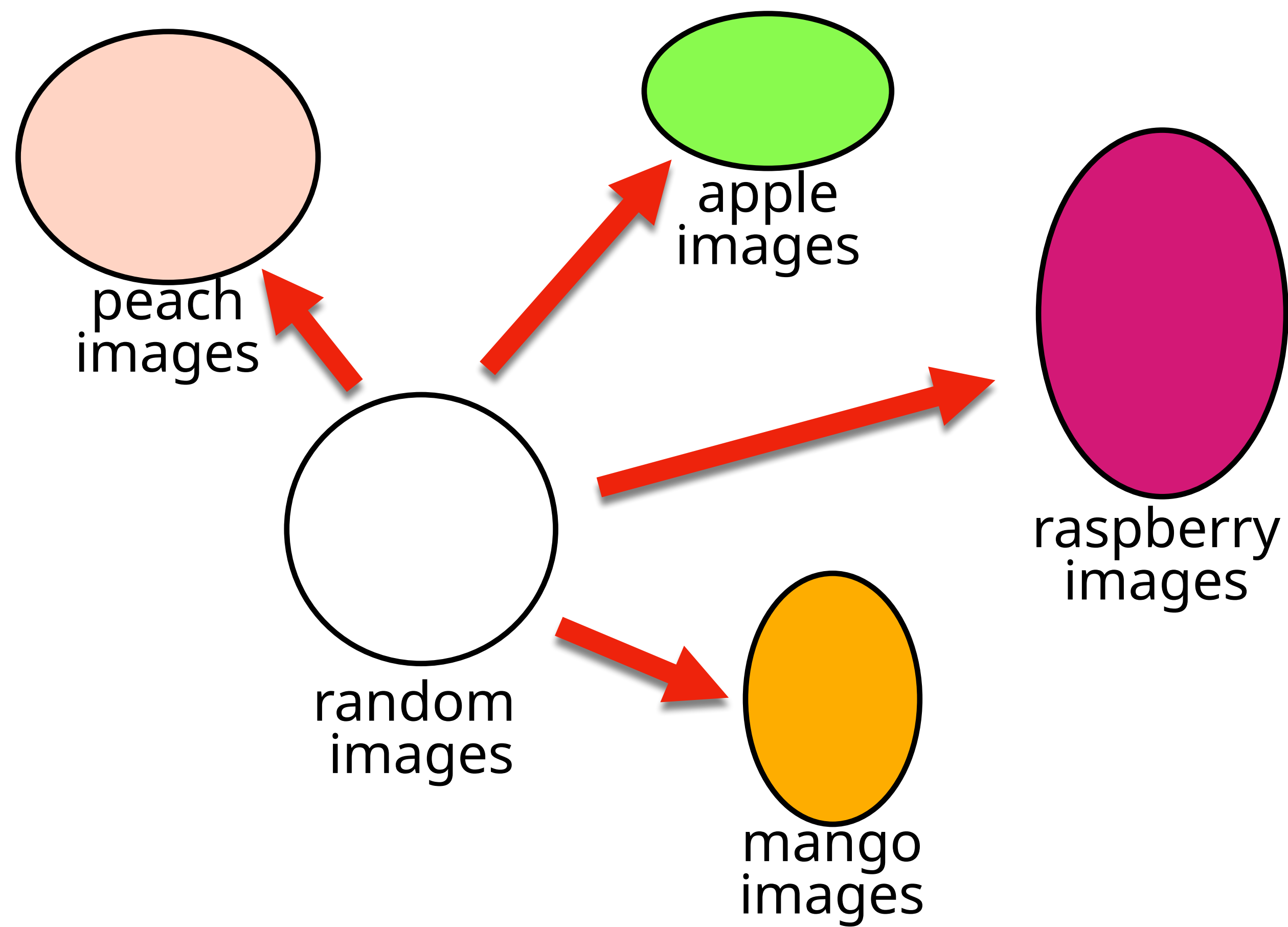
Training
slide from Steve Seitz's [video](#)



