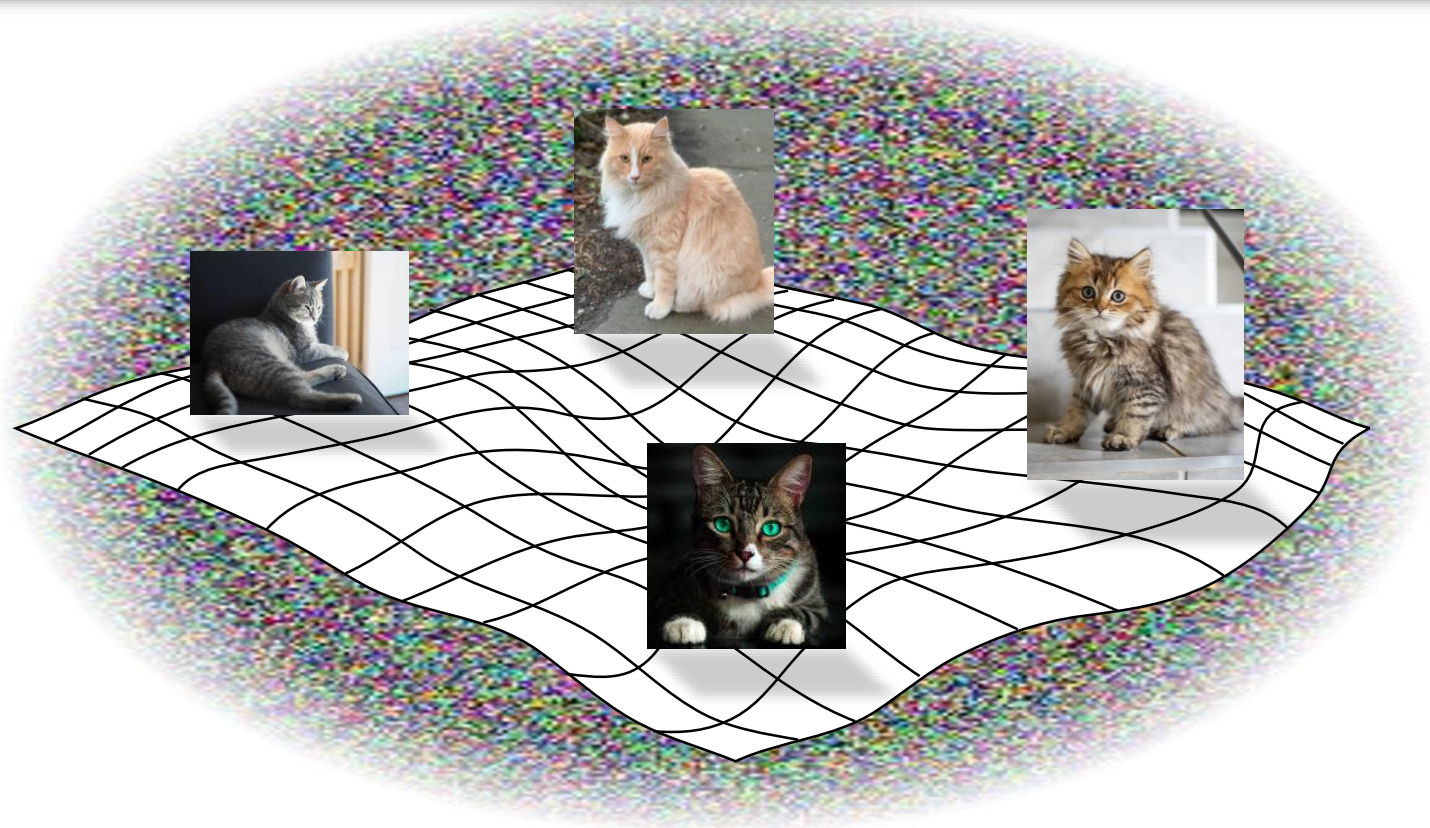


Image Manifolds & Image Generation



Agenda

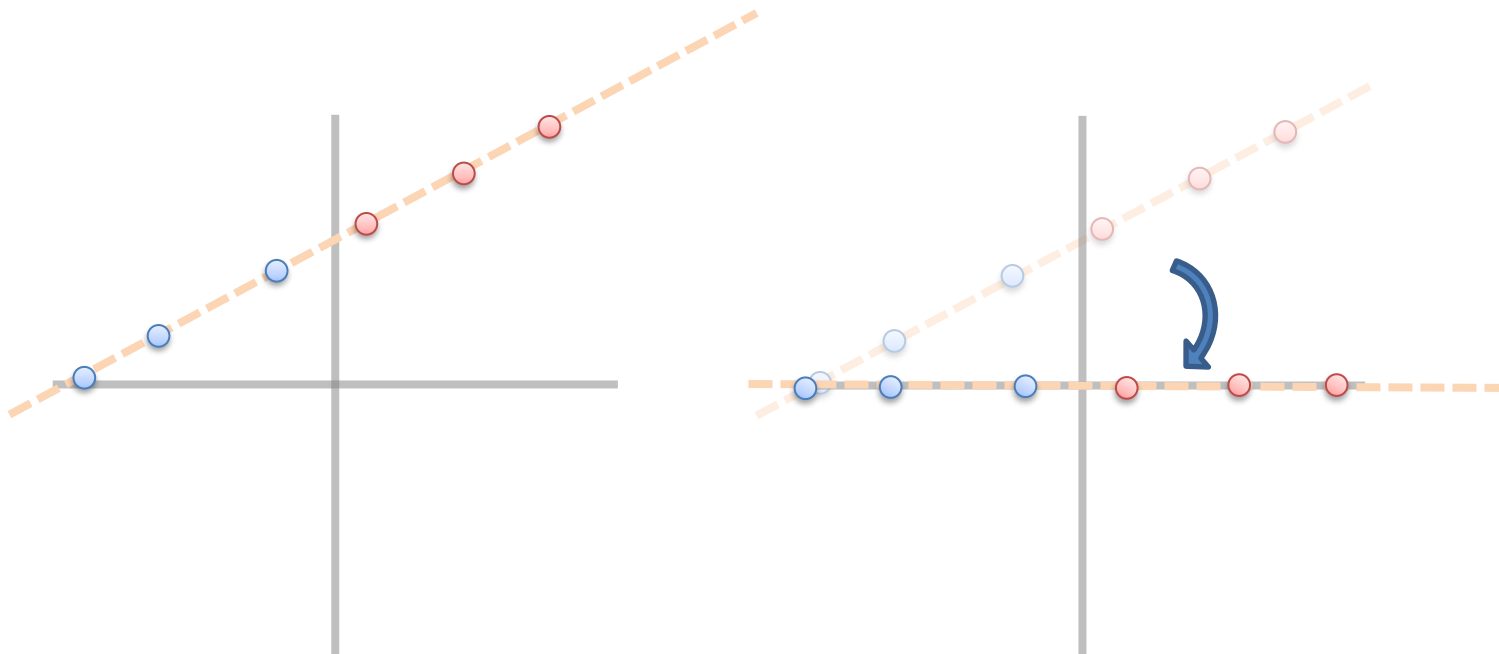
- The manifold of natural images
- Image-to-image methods and GANs
- Image synthesis methods
- diffusion models

By Abe Davis

DIMENSIONALITY REDUCTION

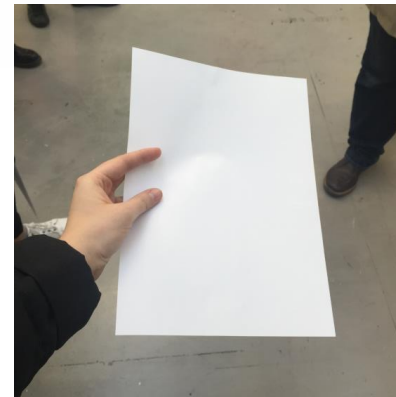
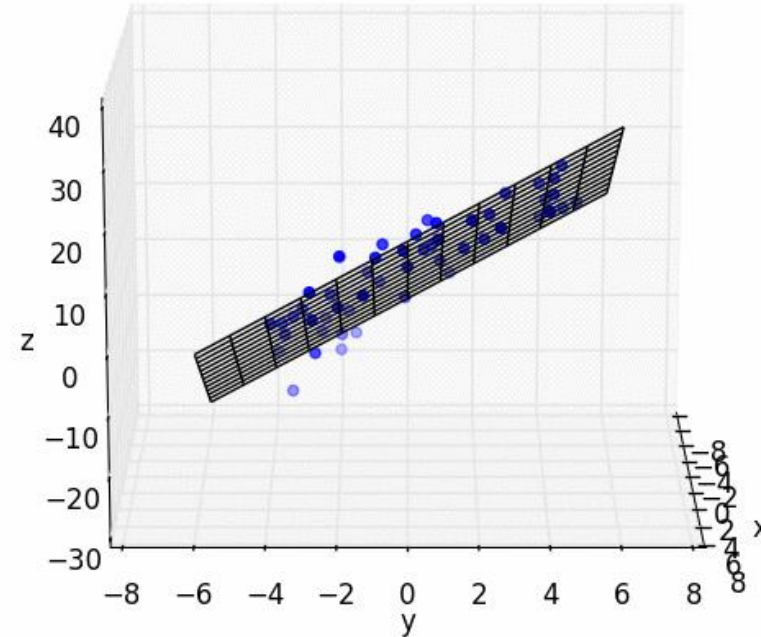
Linear Dimensionality Reduction: 2D->1D

- Consider a bunch of data points in 2D
- Let's say these points lie along a line
- If so, we can translate and rotate our data so that it is 1D



Linear Dimensionality Reduction: 3D->2D

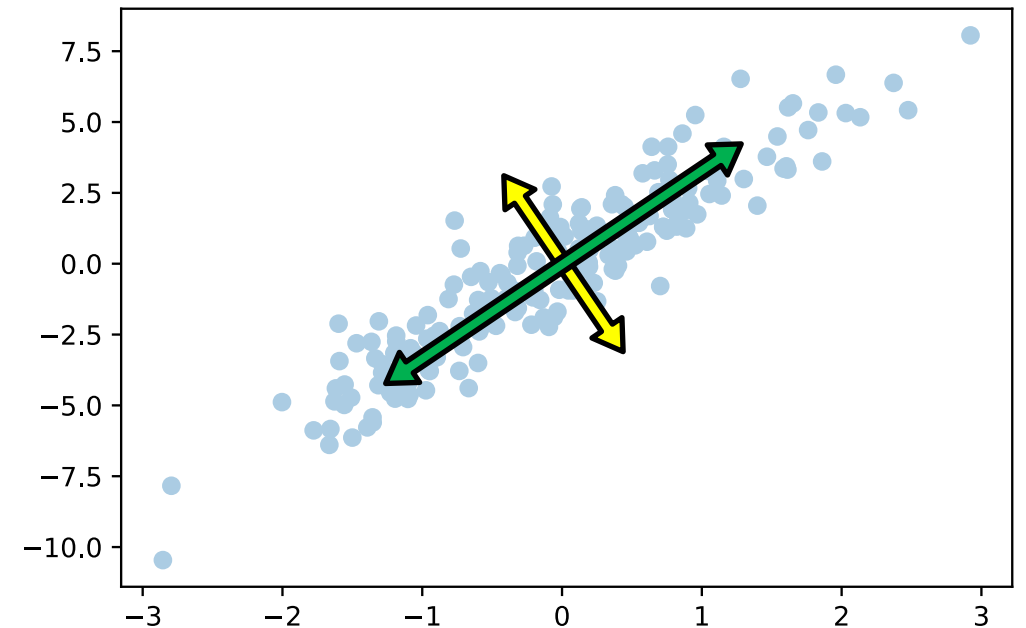
- Similar to 1D case, we can fit a plane to the data, and transform our coordinate system so that plane becomes the x-y plane
- “Plane fitting”
- Now we only need to store two numbers for each point (and the plane parameters)
- More generally: look for the 2D subspace that best fits the data, and ignore the remaining dimensions



Think of this as data that sits on a flat sheet of paper, suspended in 3D space. We will come back to this analogy in a couple slides...

Generalizing Linear Dimensionality Reduction

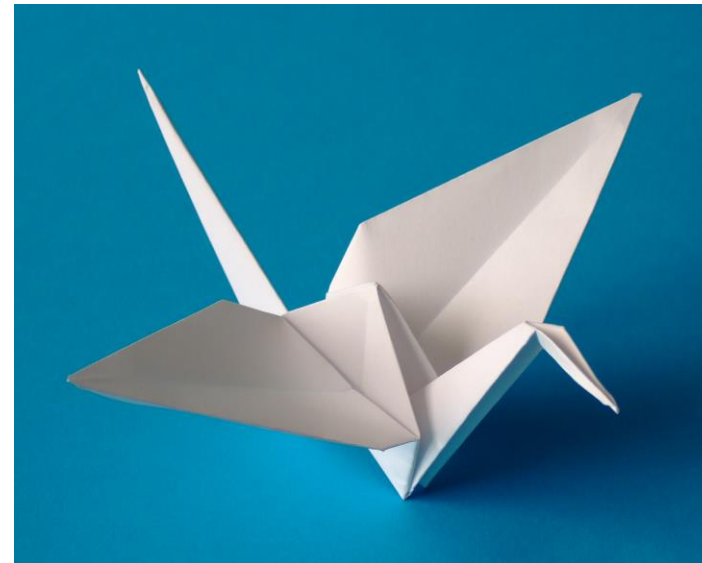
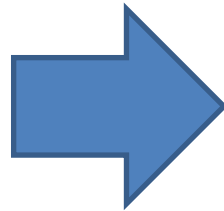
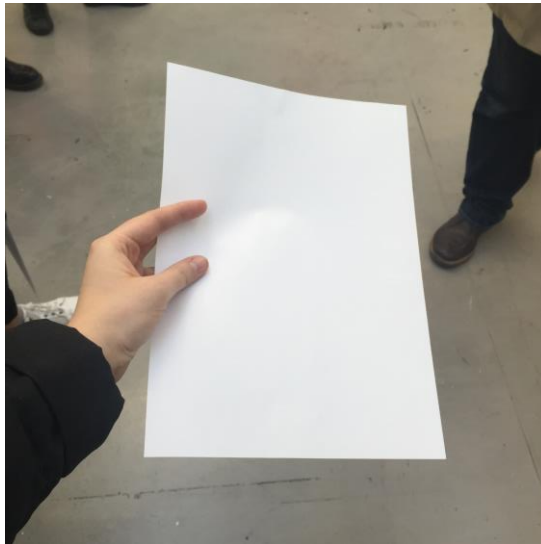
- **Principal Components Analysis (PCA)**: find and order orthogonal axes by how much the data varies along each axis.
- The axes we find (ordered by variance of our data) are called **principal components**.
- Dimensionality reduction can be done by using only the first k principal components



Side Note: principal components are closely related to the eigenvectors of the covariance matrix for our data

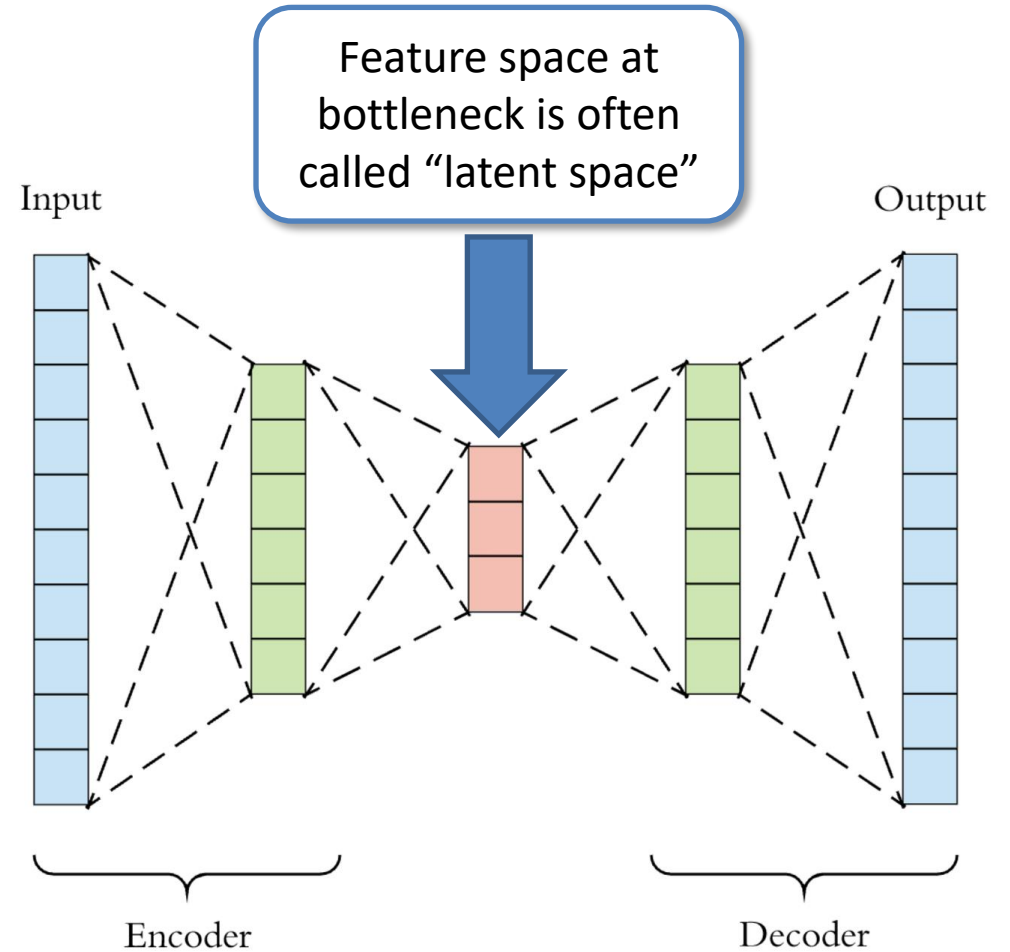
Manifolds

- Think of a piece of paper as a 2D subspace
- If we bend & fold it, it's still locally a 2D subspace...
- A "manifold" is the generalization of this concept to higher dimensions...



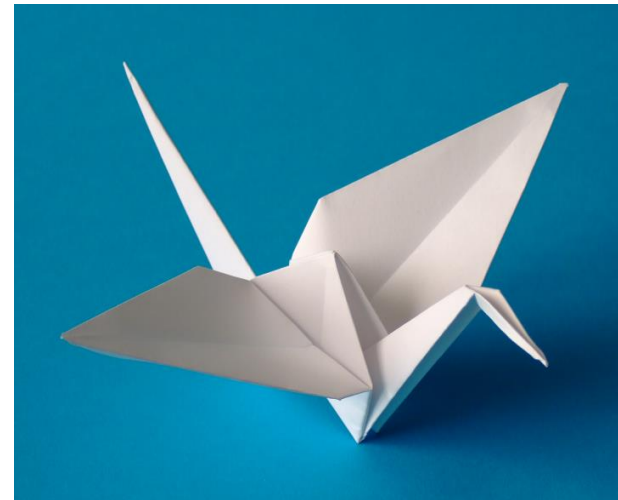
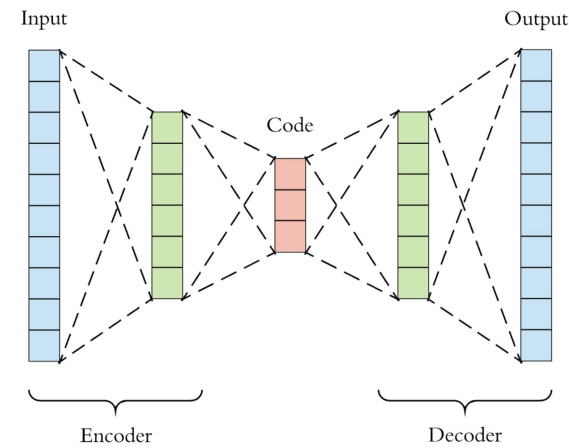
Autoencoders: Dimensionality Reduction for Manifolds

- Learn a non-linear (deep network) transformation into some lower-dimensional space (encoder)
- Learn a transformation from lower-dimensional space back to original content (decoder)
- Loss function measures difference between input & output
- **Self Supervised!**
 - No labels required! Signal is just from learning to compress data



Autoencoders: Dimensionality Reduction for Manifolds

- Transformations that reduce dimensionality **cannot be invertible** in general
- An autoencoder tries to learn a transformation that is **invertible for points on some manifold**



By Abe Davis

IMAGE MANIFOLDS

The Space of All Images

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



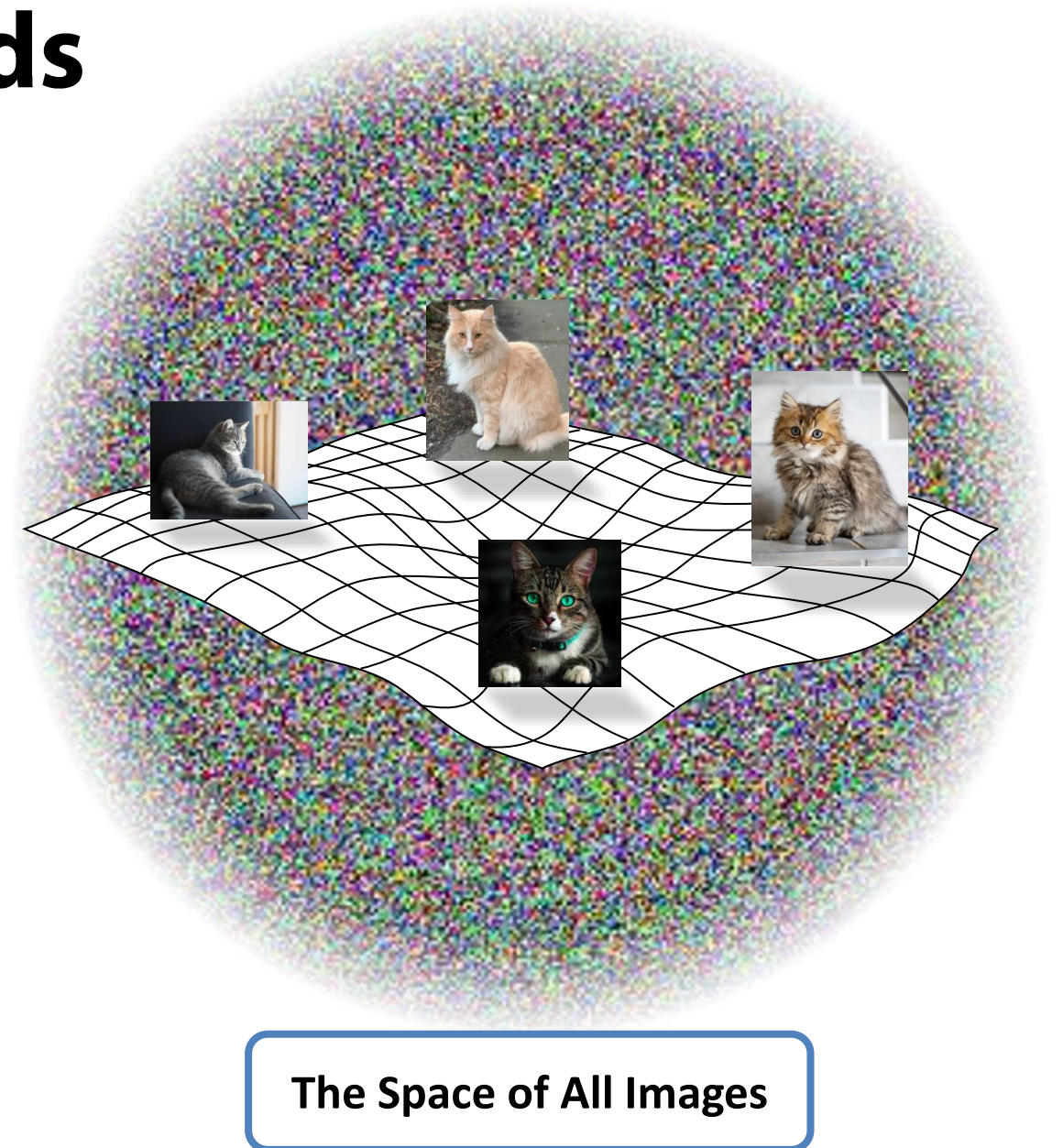
Question:

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

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

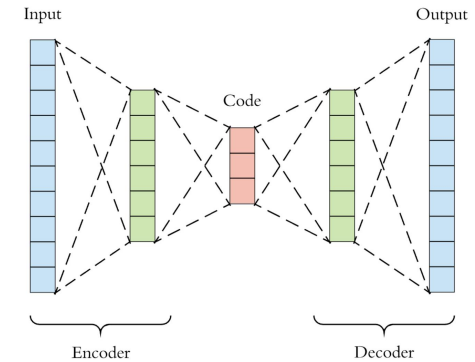
Natural Image Manifolds

- Most images are “noise”
- “Meaningful” images tend to form some manifold within the space of all images
- Images of a particular class fall on manifolds within that manifold...



Denoising & the “Nullspace” of Autoencoders

- The autoencoder tries to learn a dimensionality reduction that is invertible for our data (data on some manifold)
- Most noise will be in the non-invertible part of image space (off the manifold)
- If we feed noisy data in, we will often get denoised data out



Input



Output



Noisy Input

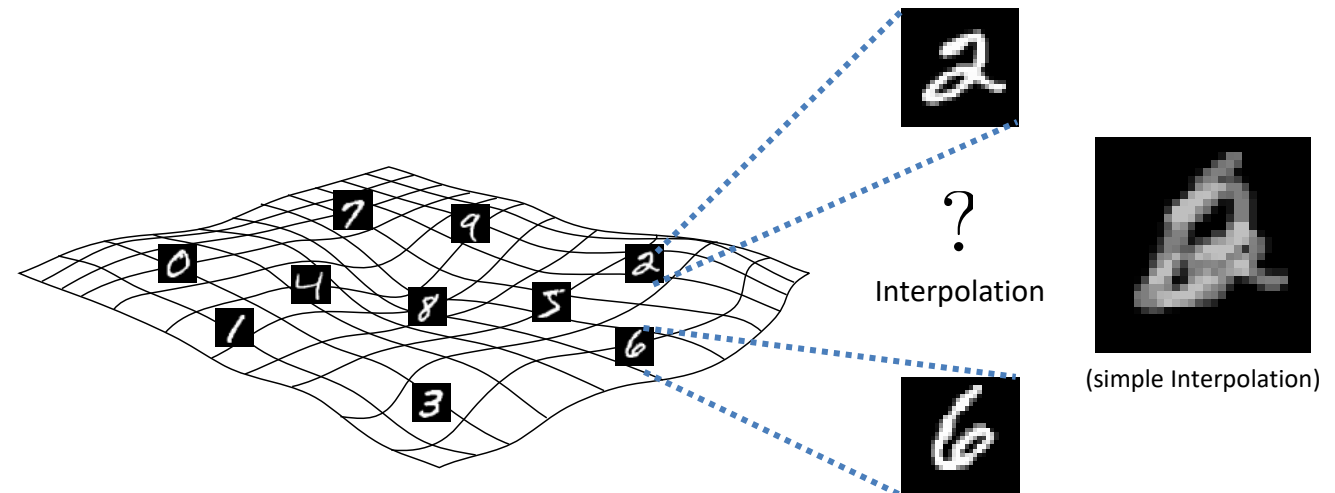
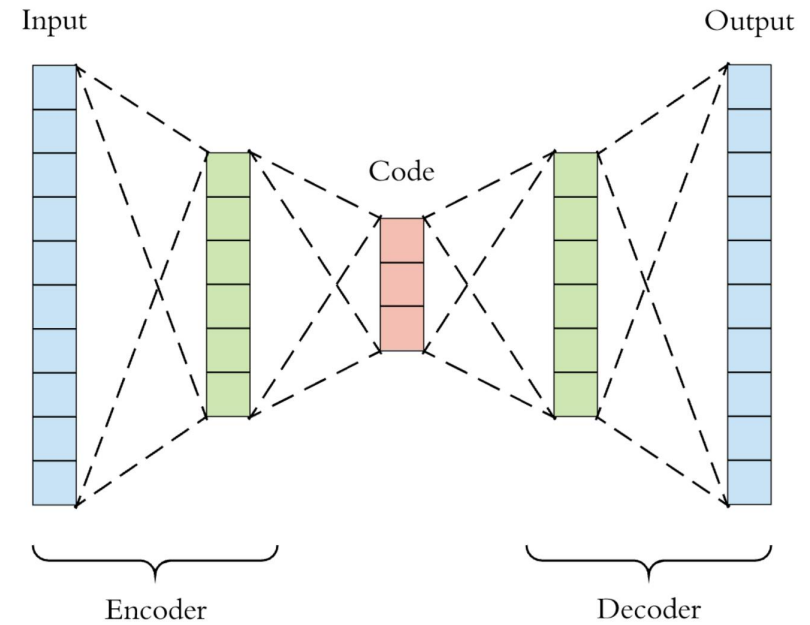


Output



Problem

- Autoencoders can compress because data sits on a manifold
- This doesn't mean that every point in the latent space will be on the manifold...
- GANs (later this lecture) will learn a loss function that helps with this...



Abe Davis, with slides from Jin Sun, Phillip Isola, and Richard Zhang

IMAGE-TO-IMAGE APPLICATIONS

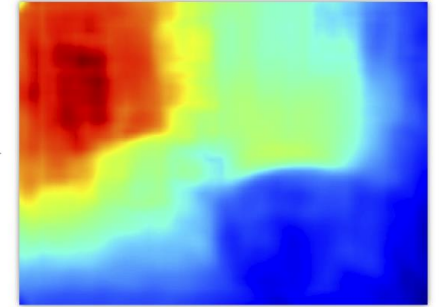
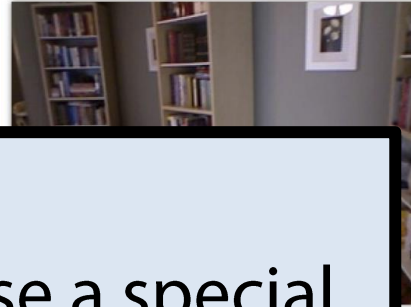
Image prediction (“structured prediction”)

Object labeling



[Long et al. 2015]

Depth prediction



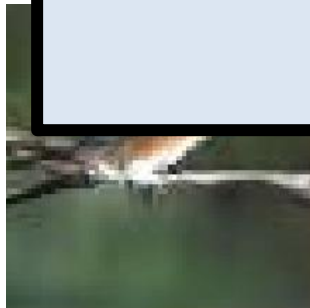
Depth Map

[Eigen et al. 2014, ...]

Recall: we often use a special architecture like a U-Net for such image-to-image mappings

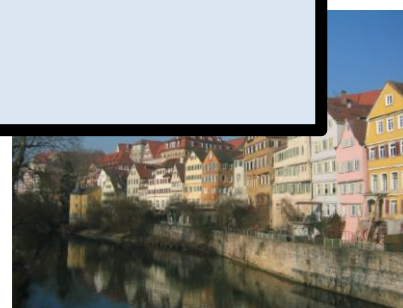
Text-to-photo

“this small bird has a pink breast and crown...”



[Reed et al. 2016, ...]

