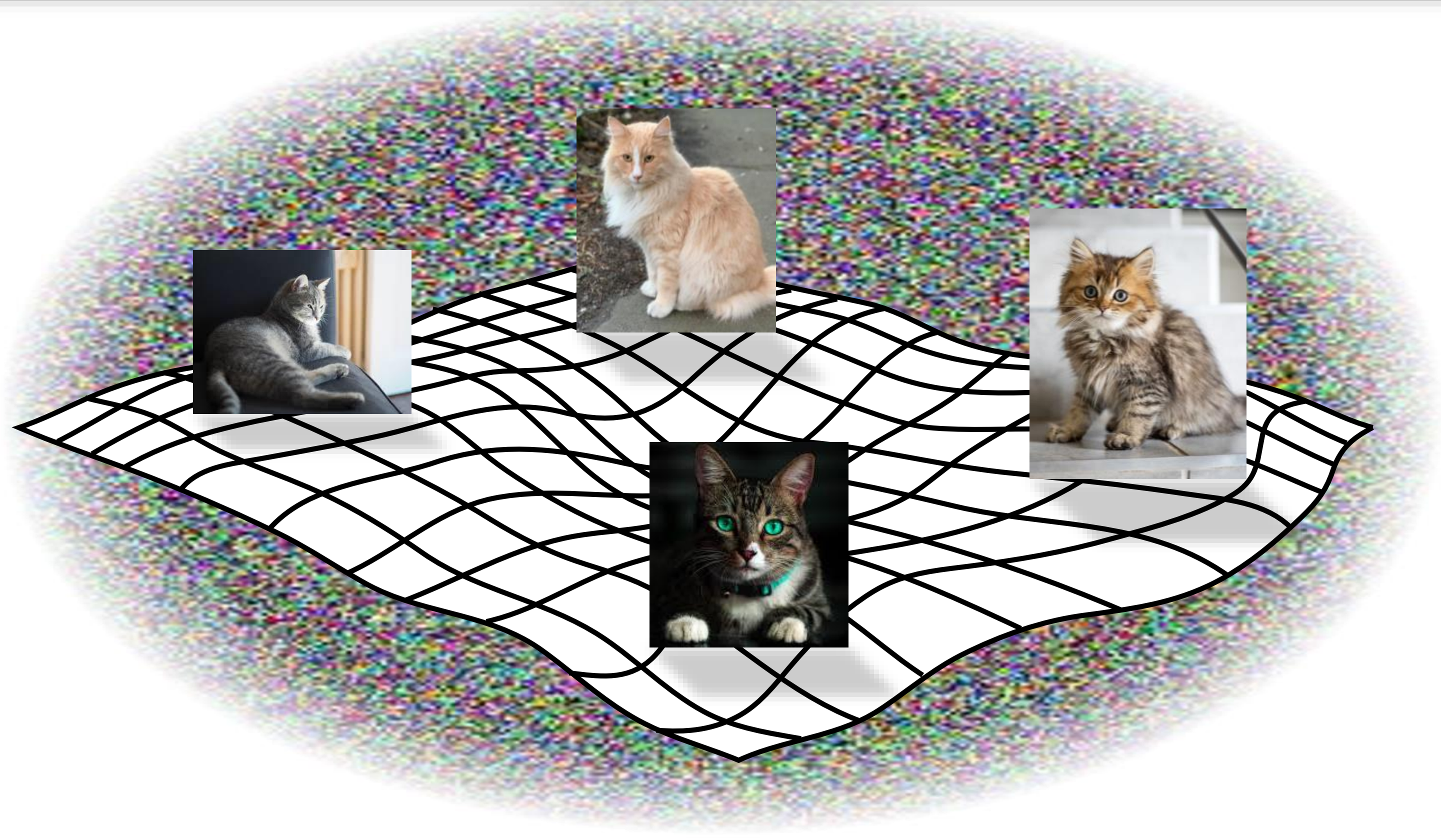
CS5670: Computer Vision

Image Manifolds & Image Synthesis (including GANS)



Most content from Abe Davis, with additional credit to Jin Sun and Phillip Isola

Announcements

- Take-home final May 12-17
- Project 5 (Convolutional Neural Networks) due Tuesday, May 11, 2021 (7:00 pm)

- Course evaluations are open this Friday, May 7 to May 17
  - We would love your feedback!
  - Small amount of extra credit for filling out
    - What you write is still anonymous, instructors only see whether students filled it out
  - Link coming soon

Agenda

- Last time:
  - How to train convolutional neural networks (CNNs)

- This time:
  - One more note on training CNNs for new tasks
  - Dimensionality reduction
  - Neural networks that produce images
  - Generative Adversarial Networks (GANs)

Transfer Learning

“You need a lot of a data if you want to train/use CNNs”

Transfer Learning

"You need a lot of a data if you want to train/use CNNs"

BUSTED

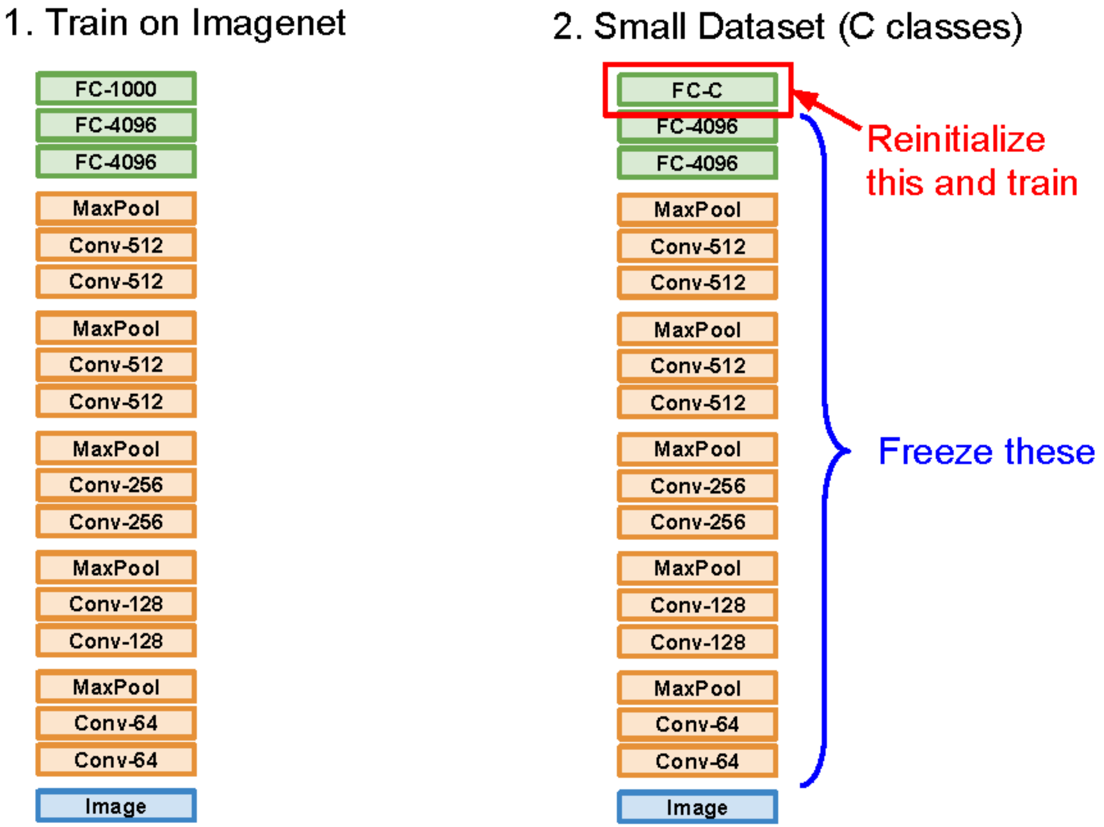
Transfer Learning with CNNs

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

1. Train on Imagenet

Transfer Learning with CNNs

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014



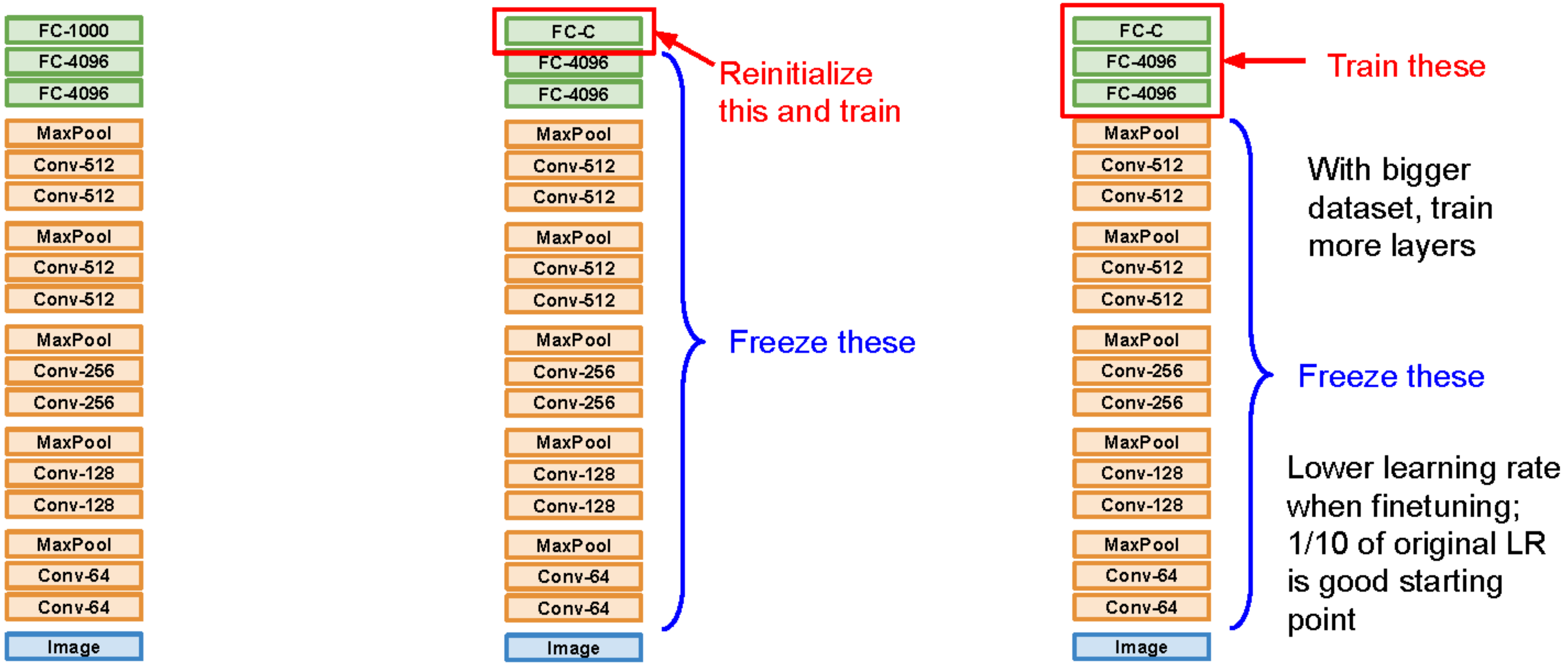
Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014
Razavian et al, "CNN Features Off-the-Shelf: An Astounding Baseline for Recognition", CVPR Workshops 2014

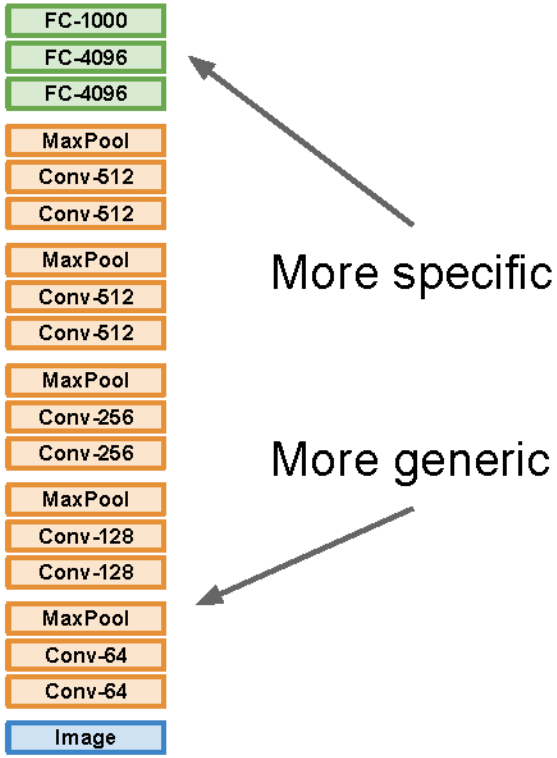
Transfer Learning with CNNs

1. Train on Imagenet

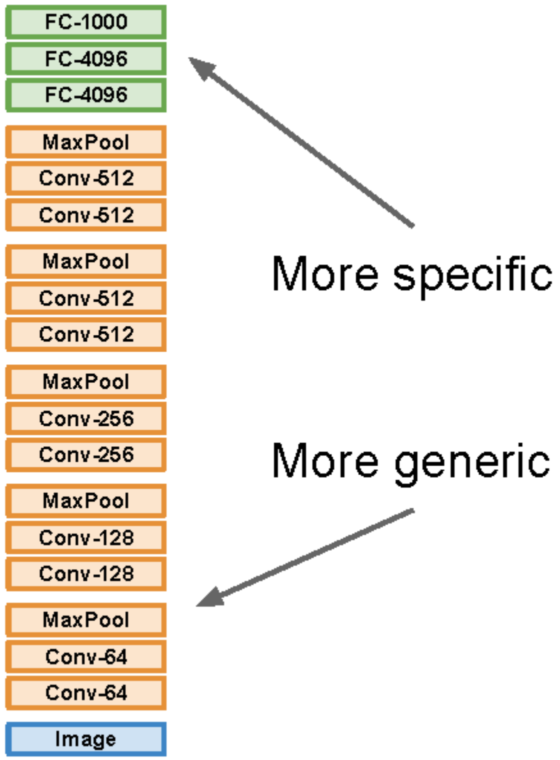
2. Small Dataset (C classes)

3. Bigger dataset

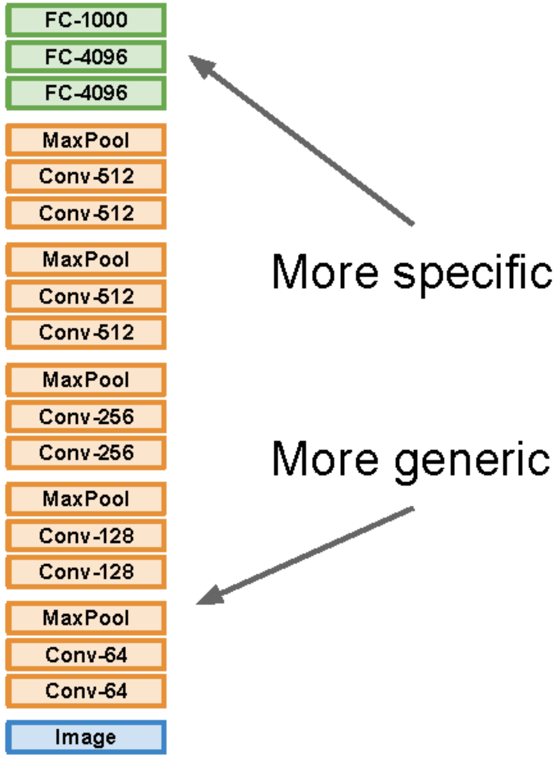




	very similar dataset	very different dataset
very little data	?	?
quite a lot of data	?	?



	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	?
quite a lot of data	Finetune a few layers	?



	very similar dataset	very different dataset
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

Transfer learning with CNNs is pervasive…
(it's the norm, not an exception)

Object Detection
(Fast R-CNN)

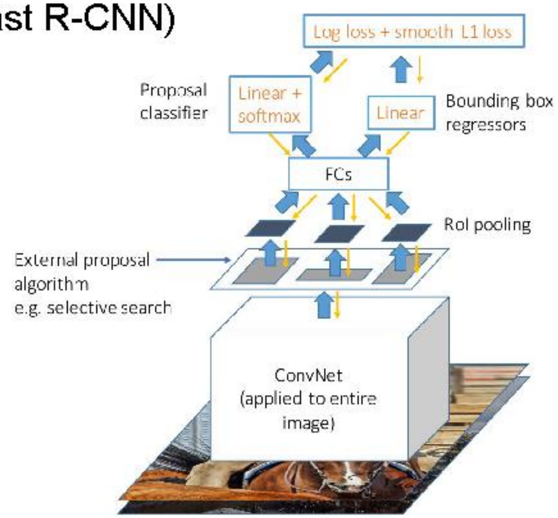
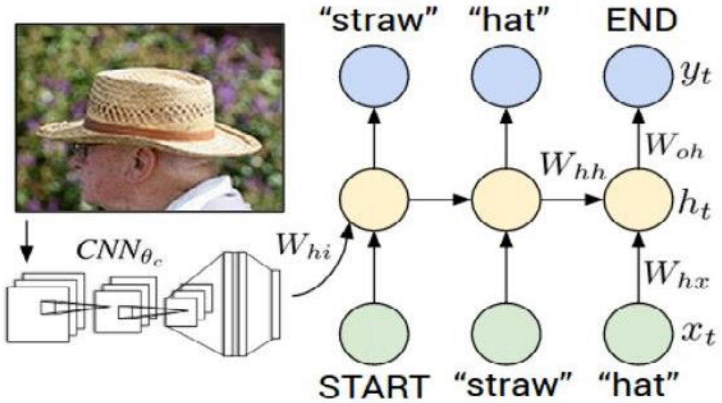


Image Captioning: CNN + RNN

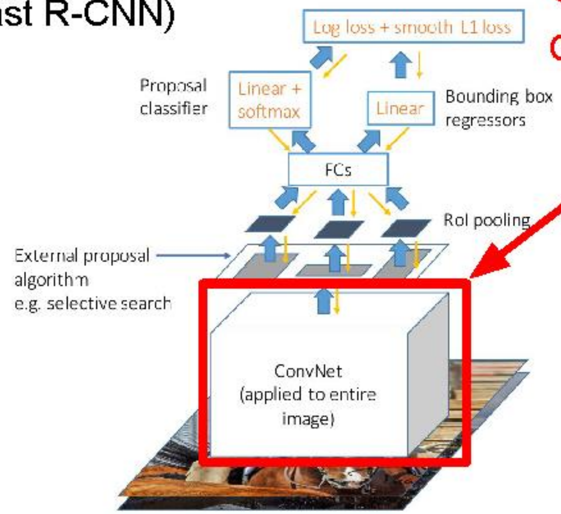


Girshick, "Fast R-CNN", ICCV 2015

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

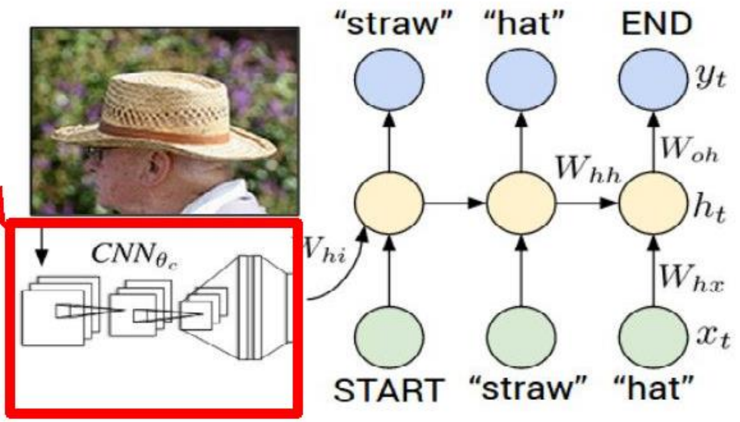
Transfer learning with CNNs is pervasive…
(it’s the norm, not an exception)

Object Detection
(Fast R-CNN)



CNN pretrained on ImageNet

Image Captioning: CNN + RNN



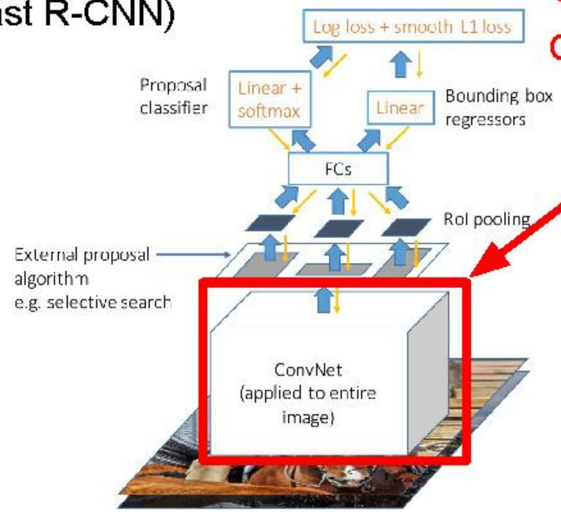
Girshick, "Fast R-CNN", ICCV 2015

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

Transfer learning with CNNs is pervasive…

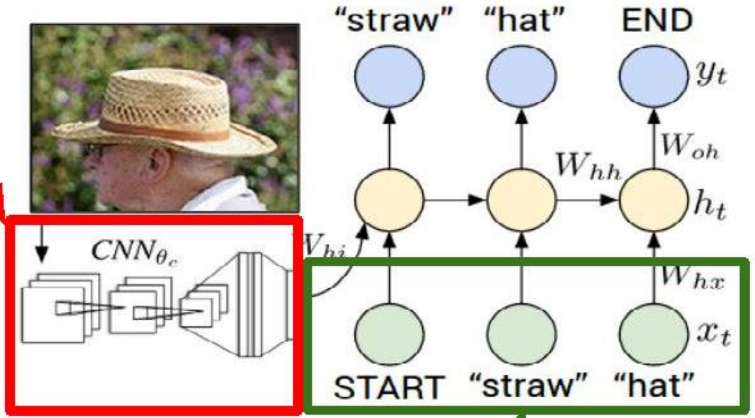
(it's the norm, not an exception)

Object Detection
(Fast R-CNN)



CNN pretrained on ImageNet

Image Captioning: CNN + RNN



Word vectors pretrained with word2vec

Girshick, "Fast R-CNN", ICCV 2015

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015

Takeaway for your projects and beyond:

Have some dataset of interest but it has < ~1M images?