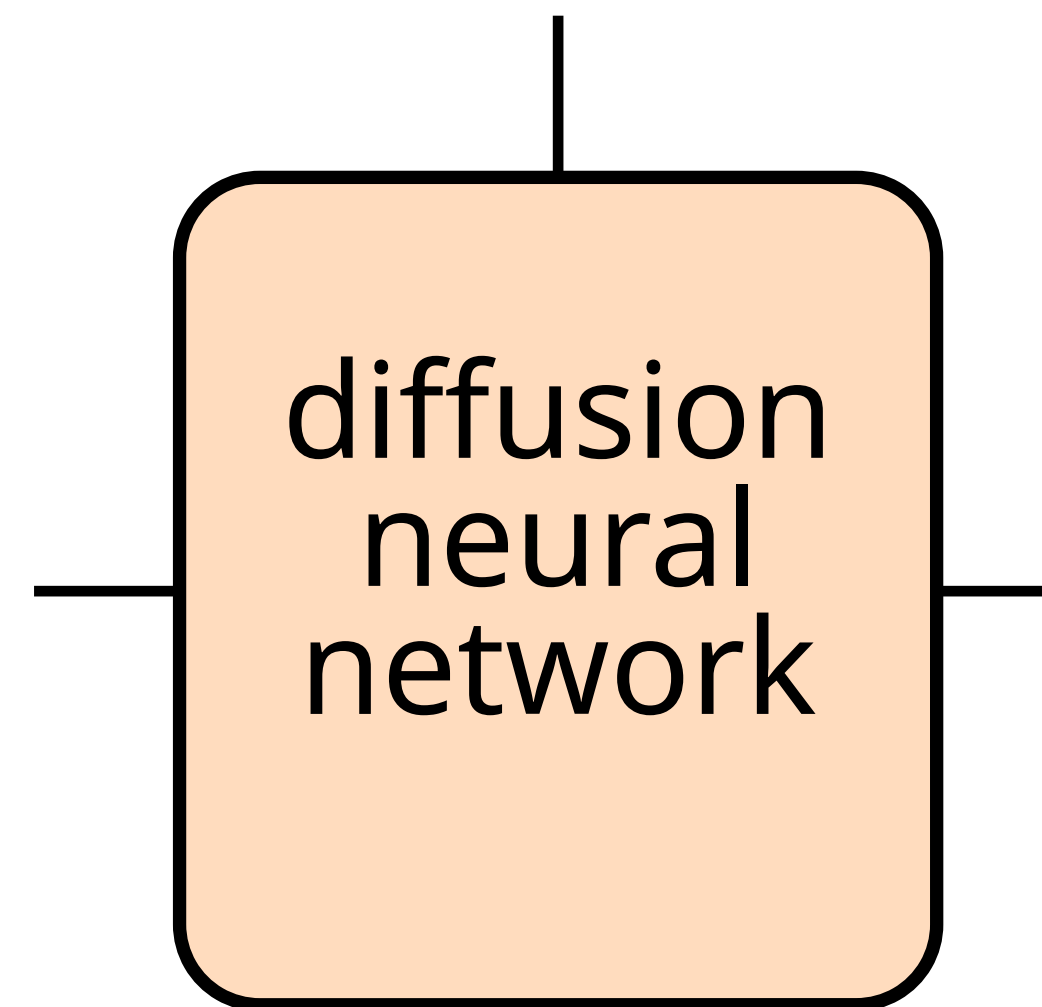


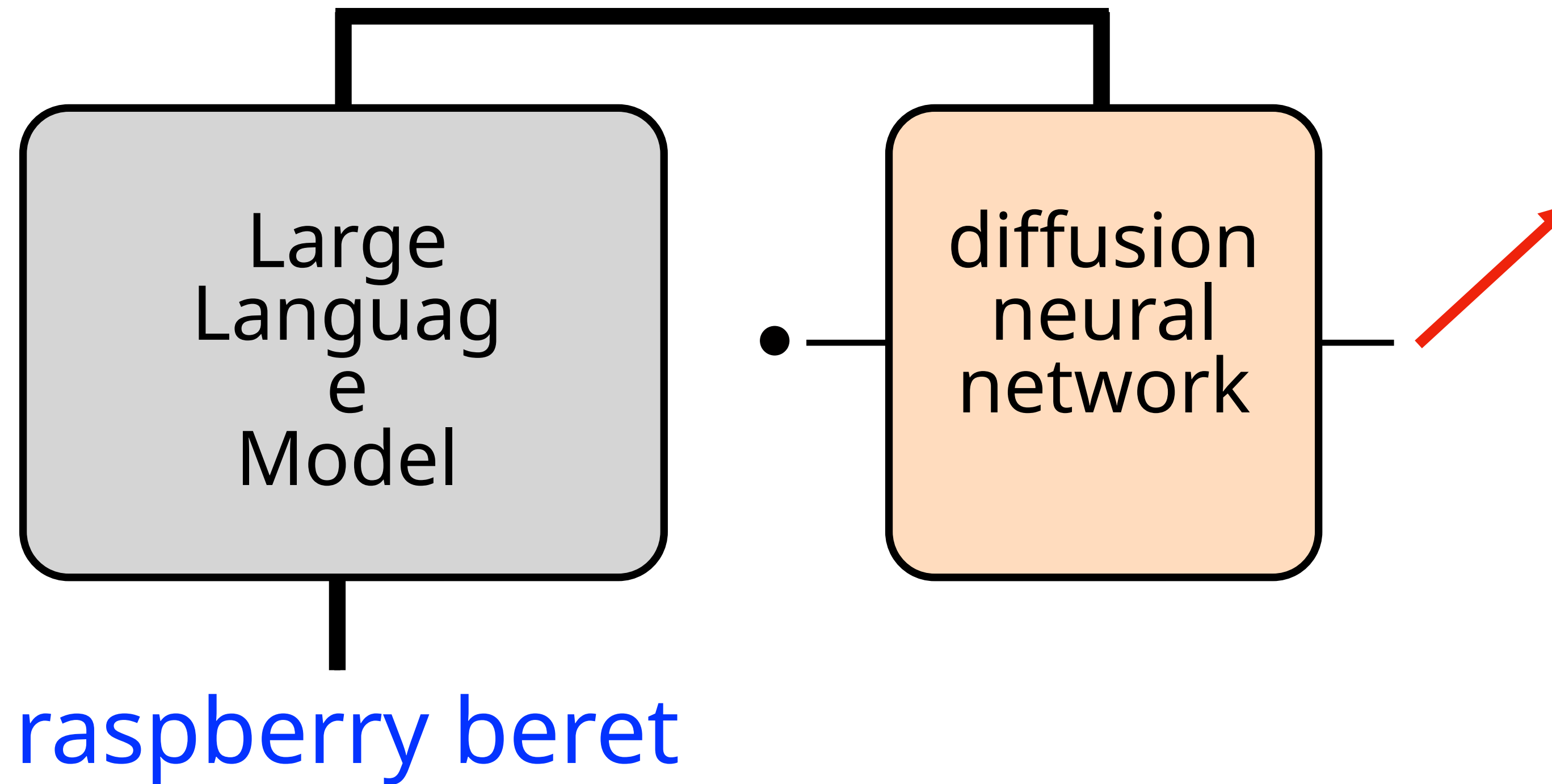






raspberry beret







raspberry beret



Imagen



Imagen

raspberry beret

slide from Steve Seitz's [video](#)

beret of raspberries



beret of raspberries

slide from Steve Seitz's [video](#)