Style transfer



[Gatys et al. 2016, ...]

x

y

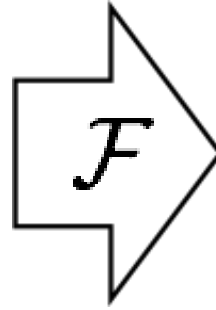
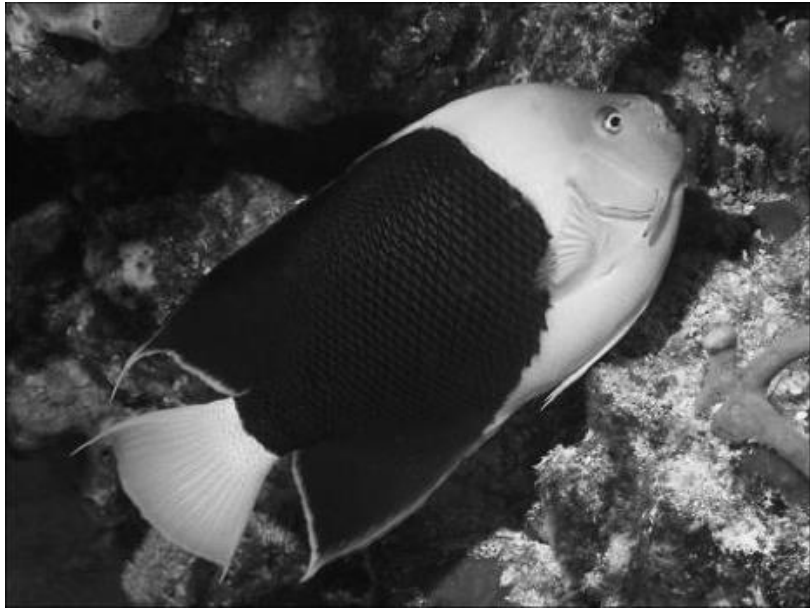
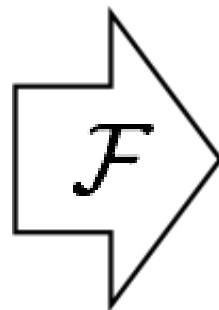


Image Colorization

x

y



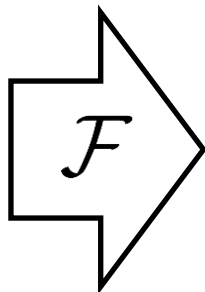
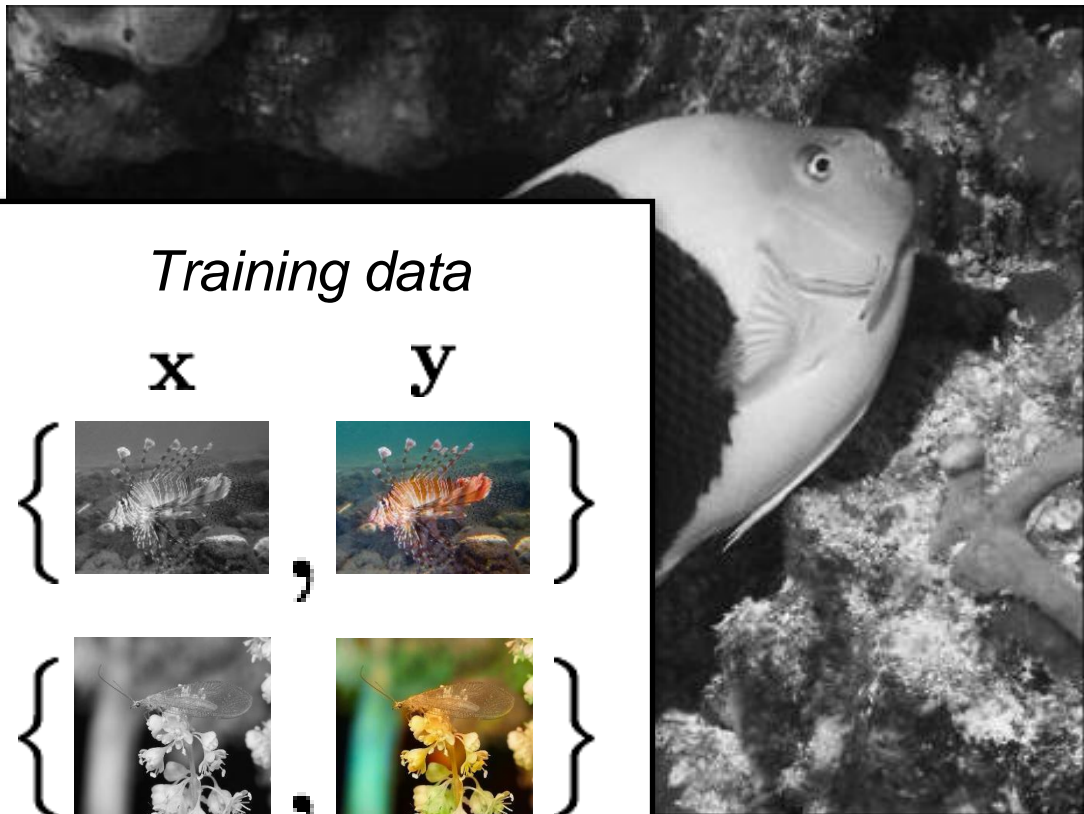
$$\arg \min_{\mathcal{F}} \mathbb{E}_{\mathbf{x}, \mathbf{y}} [L(\mathcal{F}(\mathbf{x}), \mathbf{y})]$$

“**What** should I do”

“**How** should I do it?”

x

y



Training data

x	y
{  ,  }	
{  ,  }	
{  ,  }	
⋮	

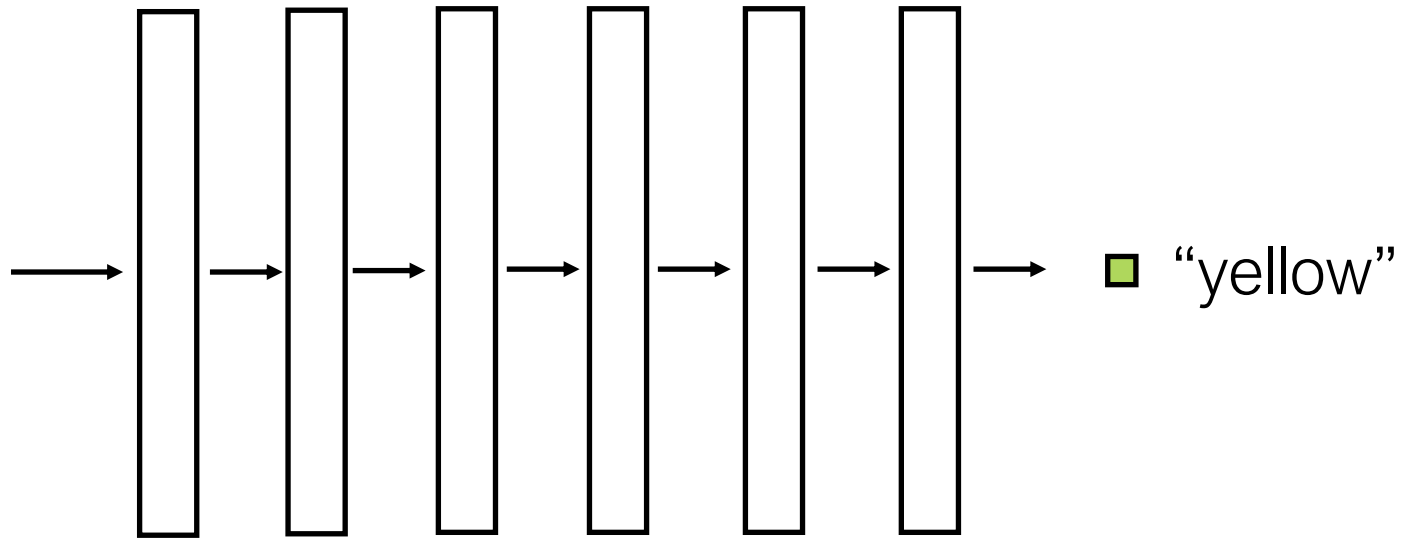
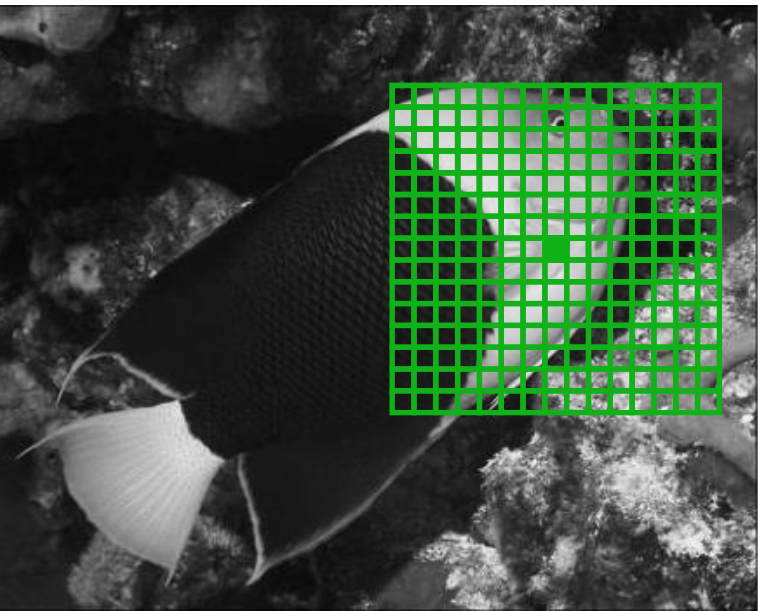
channel

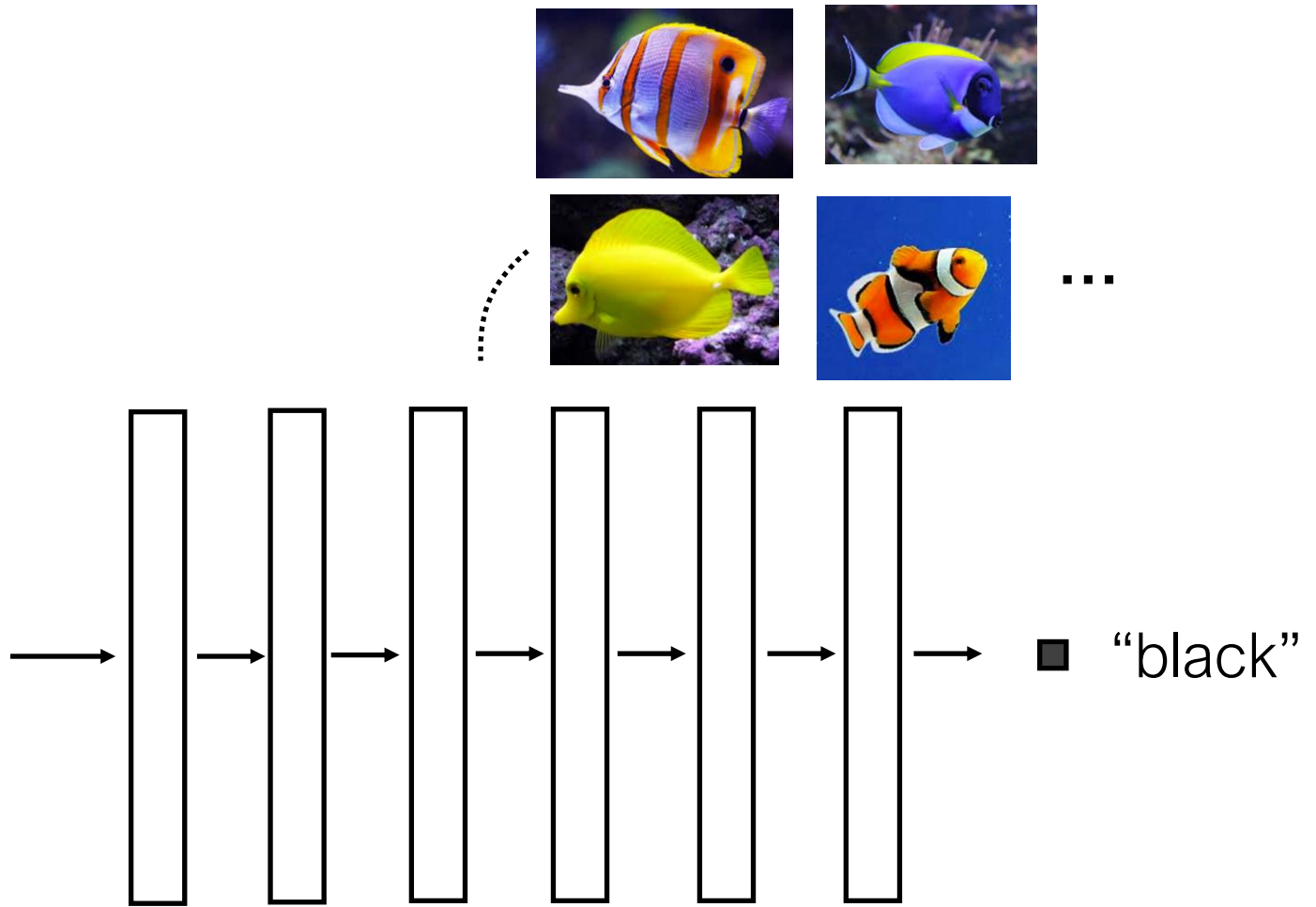
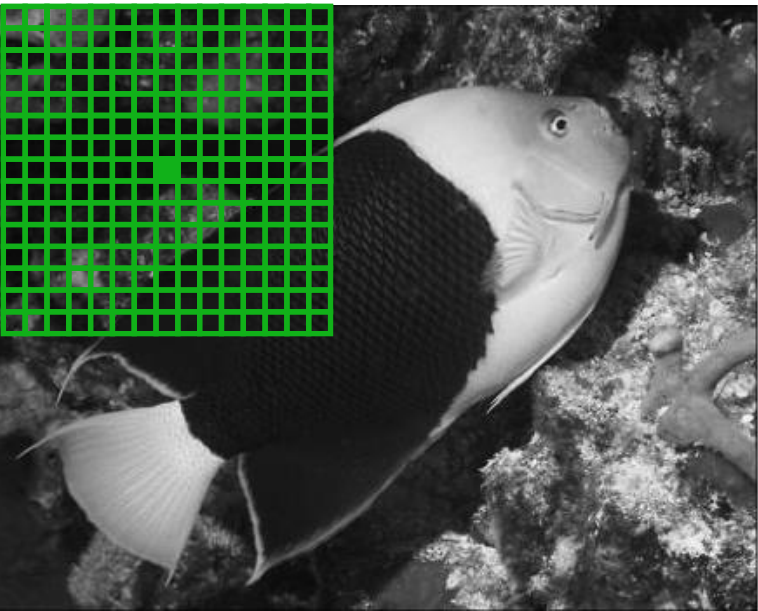
Color information: ab channels

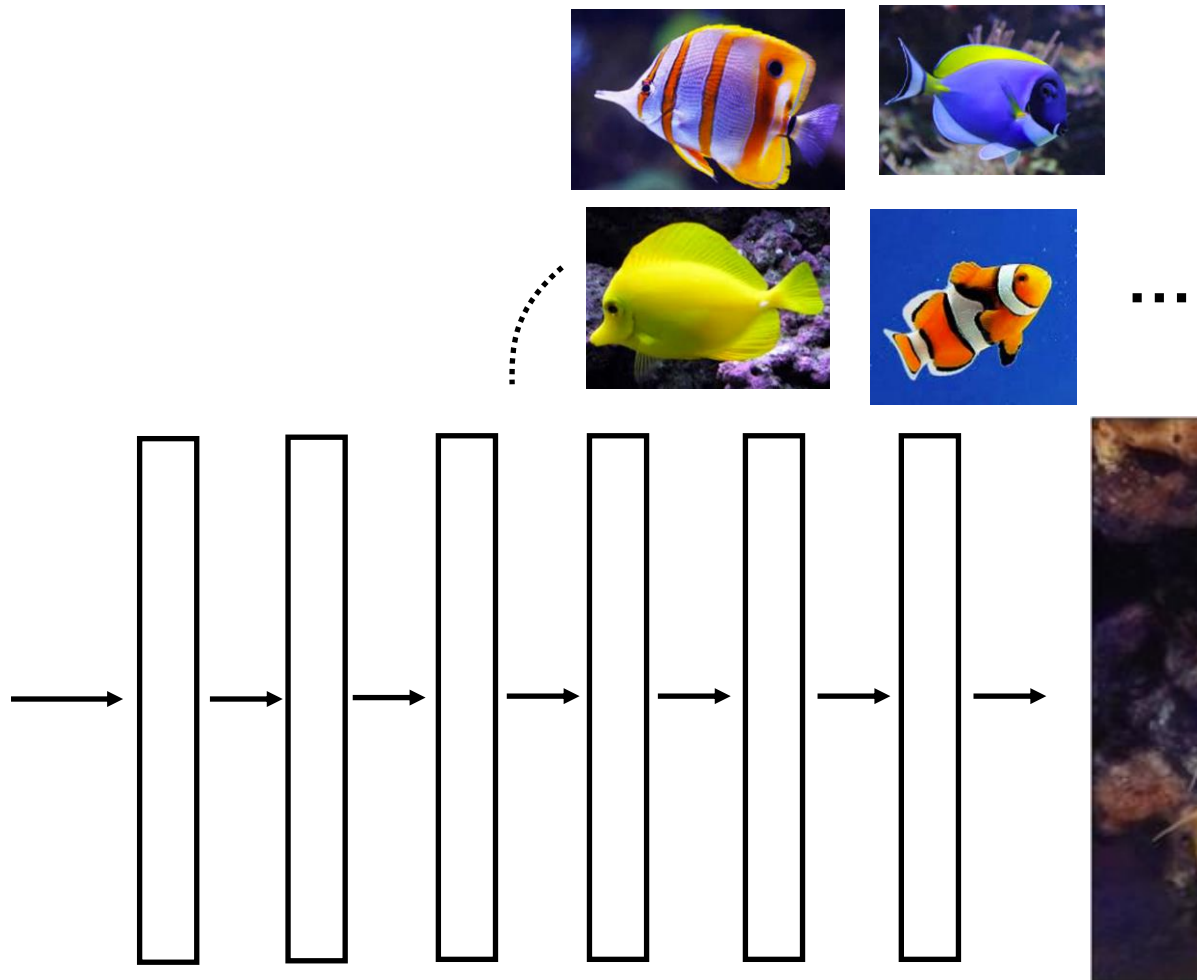
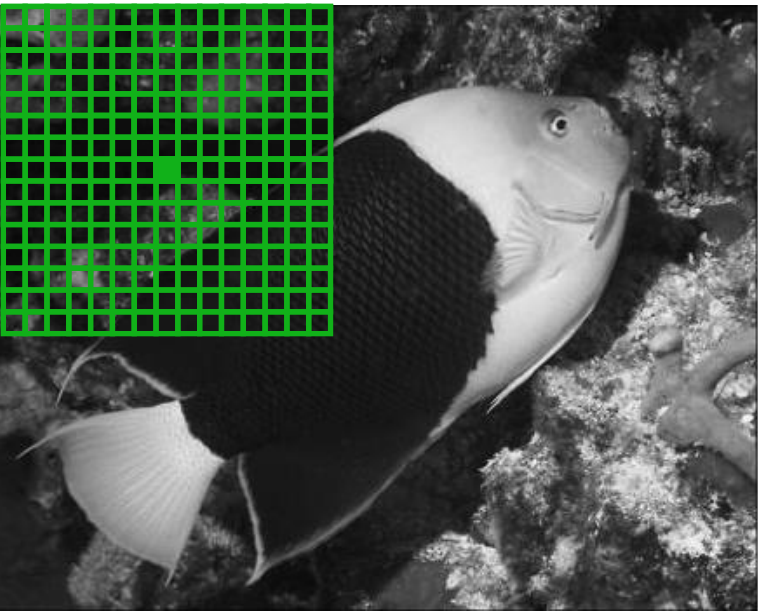
$$\arg \min_{\mathcal{F}} \mathbb{E}_{\mathbf{x}, \mathbf{y}} [L(\mathcal{F}(\mathbf{x}), \mathbf{y})]$$

Objective function
(loss)

Neural Network







from Jin Sun, Richard Zhang, Phillip Isola

Recap: basic loss functions

Prediction: $\hat{\mathbf{y}} = \mathcal{F}(\mathbf{x})$

Truth: \mathbf{y}

Classification (cross-entropy):

$$L(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_i \hat{\mathbf{y}}_i \log \mathbf{y}_i \quad \leftarrow$$

How many extra bits it takes to correct the predictions

Recap: basic loss functions

Prediction: $\hat{\mathbf{y}} = \mathcal{F}(\mathbf{x})$

Truth: \mathbf{y}

Classification (cross-entropy):

$$L(\hat{\mathbf{y}}, \mathbf{y}) = - \sum_i \hat{\mathbf{y}}_i \log \mathbf{y}_i \quad \leftarrow$$

How many extra bits it takes to correct the predictions

Least-squares regression:

$$L(\hat{\mathbf{y}}, \mathbf{y}) = \|\hat{\mathbf{y}} - \mathbf{y}\|_2 \quad \leftarrow$$

How far off we are in Euclidean distance

Designing loss functions

Input



Output (with L2 loss)



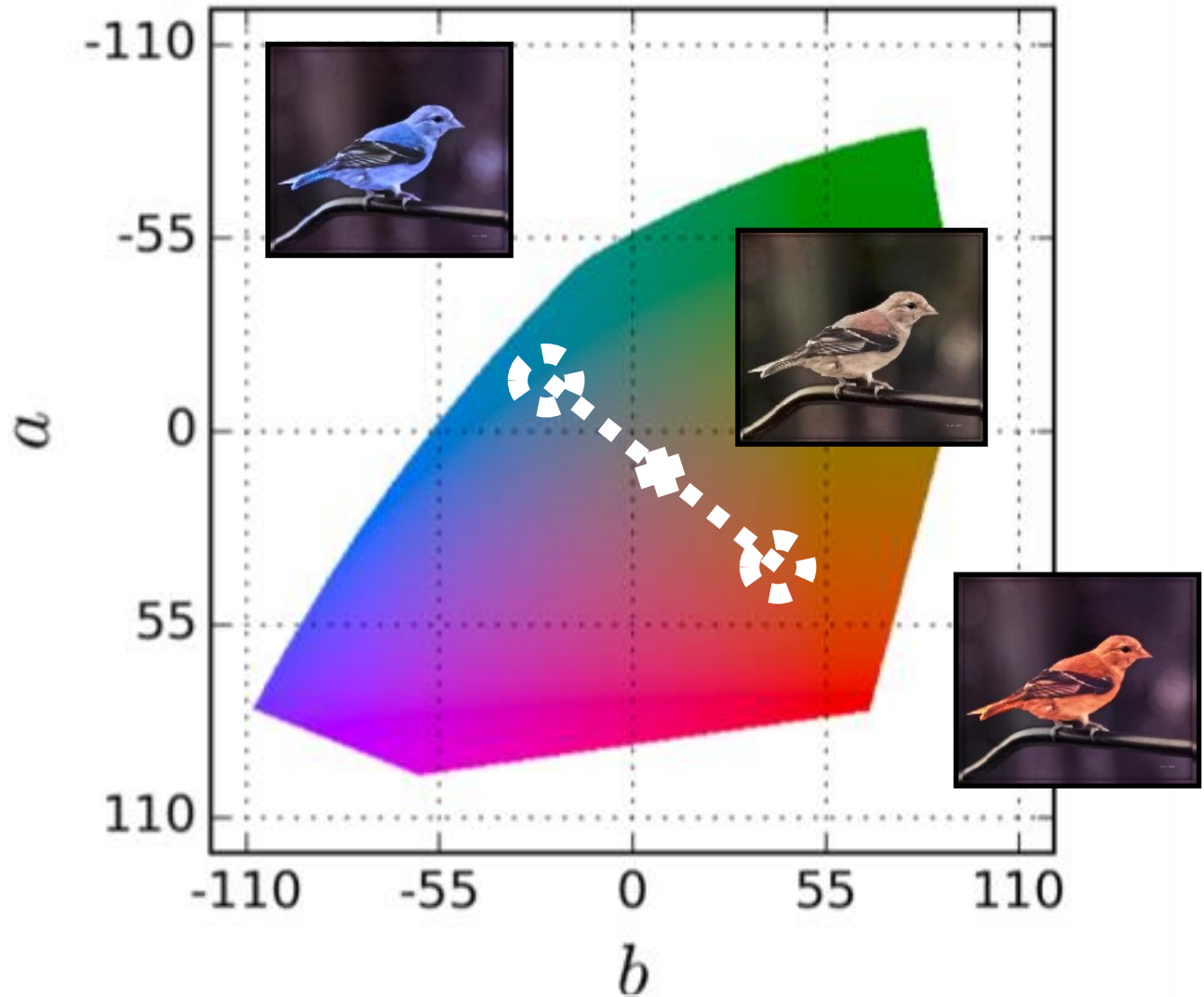
Ground truth



$$\mathbf{L}_2(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2} \sum_{h,w} \|\mathbf{Y}_{h,w} - \hat{\mathbf{Y}}_{h,w}\|_2^2 \quad (\text{L2 loss})$$



With L2 loss, predictions
“regress to the mean”,
and lack vivid colors



$$L_2(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2} \sum_{h,w} \|\mathbf{Y}_{h,w} - \hat{\mathbf{Y}}_{h,w}\|_2^2$$

Designing loss functions

Input



Zhang et al. 2016



Ground truth

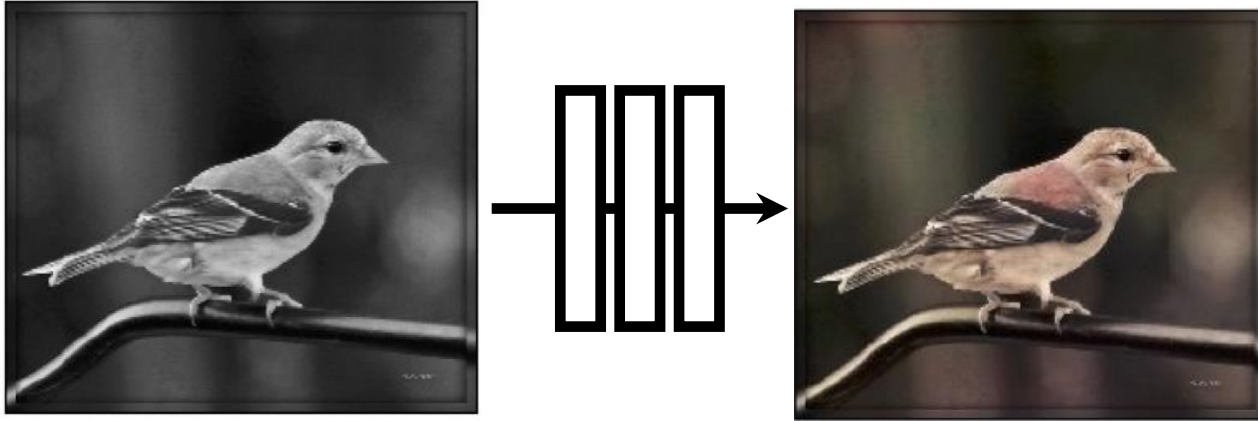


Color distribution cross-entropy loss with colorfulness enhancing term.