1. Find a very large dataset that has similar data, train a big ConvNet there
2. Transfer learn to your dataset

Deep learning frameworks provide a "Model Zoo" of pretrained models so you don't need to train your own

TensorFlow: https://github.com/tensorflow/models
PyTorch: https://github.com/pytorch/vision

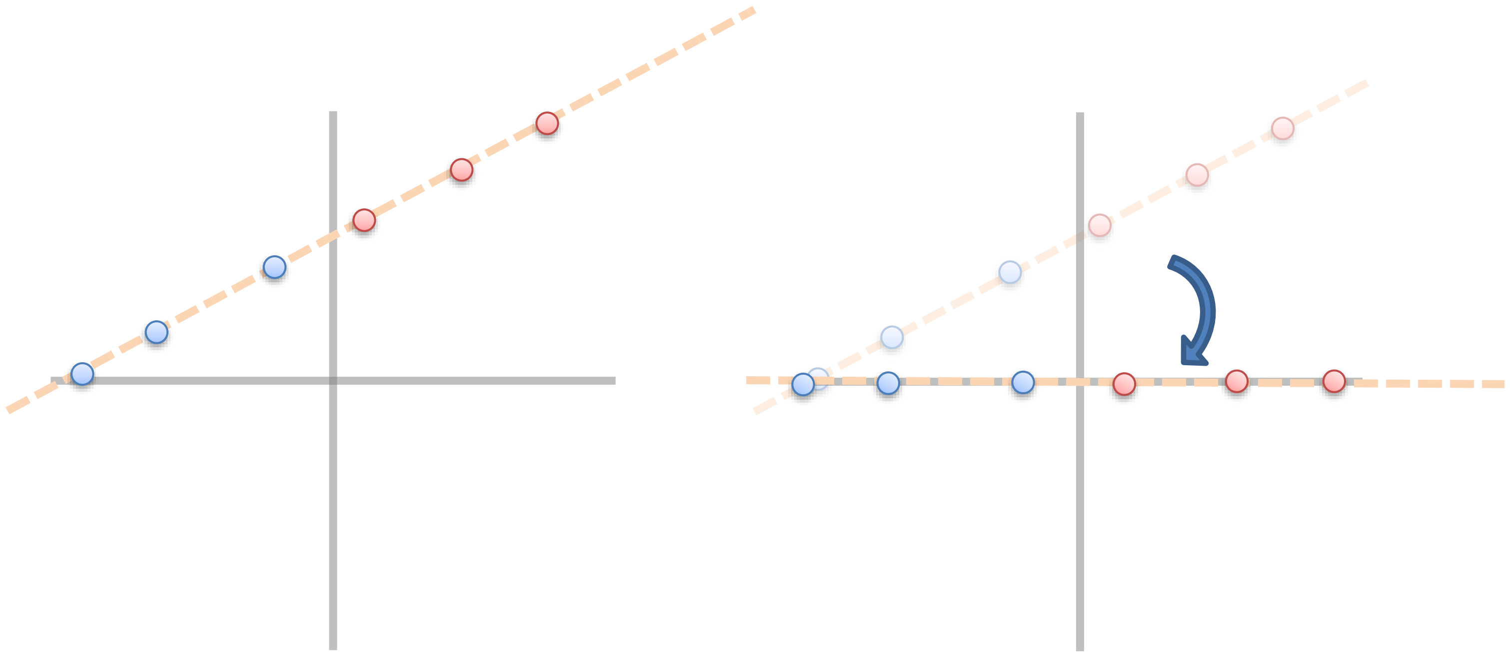
Common modern approach: start with a ResNet architecture pre-trained on ImageNet, and fine-tune on your (smaller) dataset

By Abe Davis

DIMENSIONALITY REDUCTION

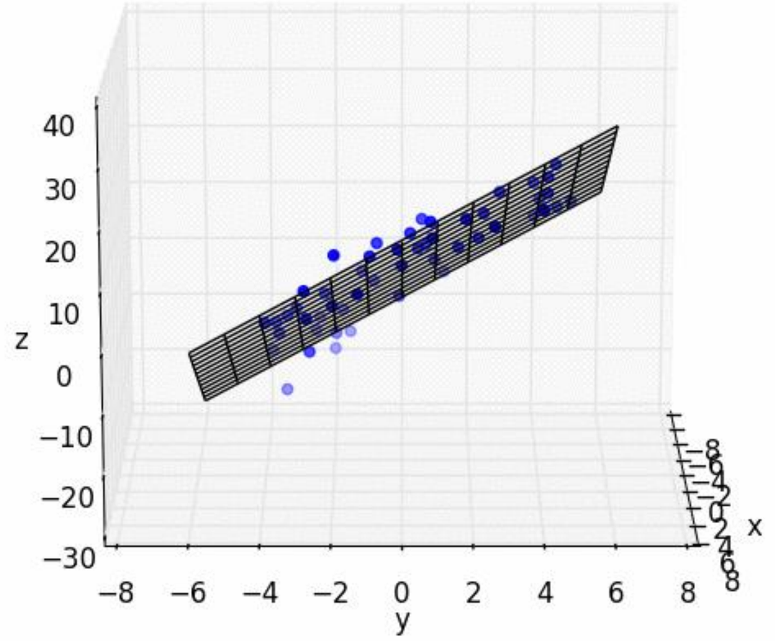
Linear Dimensionality Reduction: 2D->1D

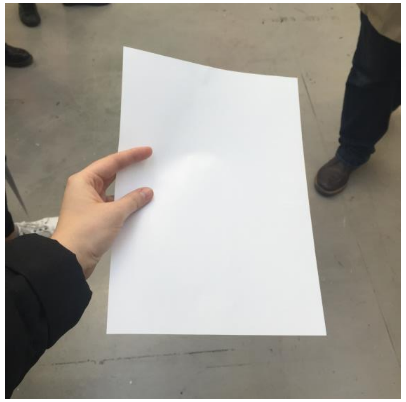
- Consider a bunch of data points in 2D
- Let’s say these points only differ along one line
- If so, we can translate and rotate our data so that it is 1D



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”
- More generally: look for the 2D subspace that best fits the data, and ignore the remaining dimensions

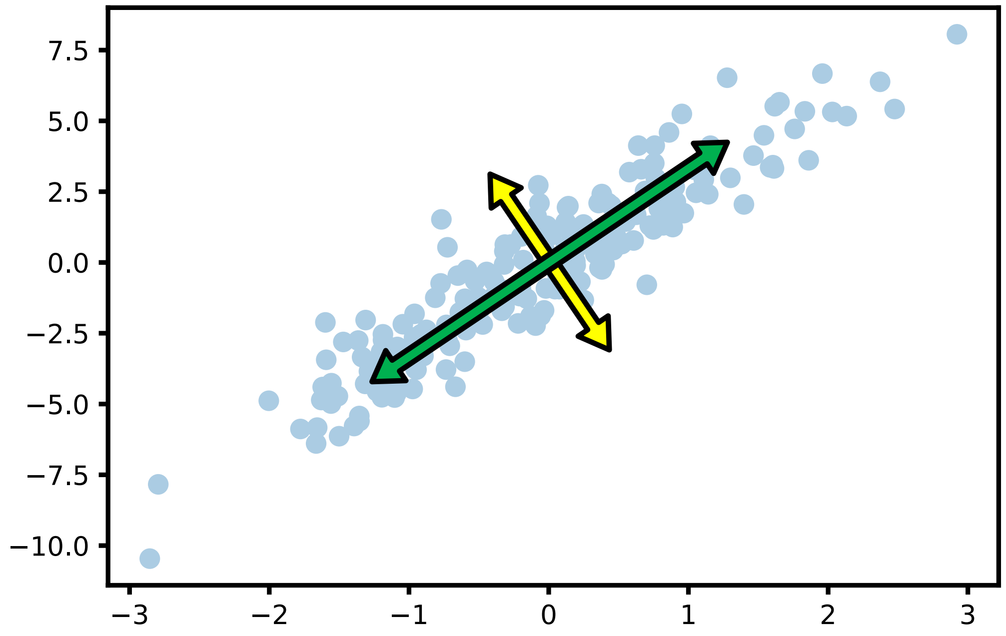




Think of this as data that sits on a flat sheet of paper, suspended in 3D space. We will come back to this analogy in a couple slides...

Generalizing Linear Dimensionality Reduction

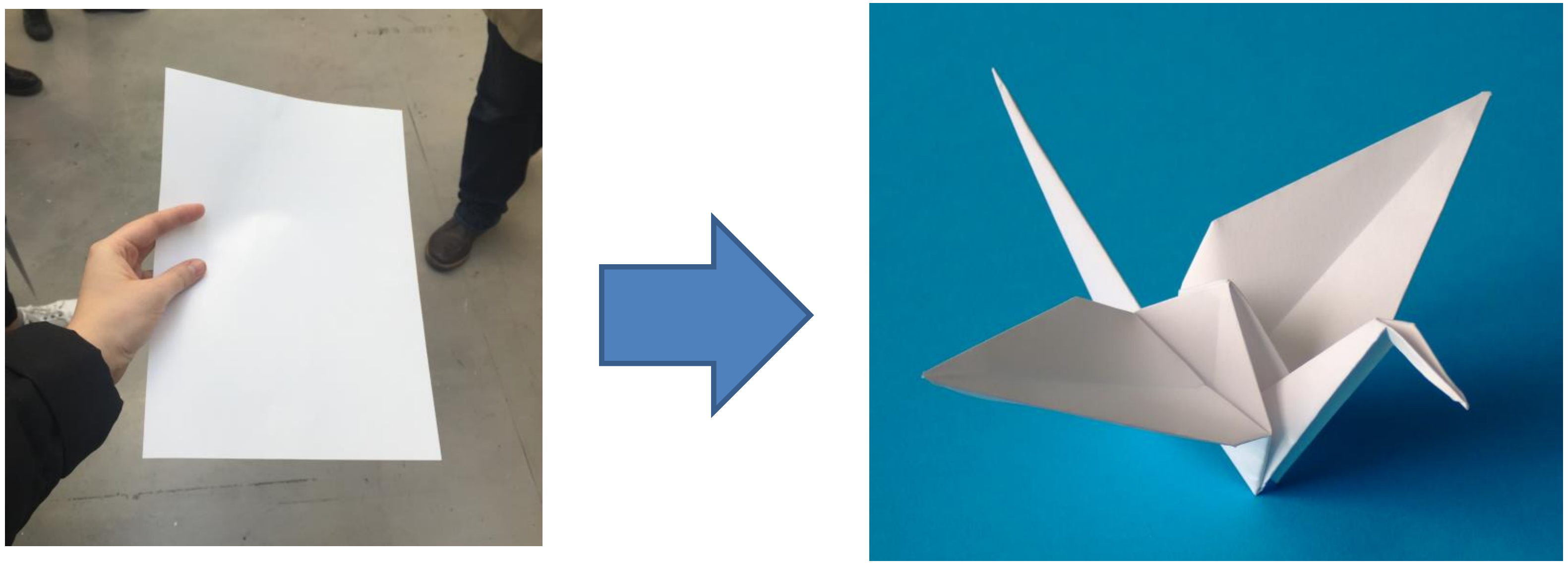
- ***Principal Components Analysis (PCA)***: find and order orthogonal axes by how much the data varies along each axis.
- The axes we find (ordered by variance of our data) are called ***principal components***.
- Dimensionality reduction can be done by using only the first *k* principal components



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

Manifolds

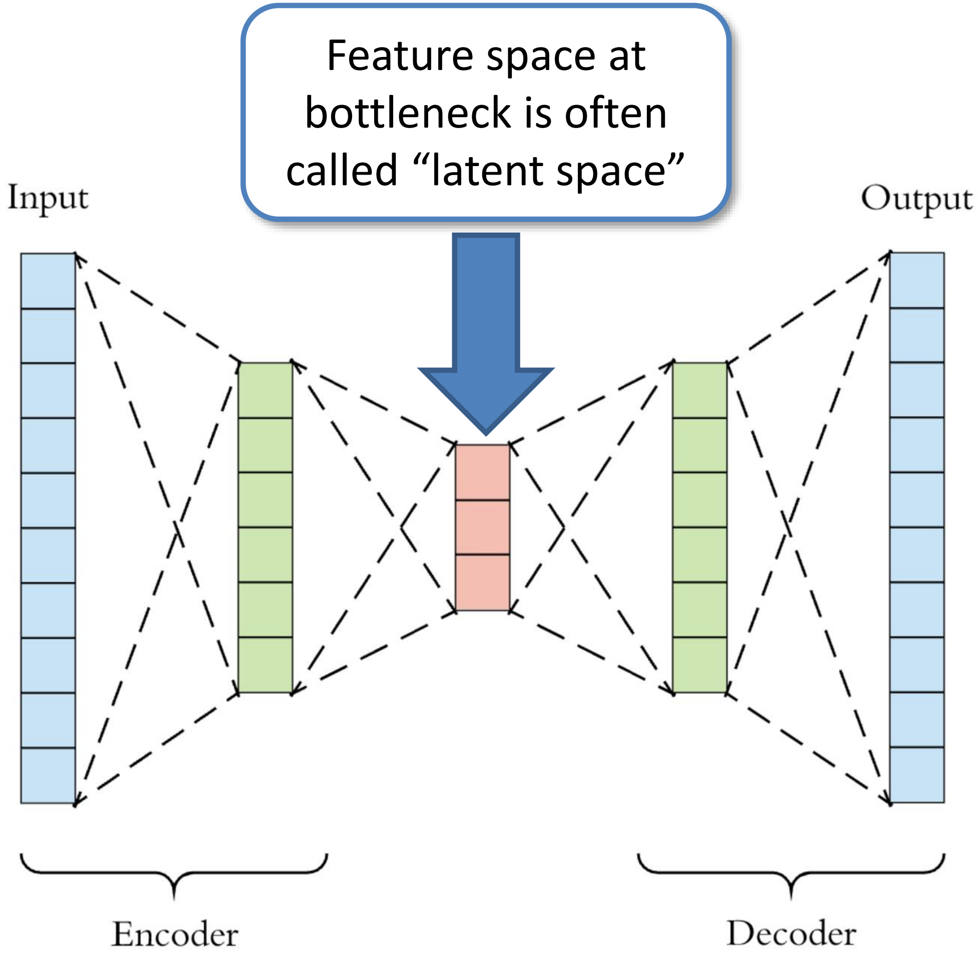
- Think of a piece of paper as a 2D subspace
- If we bend & fold it, it’s still locally a 2D subspace…
- A “manifold” is the generalization of this concept to higher dimensions…



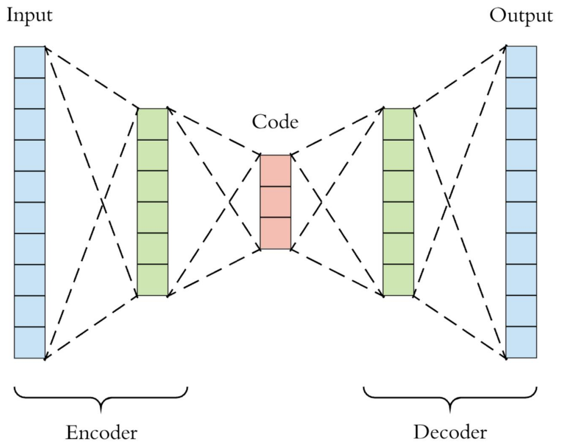
Autoencoders: Dimensionality Reduction for Manifolds

- Learn a non-linear 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

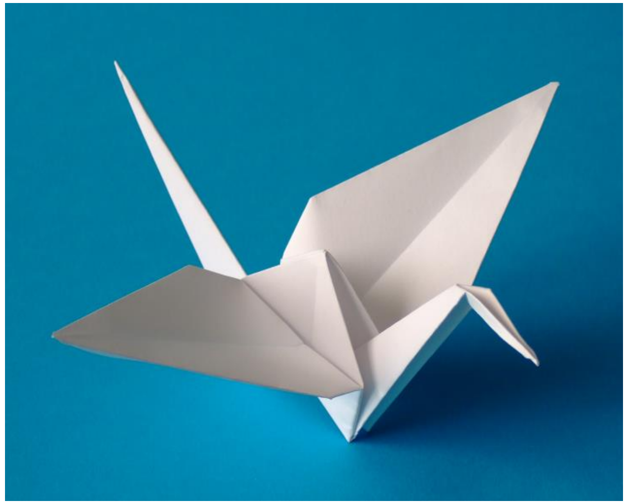
- **Unsupervised**
  - No labels required!