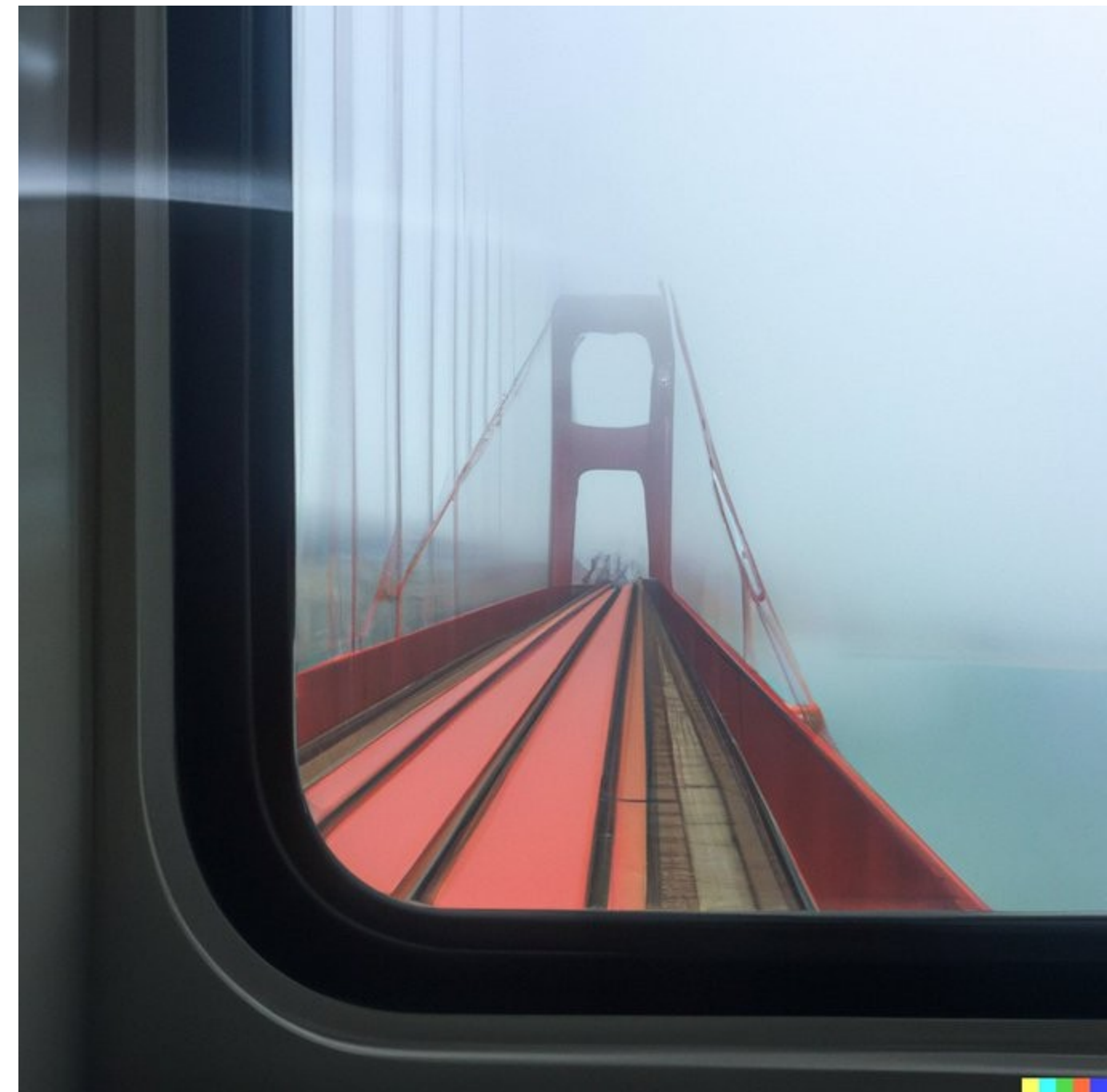
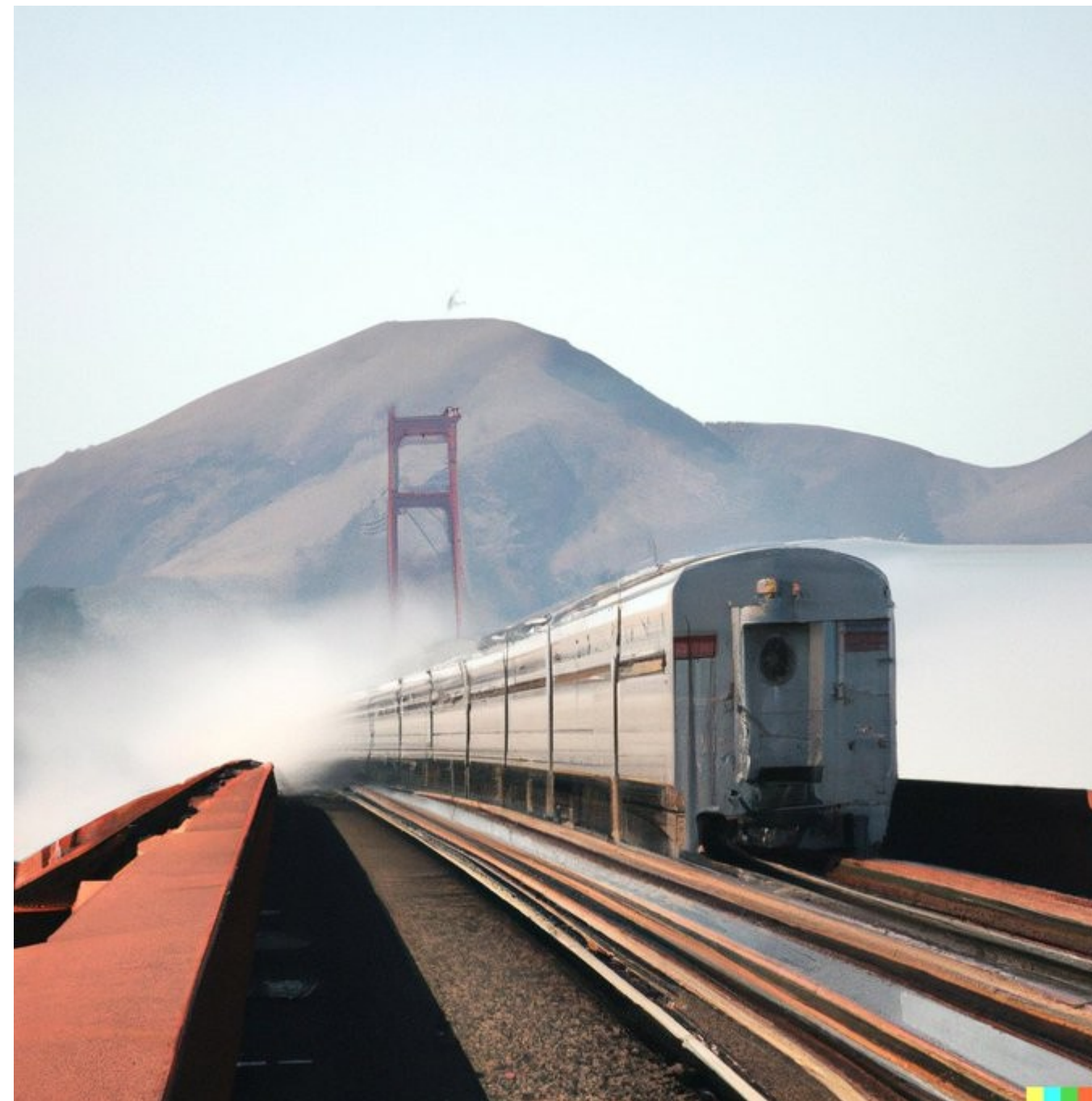