[Zhang, Isola, Efros, ECCV 2016]

Designing loss functions

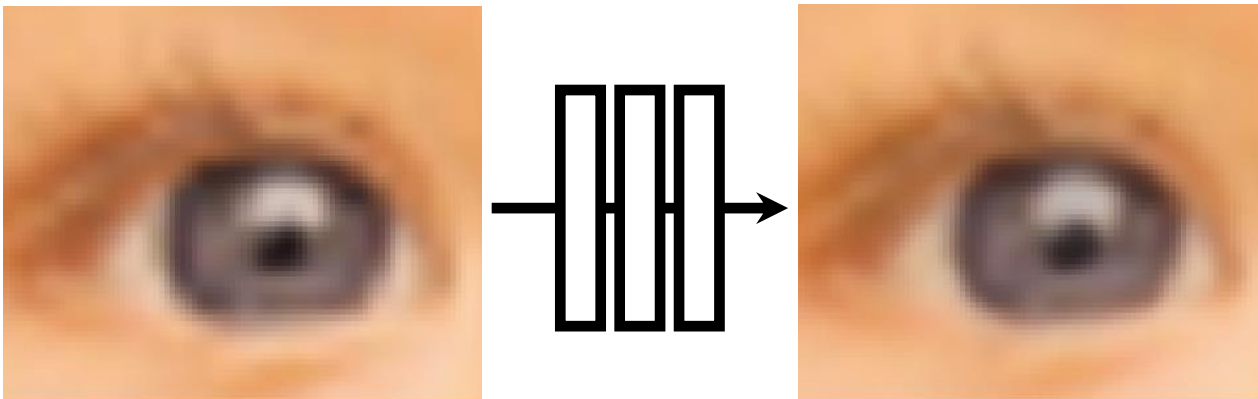
Image colorization



[Zhang, Isola, Efros, ECCV 2016]

L2 regression

Super-resolution

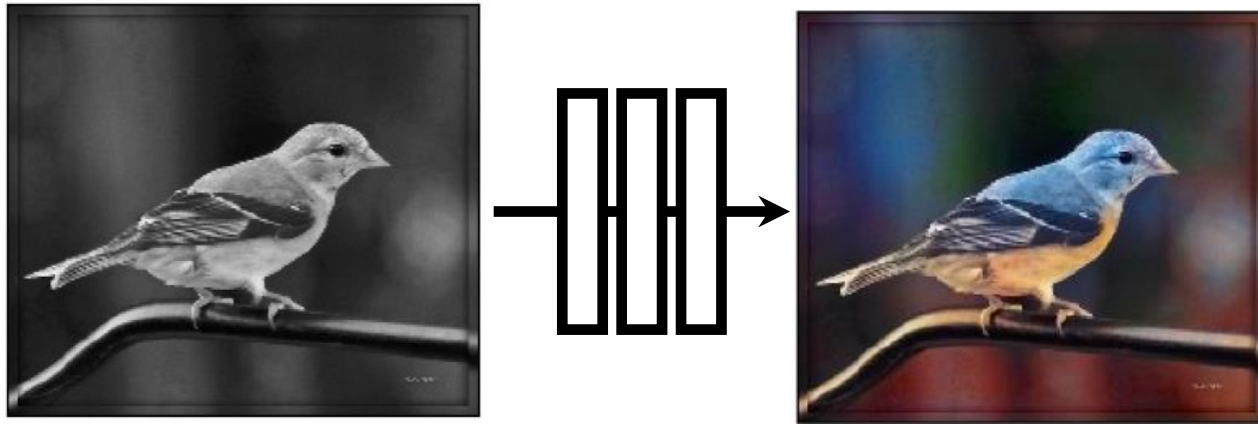


[Johnson, Alahi, Li, ECCV 2016]

L2 regression

Designing loss functions

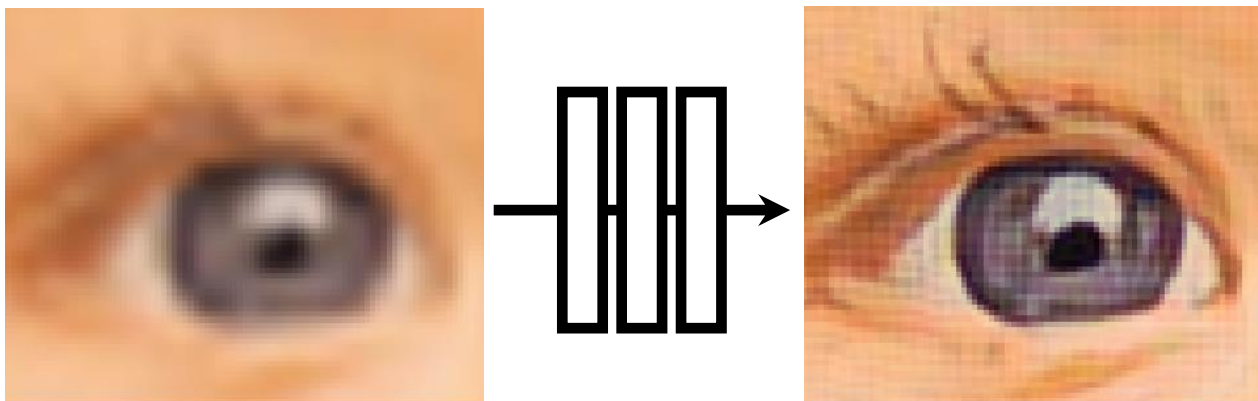
Image colorization



[Zhang, Isola, Efros, ECCV 2016]

Cross entropy objective,
with colorfulness term

Super-resolution

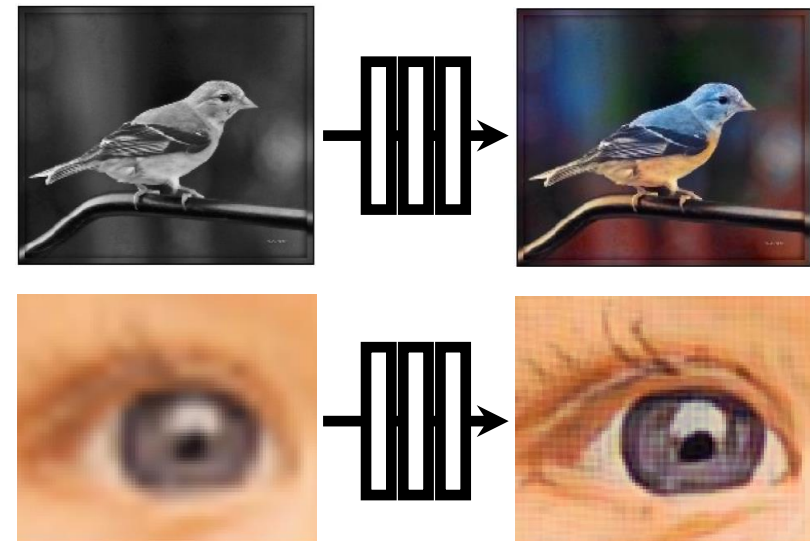
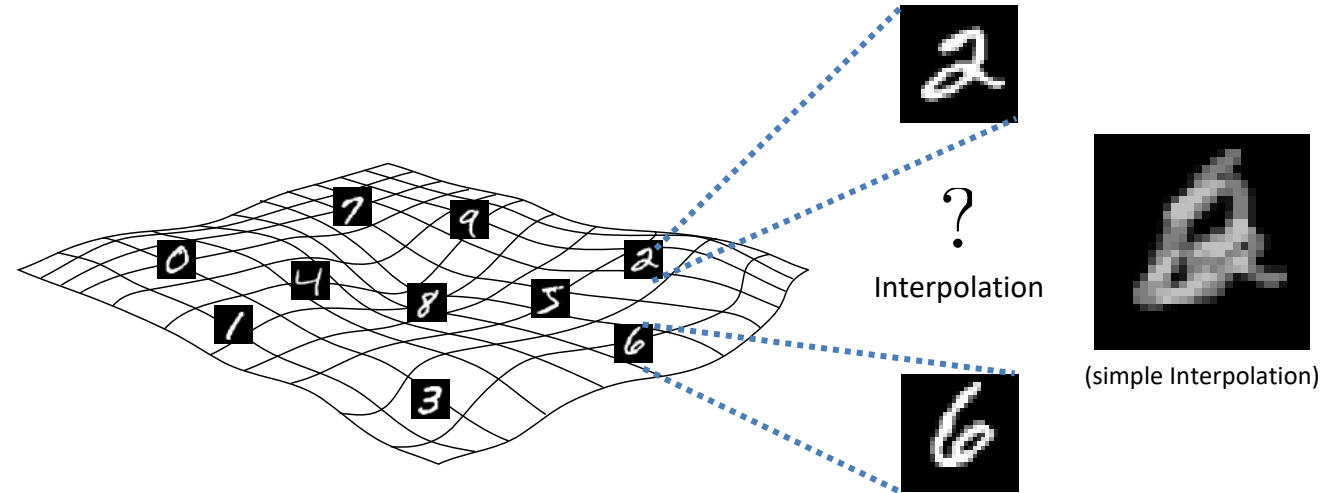


[Johnson, Alahi, Li, ECCV 2016]

Deep feature covariance
matching objective

Better Loss Function: Sticking to the Manifold

- How do we design a loss function that penalizes images that aren't on the image manifold?
- Key insight: we will *learn* our loss function by training a network to discriminate between images that are on the manifold and images that aren't

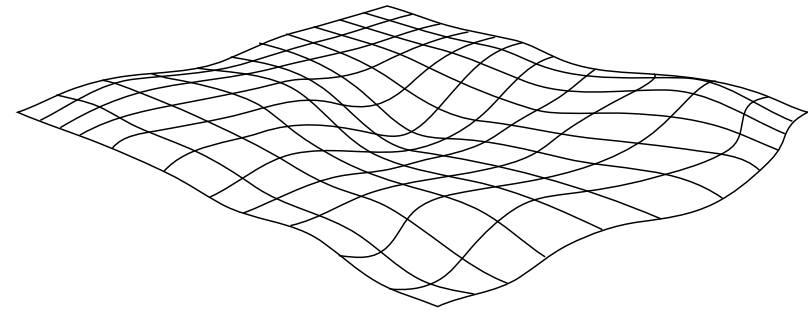


Abe Davis, with slides from Jin Sun and Phillip Isola

PART 3: GENERATIVE ADVERSARIAL NETWORKS (GANS)

Generative Adversarial Networks (GANs)

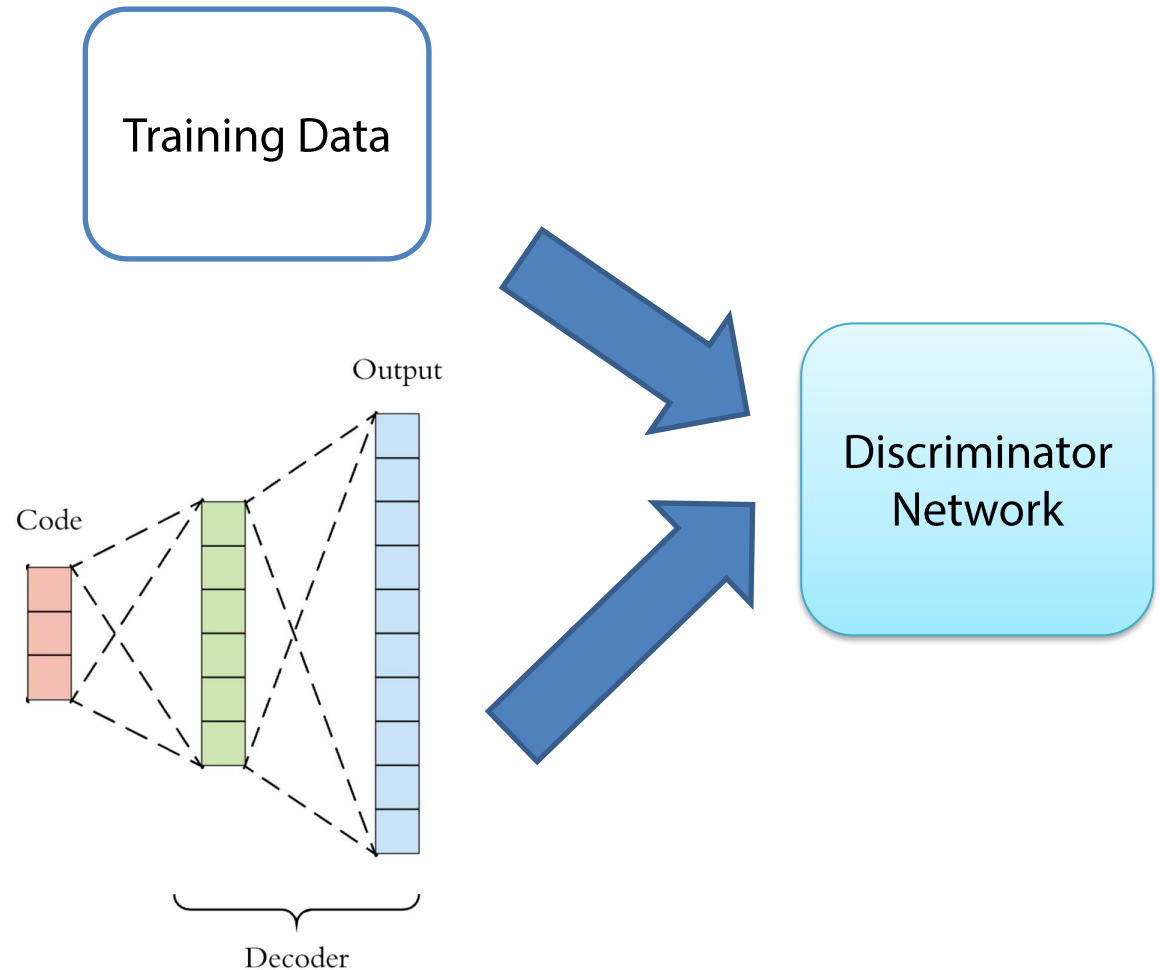
- Basic idea: Learn a mapping from some latent space to images on a particular manifold



- Example of a **Generative Model**:
 - We can think of classification as a way to compute some $P(x)$ that tells us the probability that image x is a member of a class.
 - Rather than simply evaluating this distribution, a generative model tries to learn a way to sample from it

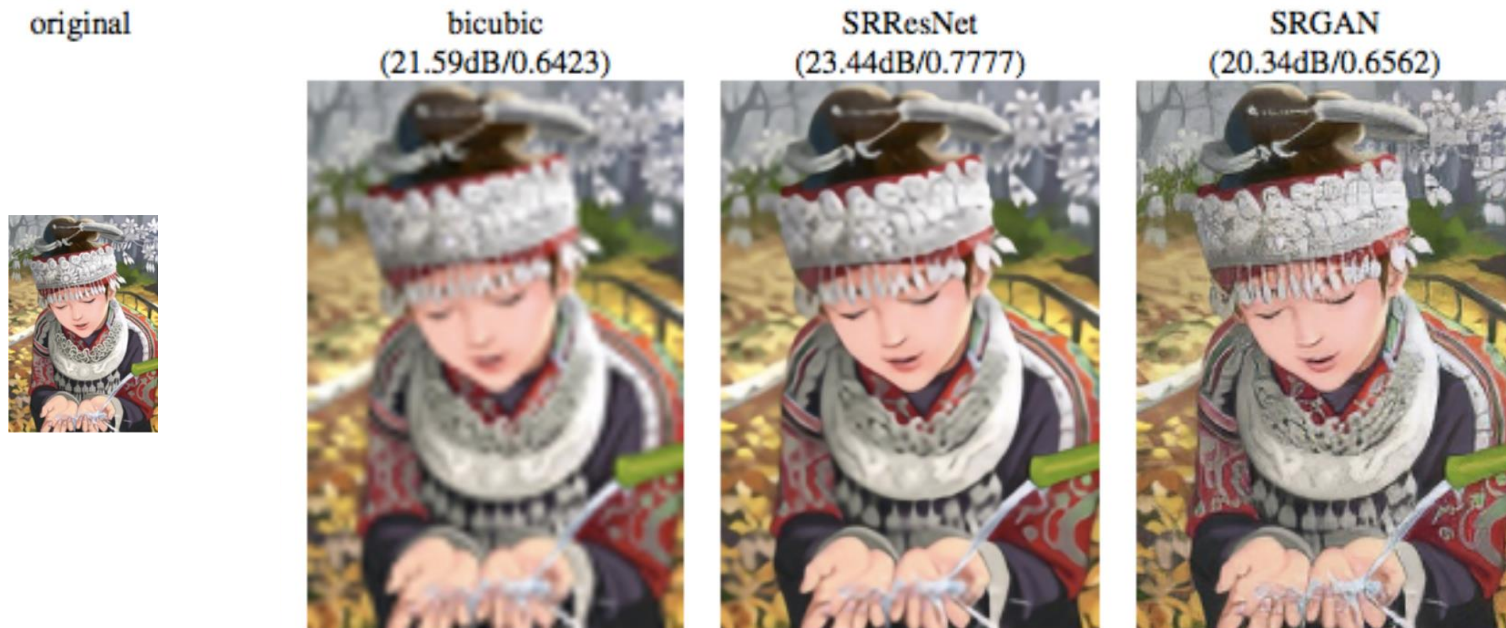
Generative Adversarial Networks (GANs)

- Generator network has similar structure to the decoder of our autoencoder
 - Maps from some latent space to images
- We train it in an adversarial manner against a discriminator network
 - Generator takes image noise, and tries to create output indistinguishable from training data
 - Discriminator tries to distinguish between generator output and training data



First: Conditional GANs

- Generate samples from a *conditional distribution* (conditioned on some other input)
- Example: generate high-resolution image conditioned on low resolution input

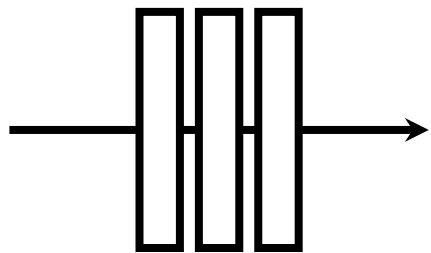


[Ledig et al 2016]

\mathbf{x}

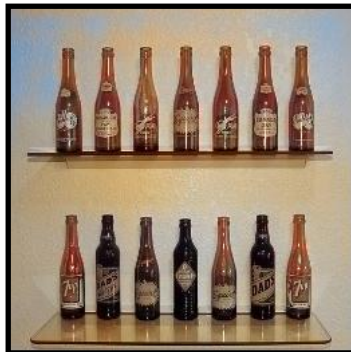


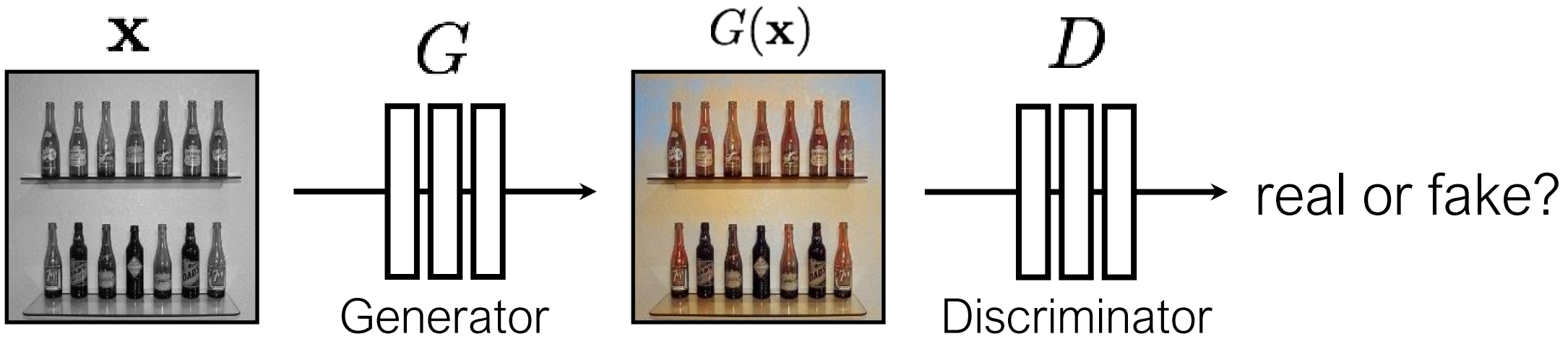
G



Generator

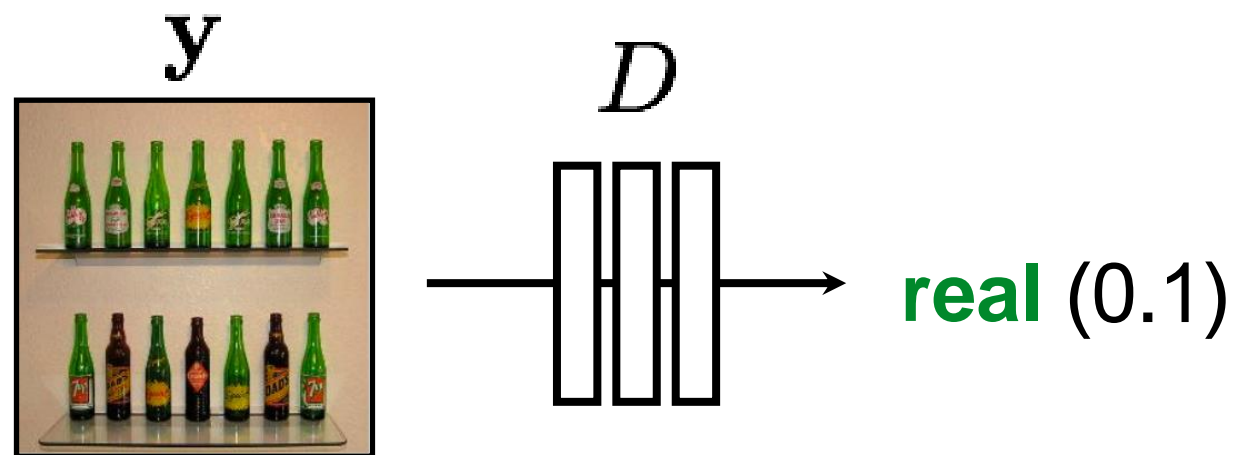
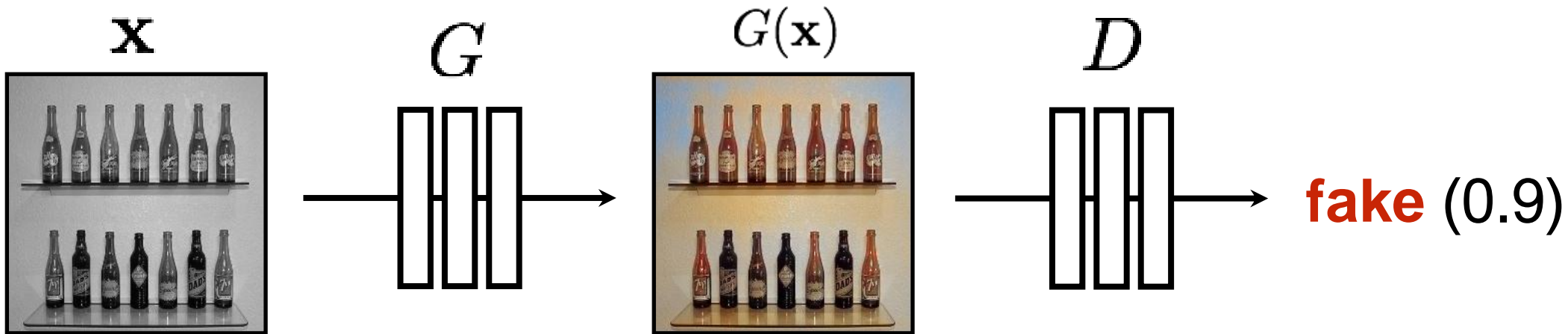
$G(\mathbf{x})$





G tries to synthesize fake images that fool **D**

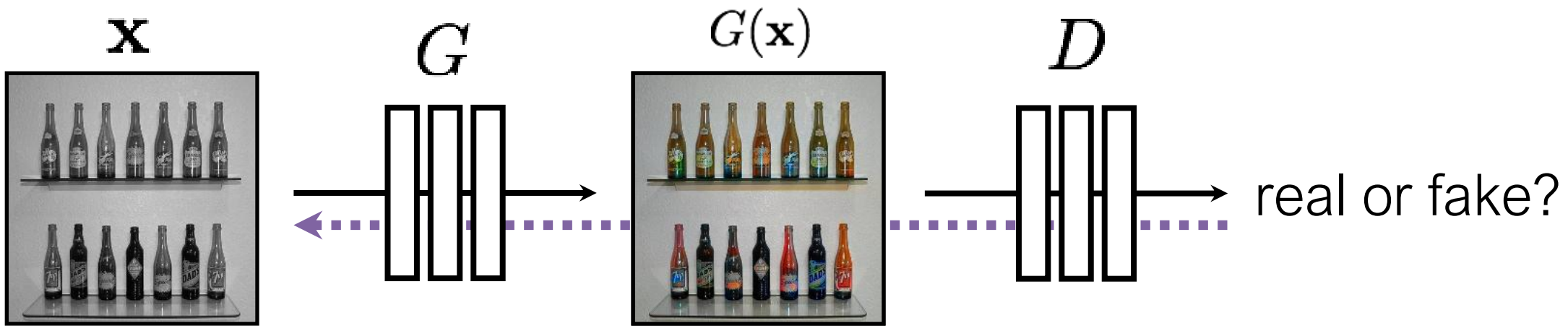
D tries to identify the fakes



(Identify generated images as fake)

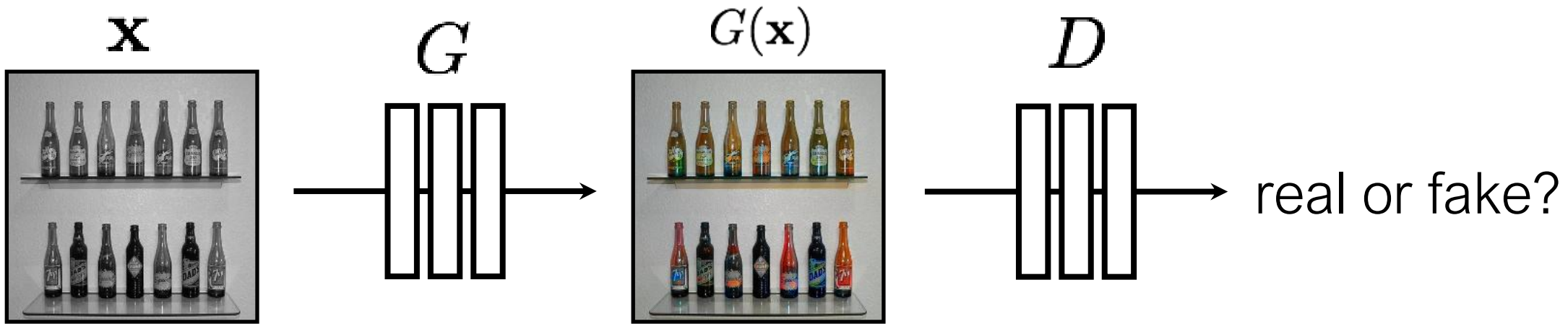
(Identify training images as real)

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



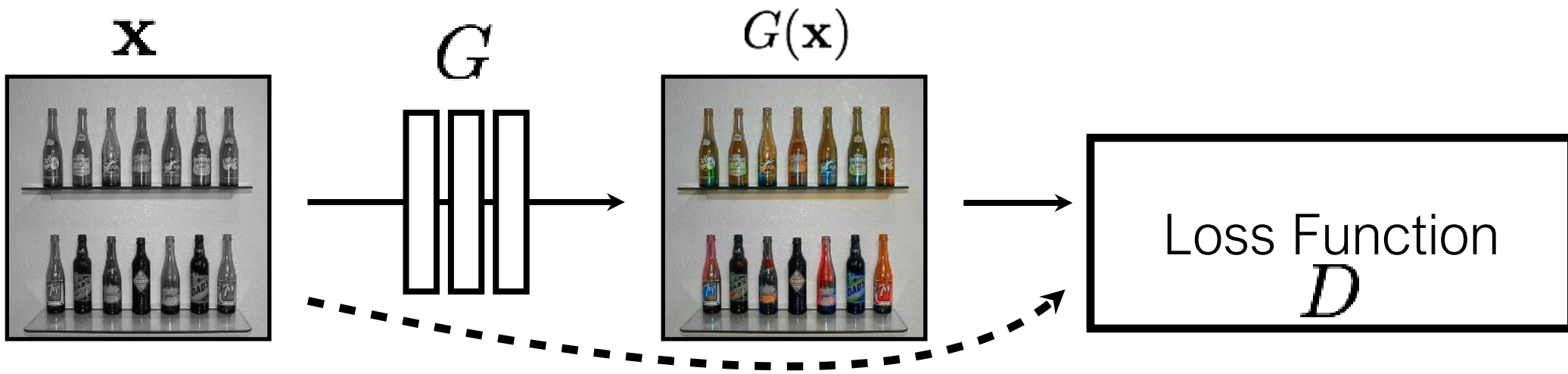
G tries to synthesize fake images that *fool* **D**:

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



G tries to synthesize fake images that *fool* the *best* **D**:

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

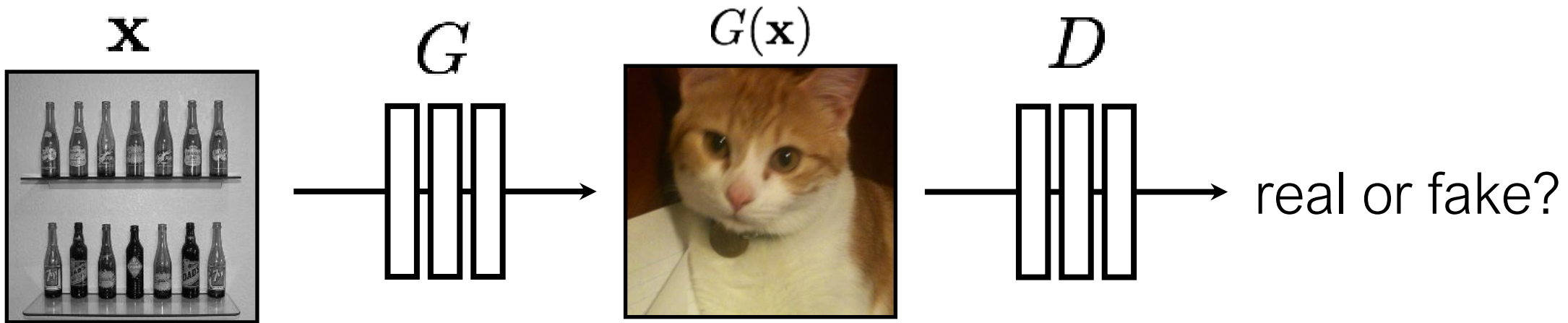


G's perspective: **D** is a loss function.

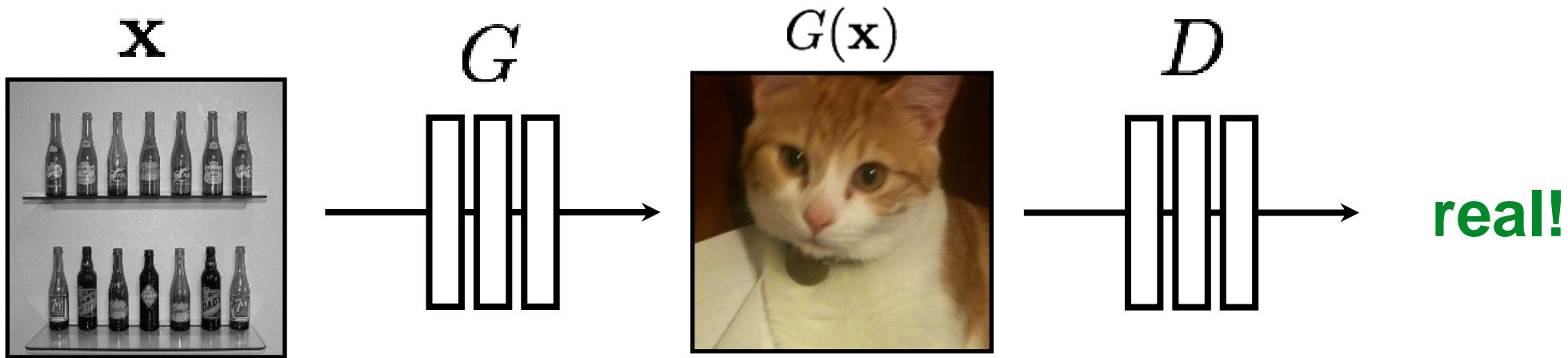
Rather than being hand-designed, it is *learned*.