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

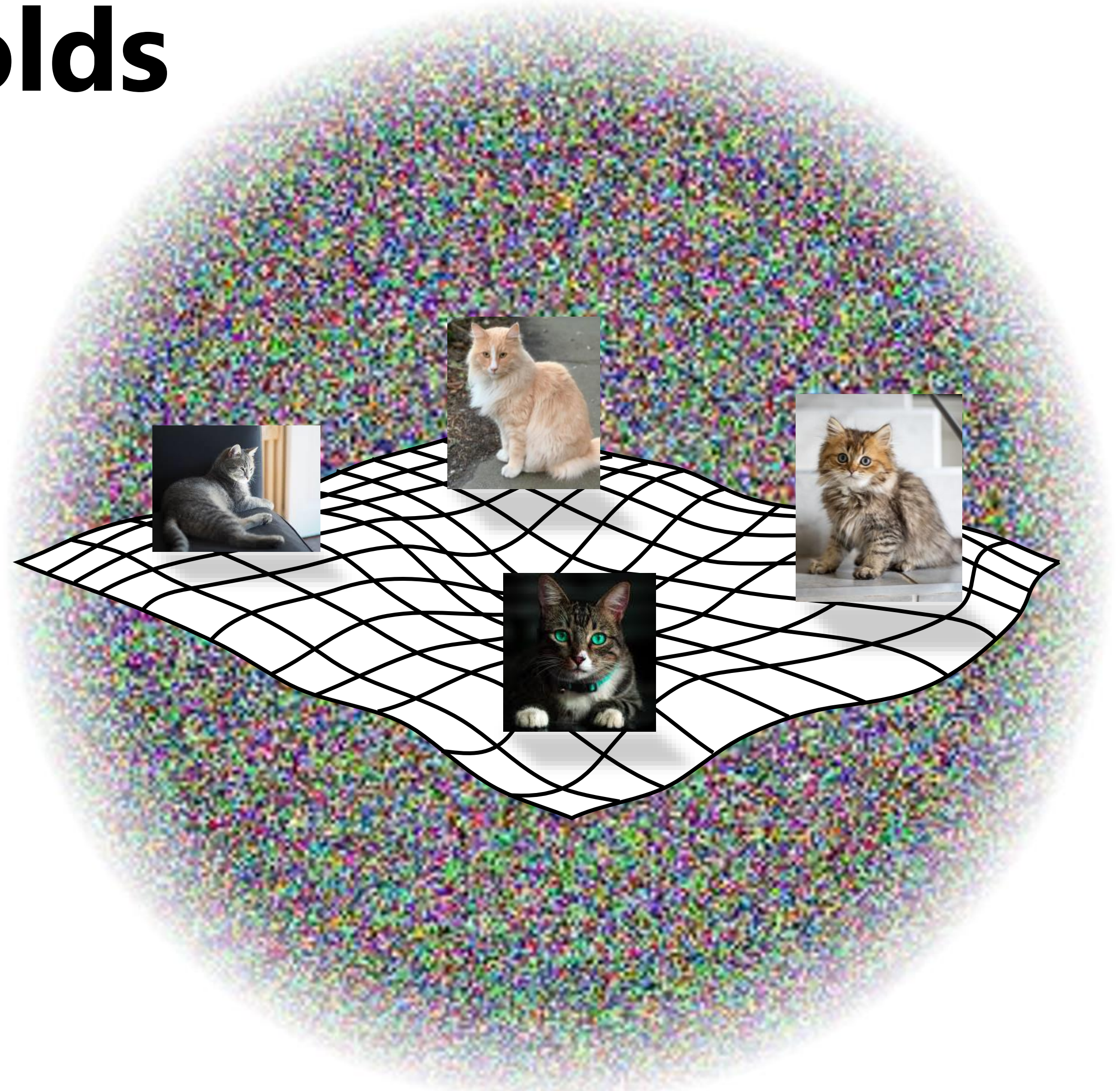
- Lets consider the space of all 100x100 images
- Now lets randomly sample that space…
- Conclusion: Most images are noise

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

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

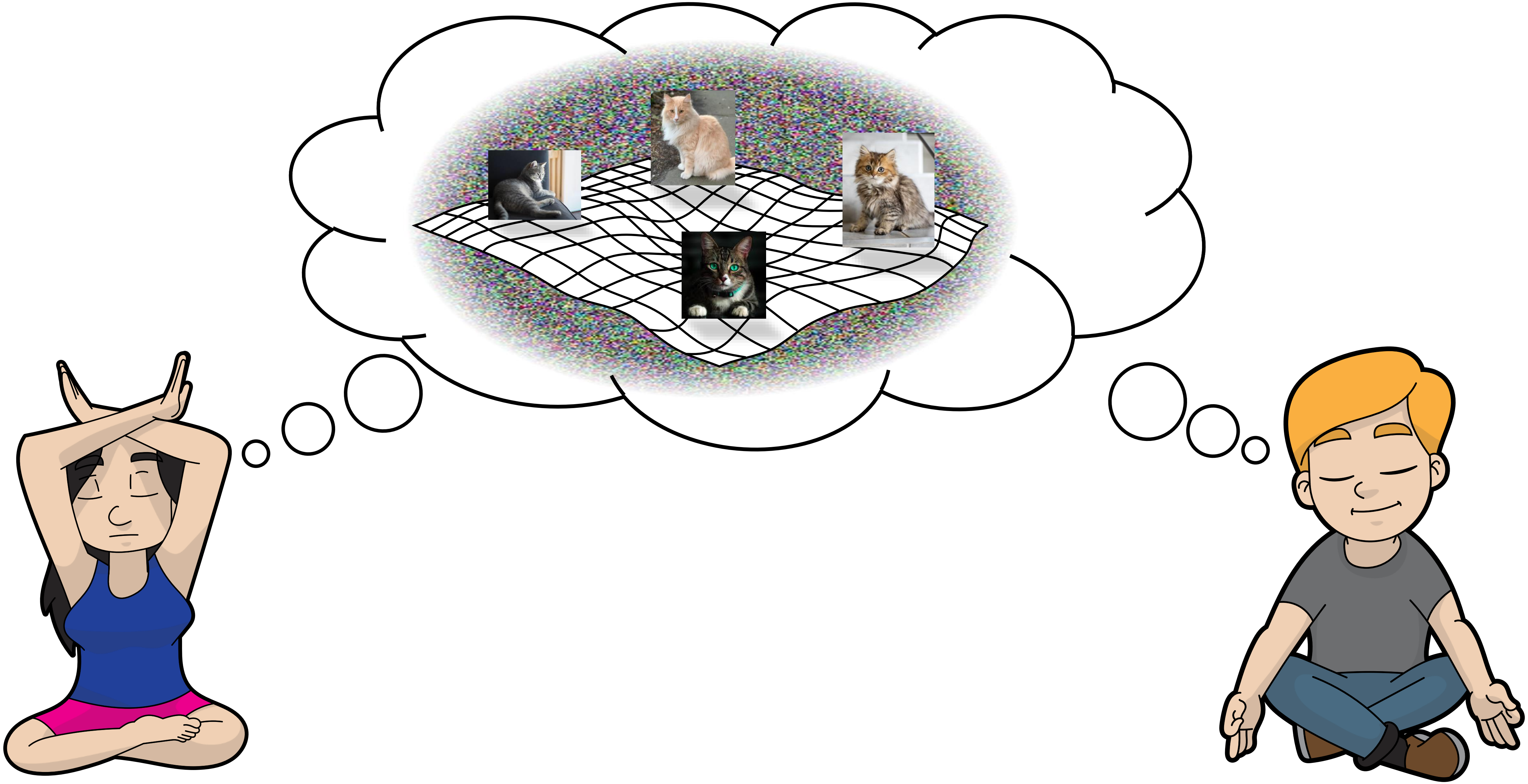
Natural Image Manifolds

- 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…



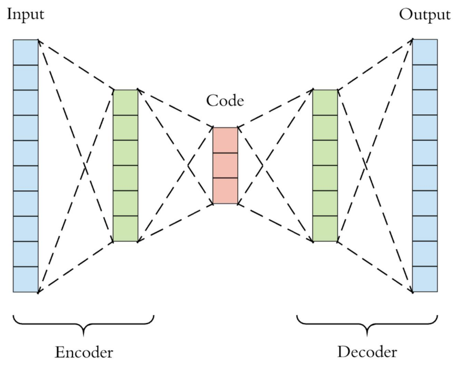
The Space of All Images

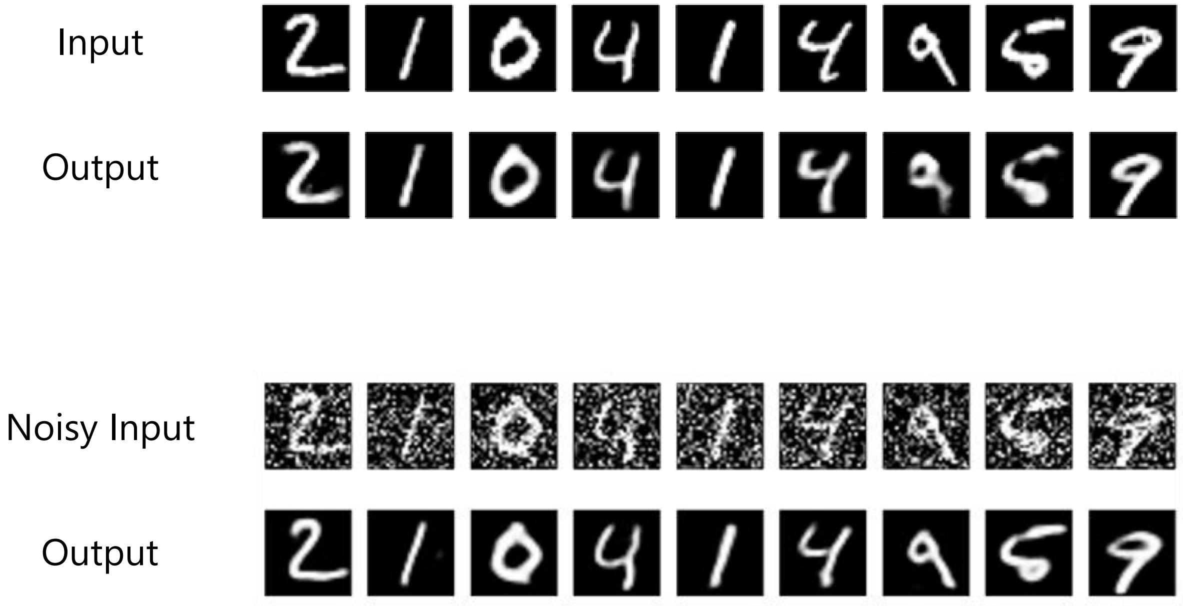
Natural Image Manifolds



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

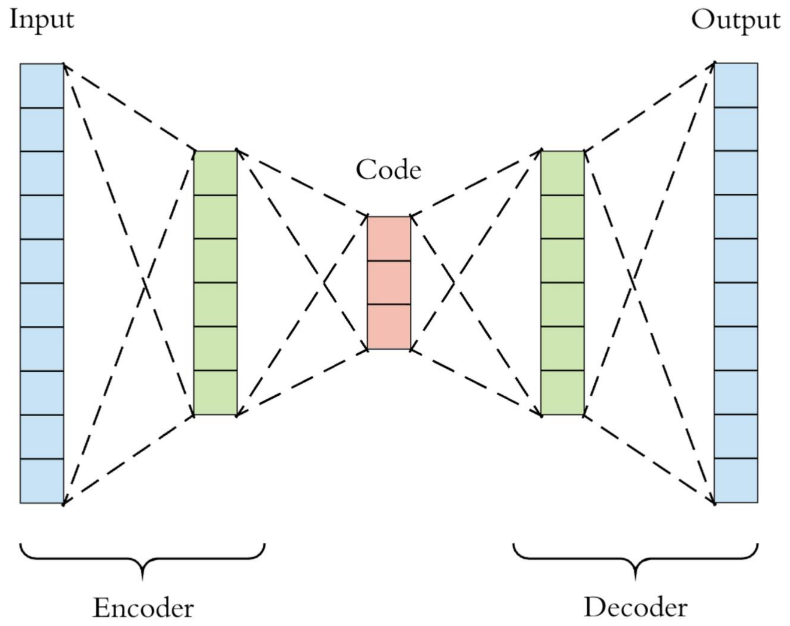


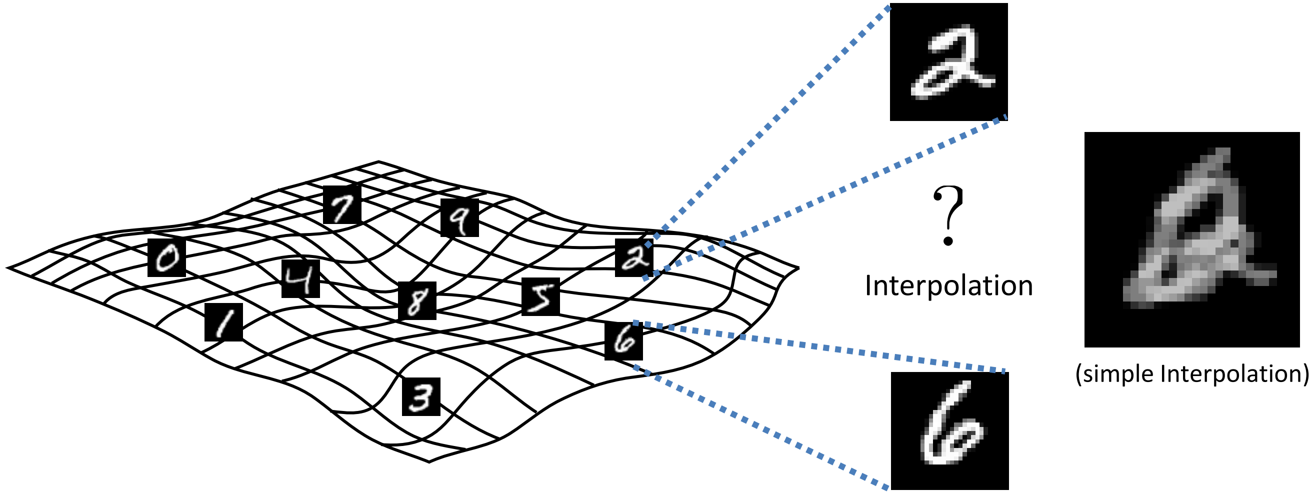


Examples from: https://blog.keras.io/building-autoencoders-in-keras.html

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…



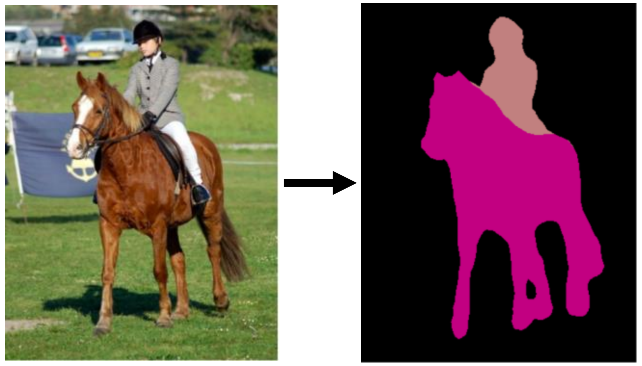


Abe Davis, with slides from Jin Sun, Phillip Isola, and Richard Zhang

IMAGE-TO-IMAGE APPLICATIONS

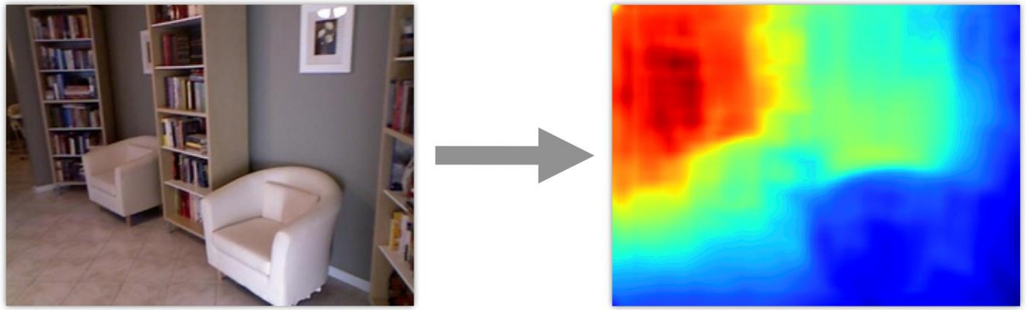
Image prediction (“structured prediction”)

Object labeling



[Long et al. 2015, …]

Depth prediction



Single RGB Image   Depth Map

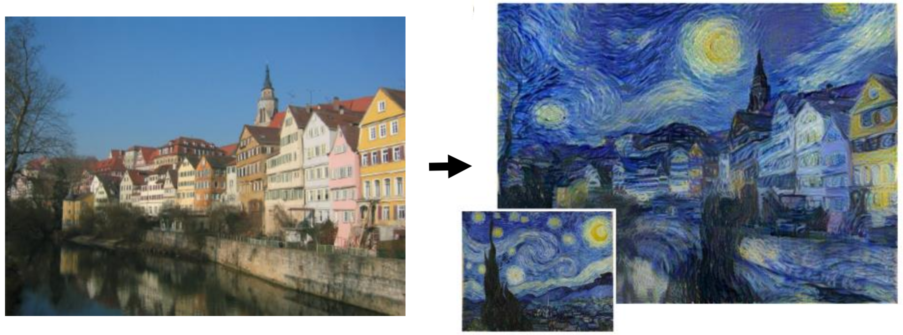
[Eigen et al. 2014, …]

Text-to-photo

“this small bird
has a pink breast →
and crown…”

[Reed et al. 2016, …]

Style transfer



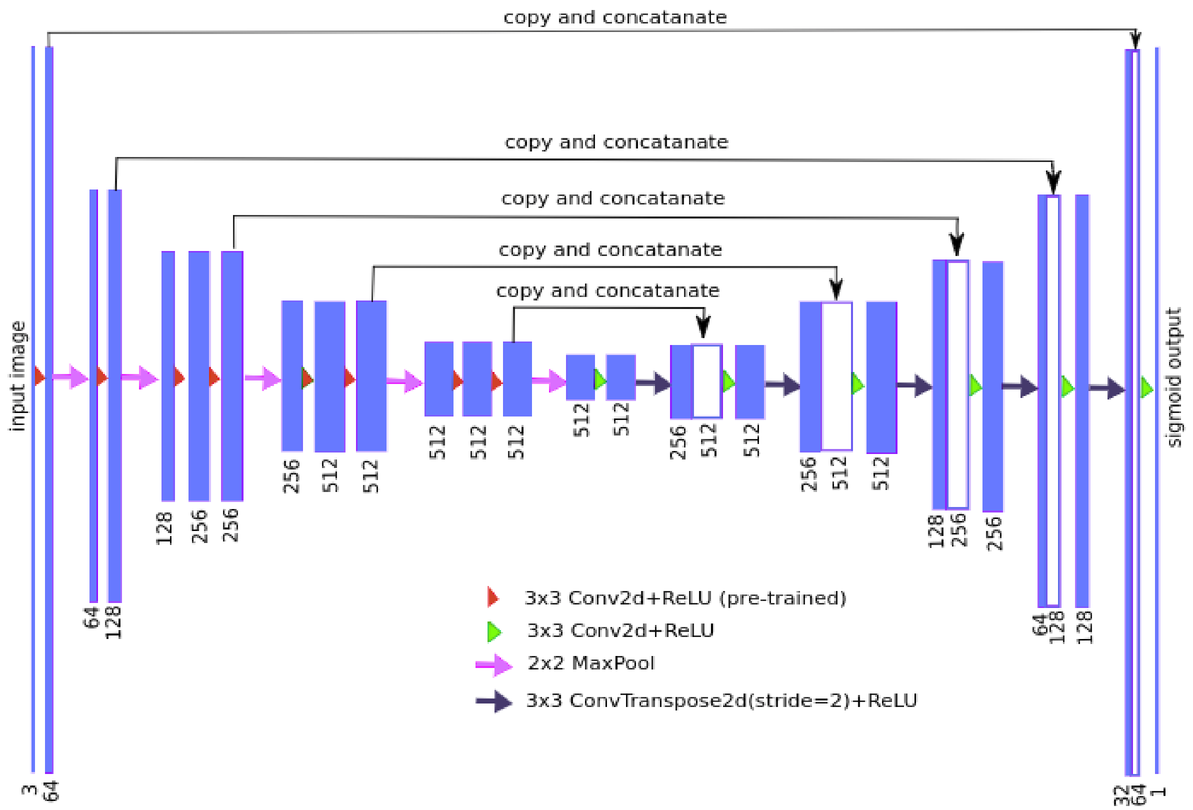
[Gatys et al. 2016, …]

Image classification vs. image translation

- For image classification, we map an image to a label (e.g., "cat")
- For image prediction/translation tasks, we map an image to another image-shaped thing (e.g., a depth map)
- What kind of convolutional neural network architecture can do this?