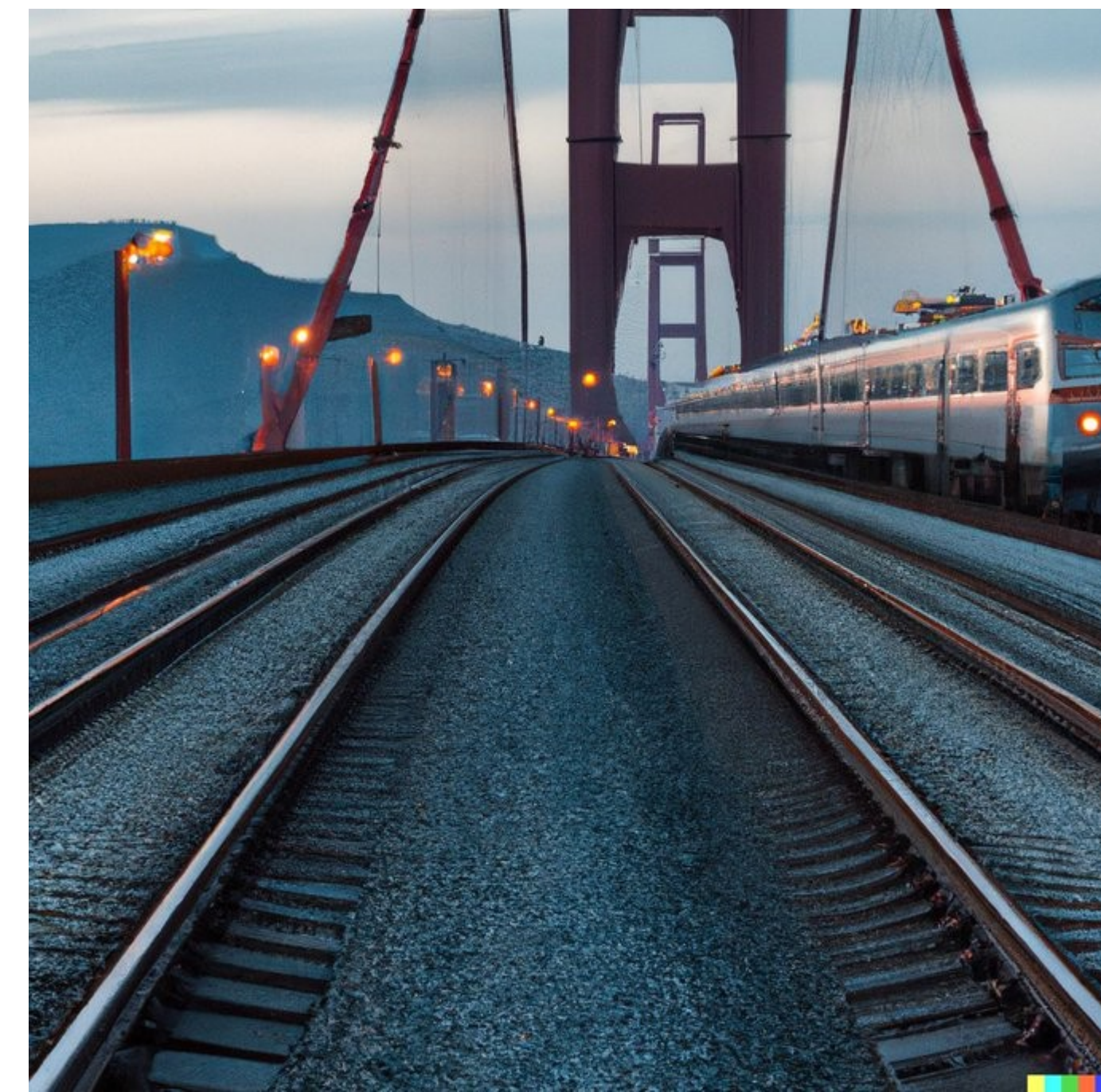