[Goodfellow et al., 2014]

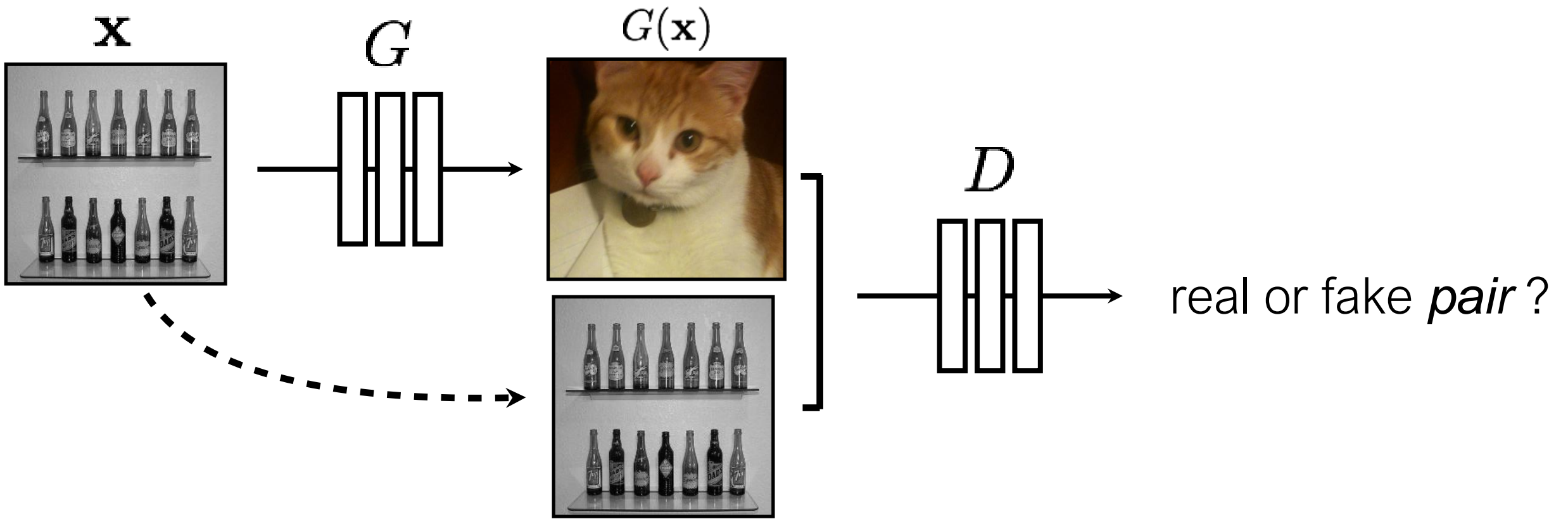
[Isola et al., 2017]



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



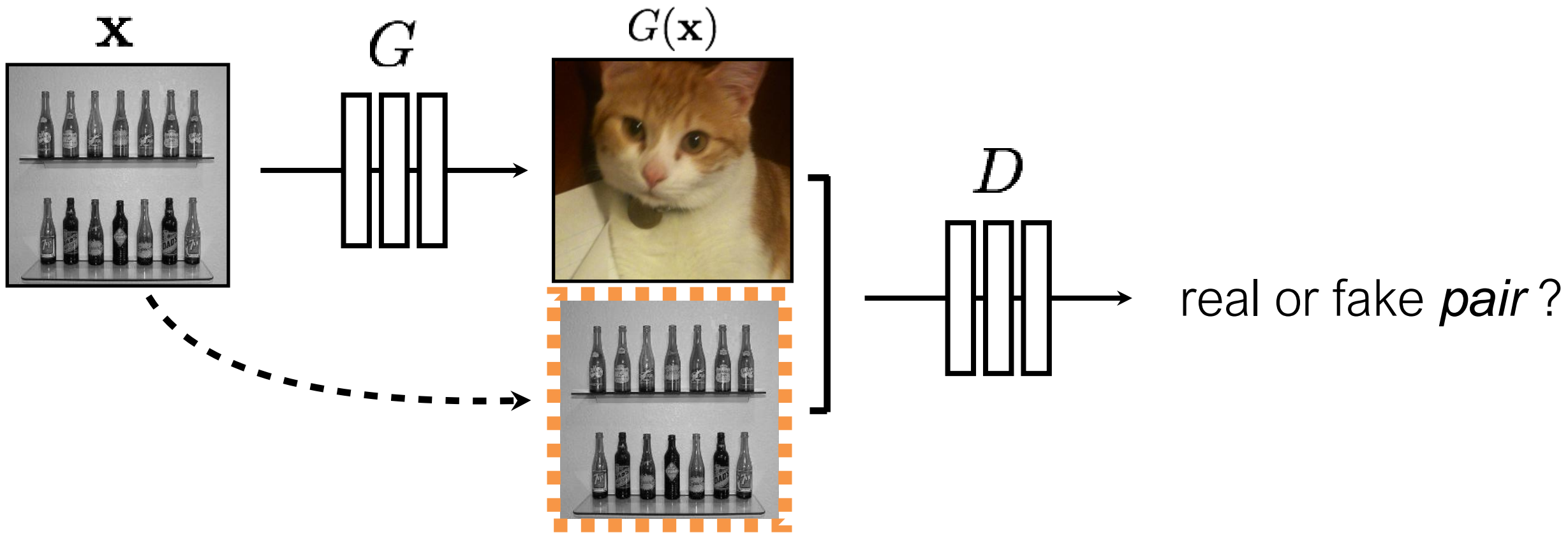
$$\arg \min_G \max_D \mathbb{E}_{\mathbf{x}, \mathbf{y}} [\log D(G(\mathbf{x})) + \log(1 - D(\mathbf{y}))]$$



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

[Goodfellow et al., 2014]

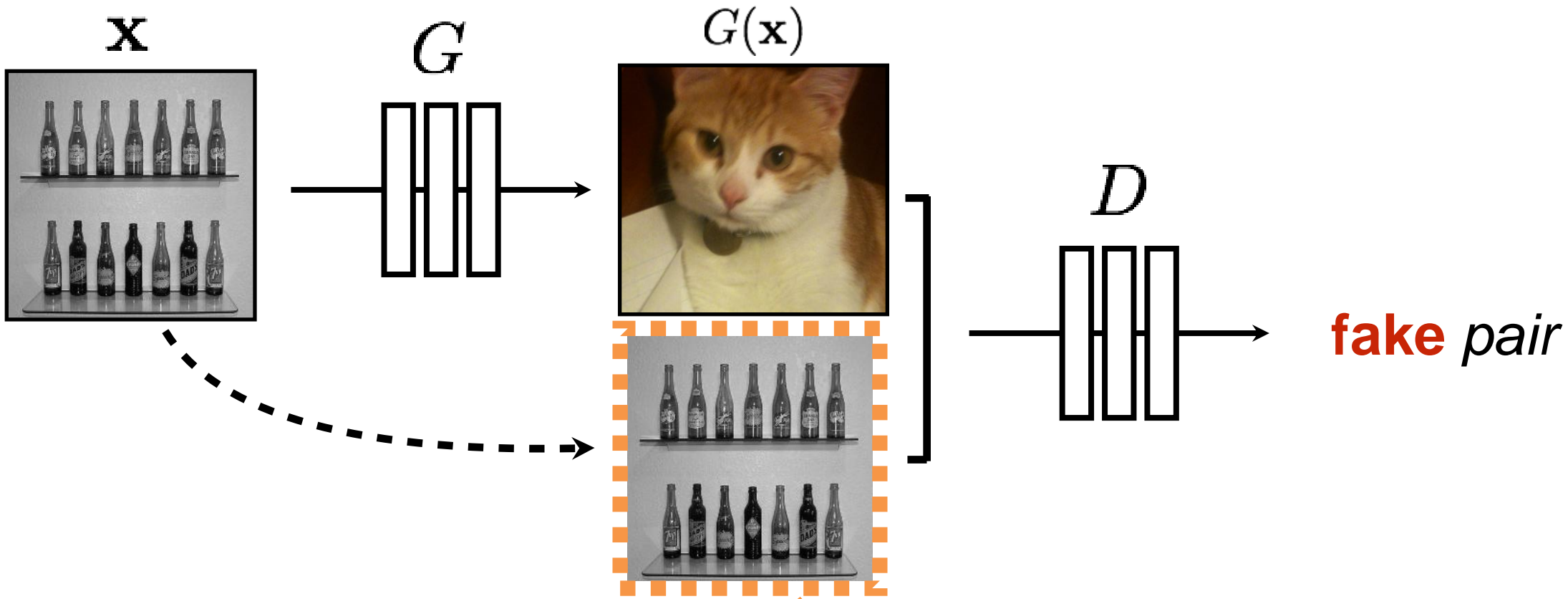
[Isola et al., 2017]



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

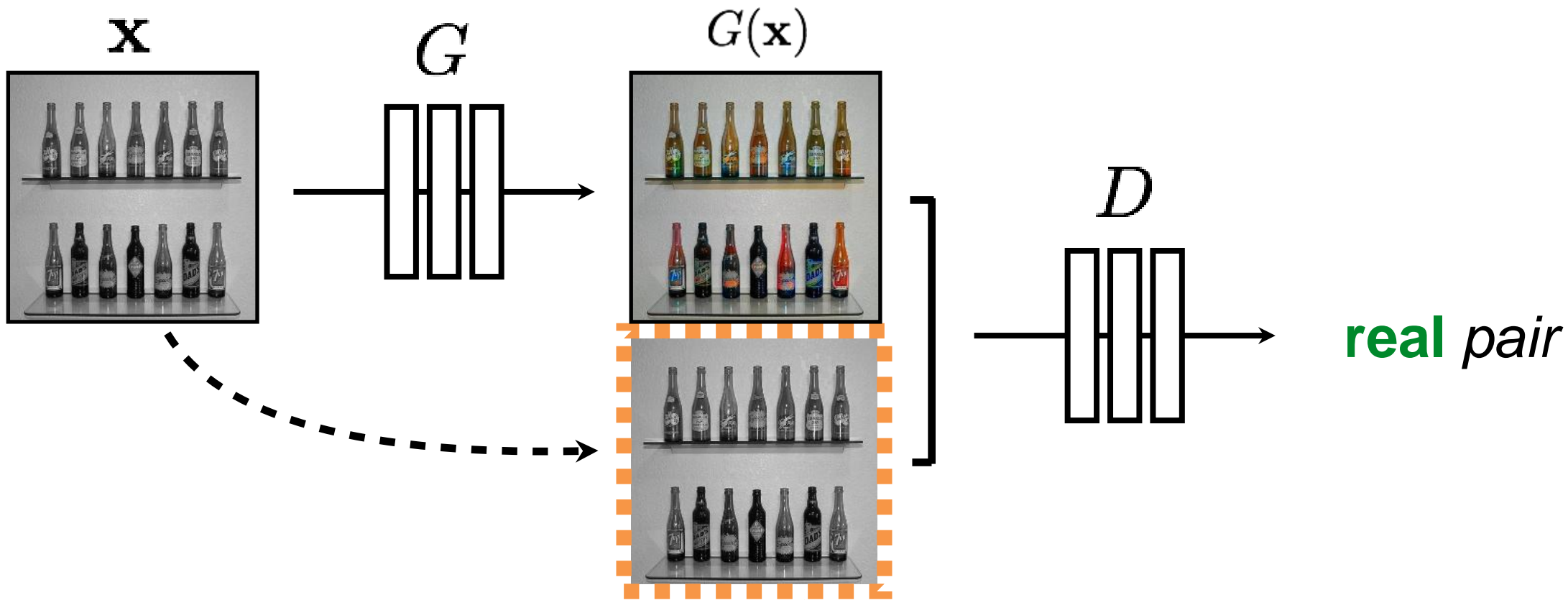
[Goodfellow et al., 2014]

[Isola et al., 2017]



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

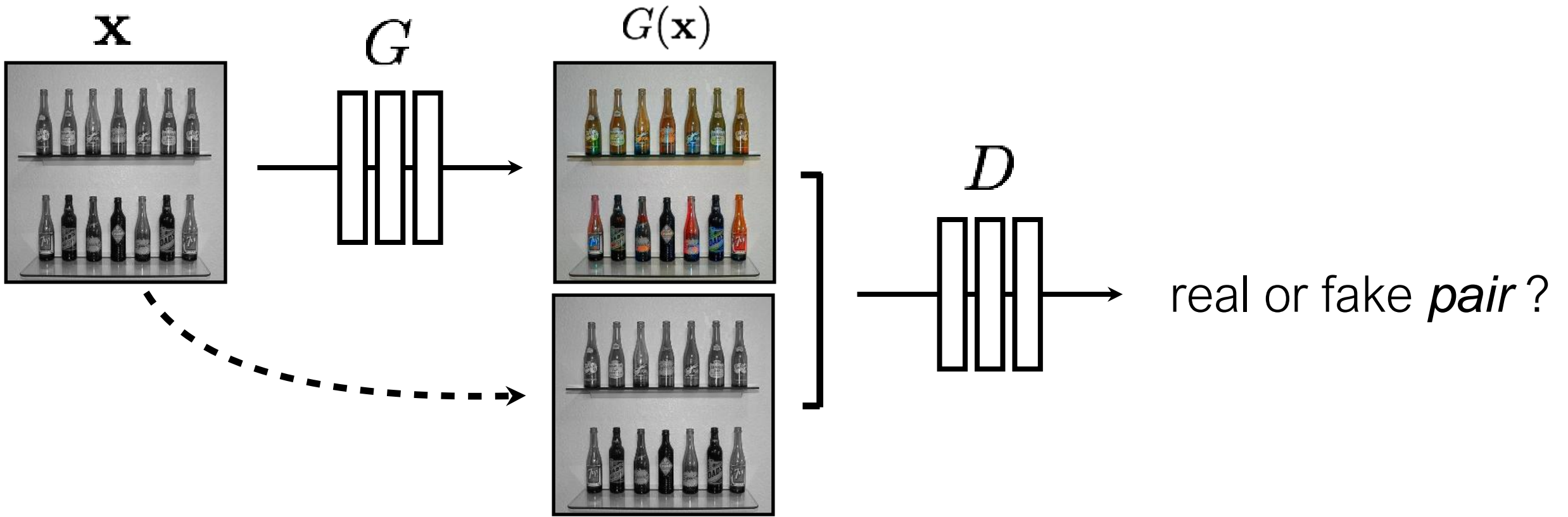
[Goodfellow et al., 2014]
 [Isola et al., 2017]



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

[Goodfellow et al., 2014]

[Isola et al., 2017]



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

[Goodfellow et al., 2014]

[Isola et al., 2017]

More Examples of Image-to-Image Translation with GANs

- We have pairs of corresponding training images
- Conditioned on one of the images, sample from the distribution of likely corresponding images

Segmentation to Street Image



Aerial Photo To Map



Edges to Image



BW → Color

Input

Output

Input

Output

Input

Output



Input



Output



Groundtruth

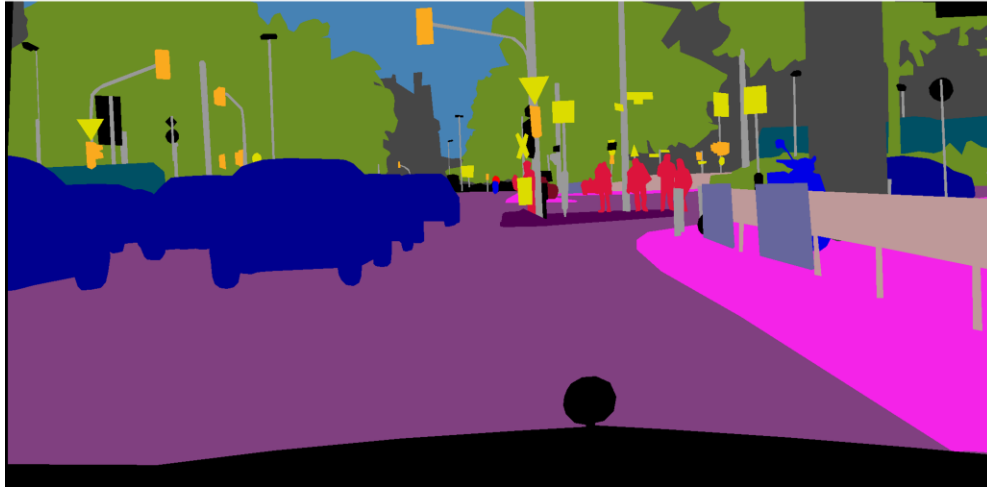


Data from
[\[maps.google.com\]](https://maps.google.com)

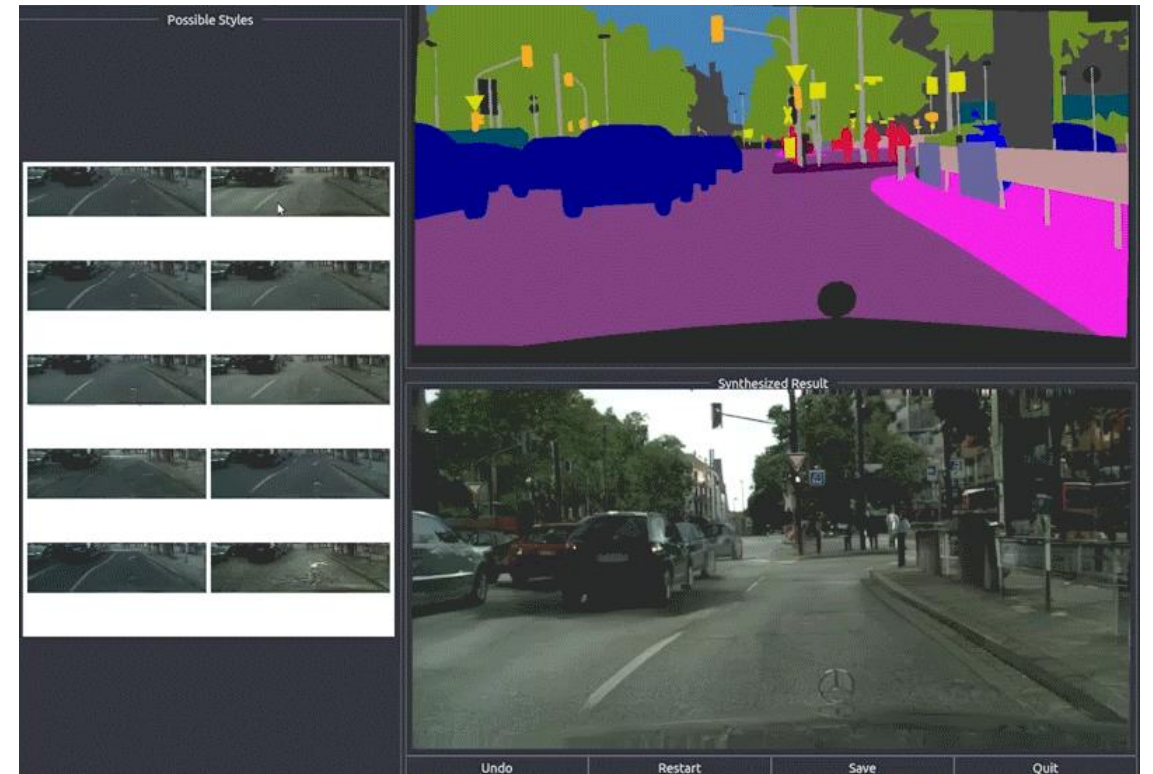


Labels → Street Views

Input labels



Synthesized image



Day → Night

Input

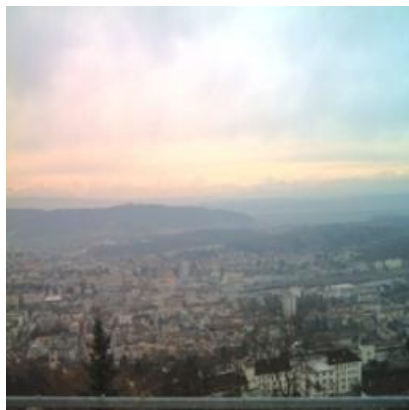
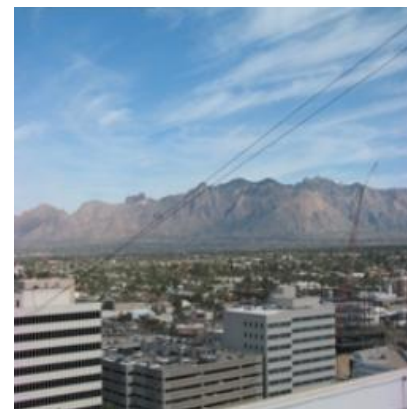
Output

Input

Output

Input

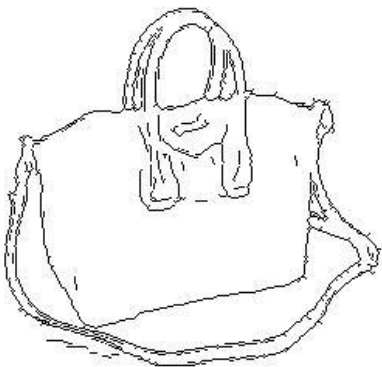
Output



Edges → Images

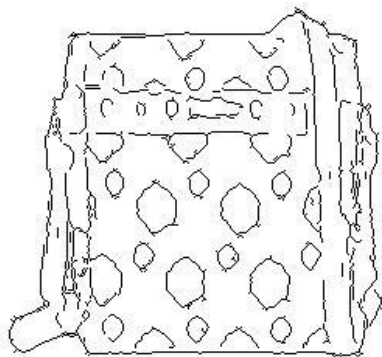
Input

Output



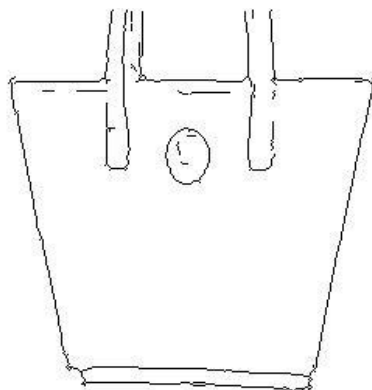
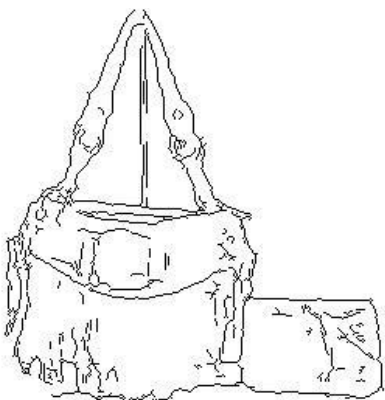
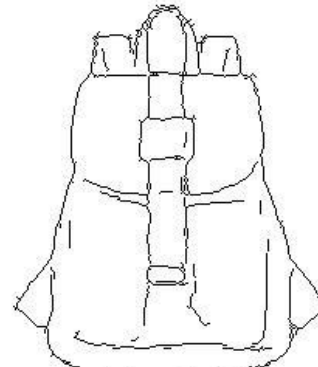
Input