U-Net

- A popular network structure to generate same-sized output
- Similar to a convolutional autoencoder, but with “skip connections” that concatenate the output of earlier layers onto later layers
- Great for learning transformations from one image to another



$\mathbf{x}$ $\xrightarrow{\mathcal{F}}$ $\mathbf{y}$

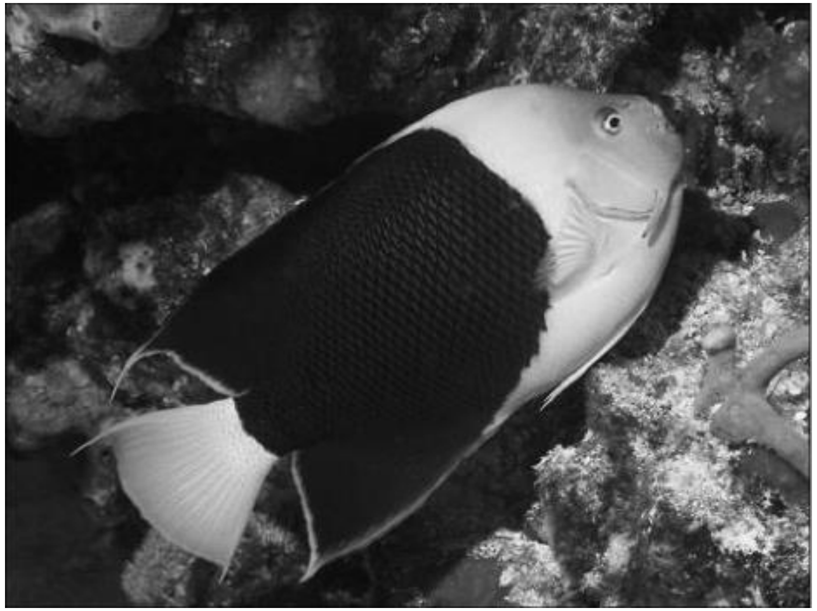
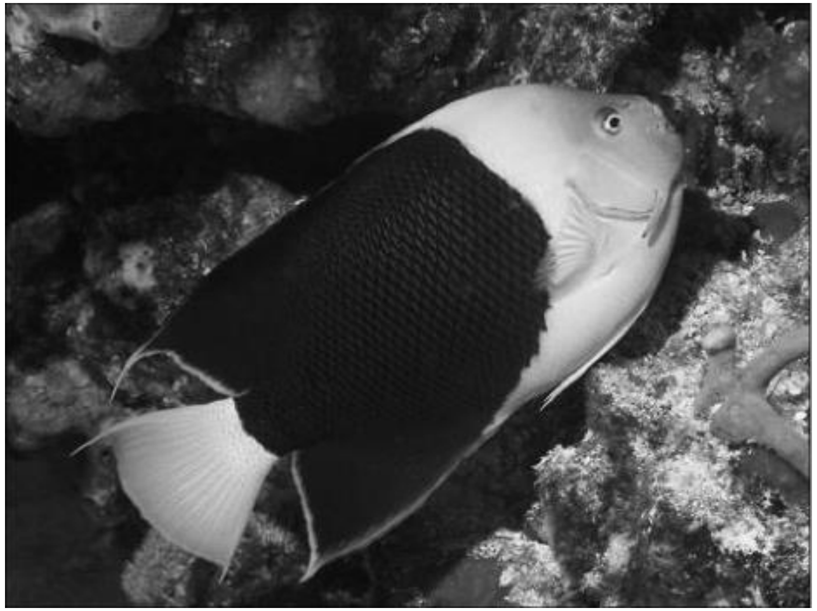


Image Colorization

from Jin Sun, Richard Zhang, Phillip Isola

$\mathbf{x}$ $\qquad\qquad$ $\mathbf{y}$

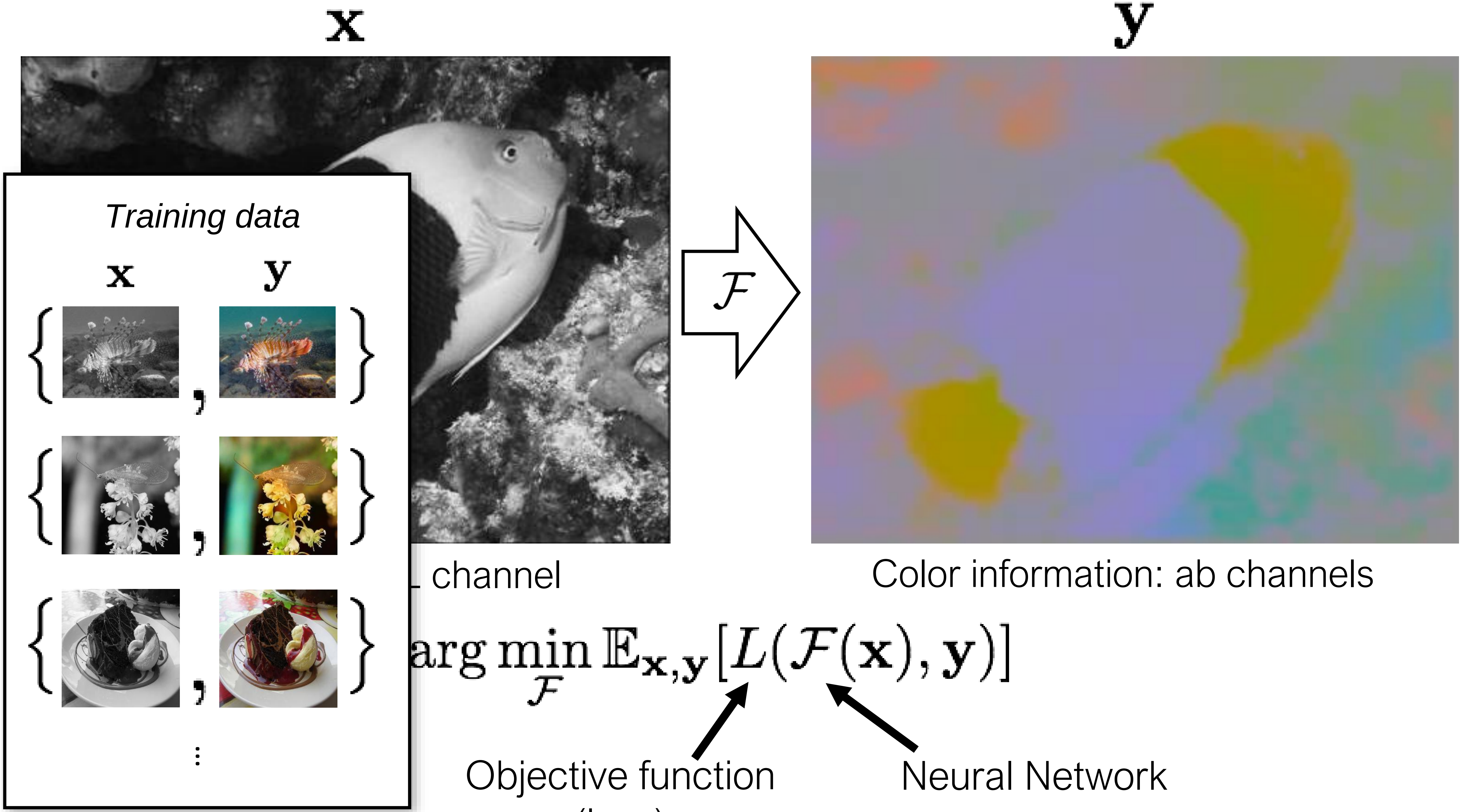


$\xrightarrow{\mathcal{F}}$

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

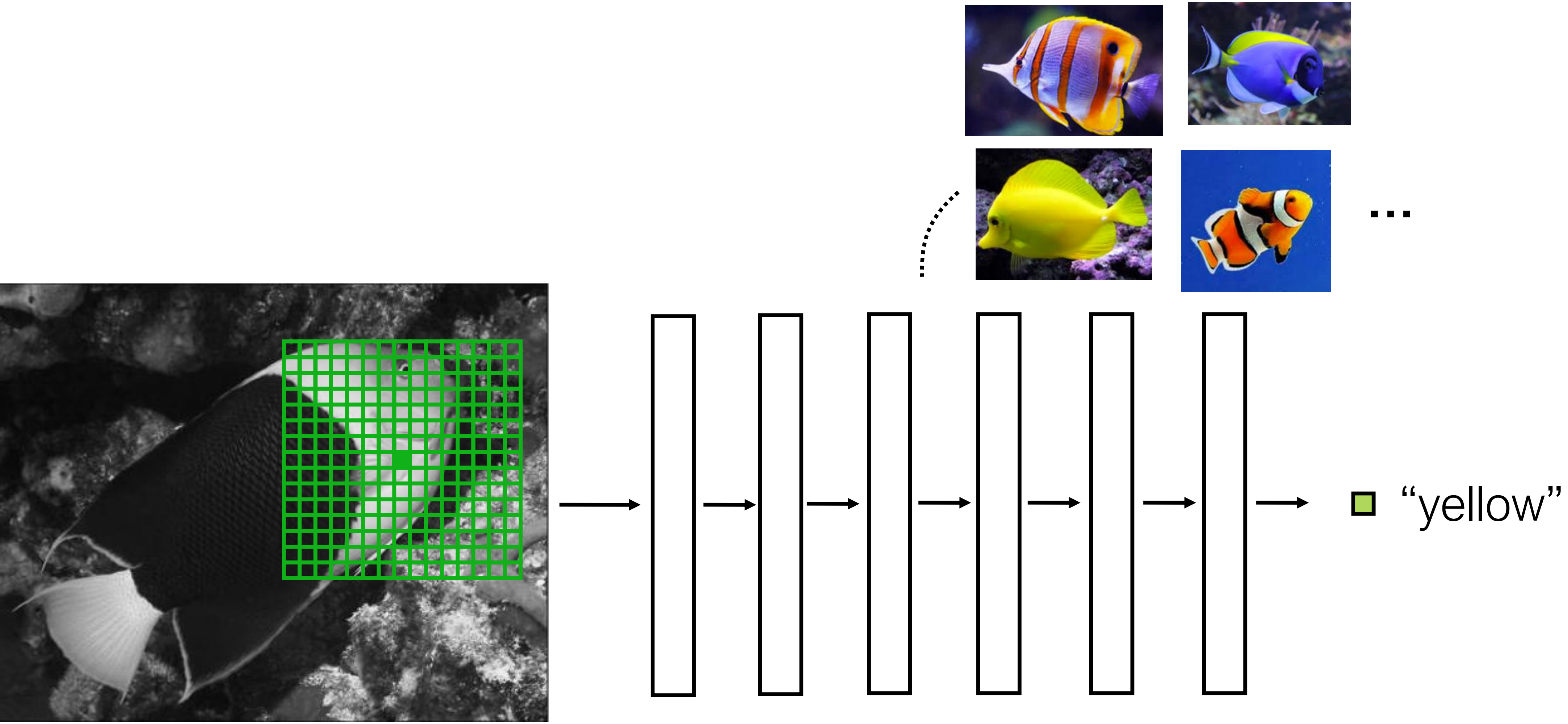
"**What** should I do" $\qquad$ "**How** should I do it?"

from Jin Sun, Richard Zhang, Phillip Isola

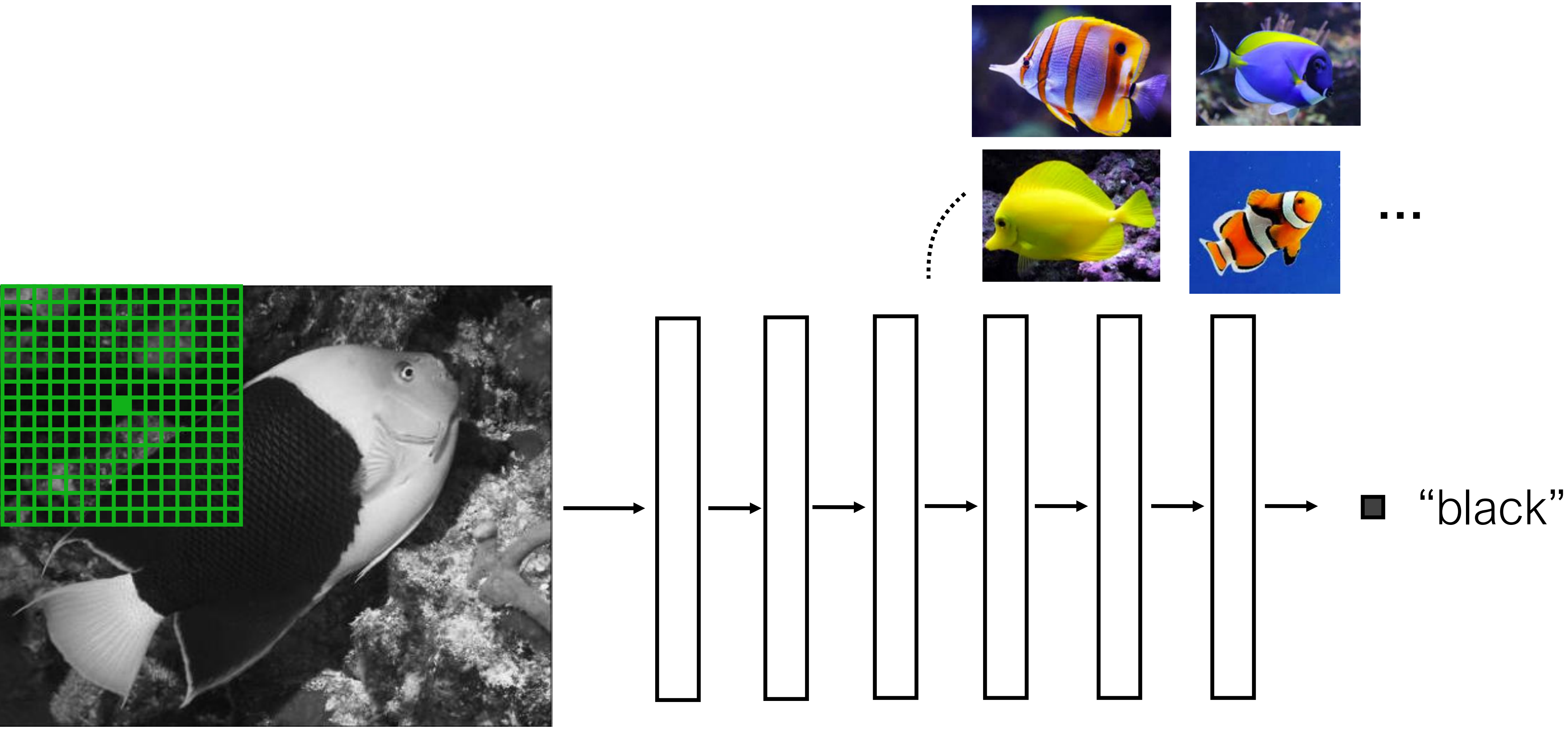


$$\arg\min_{\mathcal{F}} \mathbb{E}_{\mathbf{x},\mathbf{y}}[L(\mathcal{F}(\mathbf{x}),\mathbf{y})]$$

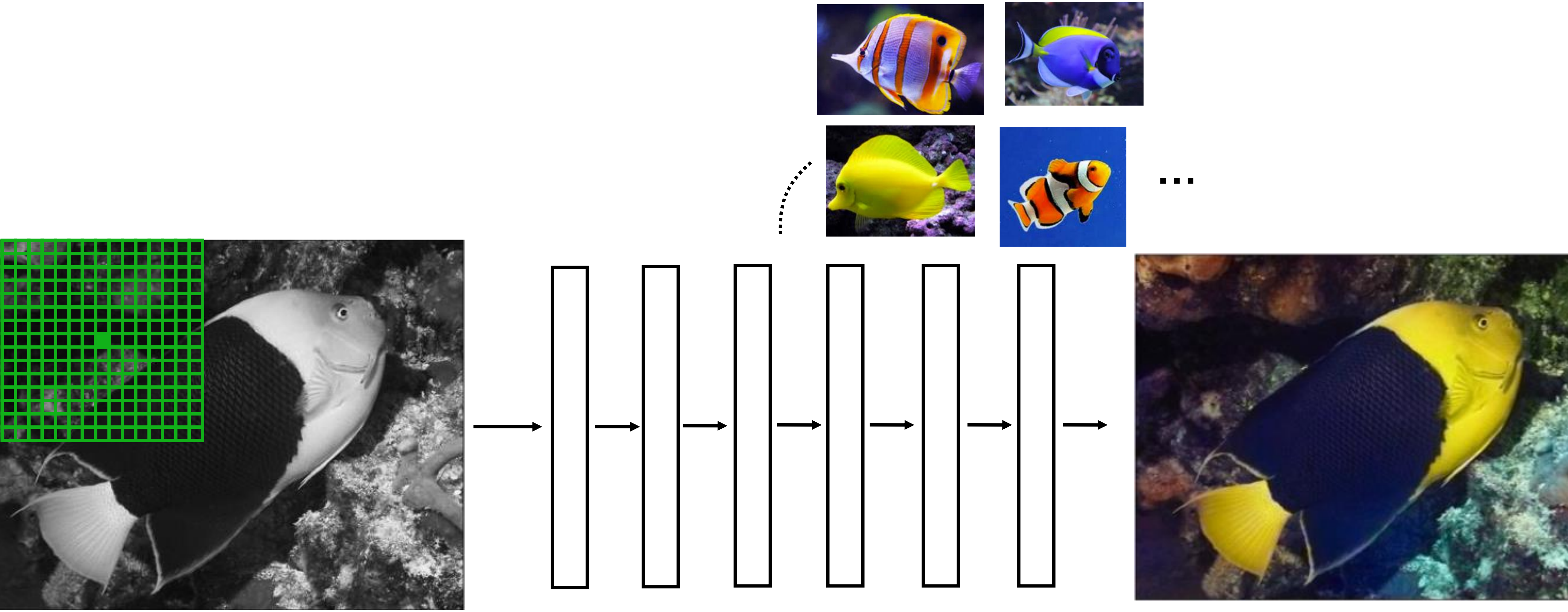
from Jin Sun, Richard Zhang, Phillip Isola



from Jin Sun, Richard Zhang, Phillip Isola



from Jin Sun, Richard Zhang, Phillip Isola



from Jin Sun, Richard Zhang, Phillip Isola

Basic loss functions

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

Truth: $\mathbf{y}$

Classification (cross-entropy):

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

← How many extra bits it takes to correct the predictions

Least-squares regression:

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

← How far off we are in Euclidean distance

from Jin Sun, Richard Zhang, Phillip Isola

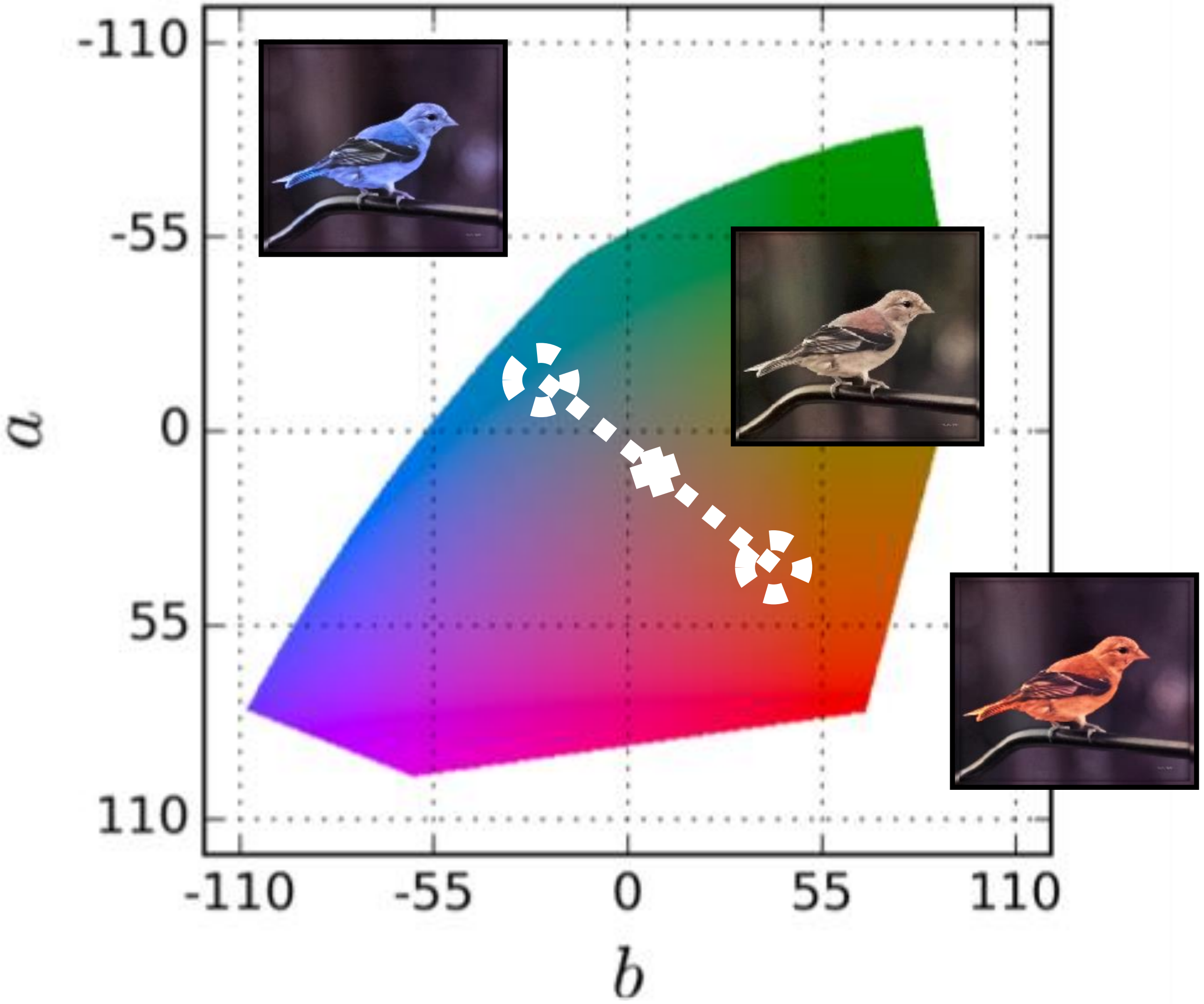
Designing loss functions

Input

Output

Ground truth

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



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

Designing loss functions

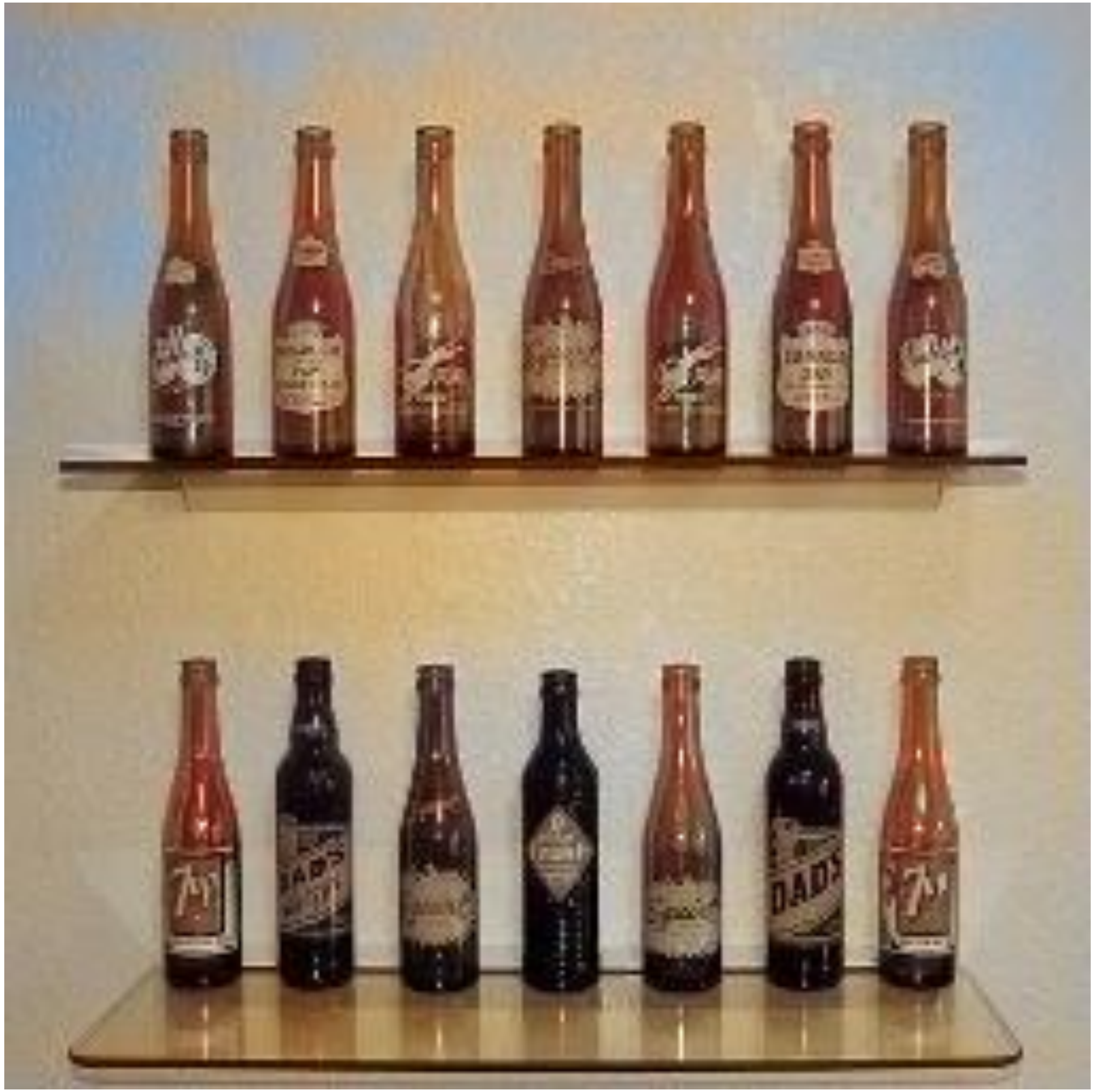
Input

Zhang et al. 2016

Ground truth

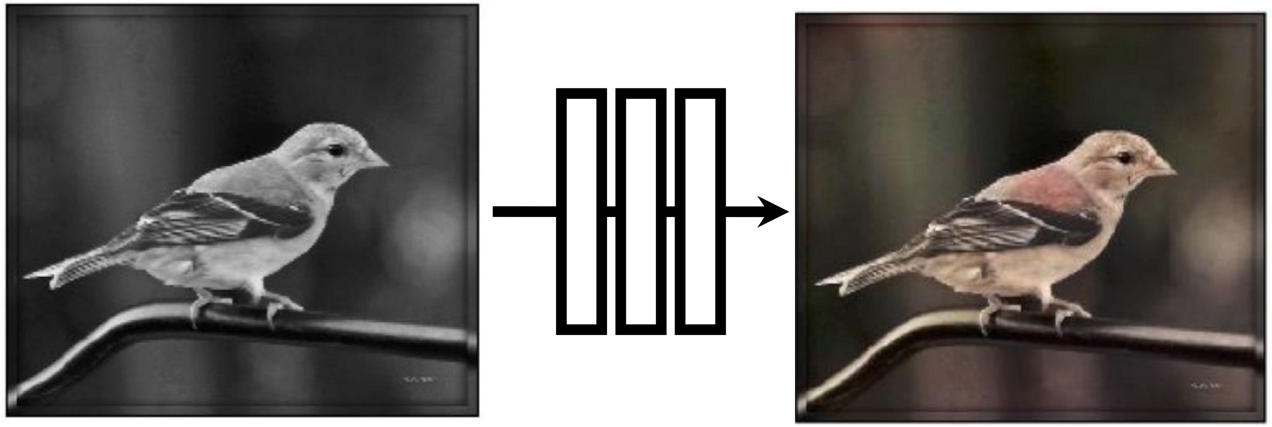
Color distribution cross-entropy loss with colorfulness enhancing term.

[Zhang, Isola, Efros, ECCV 2016]


DADS

Designing loss functions

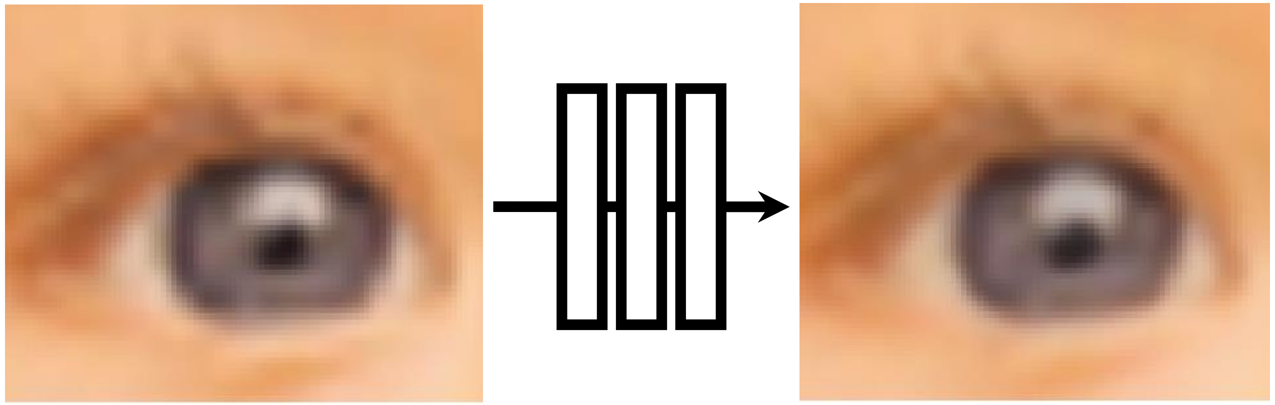
Image colorization



L2 regression

[Zhang, Isola, Efros, ECCV 2016]

Super-resolution

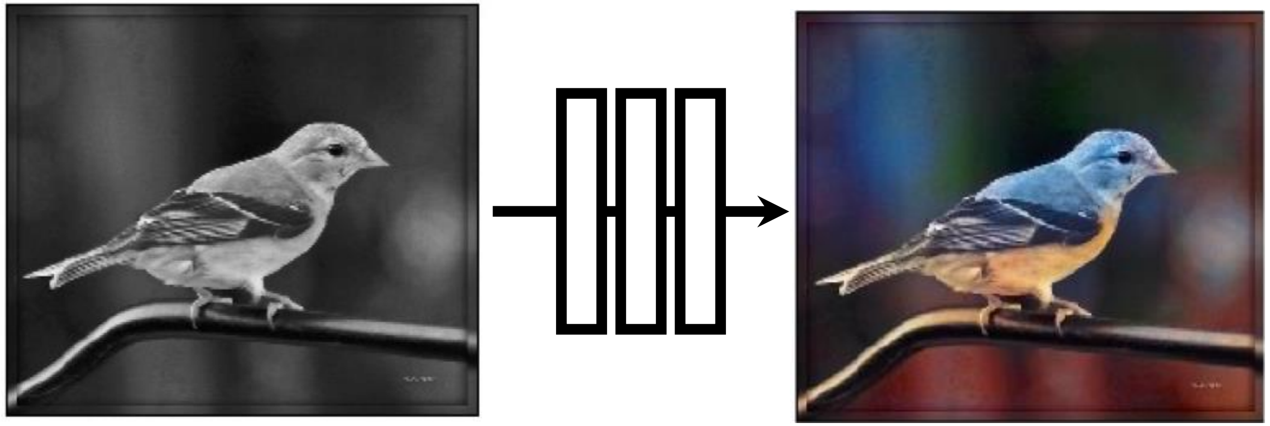


L2 regression

[Johnson, Alahi, Li, ECCV 2016]

Designing loss functions

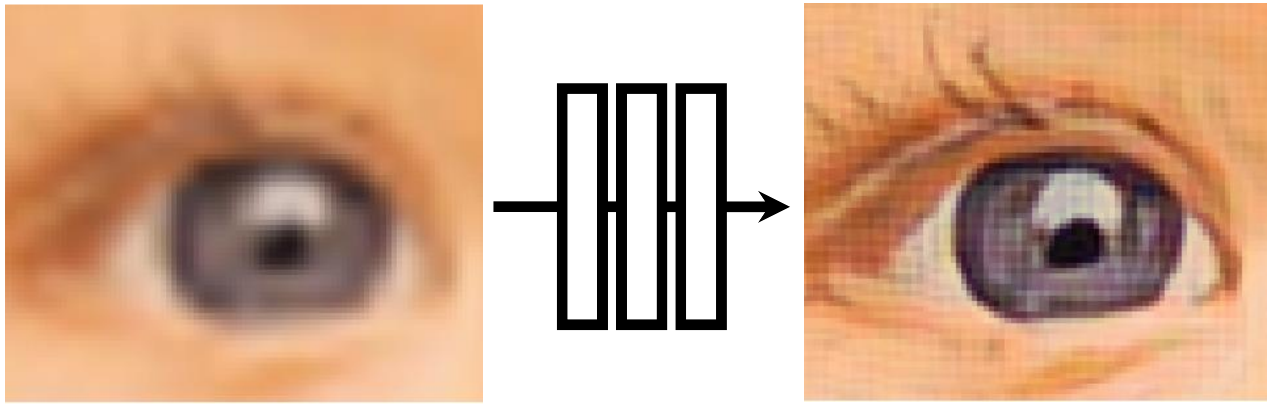
Image colorization



[Zhang, Isola, Efros, ECCV 2016]

Cross entropy objective,
with colorfulness term

Super-resolution

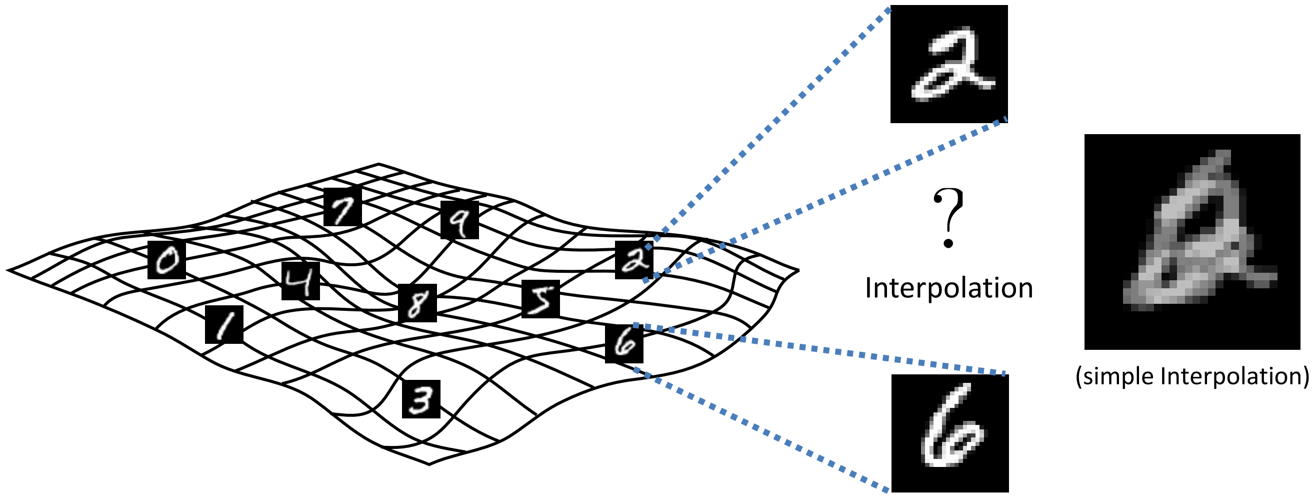


[Johnson, Alahi, Li, ECCV 2016]

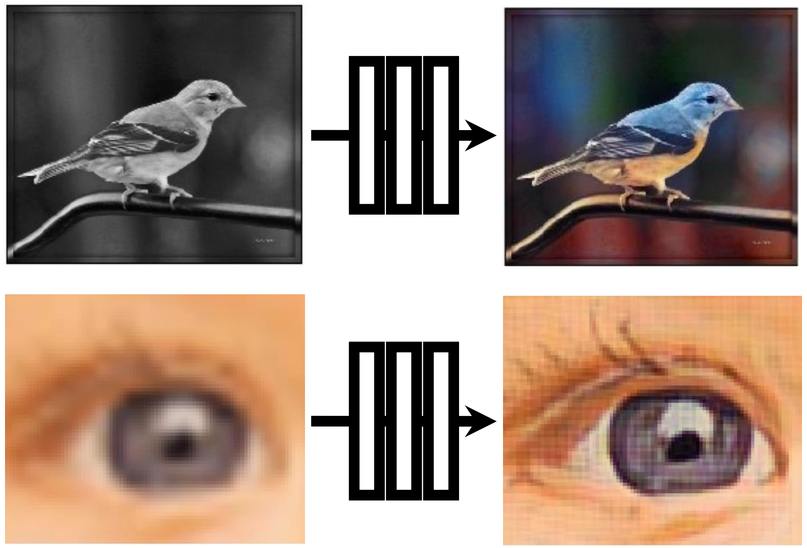
Deep feature covariance
matching objective

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

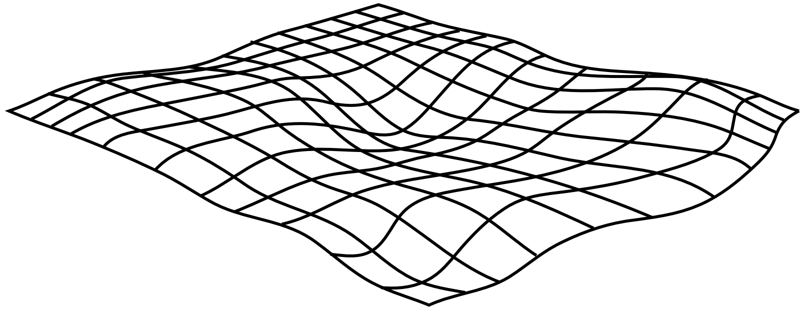


Abe Davis, with slides from Jin Sun and Phillip Isola

PART 3: GENERATIVE ADVERSARIAL NETWORKS (GANS)

Generative Adversarial Networks (GANs)

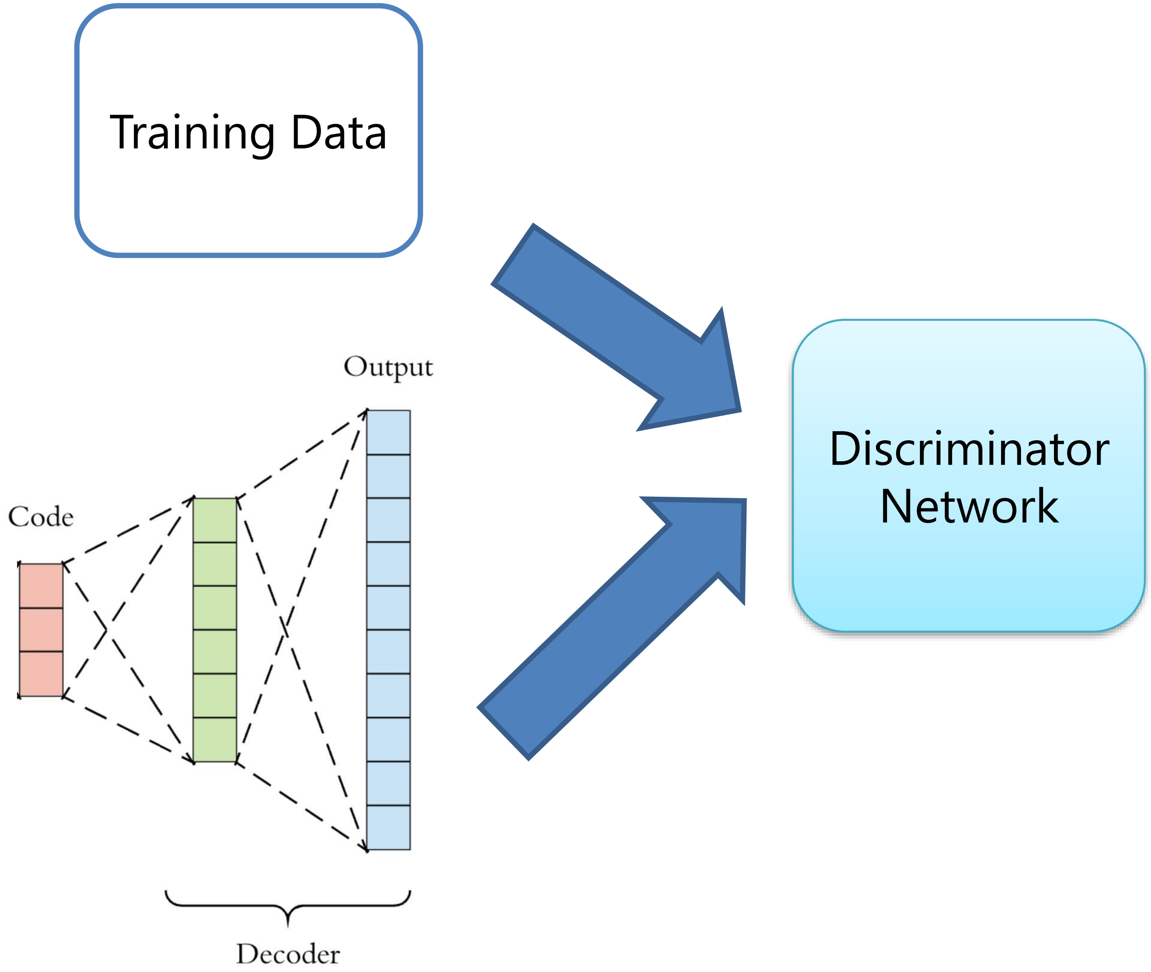
- Basic idea: Learn a mapping from some latent space to images on a particular manifold



- 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 tries to create output indistinguishable from training data
  - Discriminator tries to distinguish between generator output and training data



Example: Randomly Sampling the Space of Face Images

(Using Generative Adversarial Networks (GANs)

A

B

Which face is real?