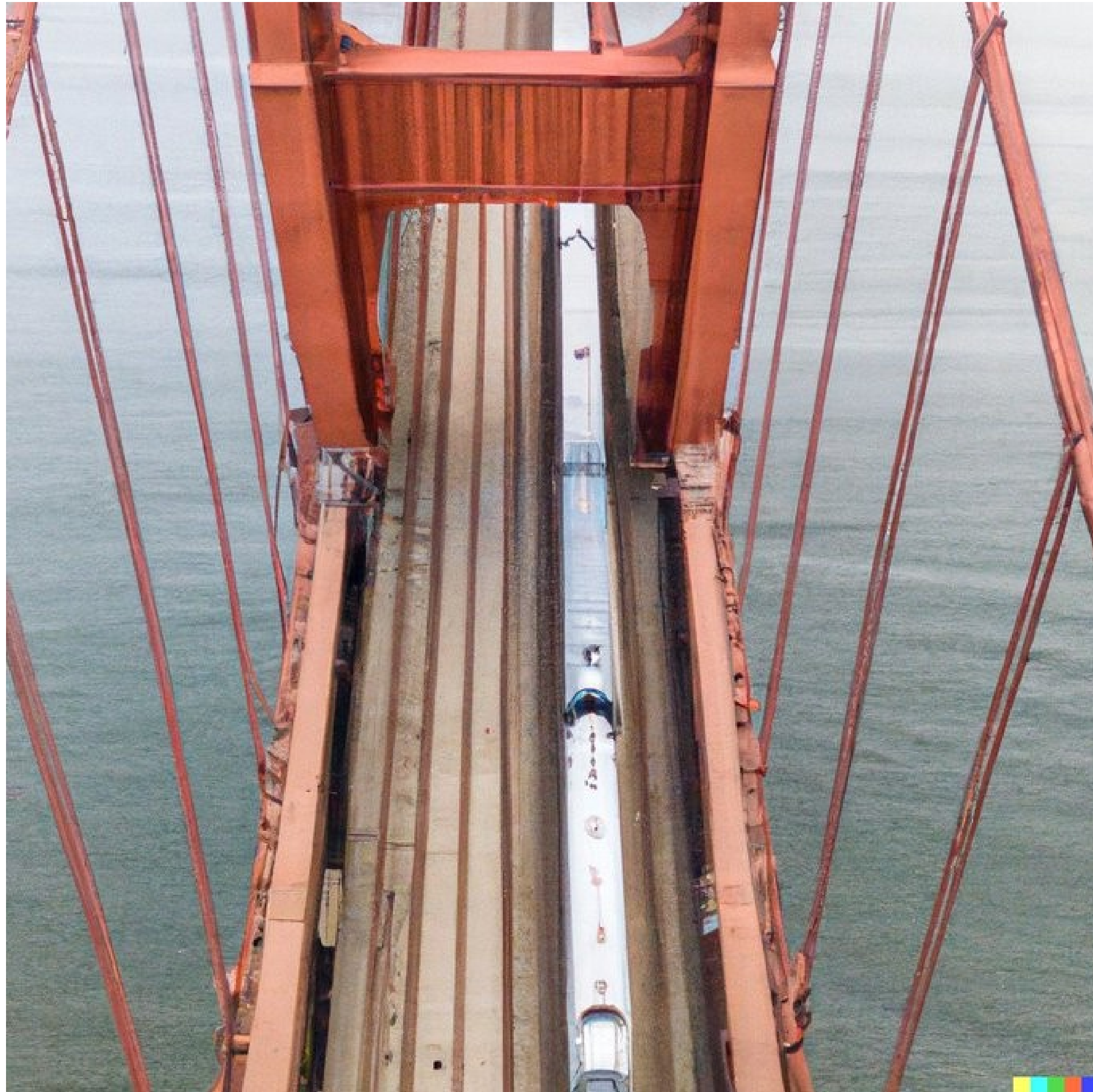
beret of raspberries