Output

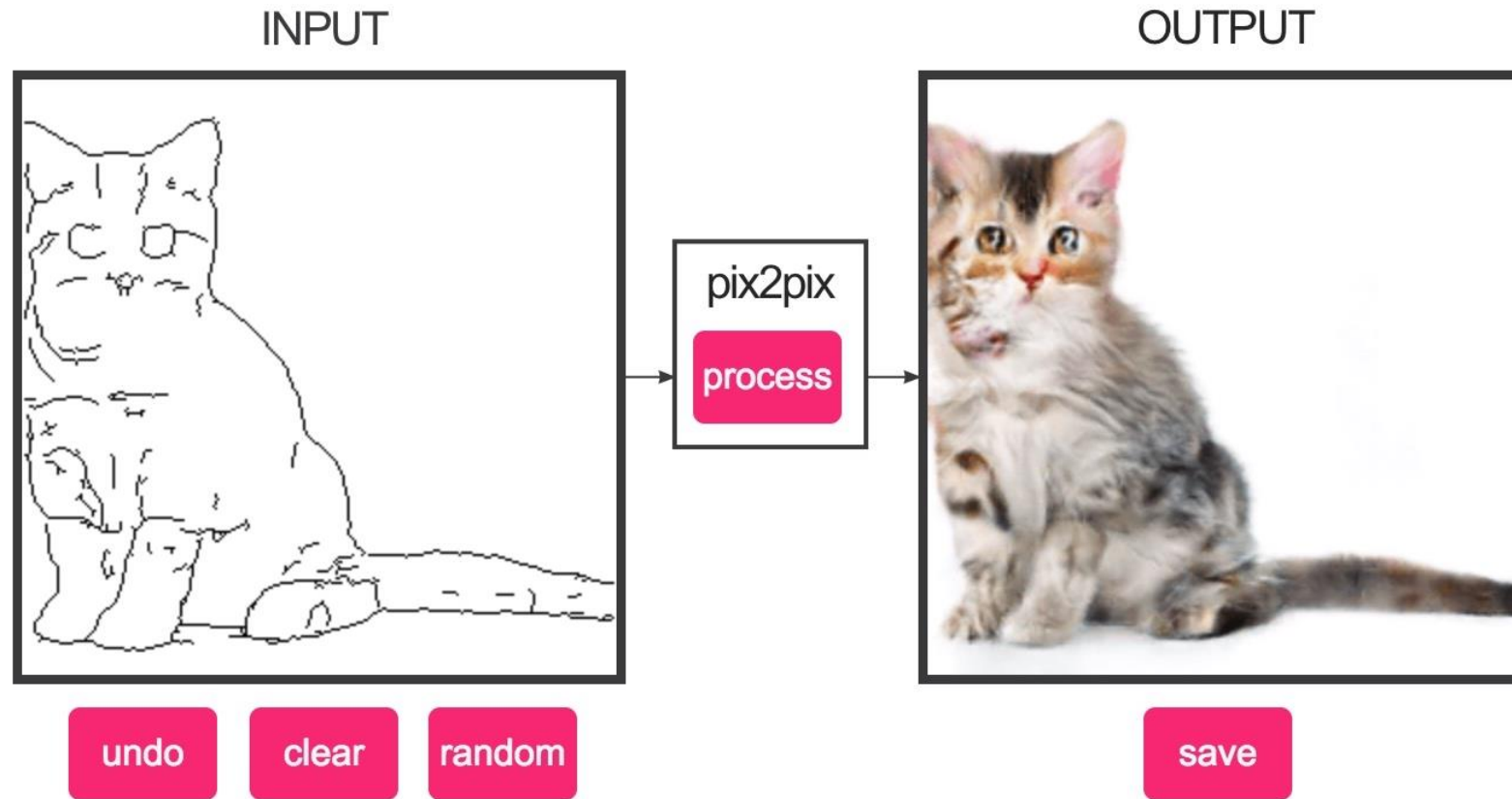


Input

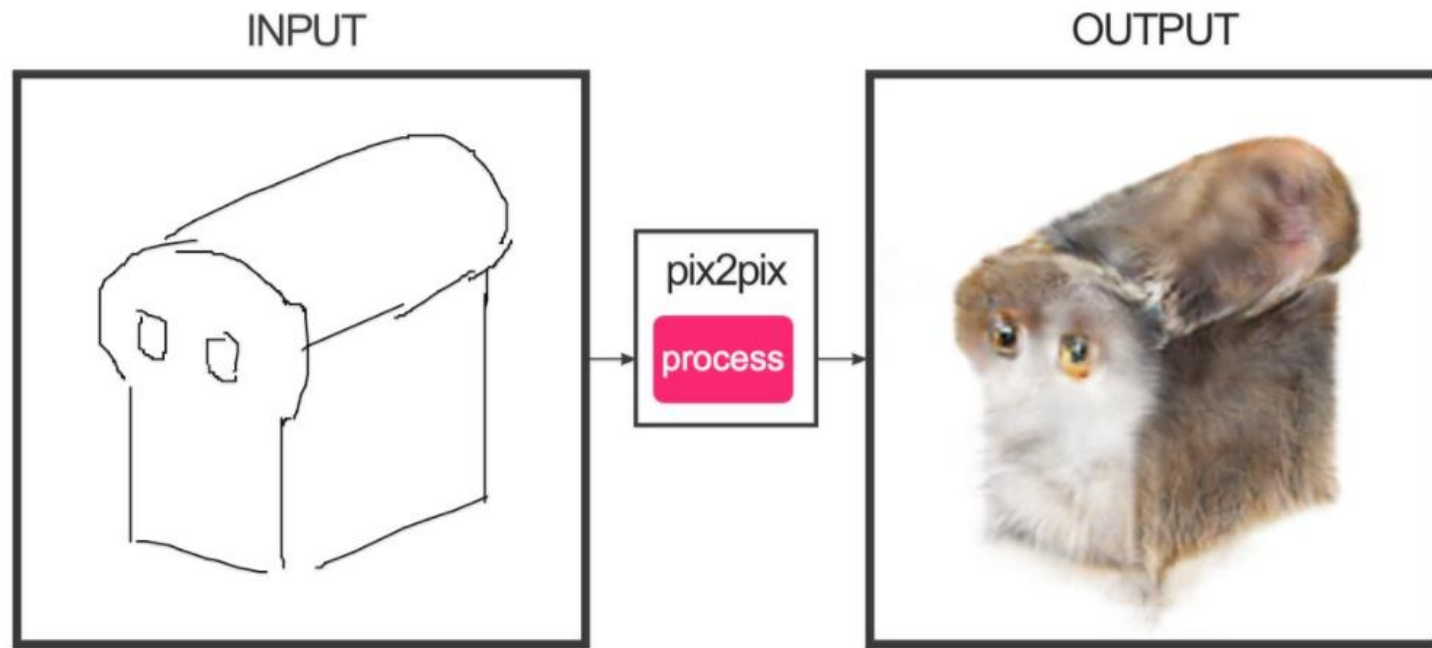
Output



Demo



<https://affinelayer.com/pixsrv/>

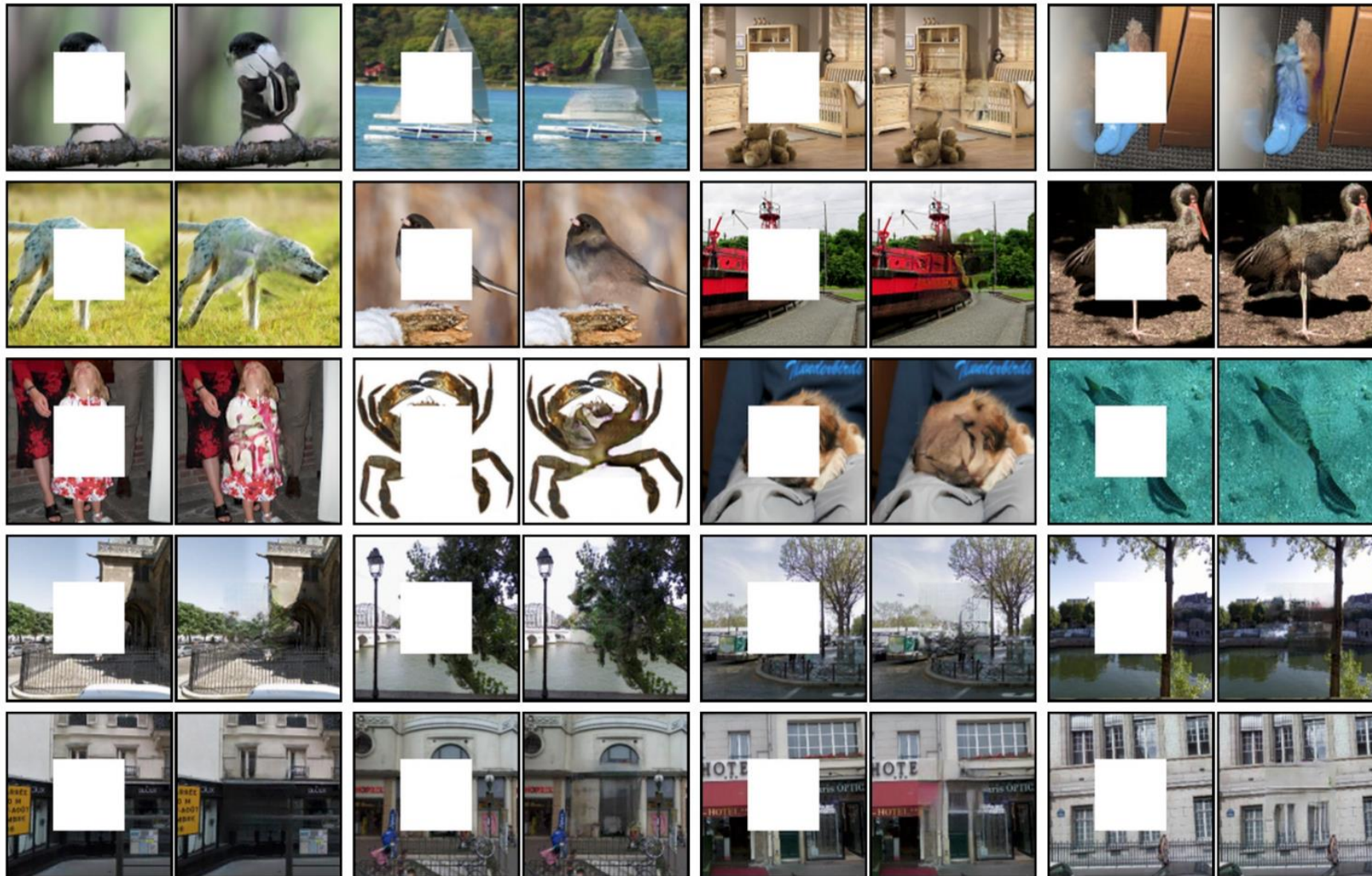


Ivy Tasi @ivymyt

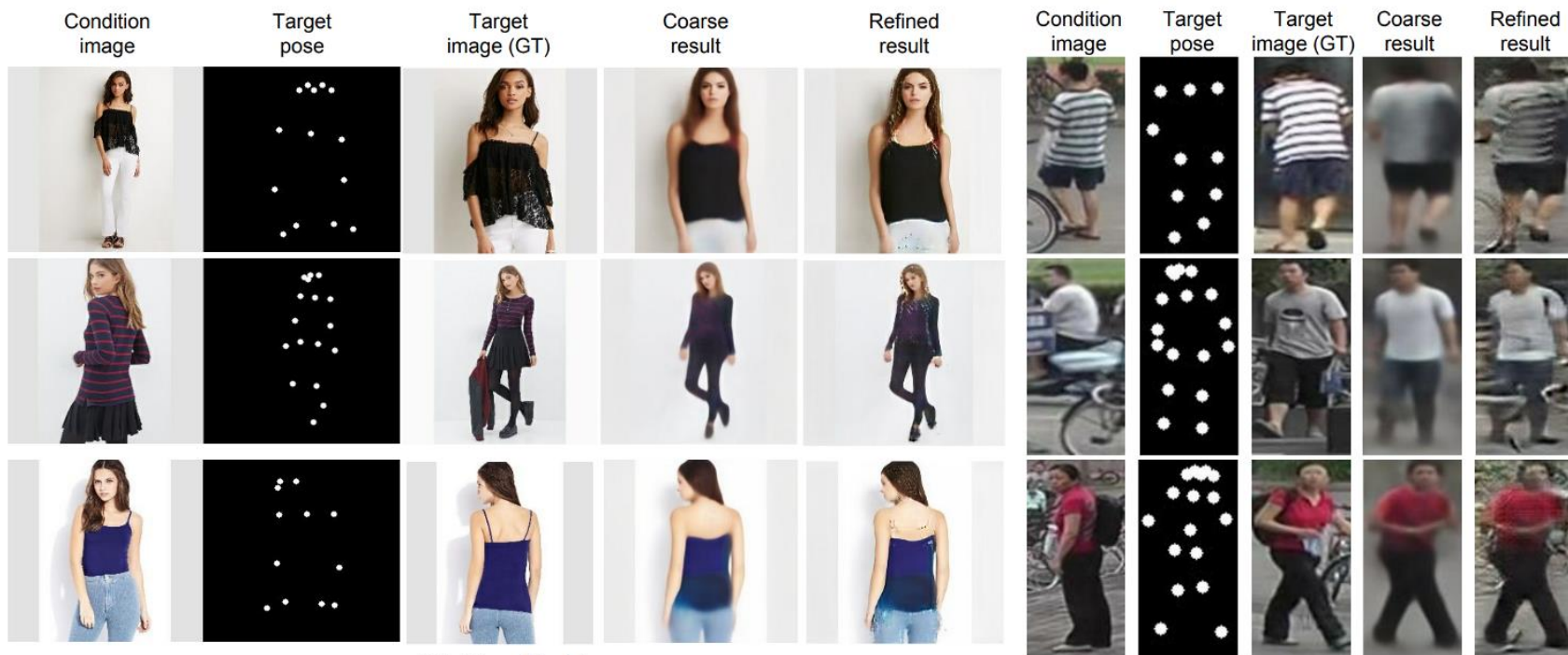


Vitaly Vidmirov @vvid

Image Inpainting



Pose-guided Generation



(a) DeepFashion

(b) Market-1501



(c) Generating from a sequence of poses

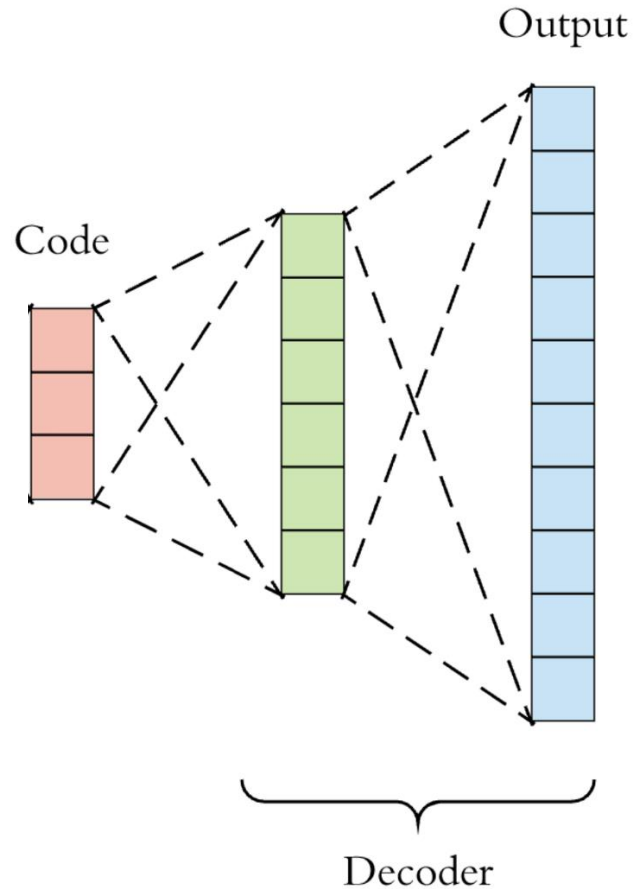
Challenges —> Solutions

- Output is high-dimensional, structured object
 - Approach: Use a deep net, D , to analyze output!
- Uncertainty in mapping; many plausible outputs
 - Approach: D only cares about “plausibility”, doesn’t hedge

Unconditional GANs: Learning an image manifold for a category



Category-specific
image dataset (FFHQ)

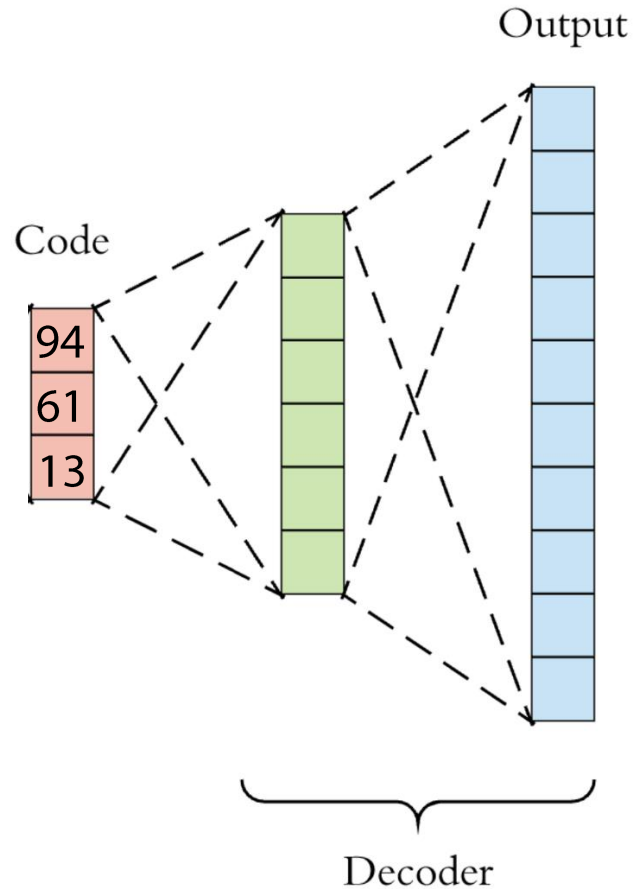


Latent code ("noise")-to-image decoder network

Unconditional GANs: Learning an image manifold for a category



Category-specific
image dataset (FFHQ)



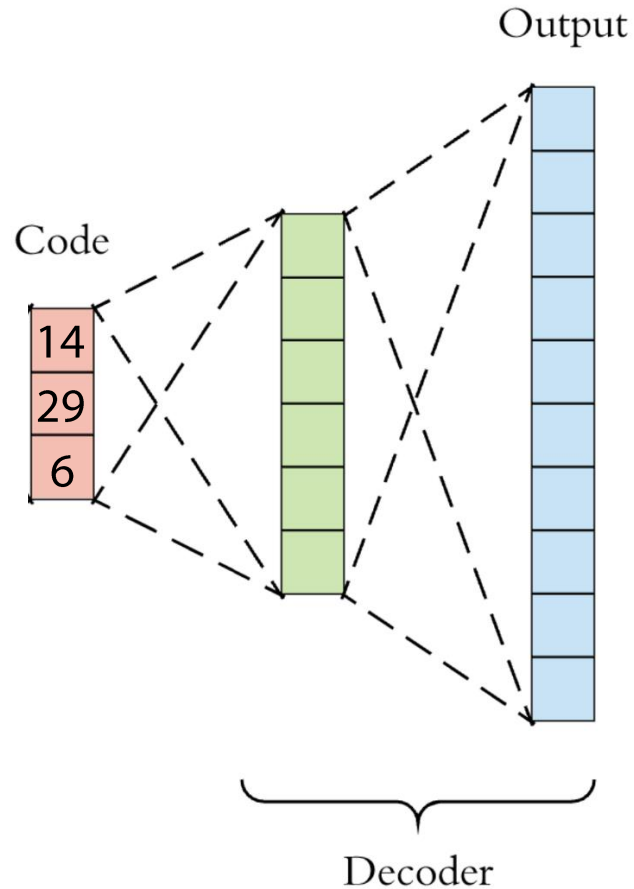
Output image

Latent code ("noise")-to-image decoder network

Unconditional GANs: Learning an image manifold for a category



Category-specific
image dataset (FFHQ)



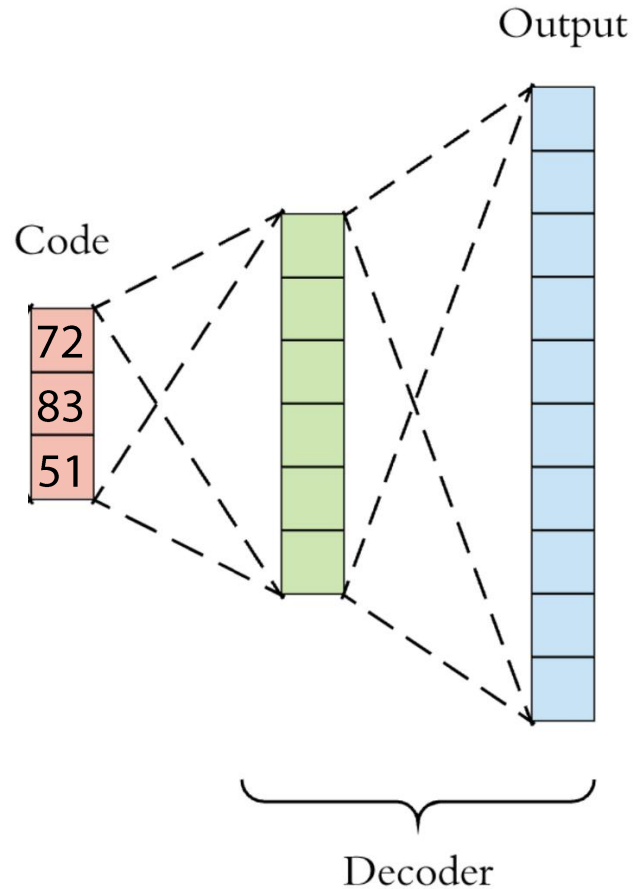
Output image

Latent code ("noise")-to-image decoder network

Unconditional GANs: Learning an image manifold for a category



Category-specific
image dataset (FFHQ)



Output image

Latent code ("noise")-to-image decoder network

Example: Randomly Sampling the Space of Face Images



A



B

[Which face is real?](#)

Example: Randomly Sampling the Space of Face Images



A



B

[Which face is real?](#)

StyleGAN



A Style-Based Generator Architecture for Generative Adversarial Networks

Tero Karras, Samuli Laine, Timo Aila

<https://github.com/NVlabs/stylegan>

StyleGAN2 [2020]



Analyzing and Improving the Image Quality of StyleGAN

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, Timo Aila

<https://github.com/NVlabs/stylegan2>

StyleGAN3 [2021]



Alias-Free Generative Adversarial Networks (StyleGAN3)

Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, Timo Aila



StyleGAN2



StyleGAN3 (Ours)

GAN models trained on animal faces: interpolating between latent codes



GAN models trained on MetFaces: interpolating between latent codes

GANs for 3D

EG3D: Efficient Geometry-aware 3D Generative Adversarial Networks

Eric Ryan Chan ^{*1,2} Connor Zhizhen Lin ^{*1} Matthew Aaron Chan ^{*1}

Koki Nagano ^{*2} Boxiao Pan ¹ Shalini De Mello ² Orazio Gallo ²

Leonidas Guibas ¹ Jonathan Tremblay ² Sameh Khamis ² Tero Karras ²

Gordon Wetzstein ¹

¹ Stanford University ² NVIDIA

* Equal contribution.



<https://nvlabs.github.io/eg3d>

Limitations

- The unconditional models above must be trained per-category:
 - We have a separate model for every category – an animal face model, broccoli model, horse model, etc...
- What if we want to generate an image from **any** description?
- -> diffusion and text-to-image models

Recall: The Space of All Images

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



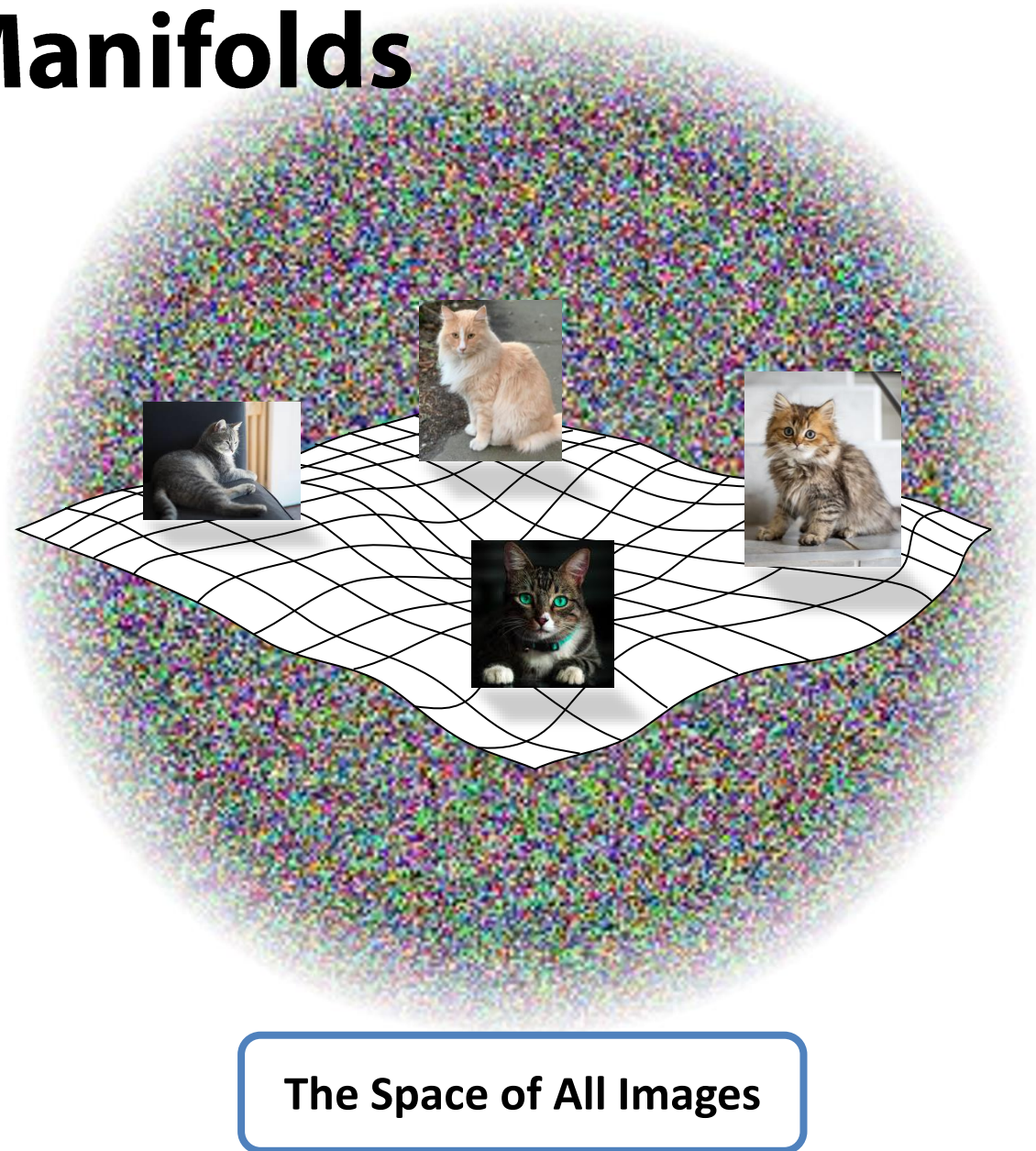
Question:

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

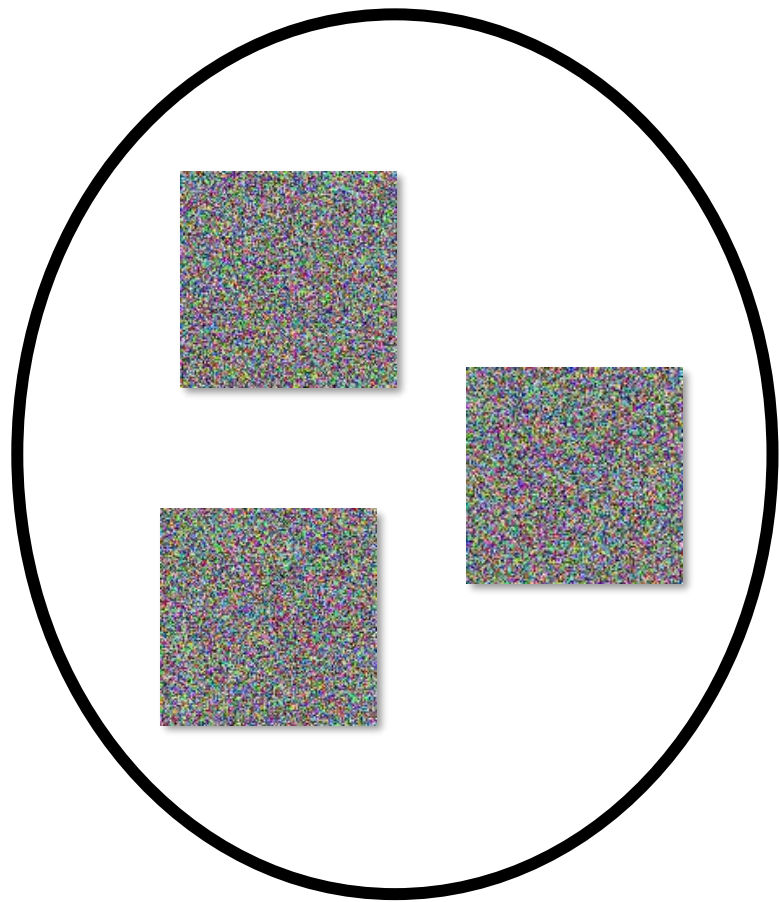
```
pixels = np.random.rand(100, 100, 3)
```

Recall: Natural Image Manifolds

- Most images are “noise”
- “Meaningful” images tend to form some manifold within the space of all images
- Images of a particular class fall on manifolds within that manifold...

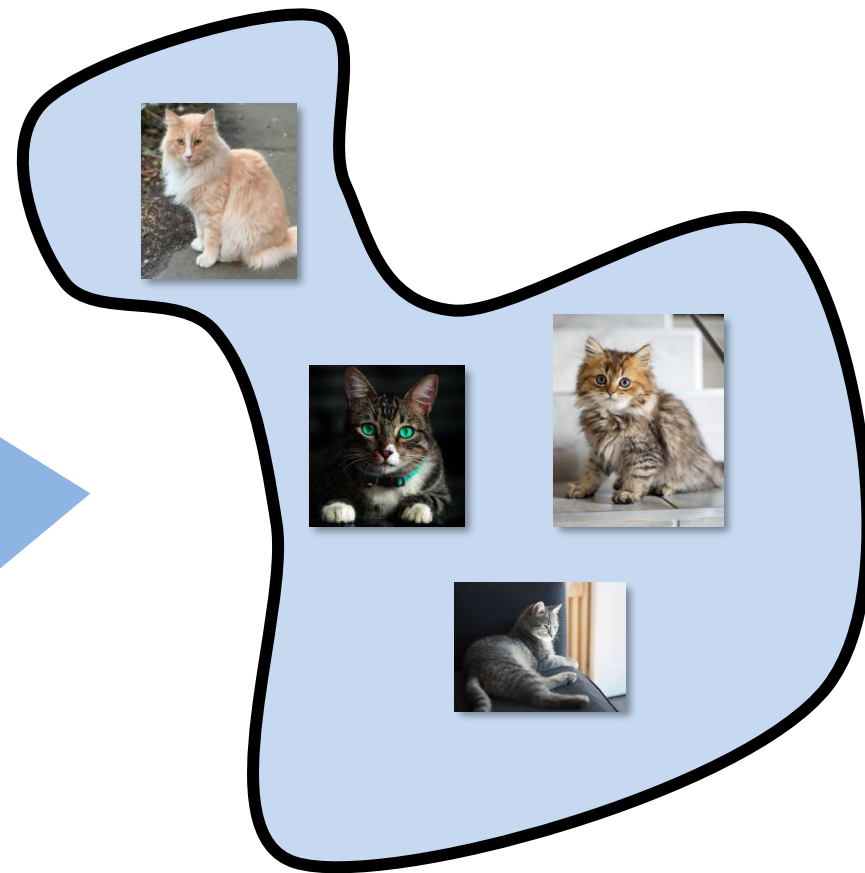


The Space of All Images

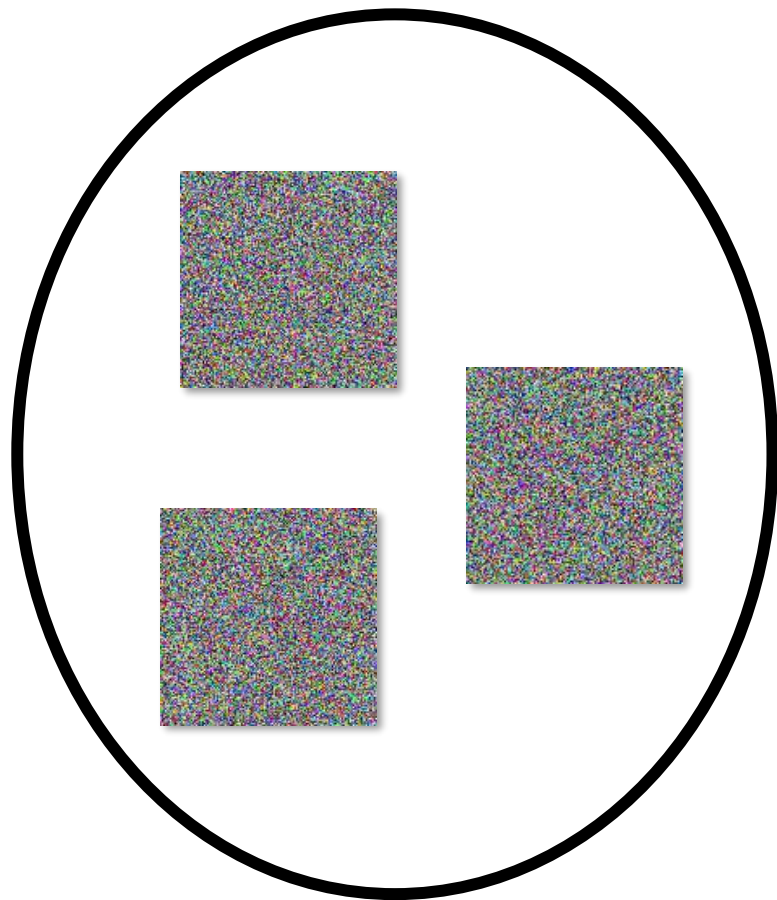


Random images

Diffusion

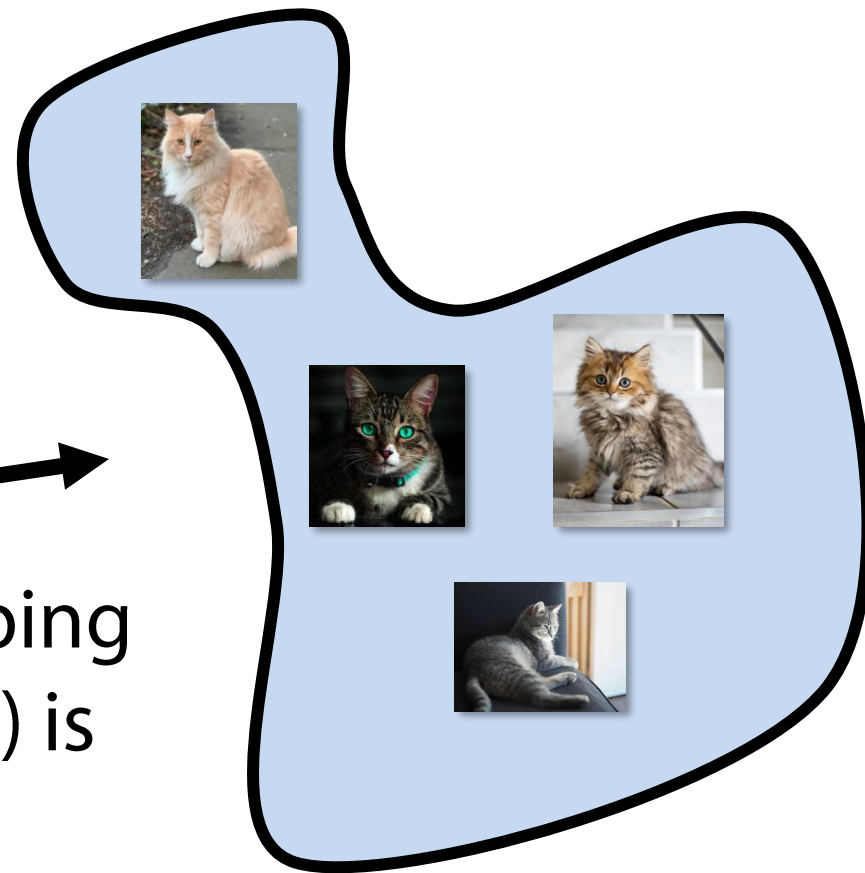
A large blue arrow points from the circle of random images to the manifold of cat images.