Example: Randomly Sampling the Space of Face Images

(Using Generative Adversarial Networks (GANs)

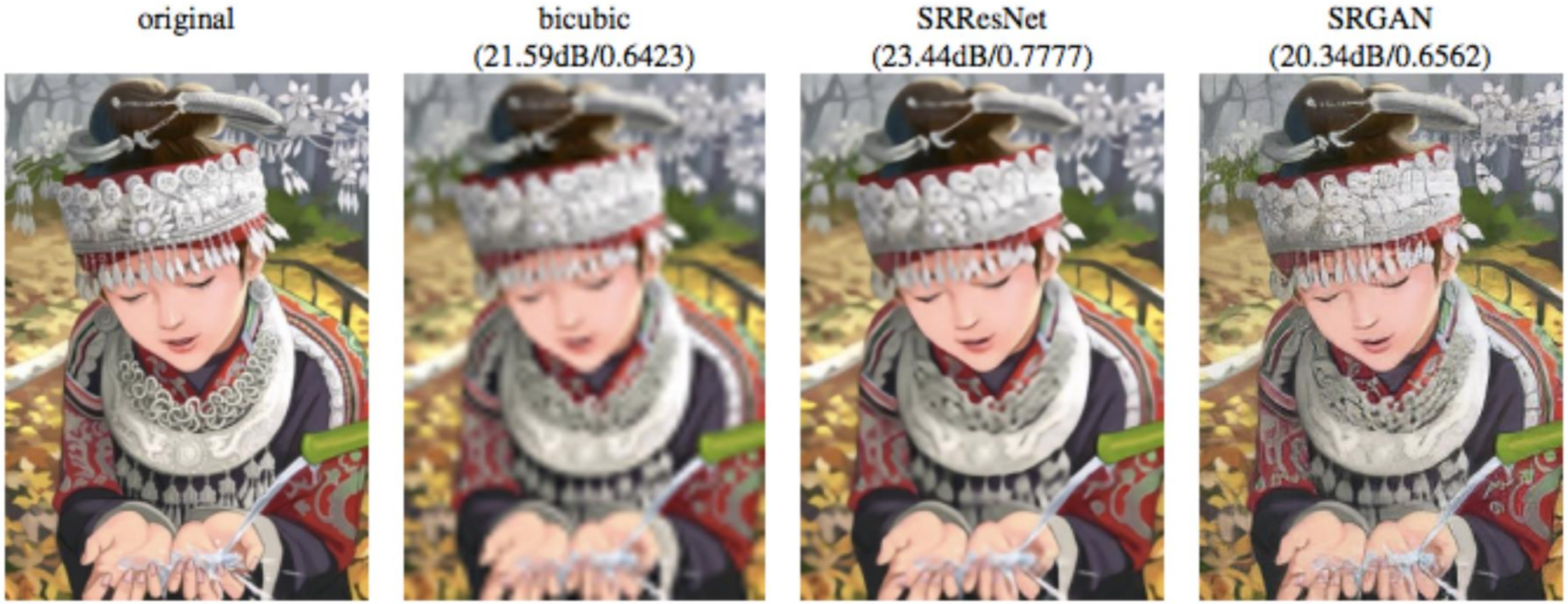
A

B

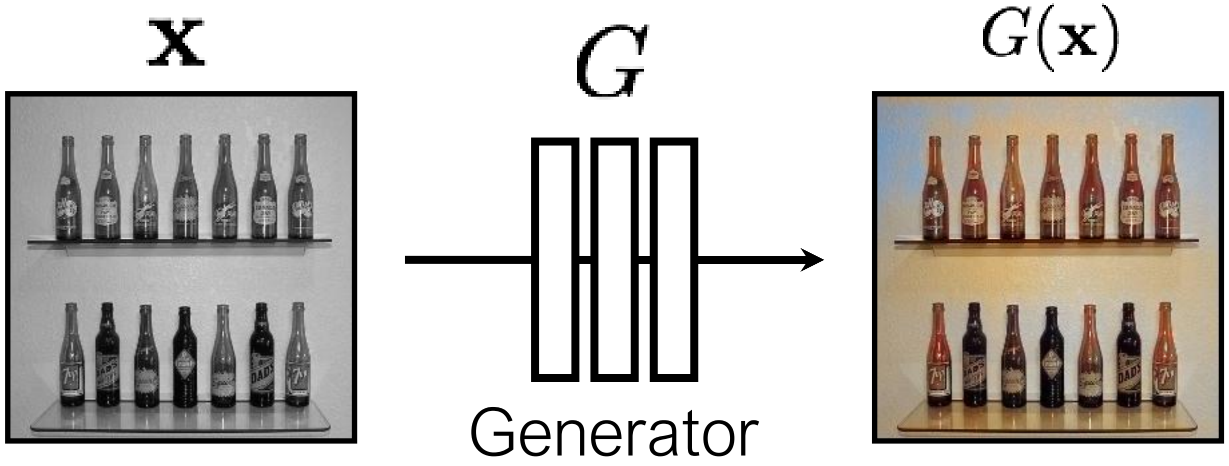
[Which face is real]?

Conditional GANs

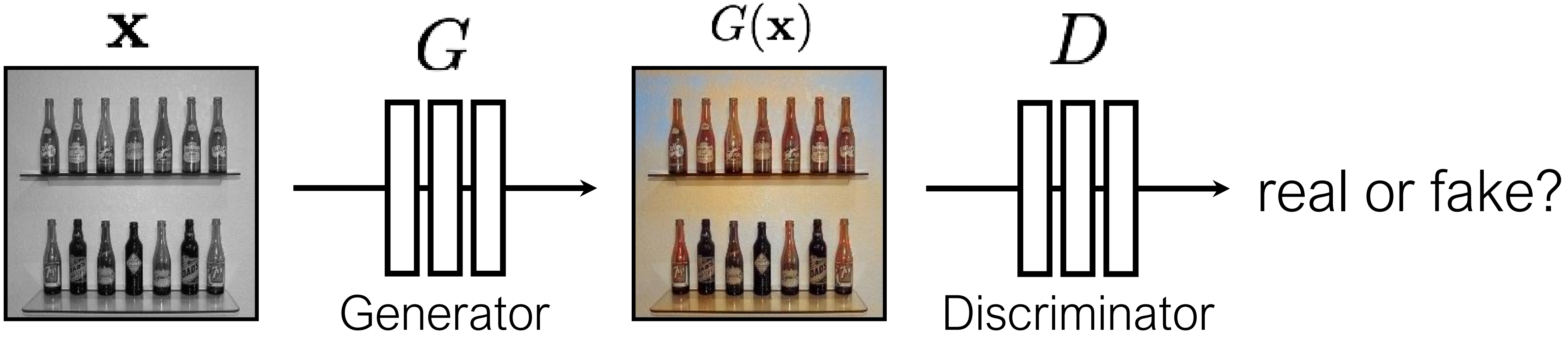
- Generate samples from a conditional distribution
- Example: generate high-resolution image conditioned on low resolution input



[Ledig et al 2016]



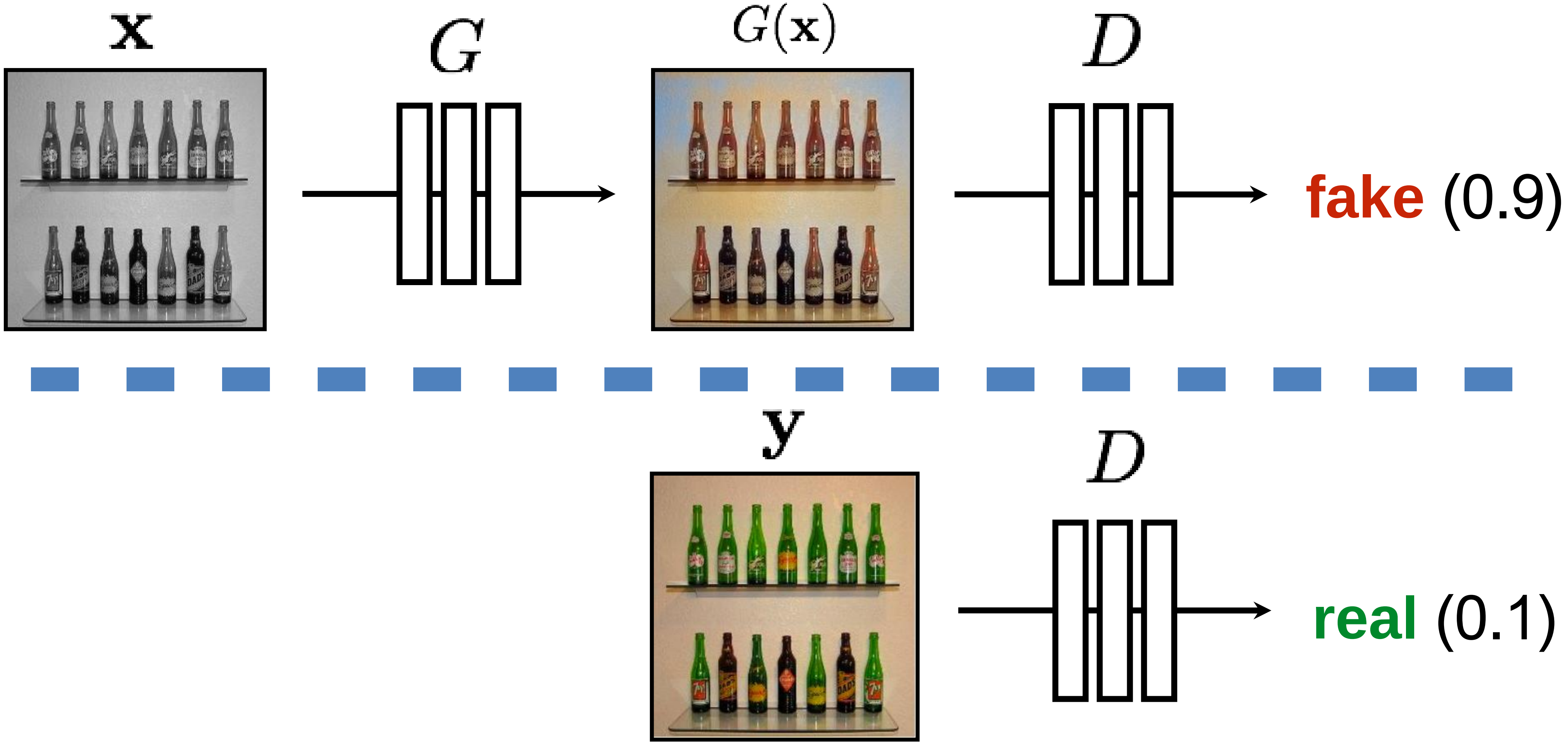
[Goodfellow et al., 2014]



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

D tries to identify the fakes

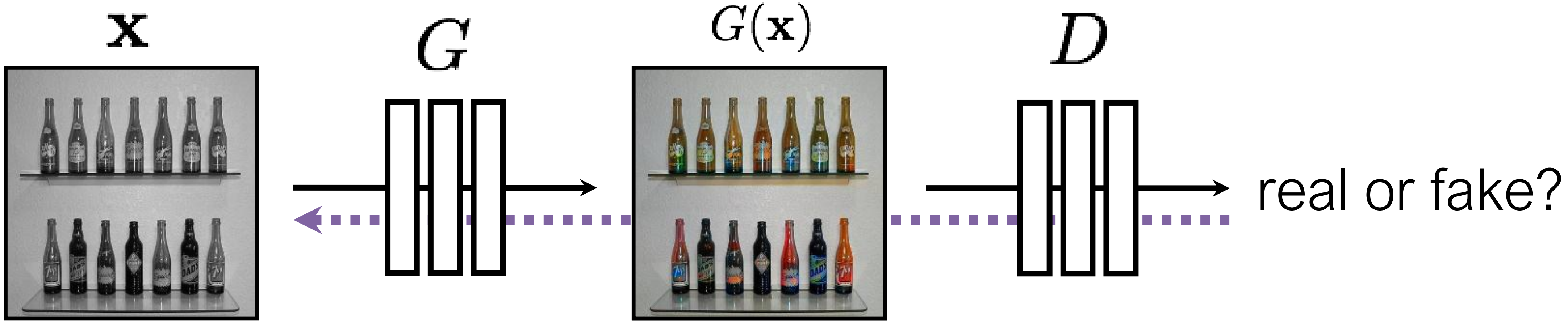
[Goodfellow et al., 2014]



(Identify generated images as fake) (Identify training images as real)

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

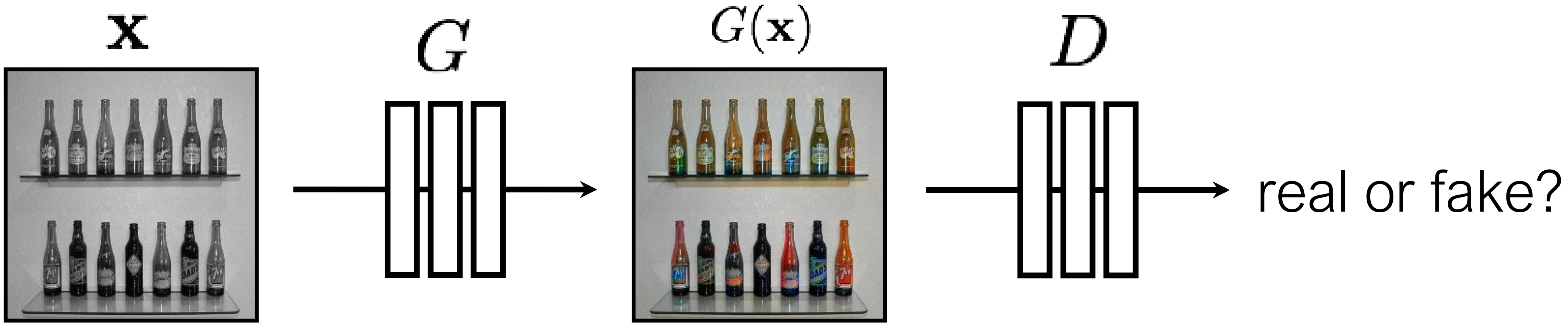
[Goodfellow et al., 2014]



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

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

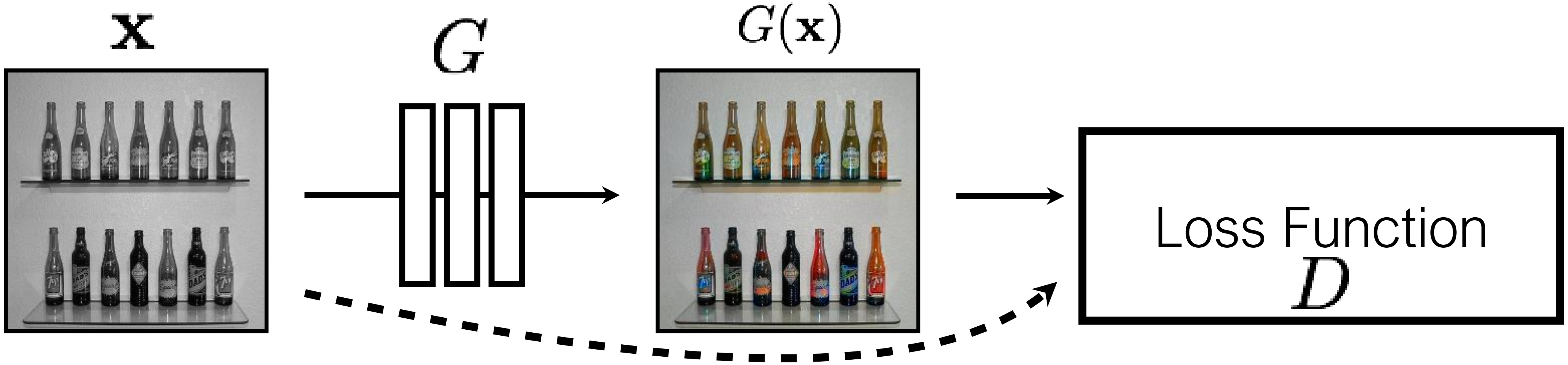
[Goodfellow et al., 2014]



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

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

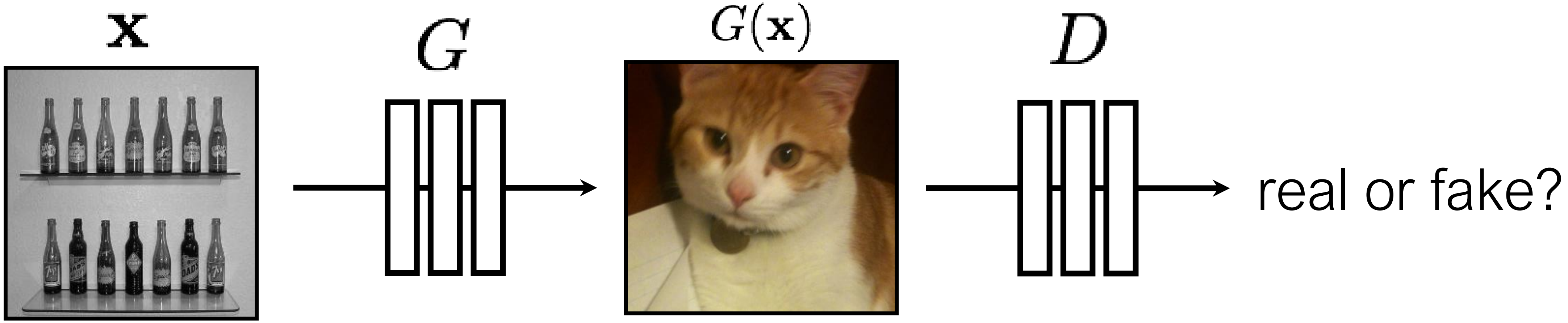
[Goodfellow et al., 2014]



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

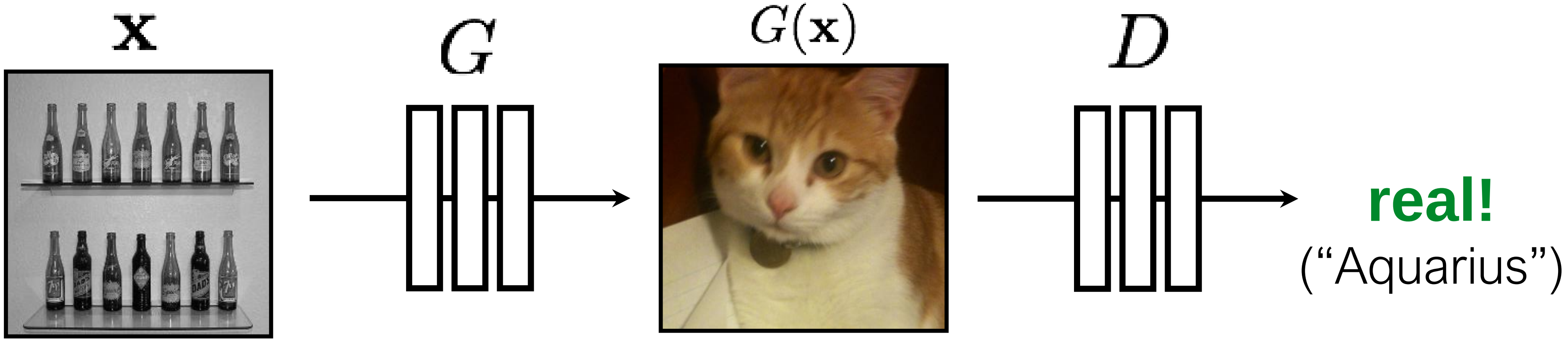
Rather than being hand-designed, it is *learned*.

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



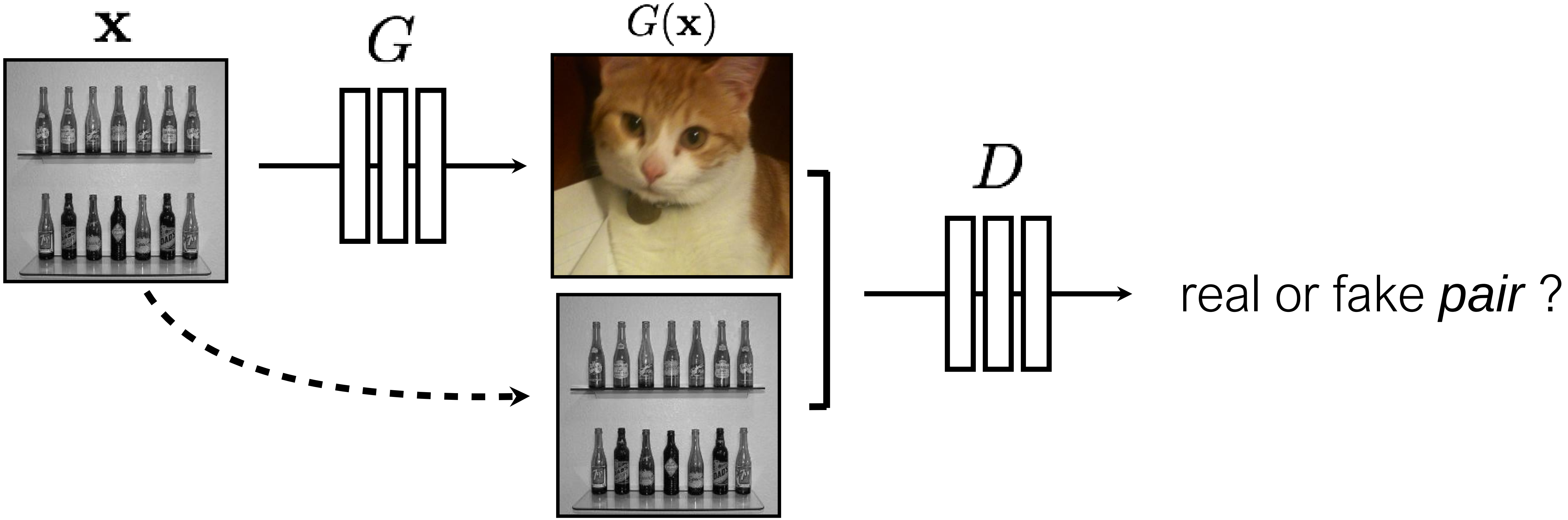
$$\arg\min_{G}\max_{D}\ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(G(\mathbf{x}))\ +\ \log(1-D(\mathbf{y}))\ ]$$

[Goodfellow et al., 2014]