slide from Steve Seitz's [video](#)



An astronaut riding a horse in a photorealistic style (Dall-E
slide from Steve Seitz's [video](#)

Impressive
compositionality:



DALL-E + Danielle Baskin

Project 3 out!



Can generative models make **Multi-View Optical Illusions?**

Salvador Dalí,
Paranoiac Face. 1937.

Illusions?



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



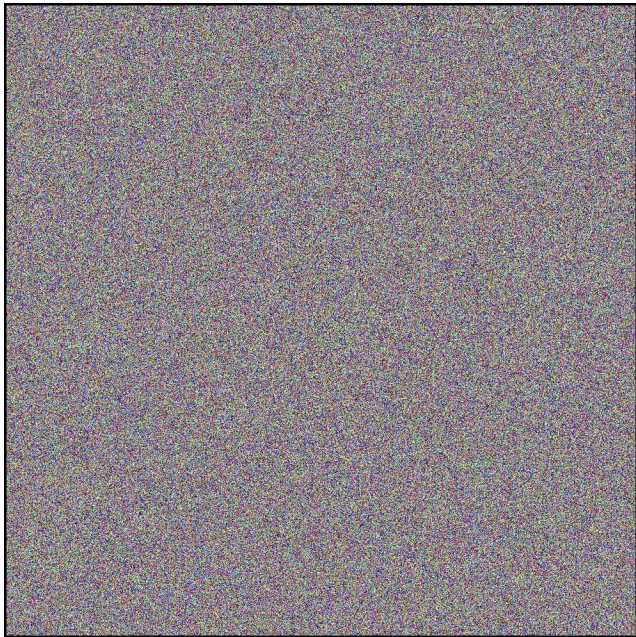
a drawing of a giraffe



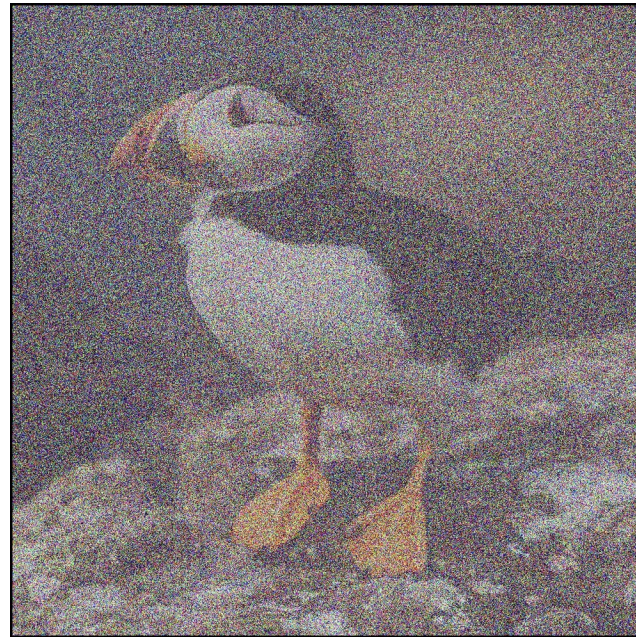
an oil painting of a
snowy mountain village