Manifold of cat images

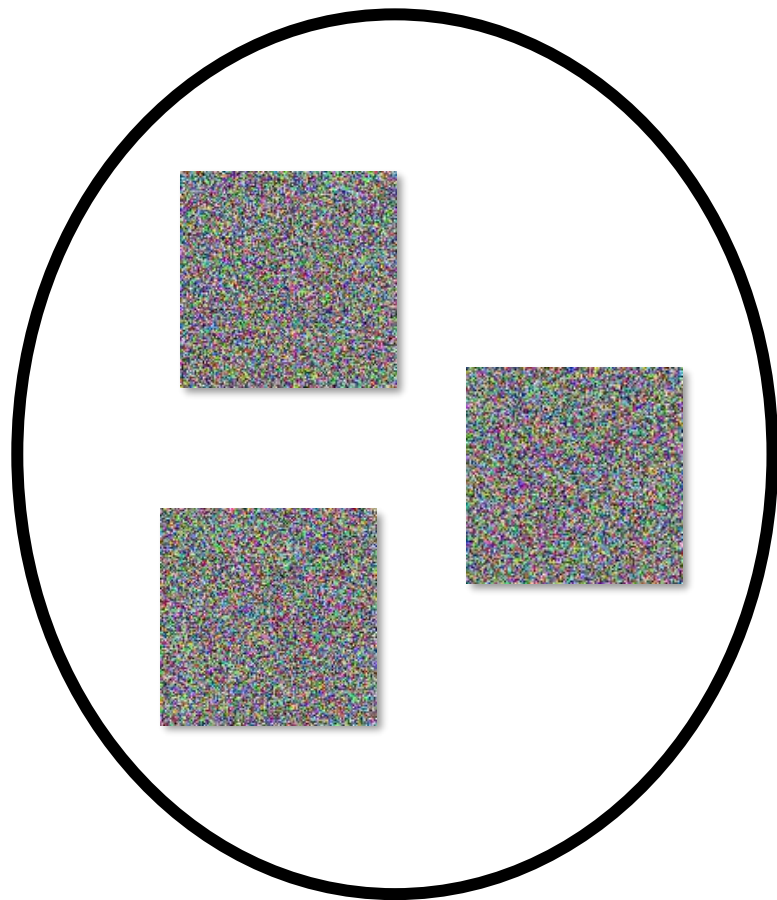


Random images

Forward mapping
(noise to cats) is
hard

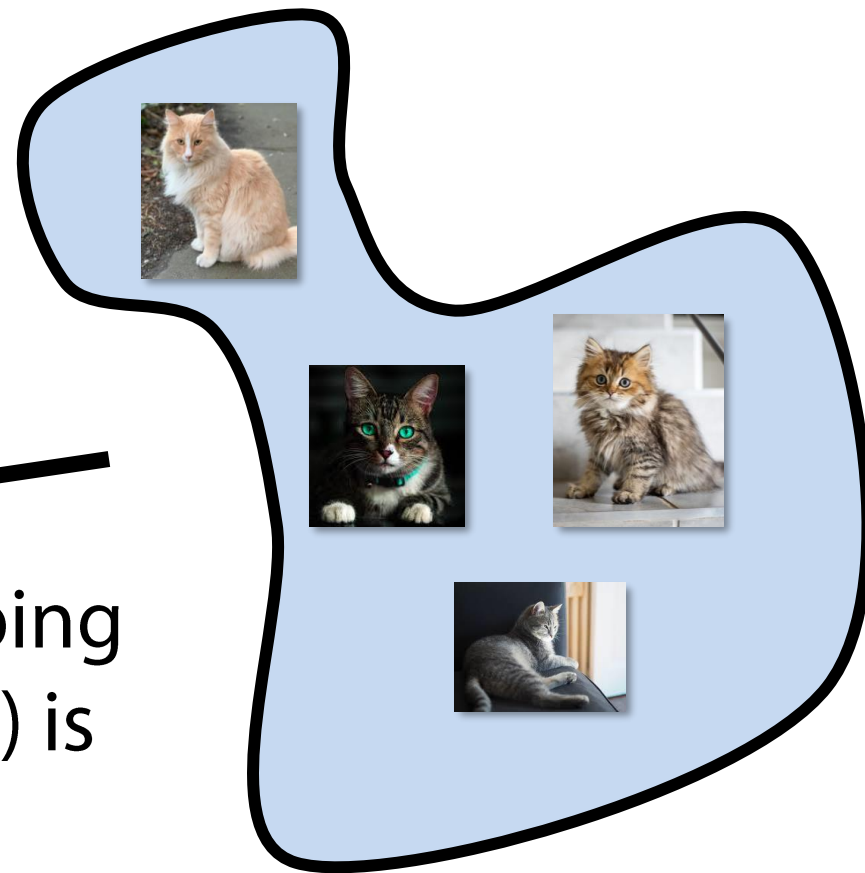


Manifold of cat images

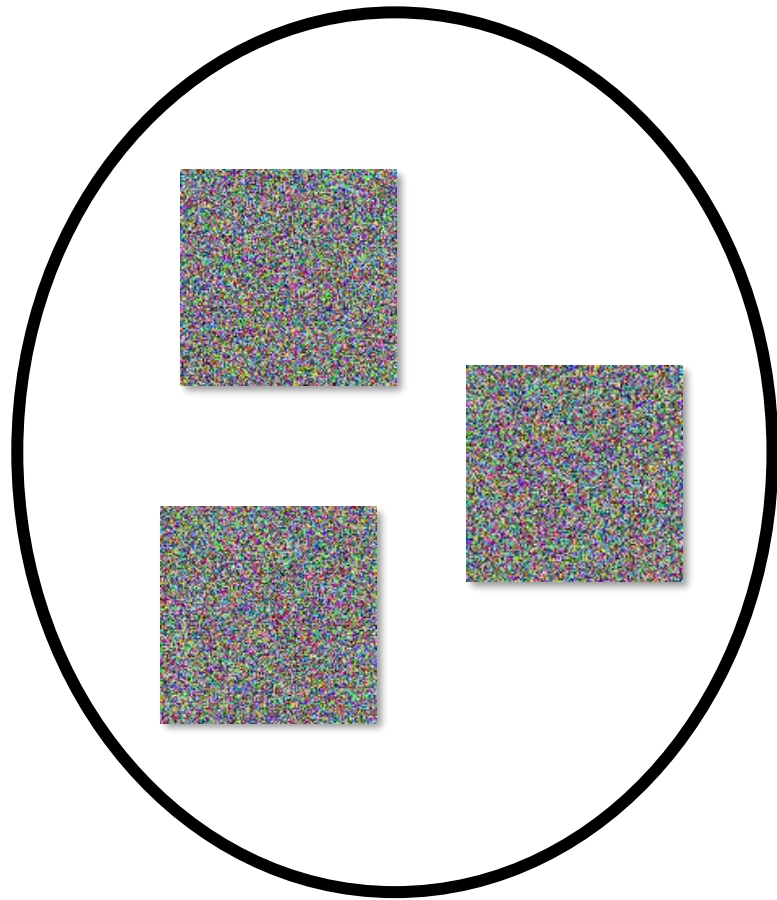


Random images

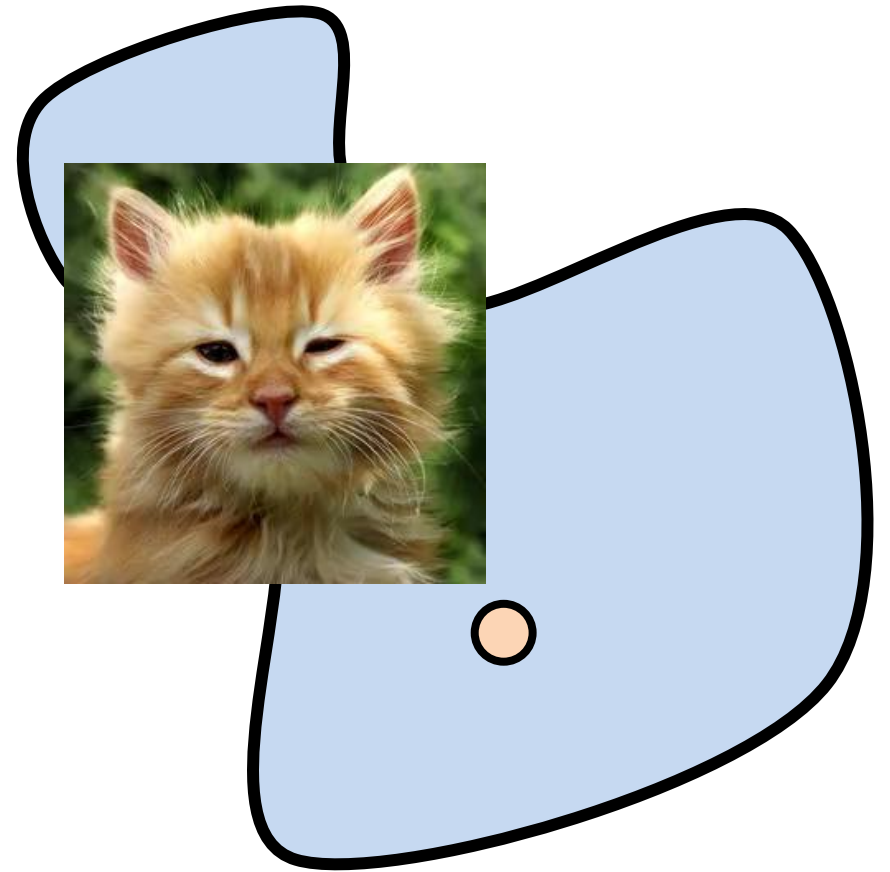
Reverse mapping
(cats to noise) is
easy



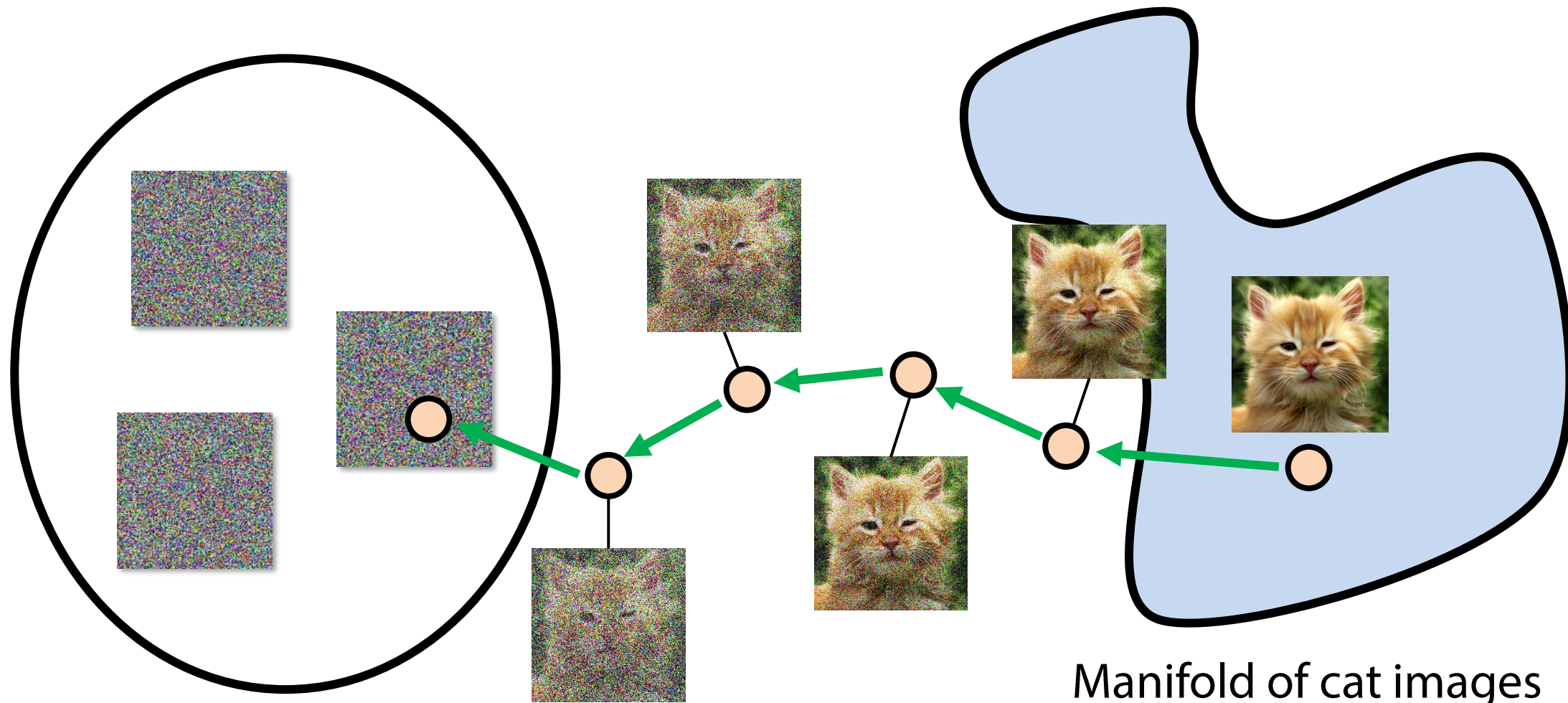
Manifold of cat images



Random images

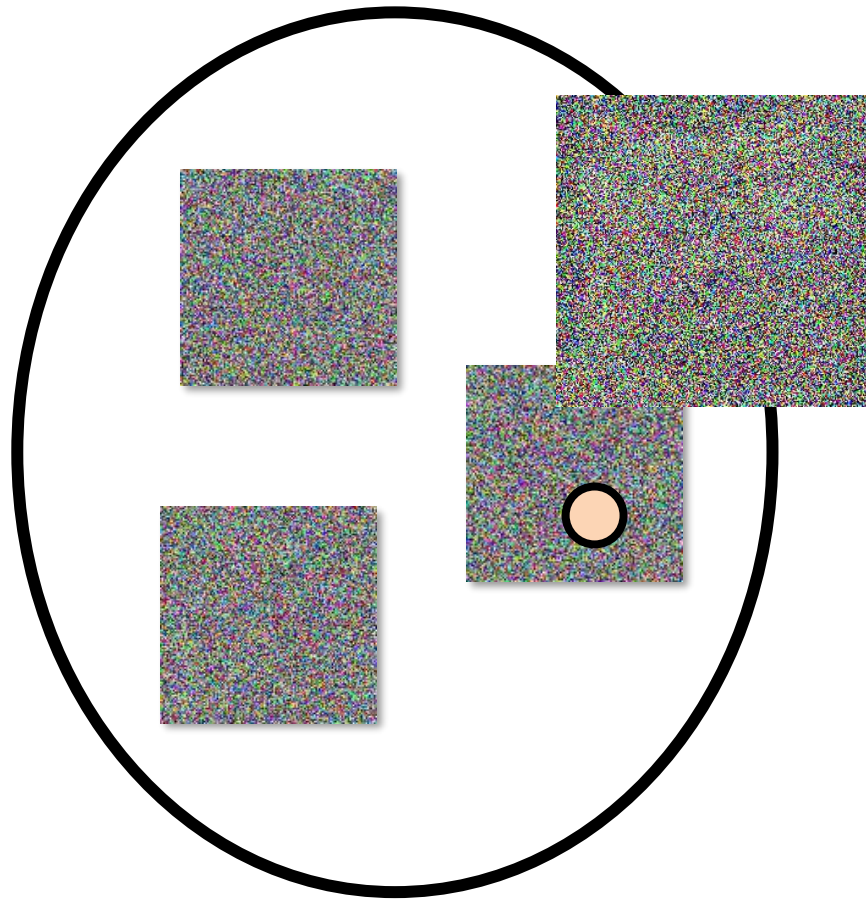


Manifold of cat images

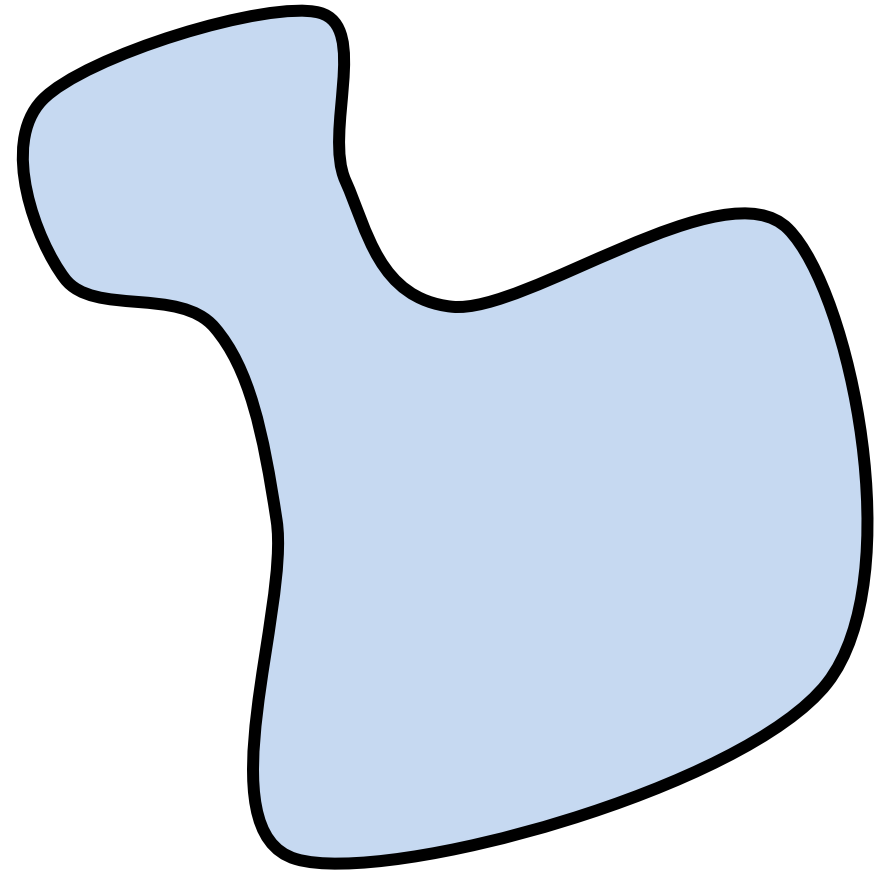


Random images

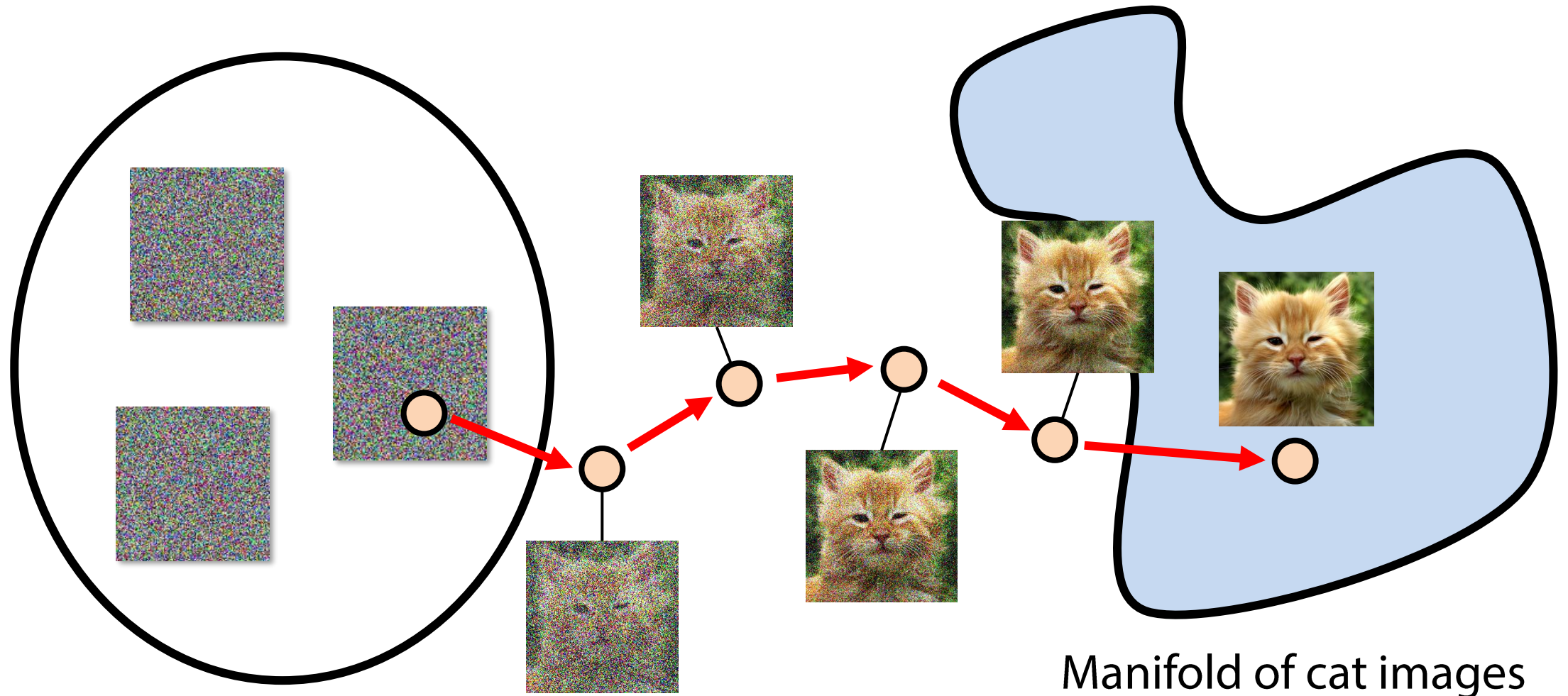
Manifold of cat images



Random images

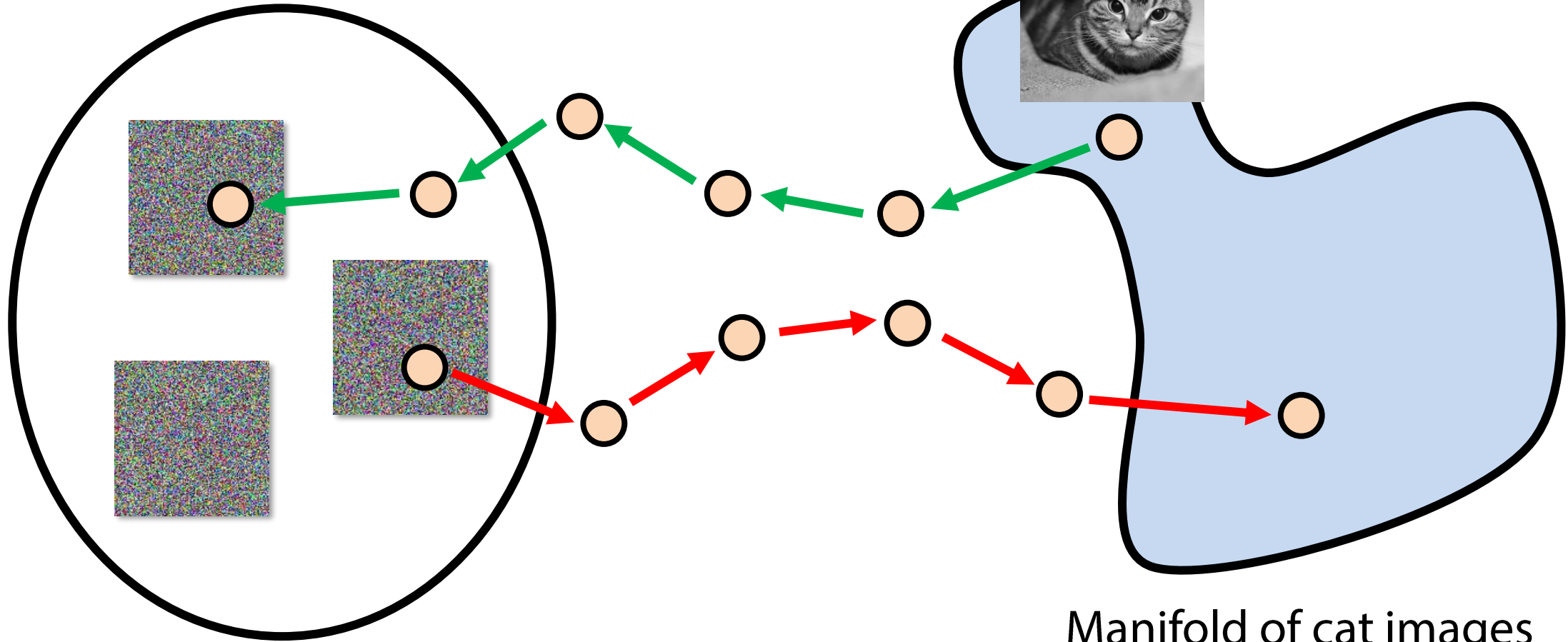


Manifold of cat images



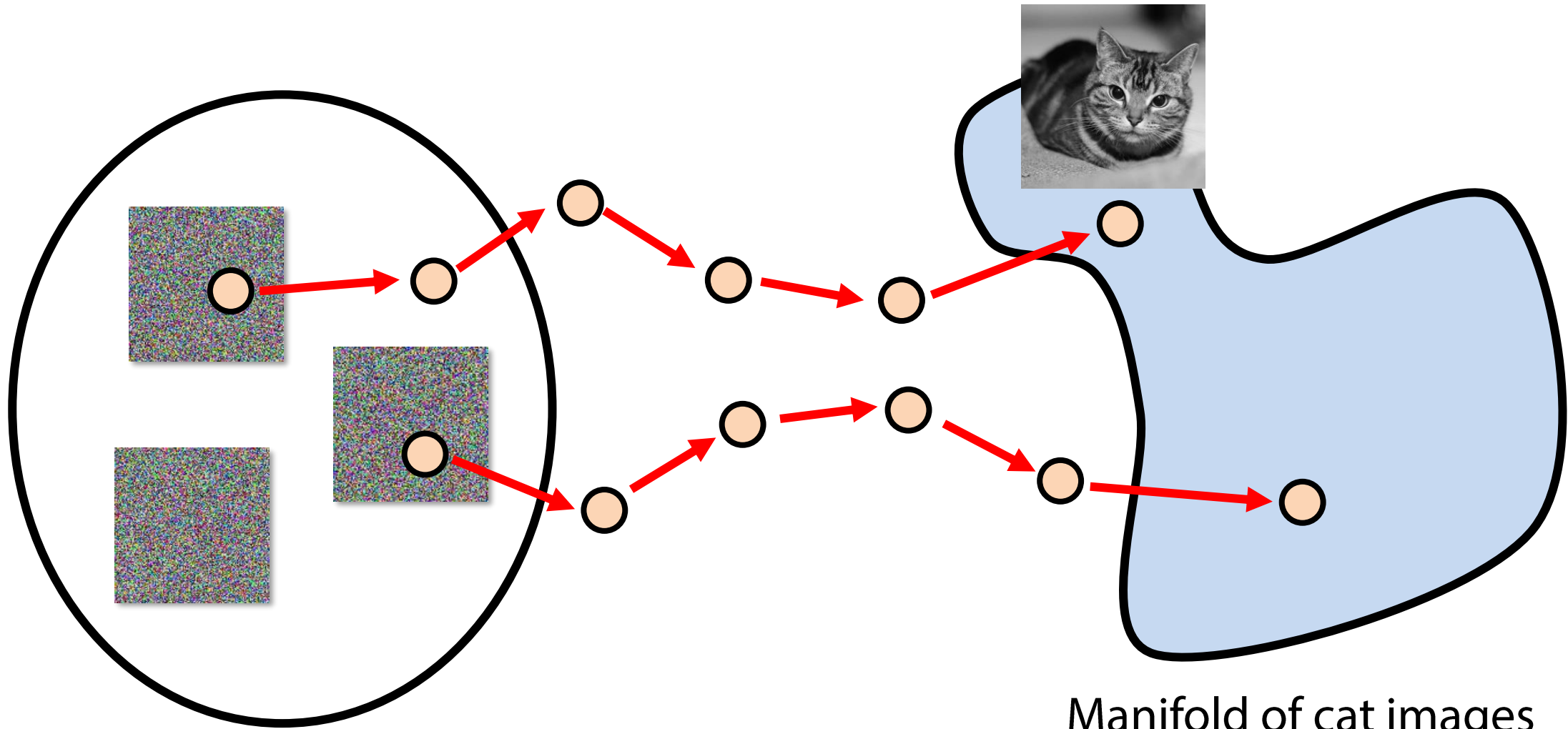
Random images

Manifold of cat images



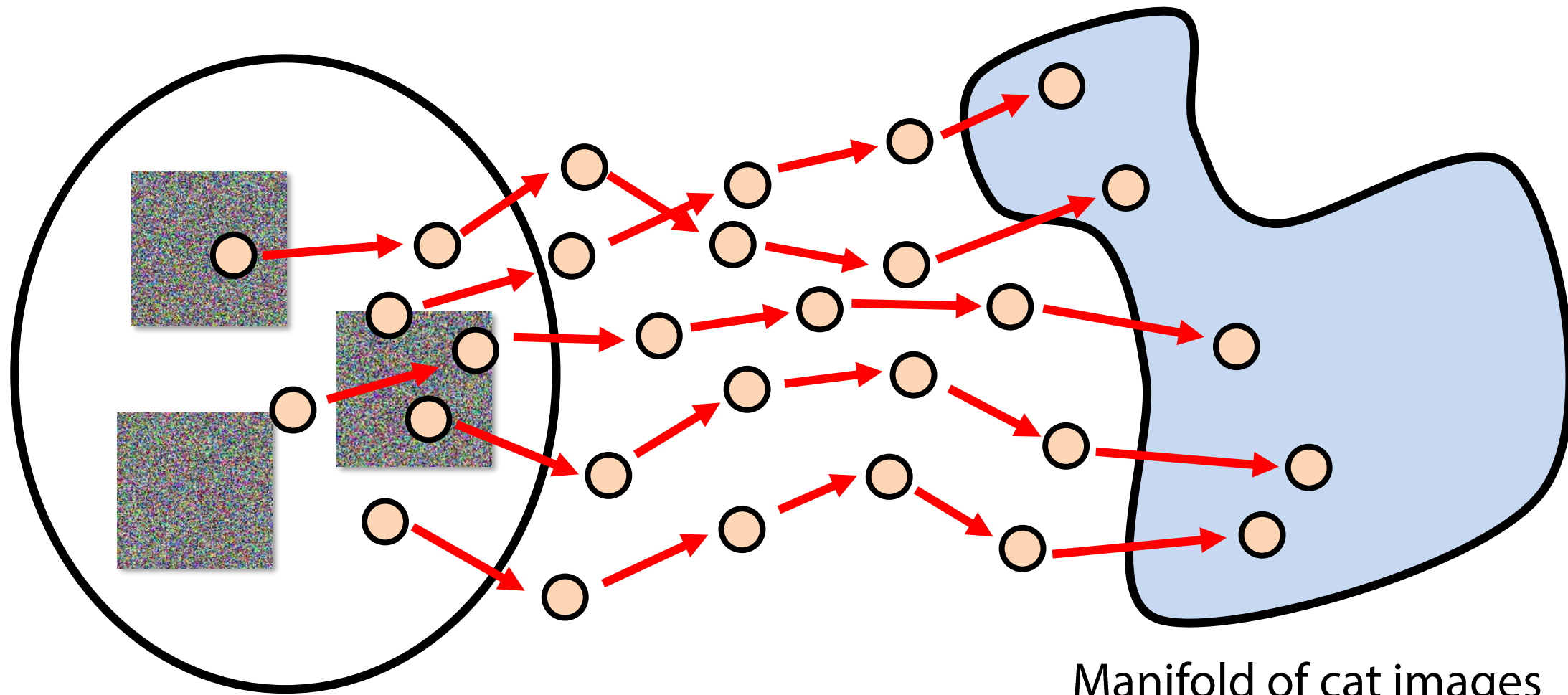
Random images

Manifold of cat images



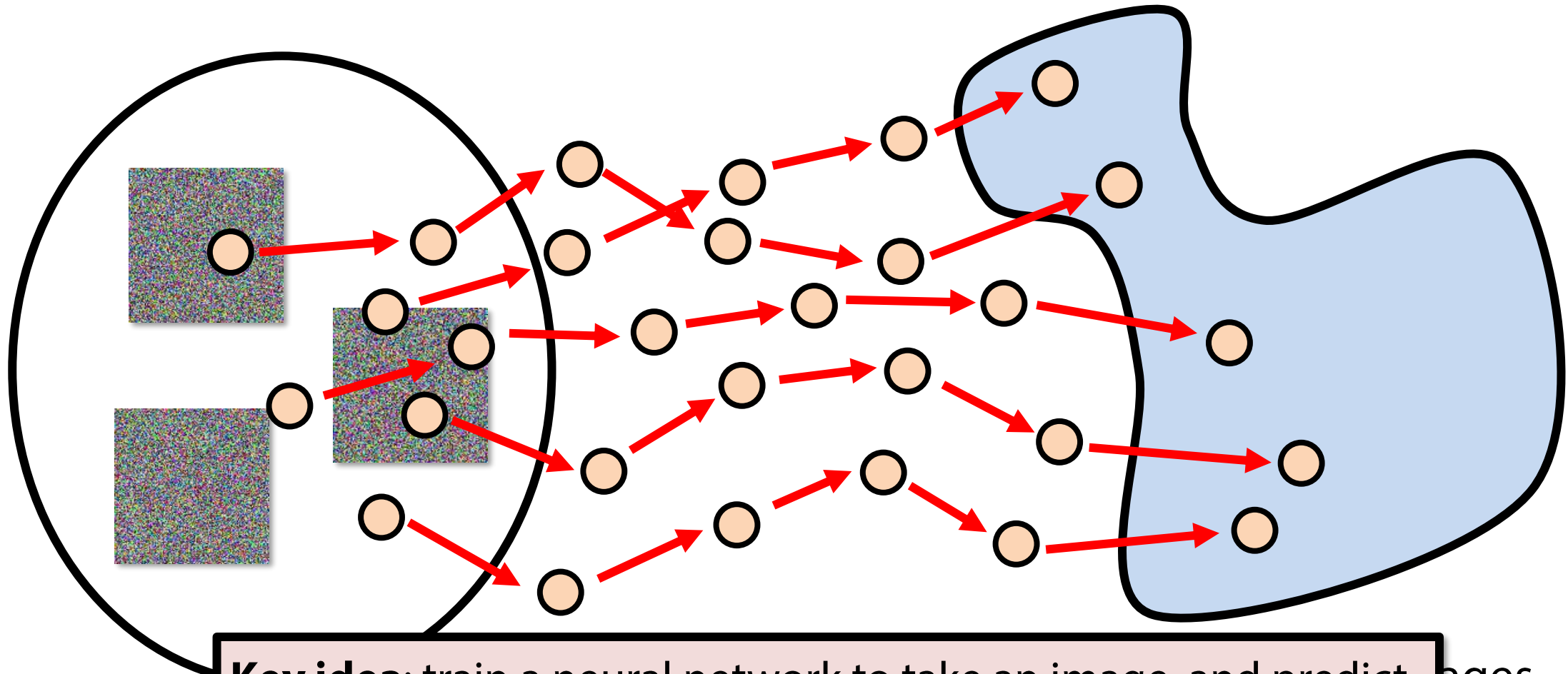
Random images

Manifold of cat images



Random images

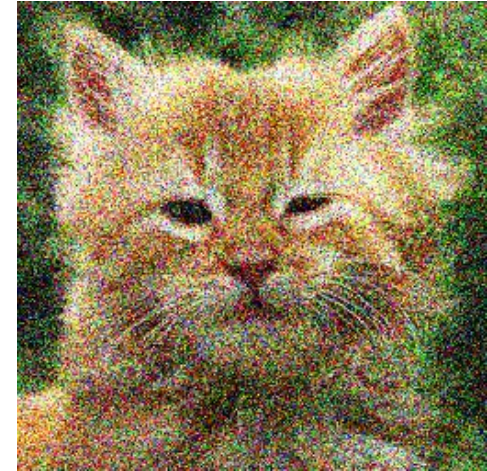
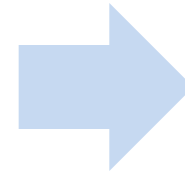
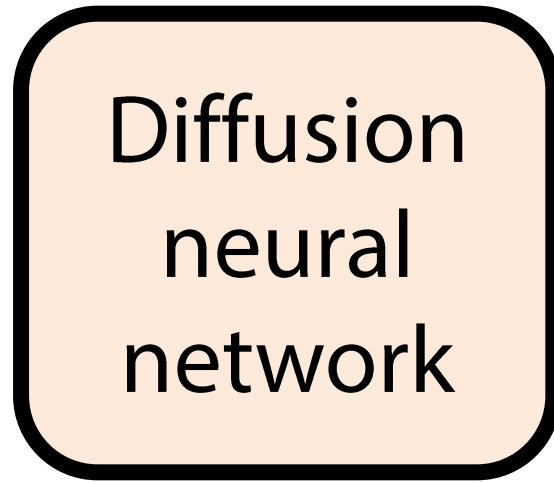
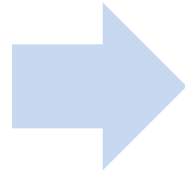
Manifold of cat images



Random

Key idea: train a neural network to take an image, and predict the corresponding arrow above; that is, predict to convert a noisy image to a slightly less noisy image that is closer to the desired image manifold, using the examples above to train.