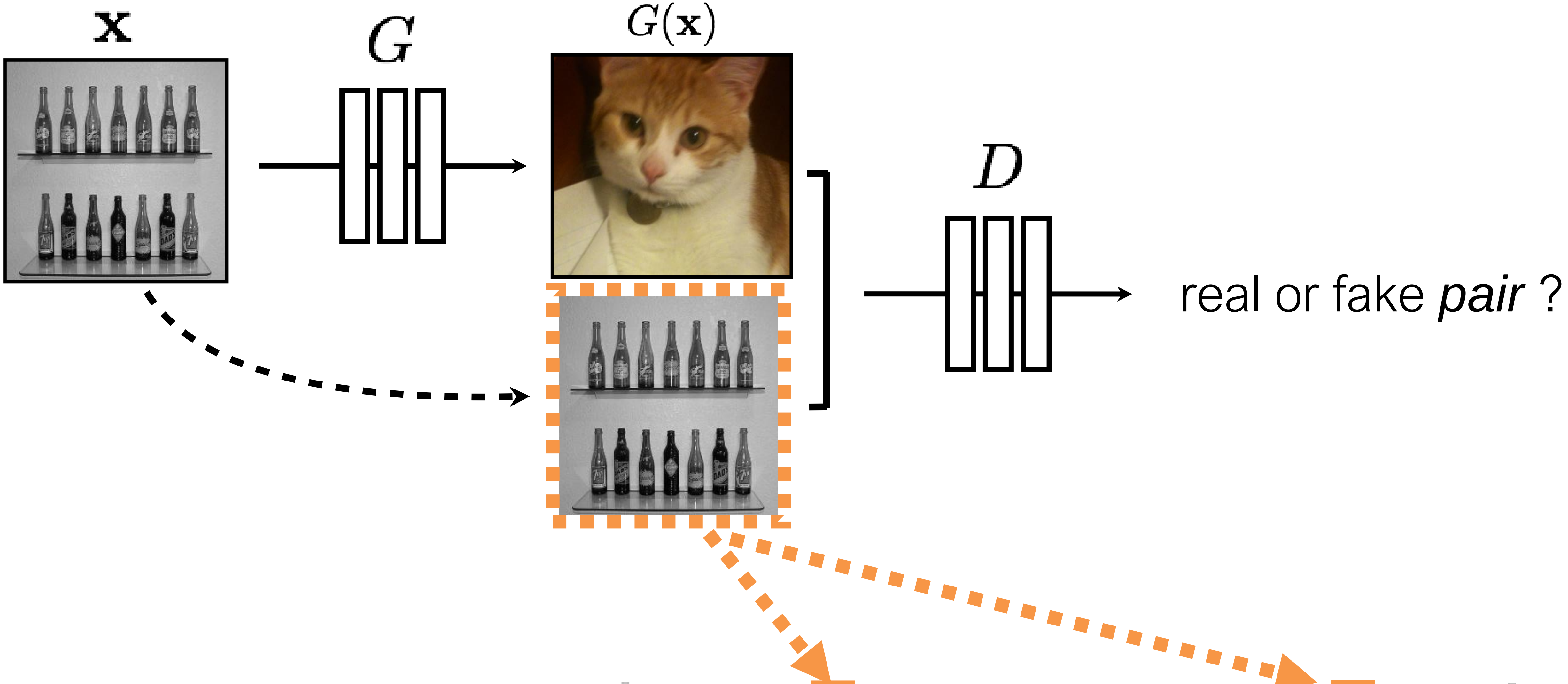
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(G(\mathbf{x})) \; + \; \log(1 - D(\mathbf{y})) \;]$$

[Goodfellow et al., 2014]



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

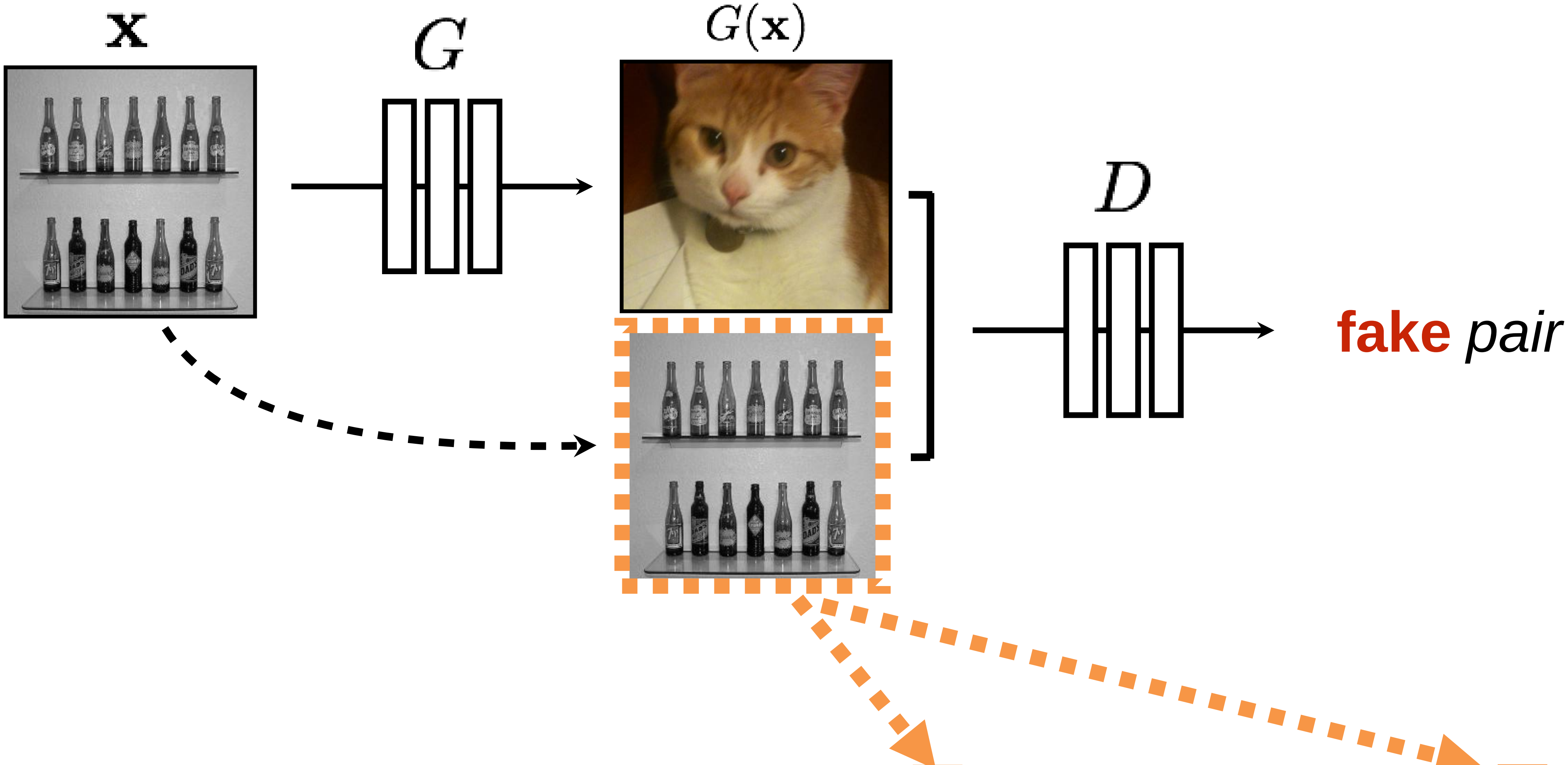
[Goodfellow et al., 2014]
[Isola et al., 2017]



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

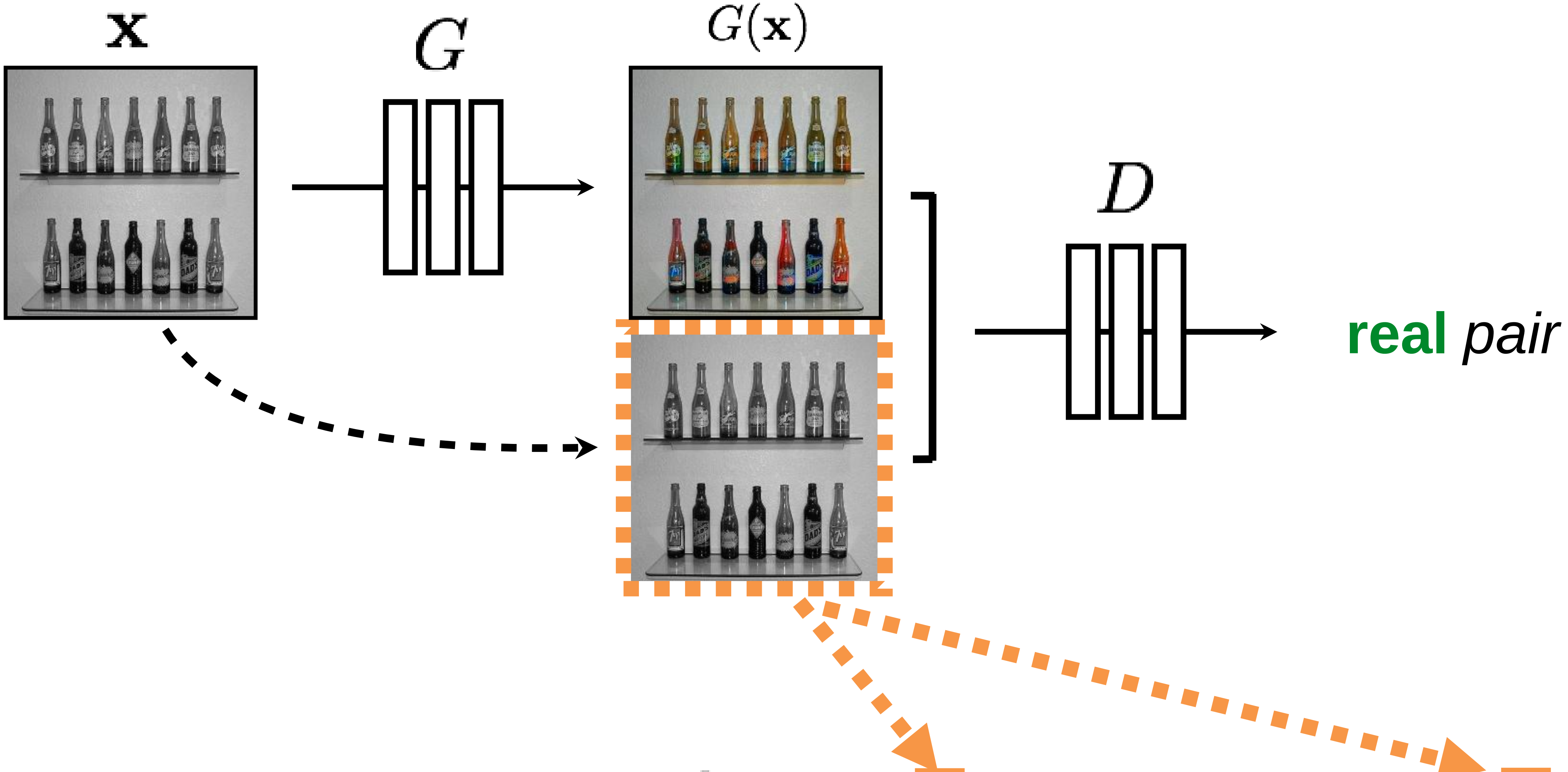
[Goodfellow et al., 2014]

[Isola et al., 2017]



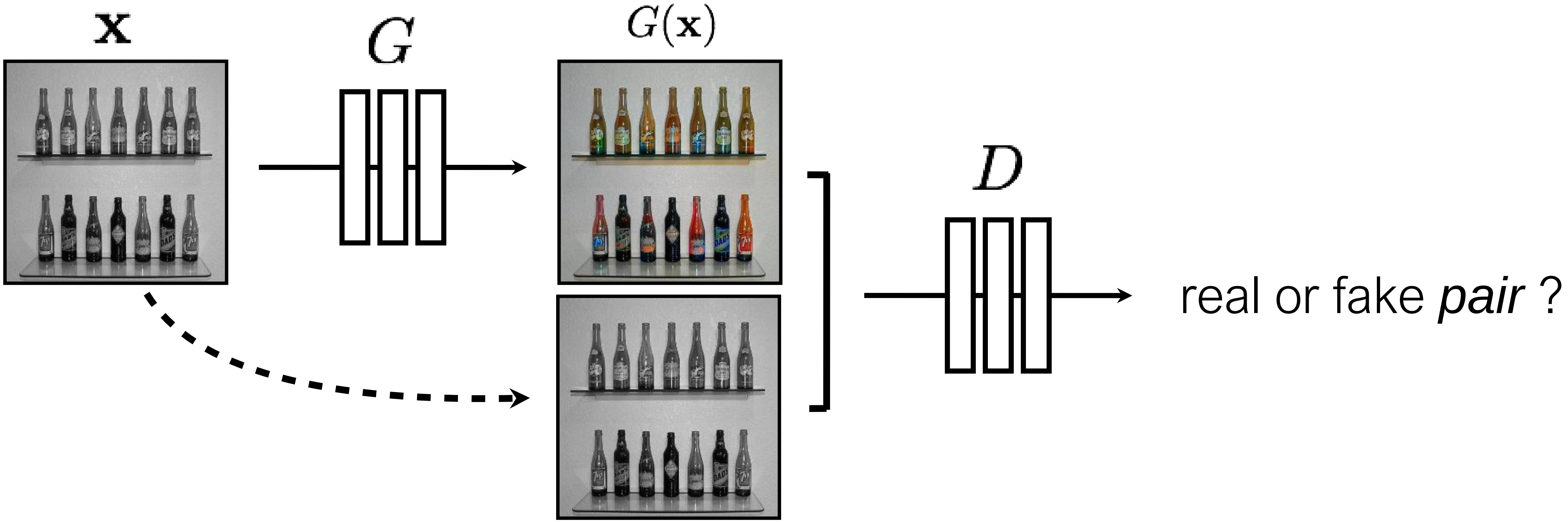
$$\arg\min_G \max_D \; \mathbb{E}_{\mathbf{x},\mathbf{y}}[\; \log D(\boxed{\mathbf{x}}, G(\mathbf{x})) + \log(1 - D(\boxed{\mathbf{x}}, \mathbf{y})) \;]$$

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



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

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



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

[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

Data from [Russakovsky et al. 2015]

Input

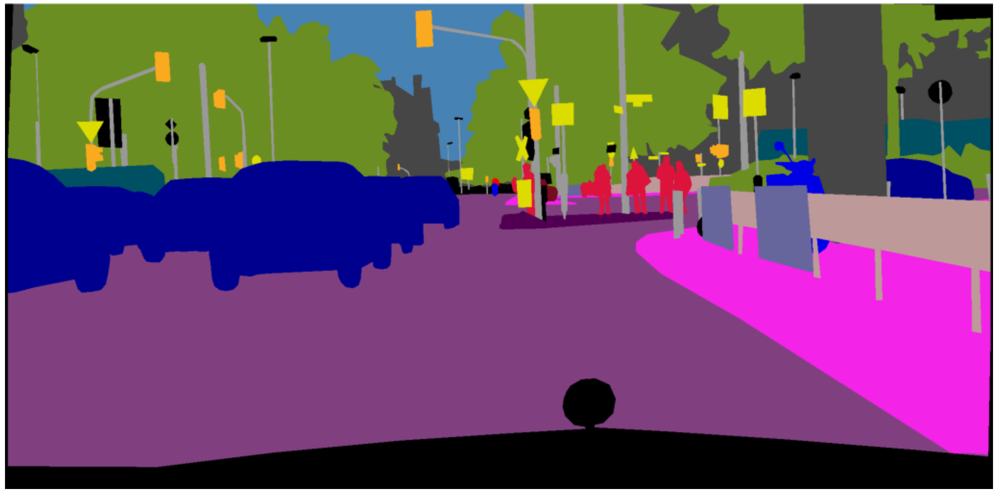
Output

Groundtruth

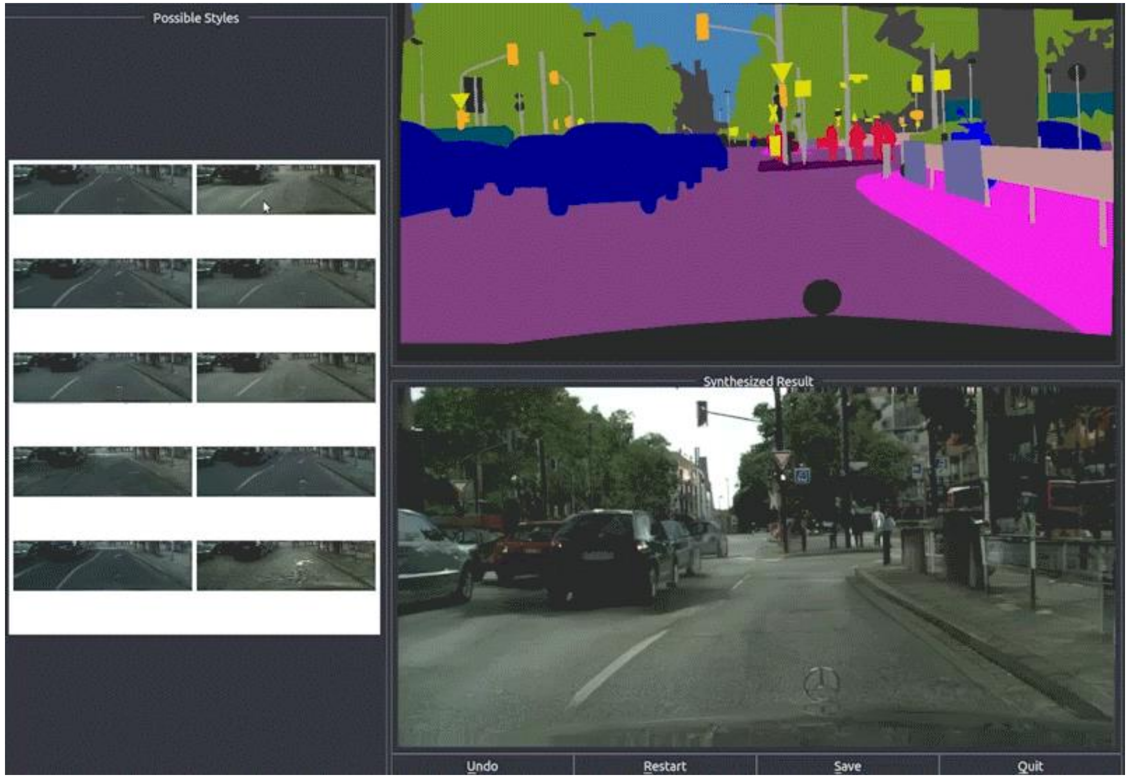
Data from
[maps.google.com]

Labels → Street Views

Input labels

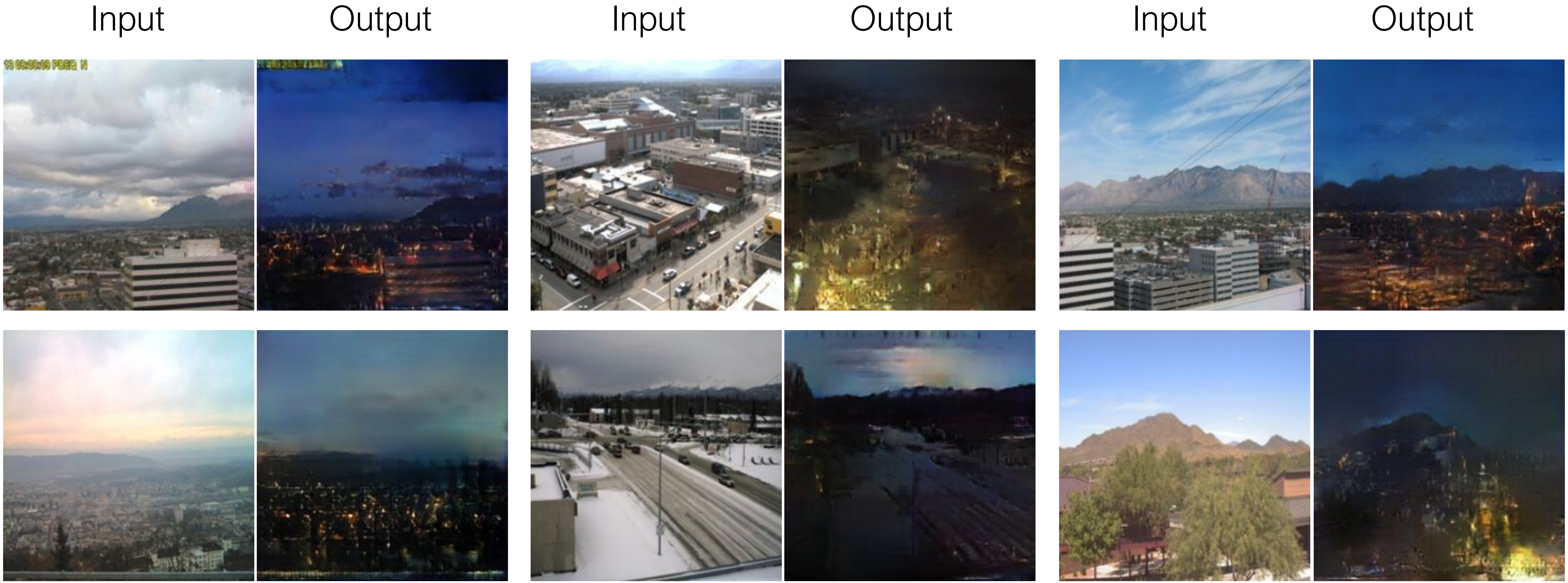


Synthesized image



Data from [Wang et al, 2018]