Diffusion Models

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

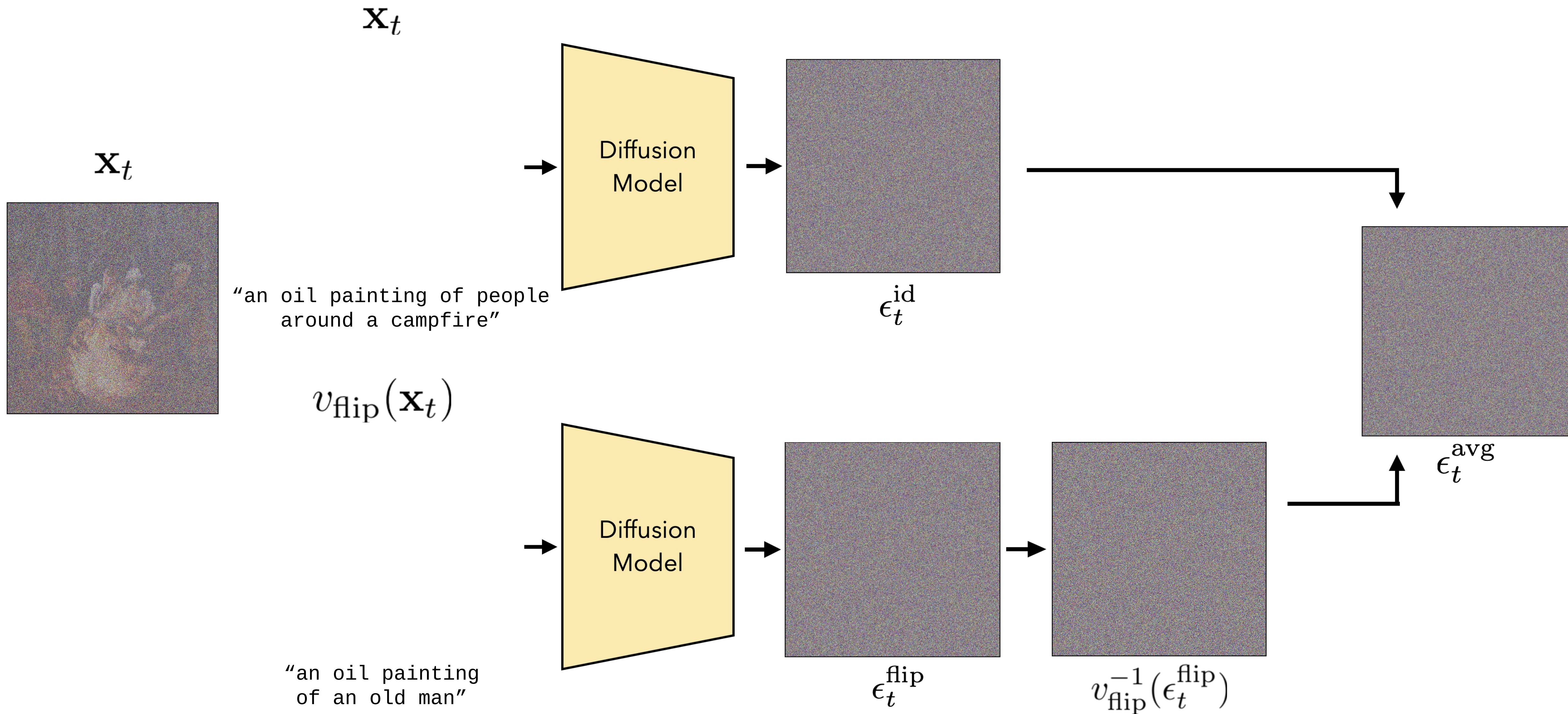


“a photo of
a puffin”



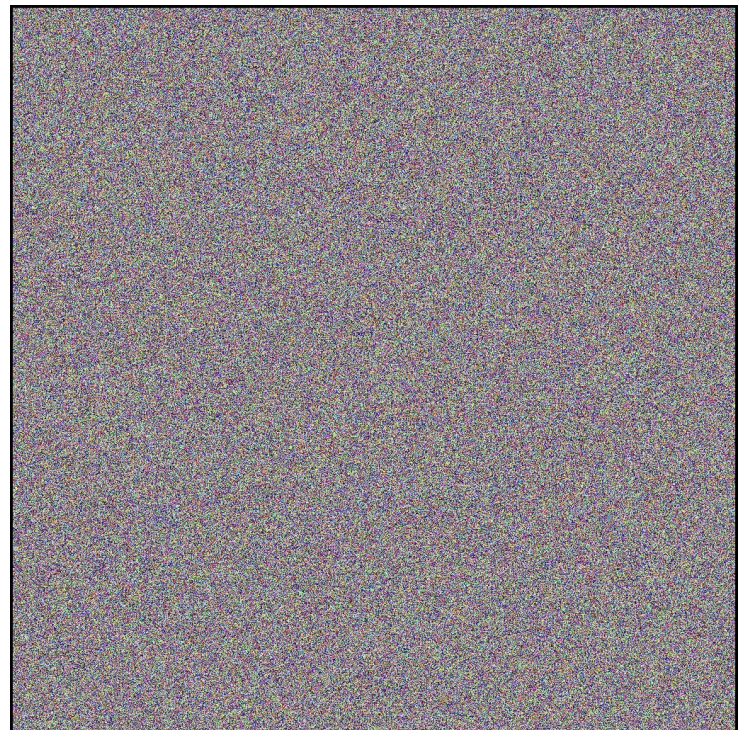
$$\mathbf{x}_0$$

Method



Method

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



$$\mathbf{x}_0$$