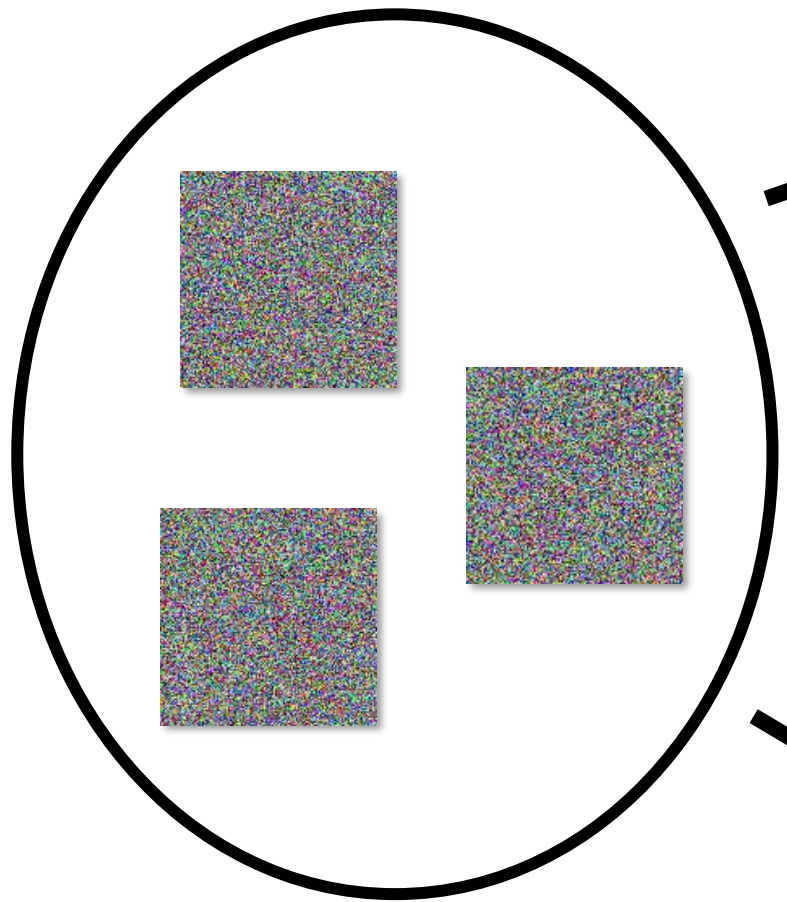
Denoising diffusion neural network



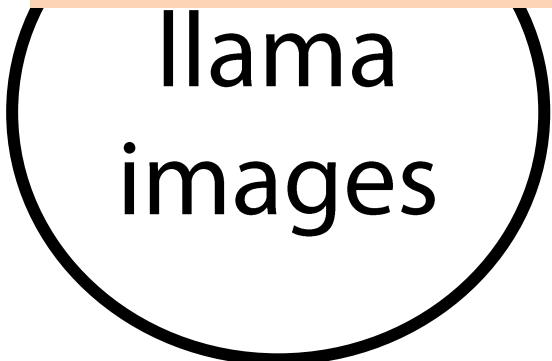
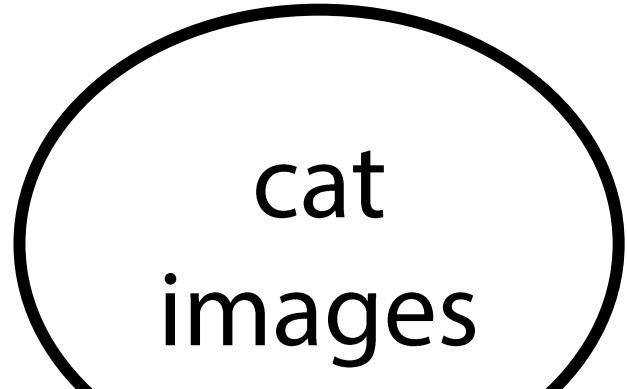
This network can be a U-Net or other suitable image-to-image network

Generating new images

- Once diffusion network has been trained, generate new images by starting with a random noise image, and iteratively applying the network to slowly remove noise, for some number of steps (e.g., 1,000 for DALL-E 2)
- “Walking from random images towards the manifold of natural images”

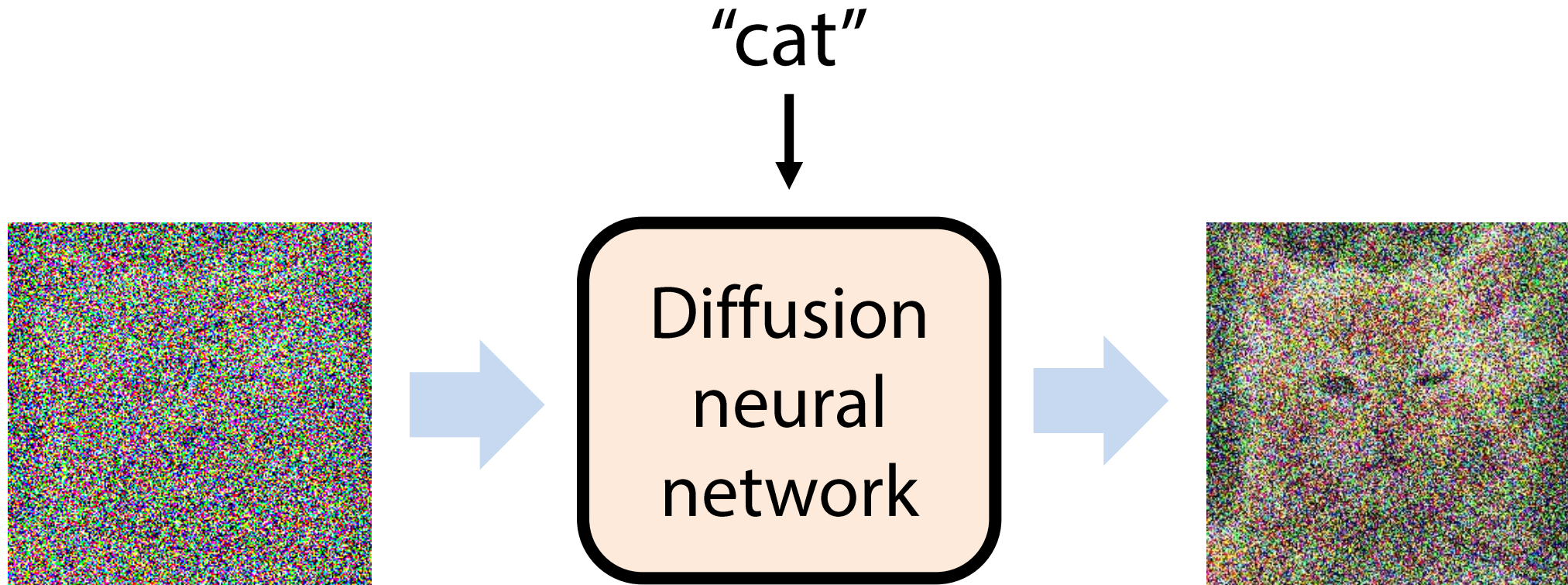


Random images

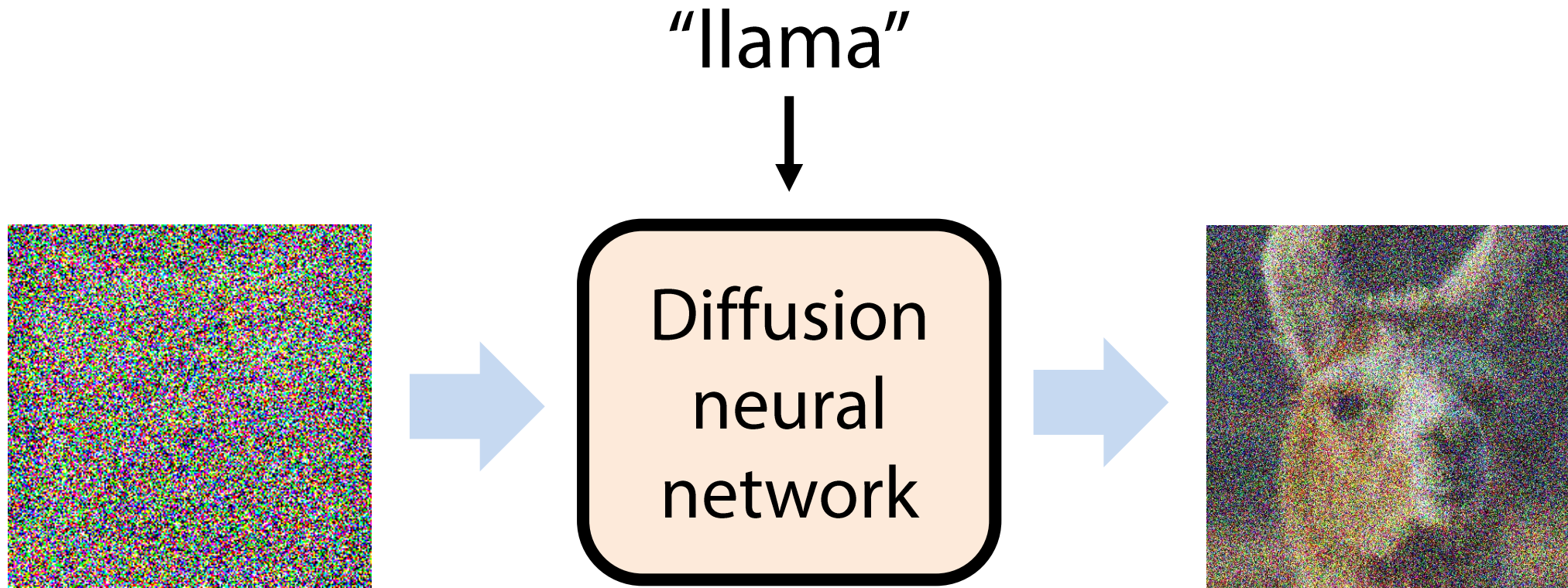


How can we avoid training a separate diffusion network for each concept?

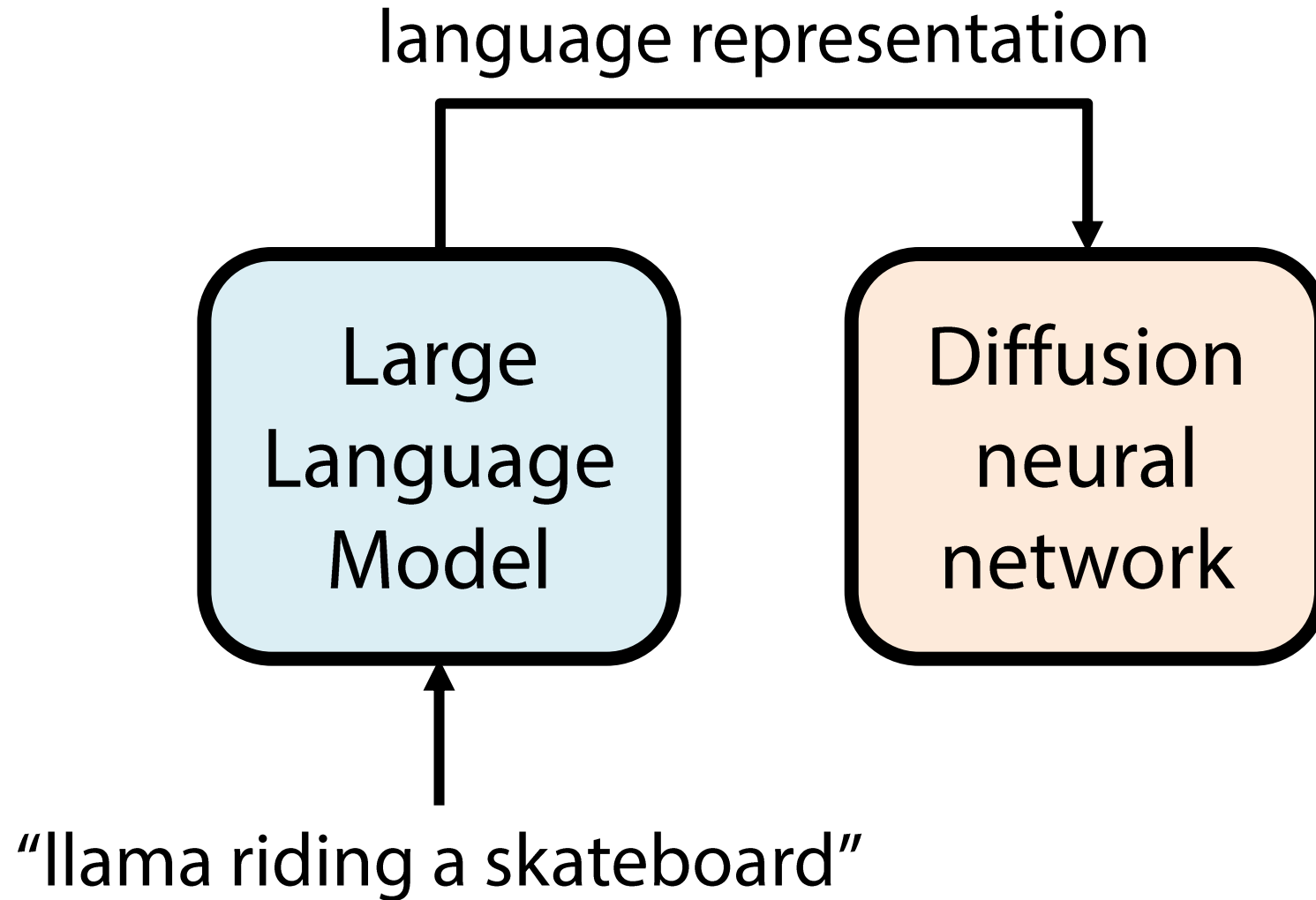
Idea 1: add a text label as conditioning



Idea 1: add a text label as conditioning



Idea 2: condition using large language model



Training on images + captions



A pack llama in the Rocky Mountain National Park 

<https://en.wikipedia.org/wiki/Llama>

DALL-E 2



“A llama riding a skateboard”



“A llama riding a skateboard captured with a DSLR”

Imagen



“Sprouts in the shape of text 'Imagen' coming out of a fairytale book.”



“A dragon fruit wearing karate belt in the snow.”

Other applications of diffusion models

- Uncropping



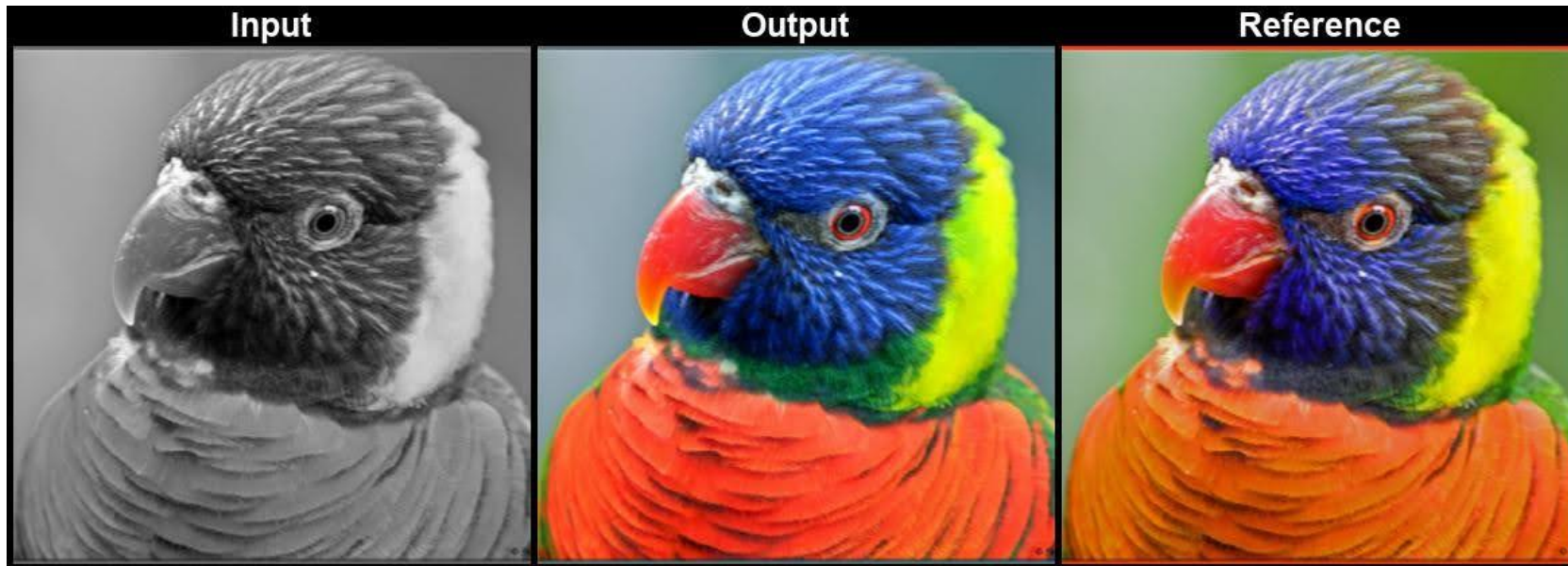
Progressively zooming out. The most zoomed-in image is the input

[Palette: Image-to-Image Diffusion Models](#)

Saharia et al. arXiv 2022.

Other applications of diffusion models

- Colorization

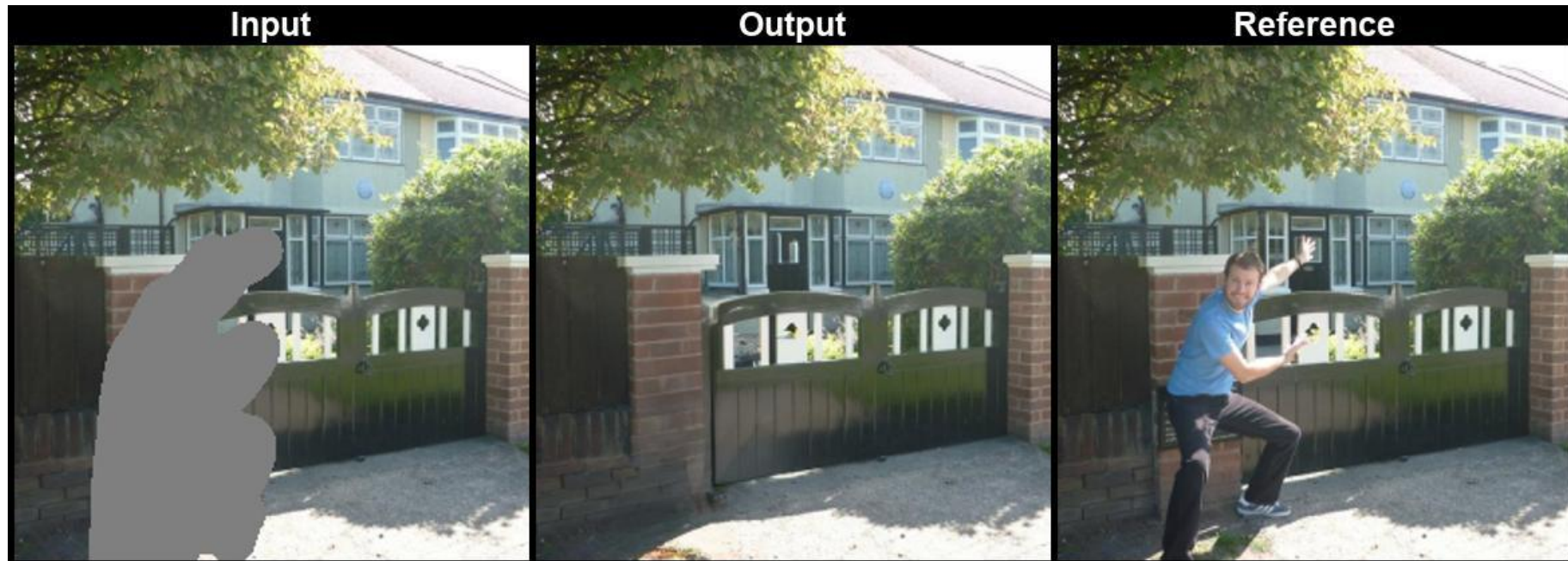


[Palette: Image-to-Image Diffusion Models](#)

Saharia et al. arXiv 2022.

Other applications of diffusion models

- Inpainting



[Palette: Image-to-Image Diffusion Models](#)

Saharia et al. arXiv 2022.

DreamFusion: Text-to-3D using 2D Diffusion



“a DSLR photo of a squirrel”

<https://dreamfusion3d.github.io/>

DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation

[Nataniel Ruiz](#) [Yuanzhen Li](#) [Varun Jampani](#) [Yael Pritch](#) [Michael Rubinstein](#) [Kfir Aberman](#)

Google Research



Input images



in the Acropolis

swimming

sleeping

in a doghouse

in a bucket

getting a haircut

It's like a photo booth, but once the subject is captured, it can be synthesized wherever your dreams take you...

[\[Paper\]](#) (new!) [\[Dataset\]](#) [\[BibTeX\]](#)

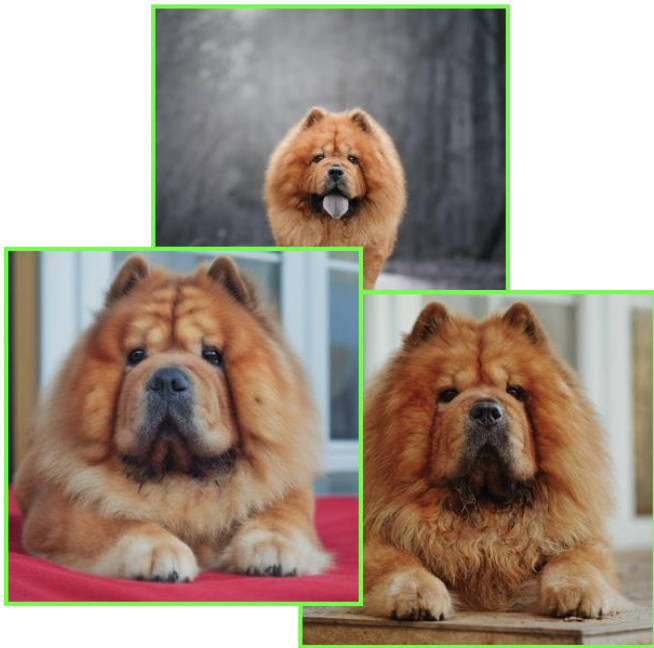
Personalized Residuals for Concept-Driven Text-to-Image Generation

Cusuh Ham, Matthew Fisher, James Hays,
Nicholas Kolkin, Yuchen Liu, Richard Zhang, Tobias Hinz

CVPR 2024

Motivation

Input images



Chef Outfit



Witch Outfit



Ironman Outfit



Nurse Outfit



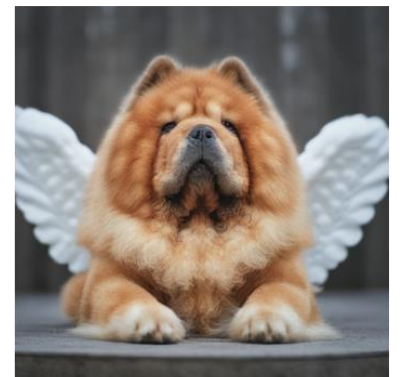
Purple Wizard Outfit



Superman Outfit

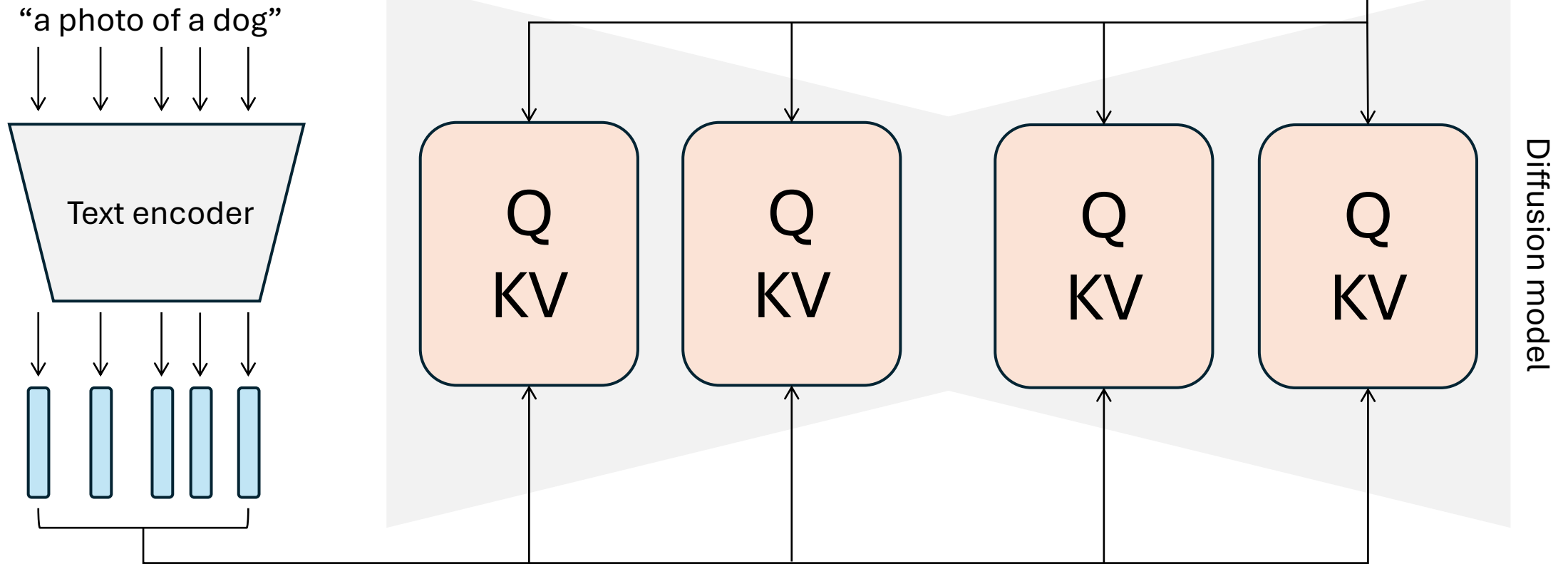
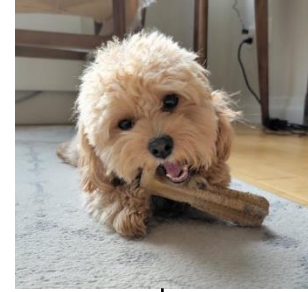


Police Outfit

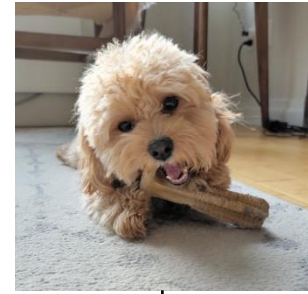


Angel Wings

Background: diffusion model

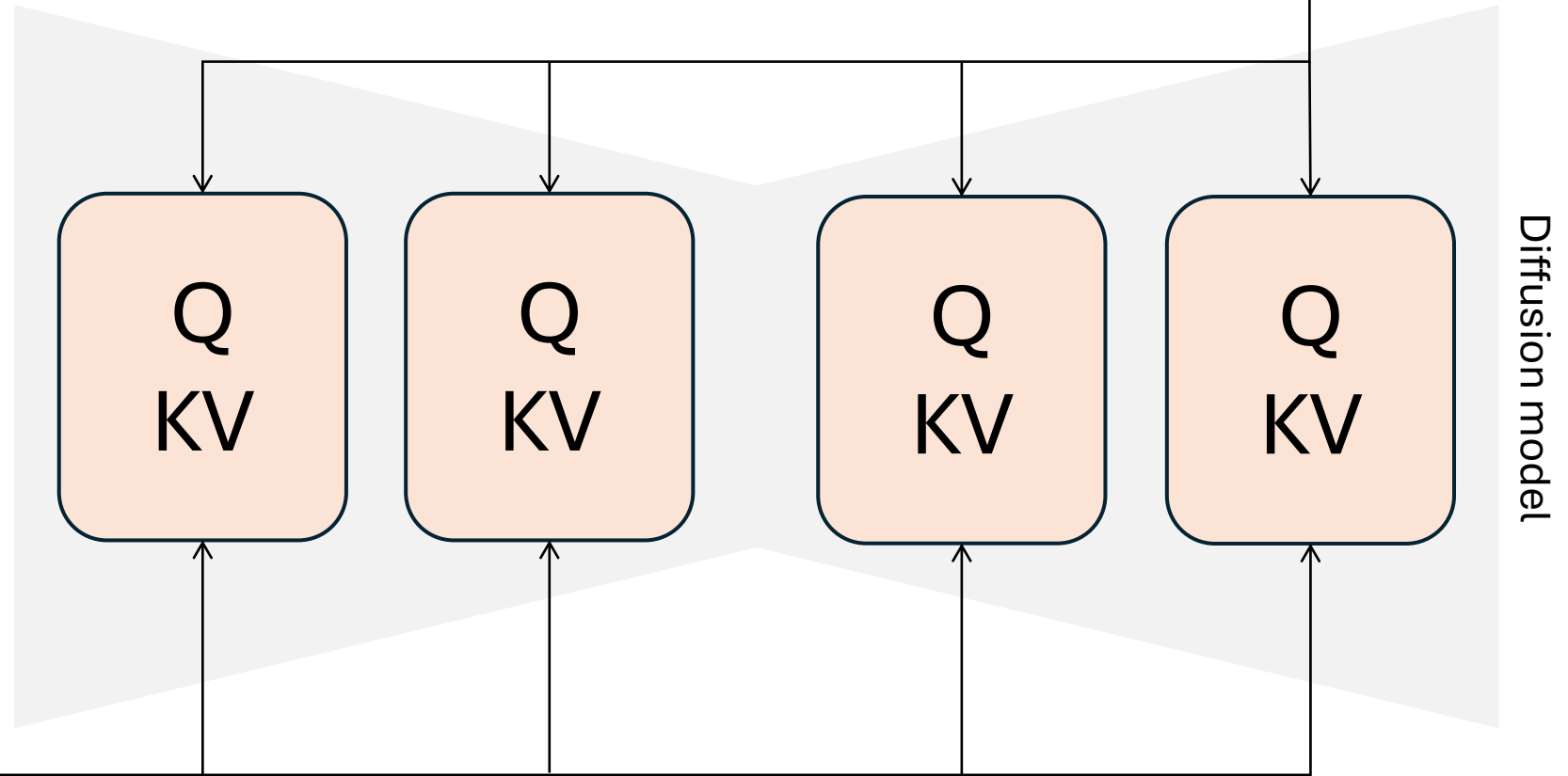
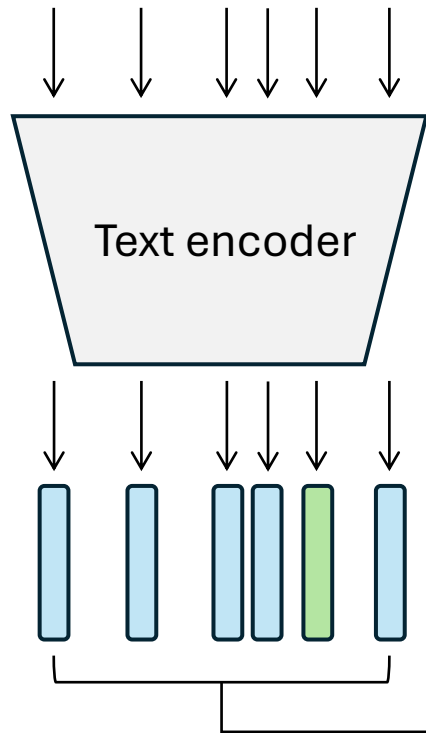