Day → Night



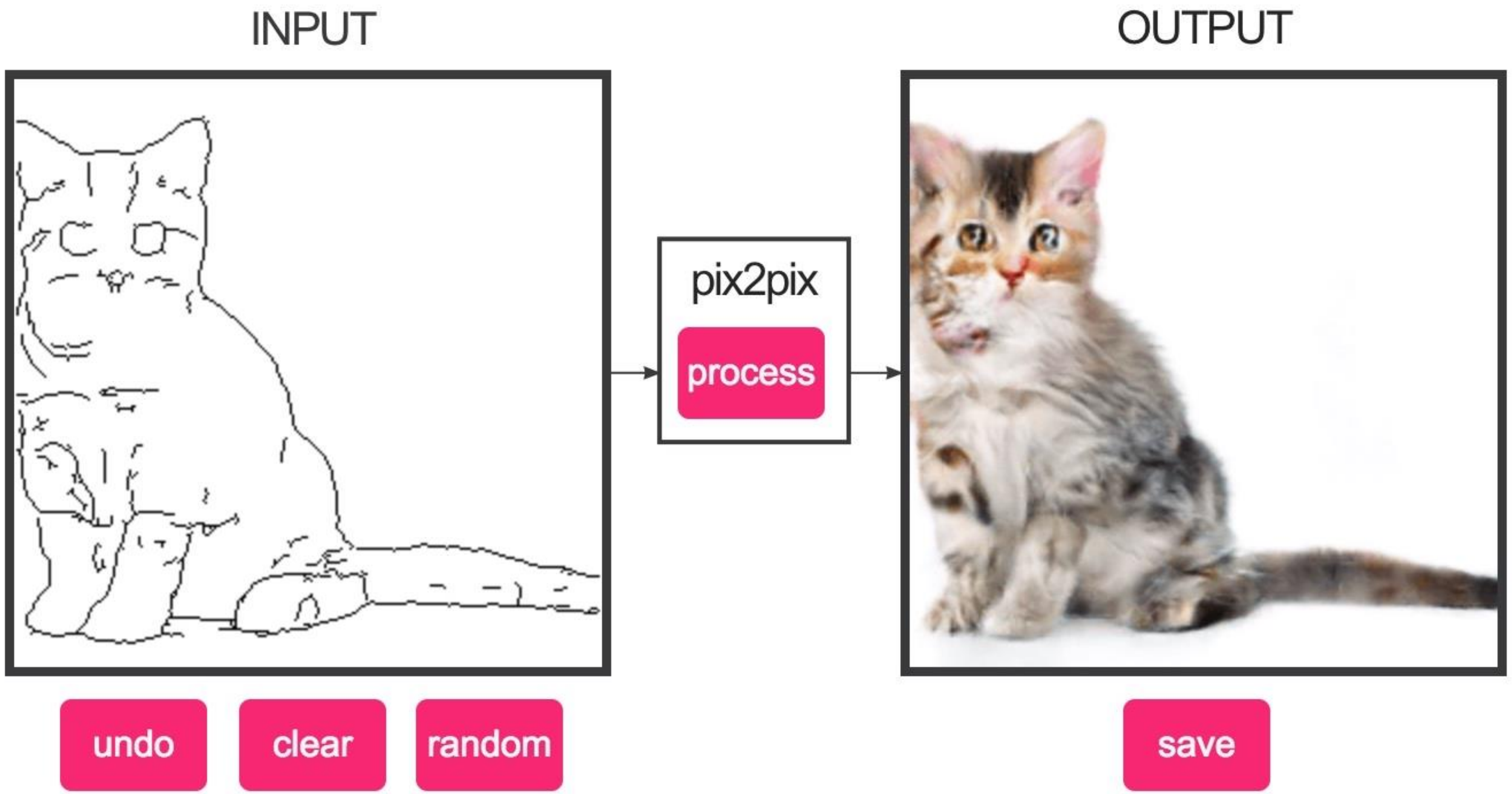
Data from [Laffont et al., 2014]

Edges → Images

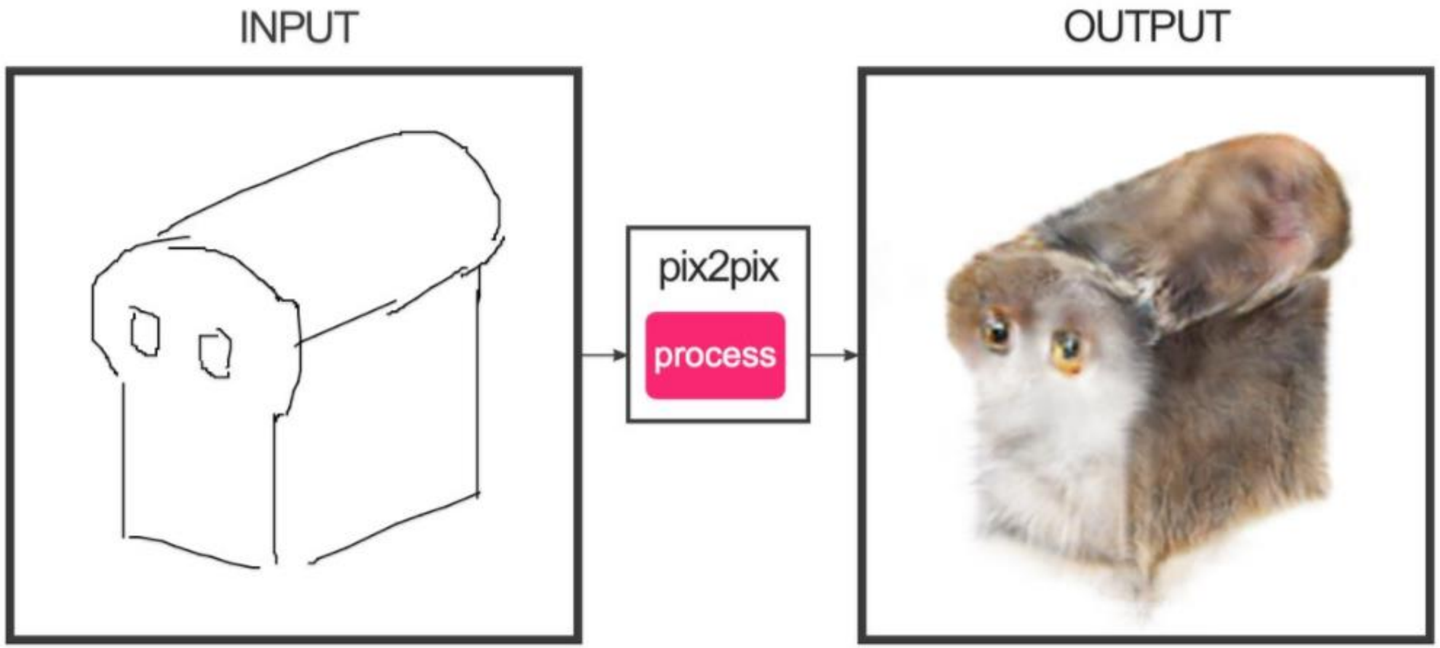
Input Output Input Output Input Output

Edges from [Xie & Tu, 2015]

Demo



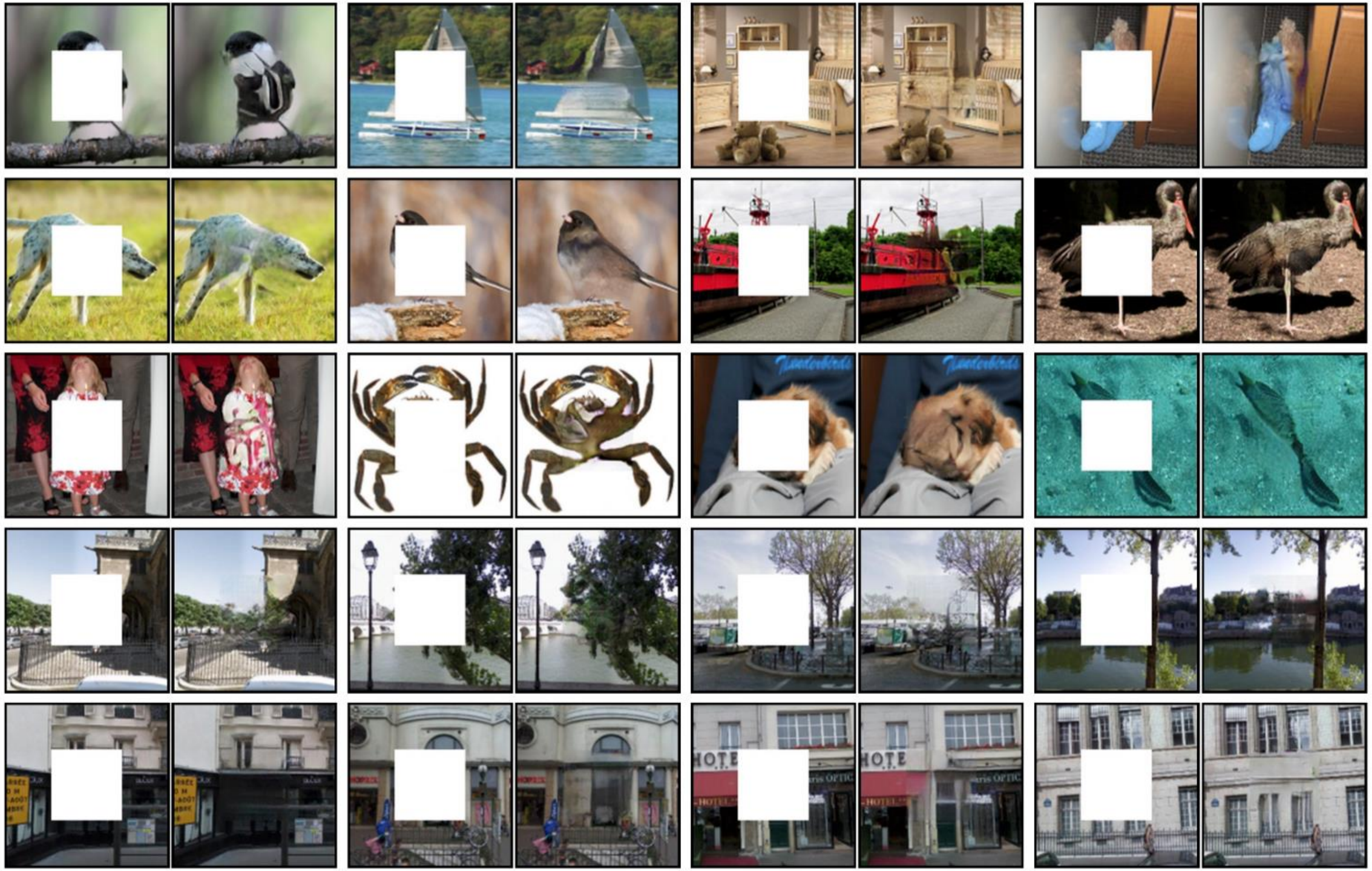
https://affinelayer.com/pixsrv/



Ivy Tasi @ivymyt

Vitaly Vidmirov @vvid

Image Inpainting



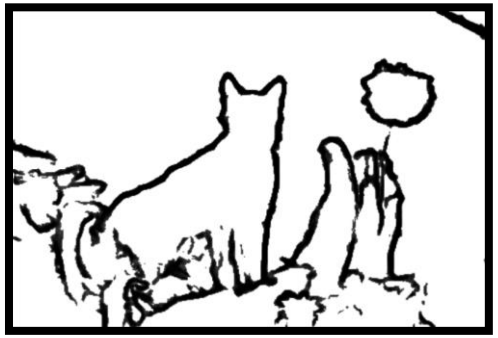
Data from [Pathak et al., 2016]

Pose-guided Generation

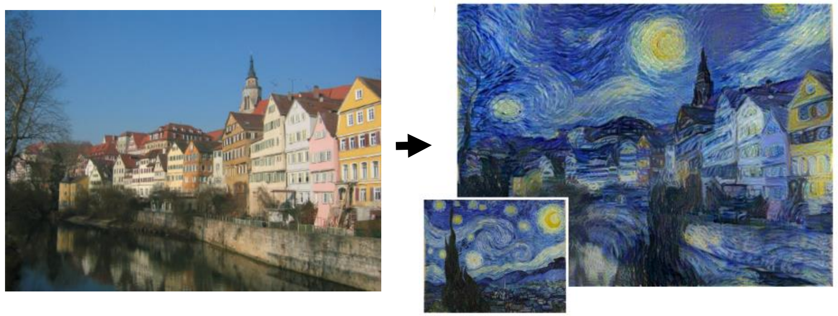
Data from [Ma et al., 2018]

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
- Lack of supervised training data
  - Approach: ?



"this small bird has a pink breast and crown…"



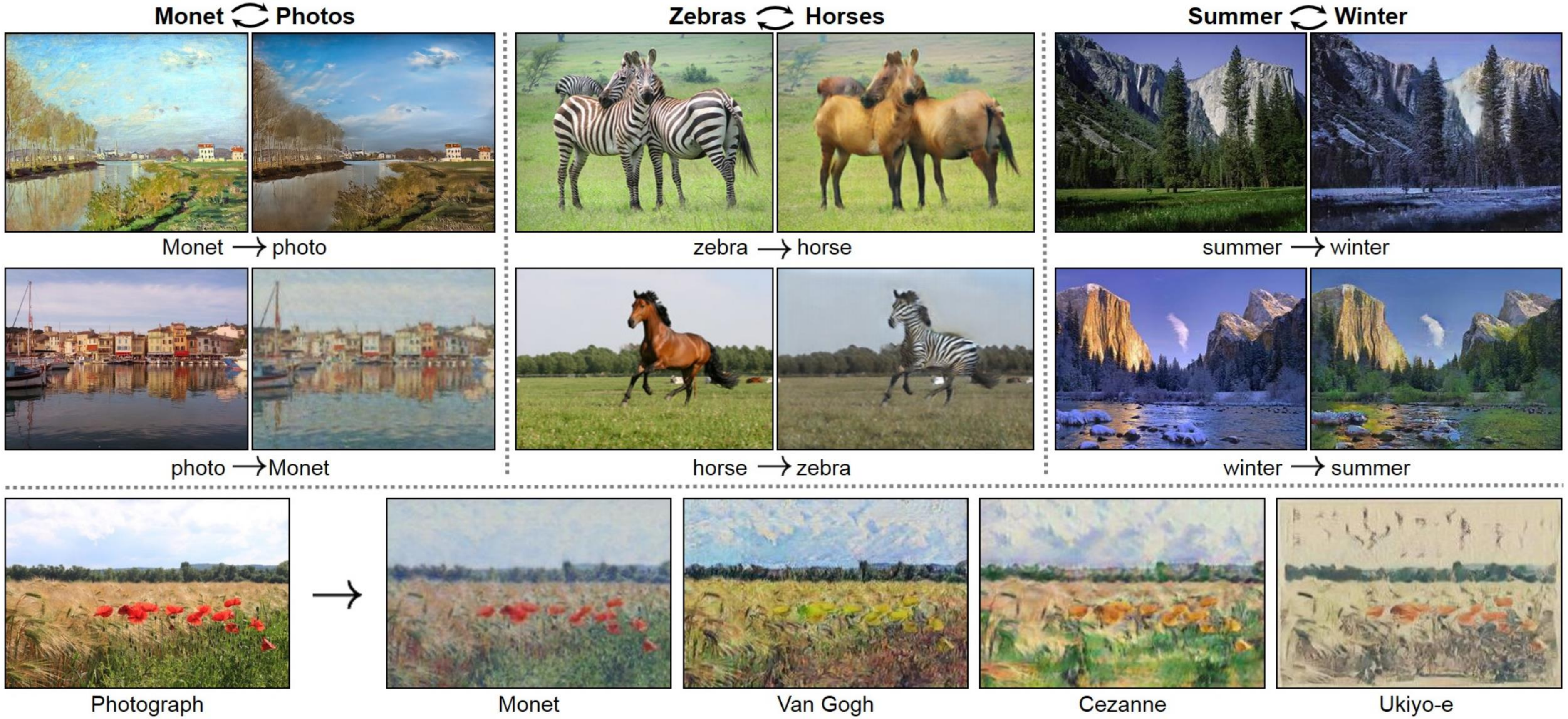
Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

Jun-Yan Zhu* Taesung Park* Phillip Isola Alexei A. Efros

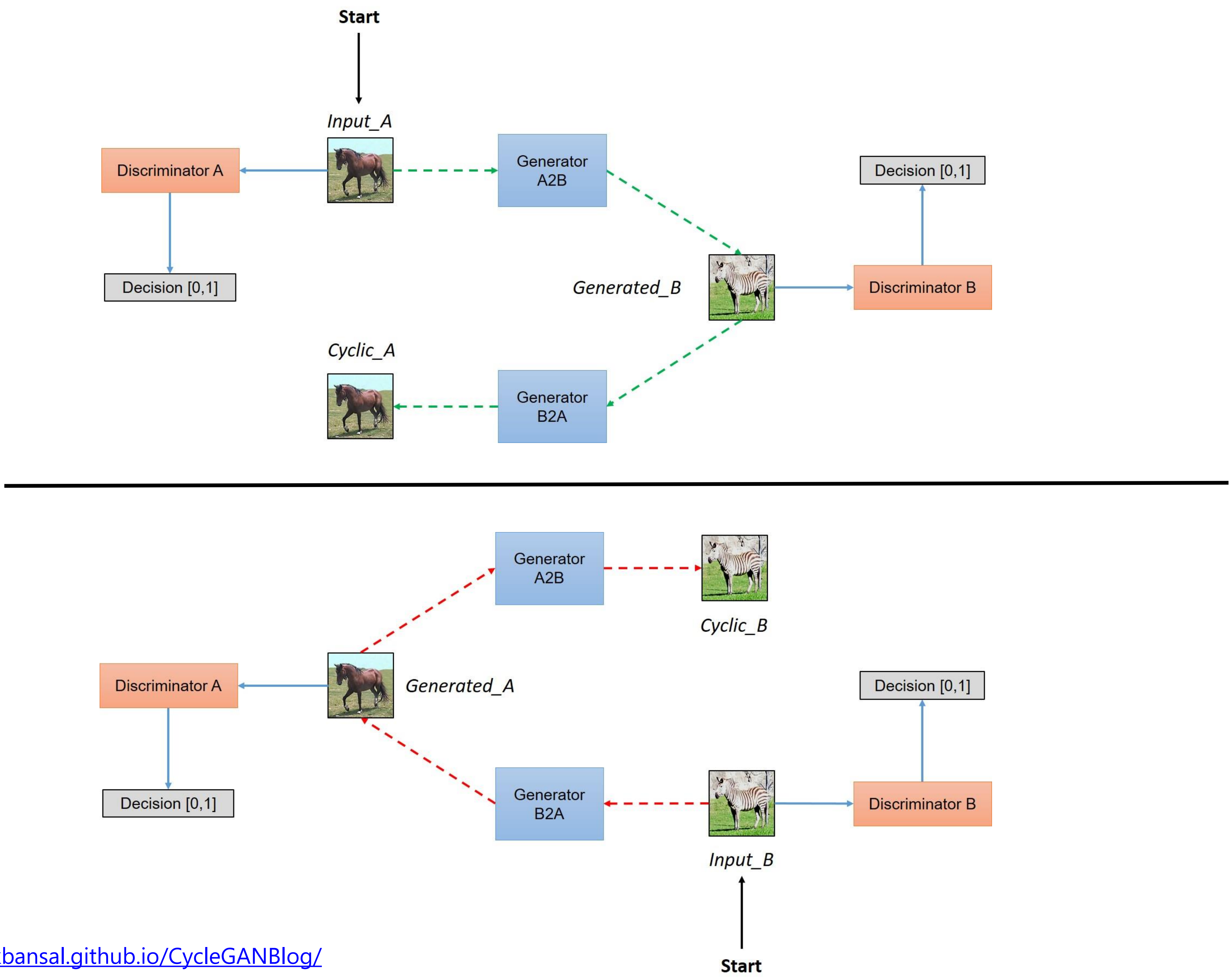
UC Berkeley

In ICCV 2017

[Paper] [Code (Torch)] [Code (PyTorch)]



https://junyanz.github.io/CycleGAN/



https://hardikbansal.github.io/CycleGANBlog/

StyleGAN

A Style-Based Generator Architecture for Generative Adversarial Networks

Tero Karras, Samuli Laine, Timo Aila

https://github.com/NVlabs/stylegan

StyleGAN2

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

Questions?