Personalization approaches



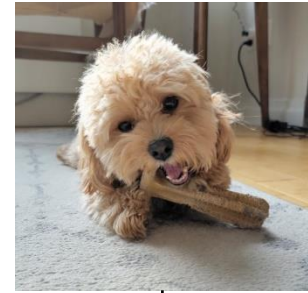
V^*

“a photo of a V^* dog”

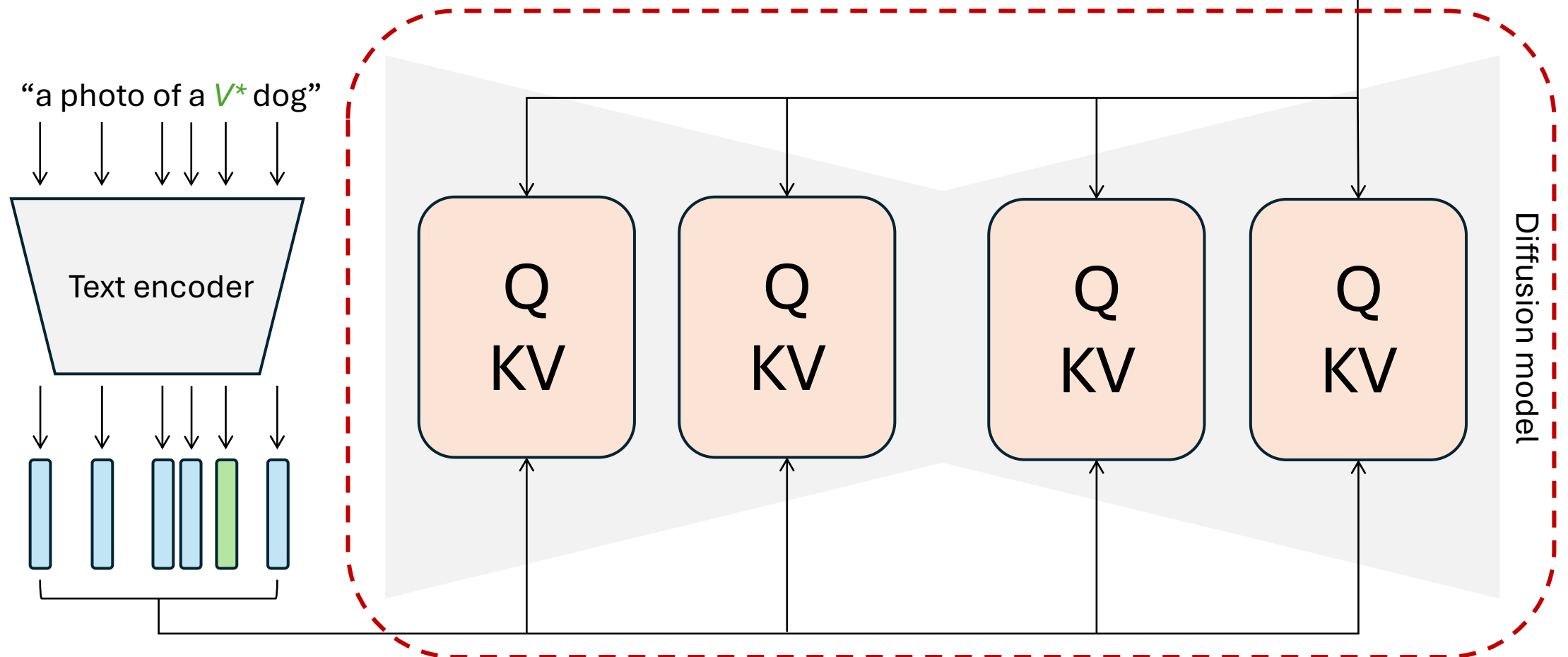


DreamBooth

- ⊘ Large # parameters
- ⊘ Requires regularization images to preserve learned prior

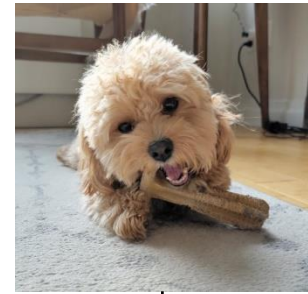


V^*



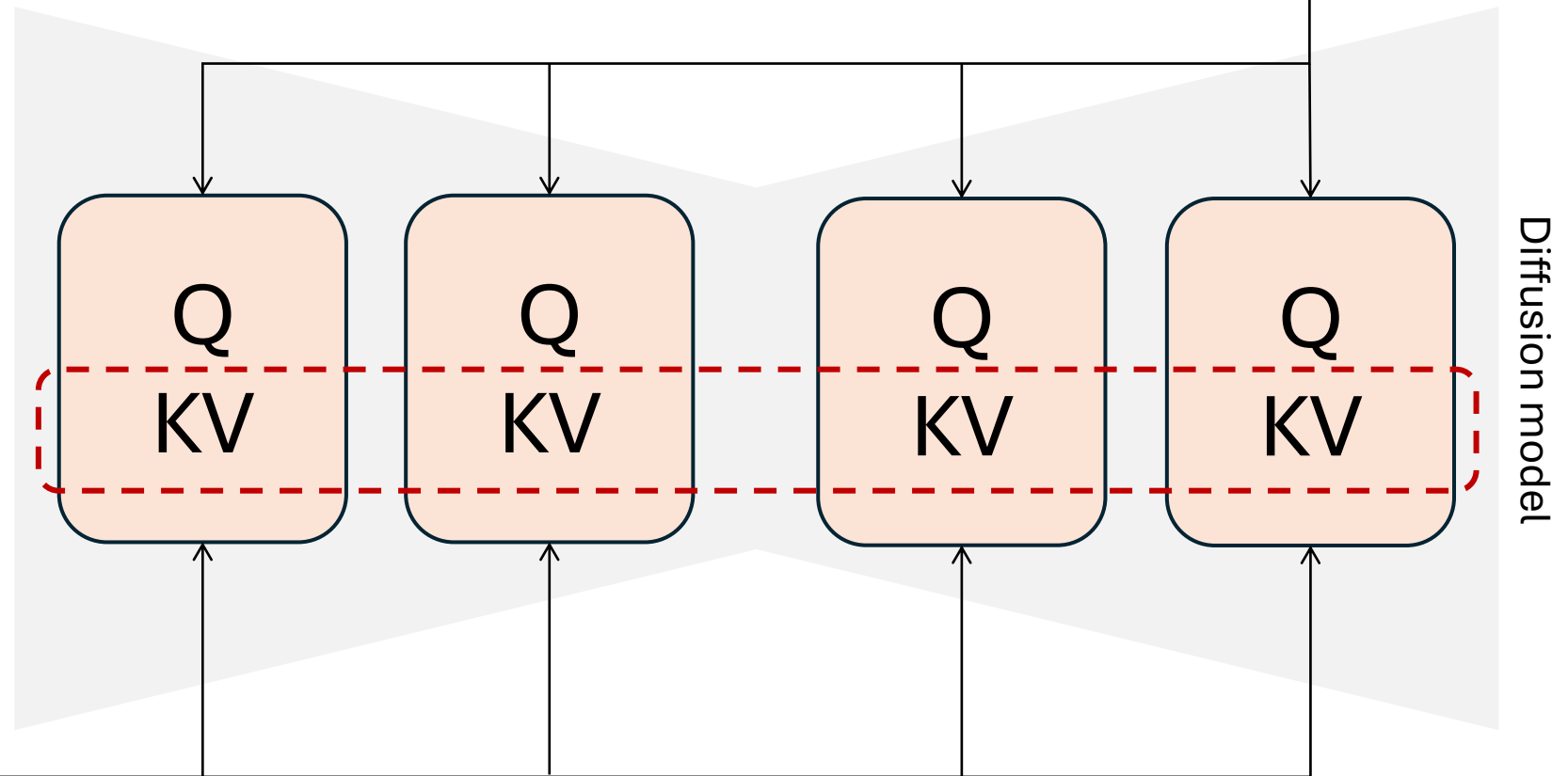
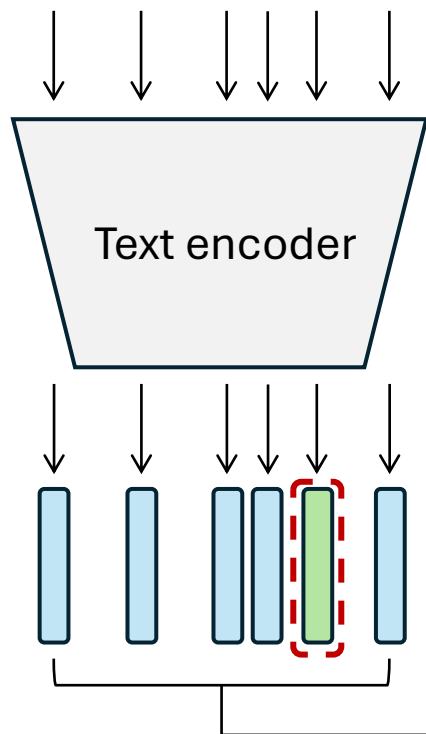
Custom Diffusion

- 😊 Fewer parameters
- 🚫 Requires regularization images



V^*

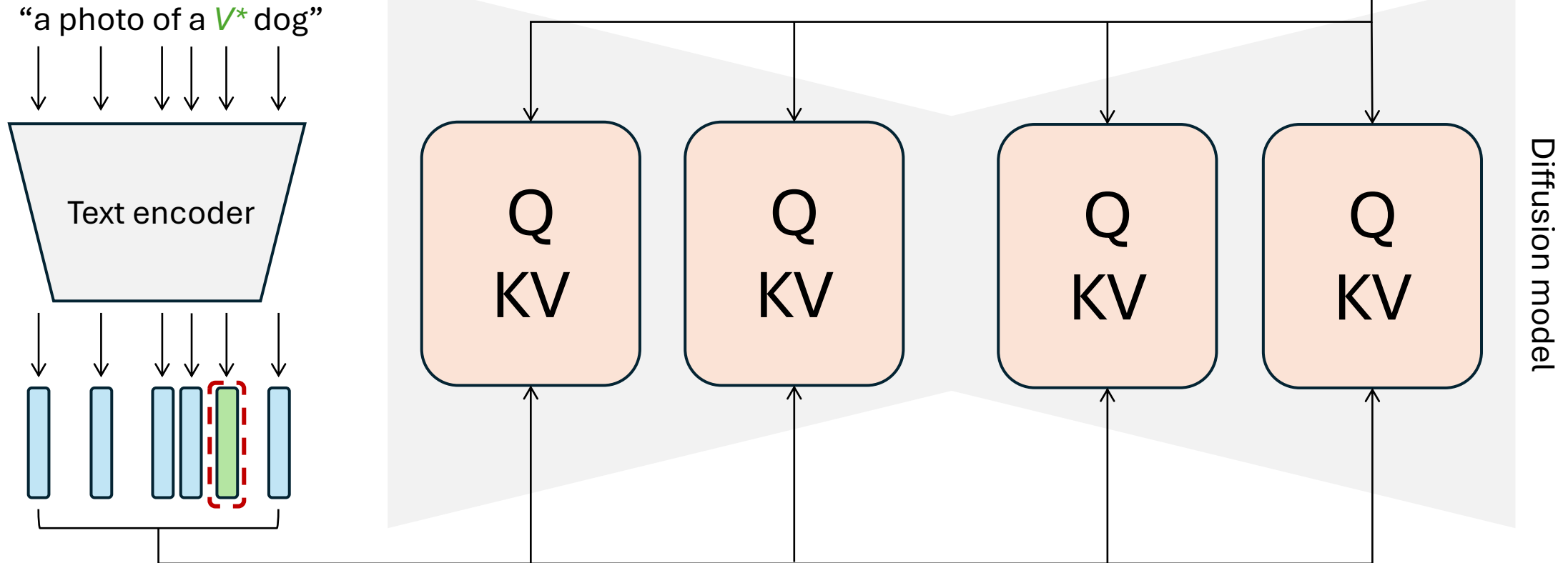
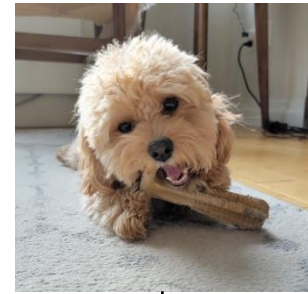
“a photo of a V^* dog”



Diffusion model

Textual Inversion

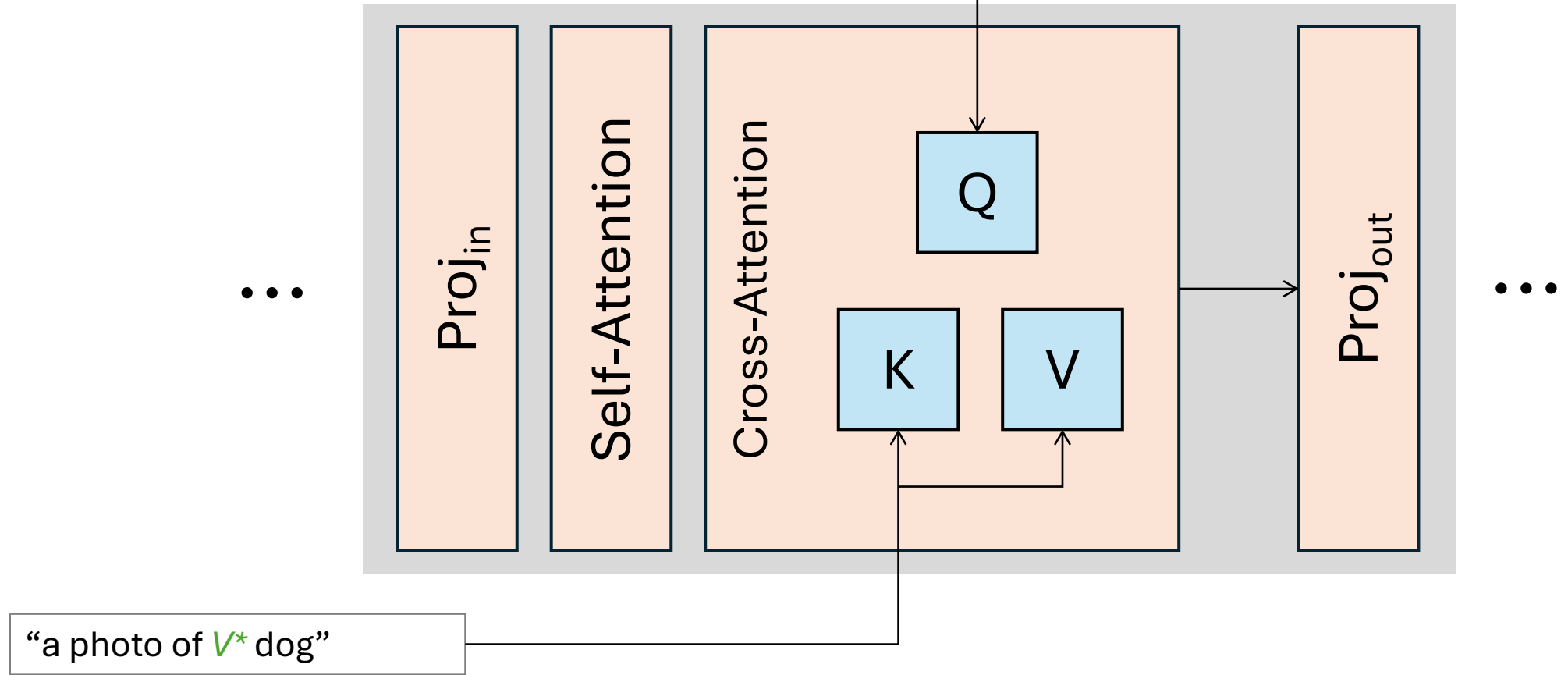
- 😊 Very few parameters
- 😊 Doesn't affect generative prior
- 🚫 Inflexible editing



Transformer blocks



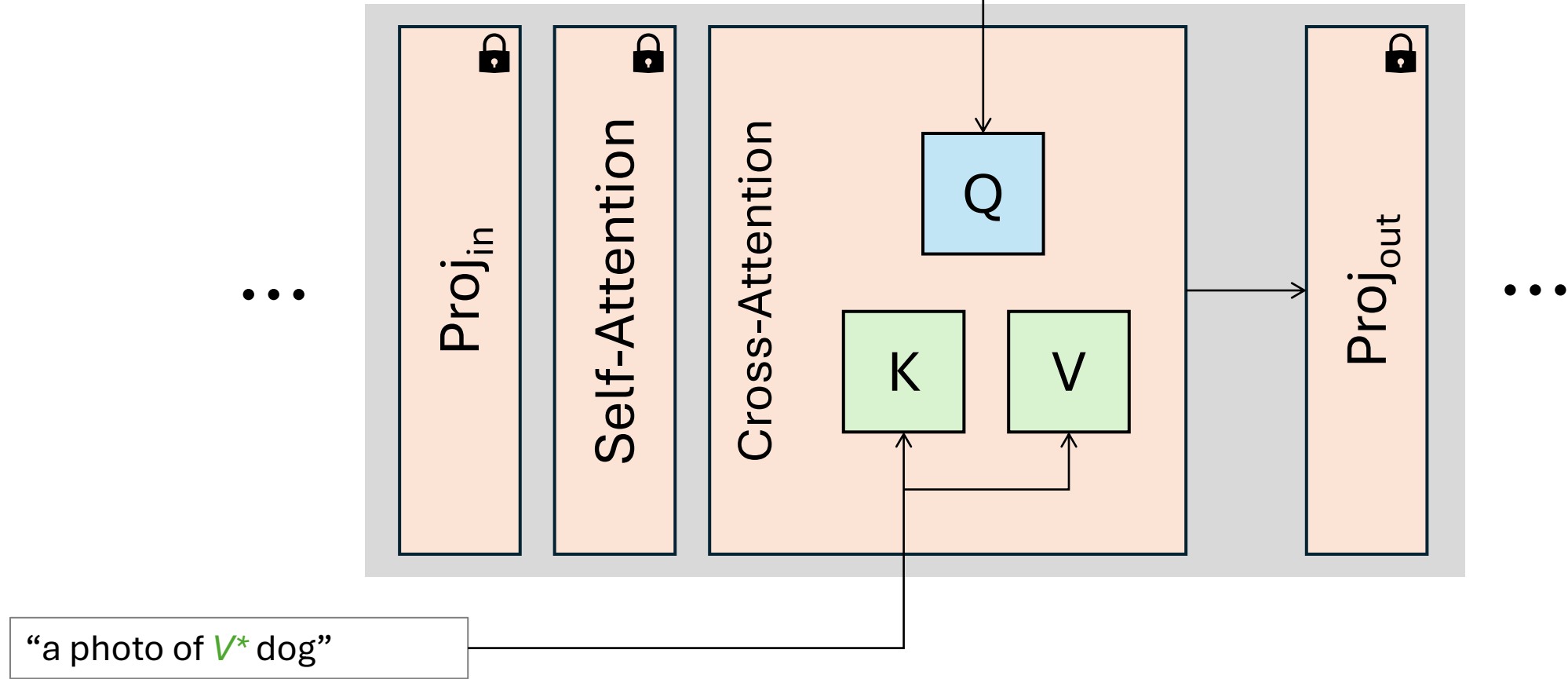
V^*



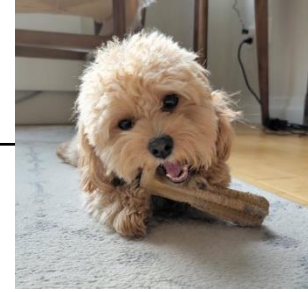
Transformer blocks



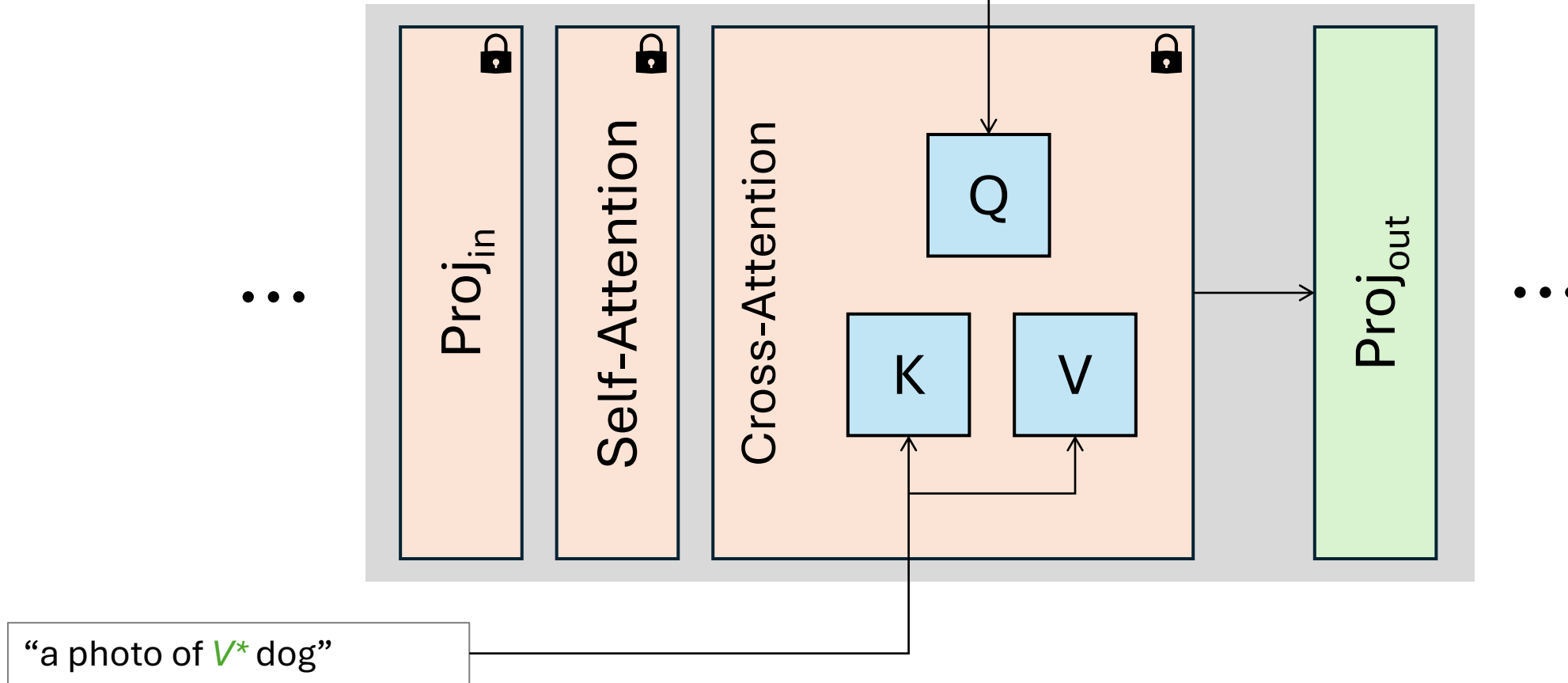
V^*



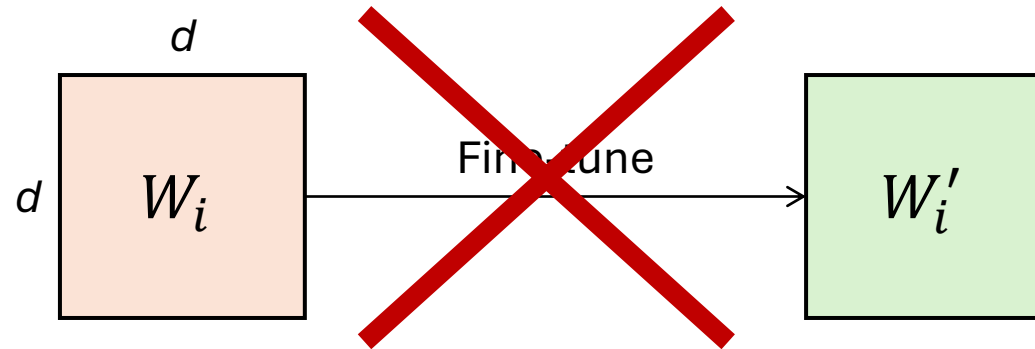
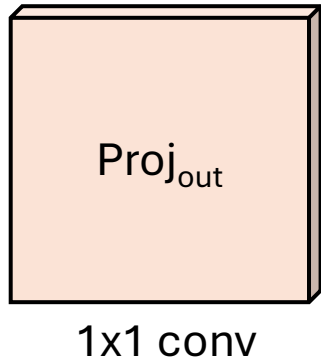
Transformer blocks



V^*

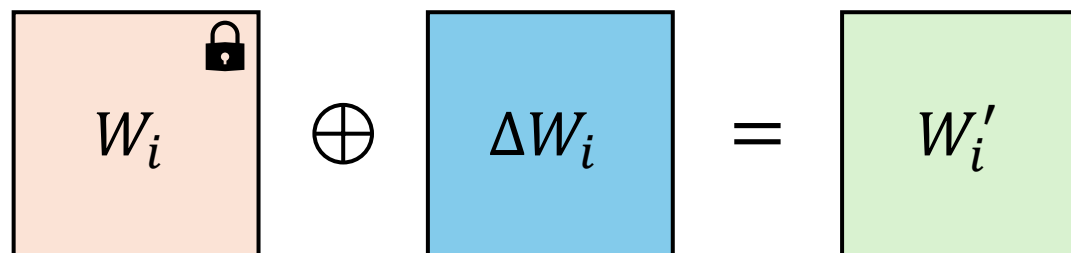
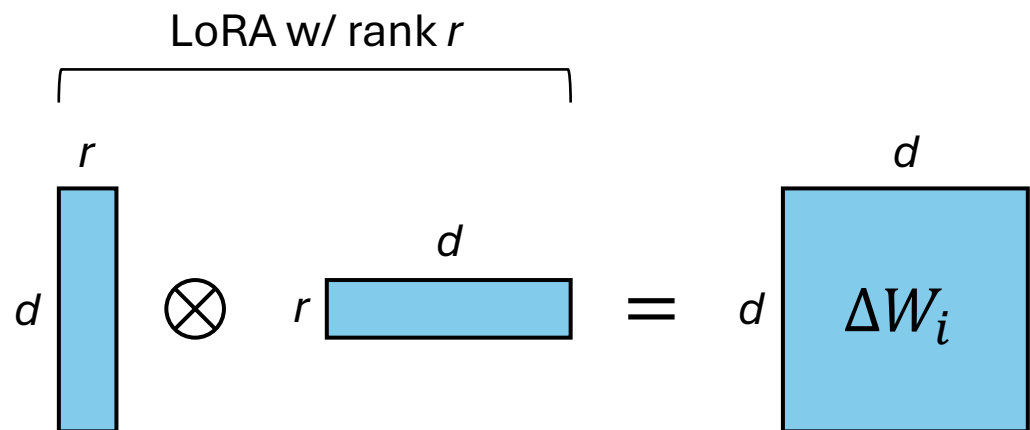


Our approach: personalized residuals



⊘ Overwrites learned prior

Our approach



Personalized residual

Method	Regularization images?	# parameters
Textual inversion	✗	768
DreamBooth	✓	983M
Custom Diffusion	✓	19M
Ours	✗	1.2M

150 iterations
~3 min on 1 A100

Concept



Ours



Textual Inversion



DreamBooth



Custom Diffusion



"V backpack on a café table with a steaming cup of coffee nearby"*



"A pink V chair"*



"Georgia O'Keeffe style V dog painting"*

Personalized Residuals



Concept



"A rusty V toy gnome in a post-apocalyptic landscape"*



Concept



"V plushie oil painting Ghibli inspired"*



Concept



"V cat wearing sunglasses"*

Personalized Residuals + LAG Sampling



Concept



"V action figure riding a motorcycle"*



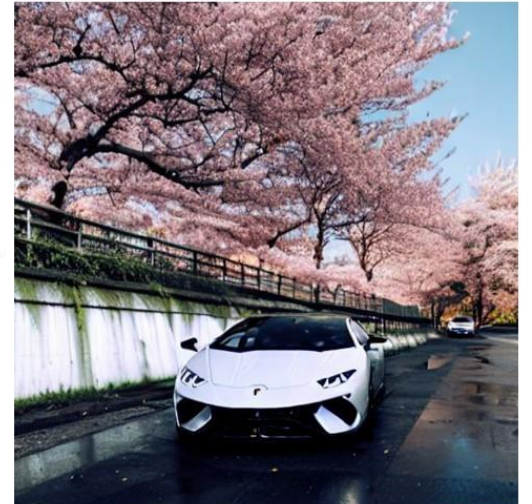
Concept



"The V lighthouse surrounded by a tranquil lake"*



Concept



"A V car resting beneath the cherry blossoms in full bloom"*

Visit the poster tomorrow night 5-6:30pm in Arch 4A-E poster #329

Comparison with GANs

- Diffusion models tend to be easier to train and more scalable
- Diffusion models tend to be slower – often many iterations of denoising are required
- However, recent work is mitigating some of these issues (with both GANs and diffusion models)