Learning-based and Image-based Rendering

Computer Graphics and Imaging
UC Berkeley CS184
Summer 2020

Agenda

- Image synthesis
  - Style Transfer
  - Relighting
  - (Image Colorization if time permits)
- Generative models
  - GAN overview
  - GAN for image synthesis

Brief Recap

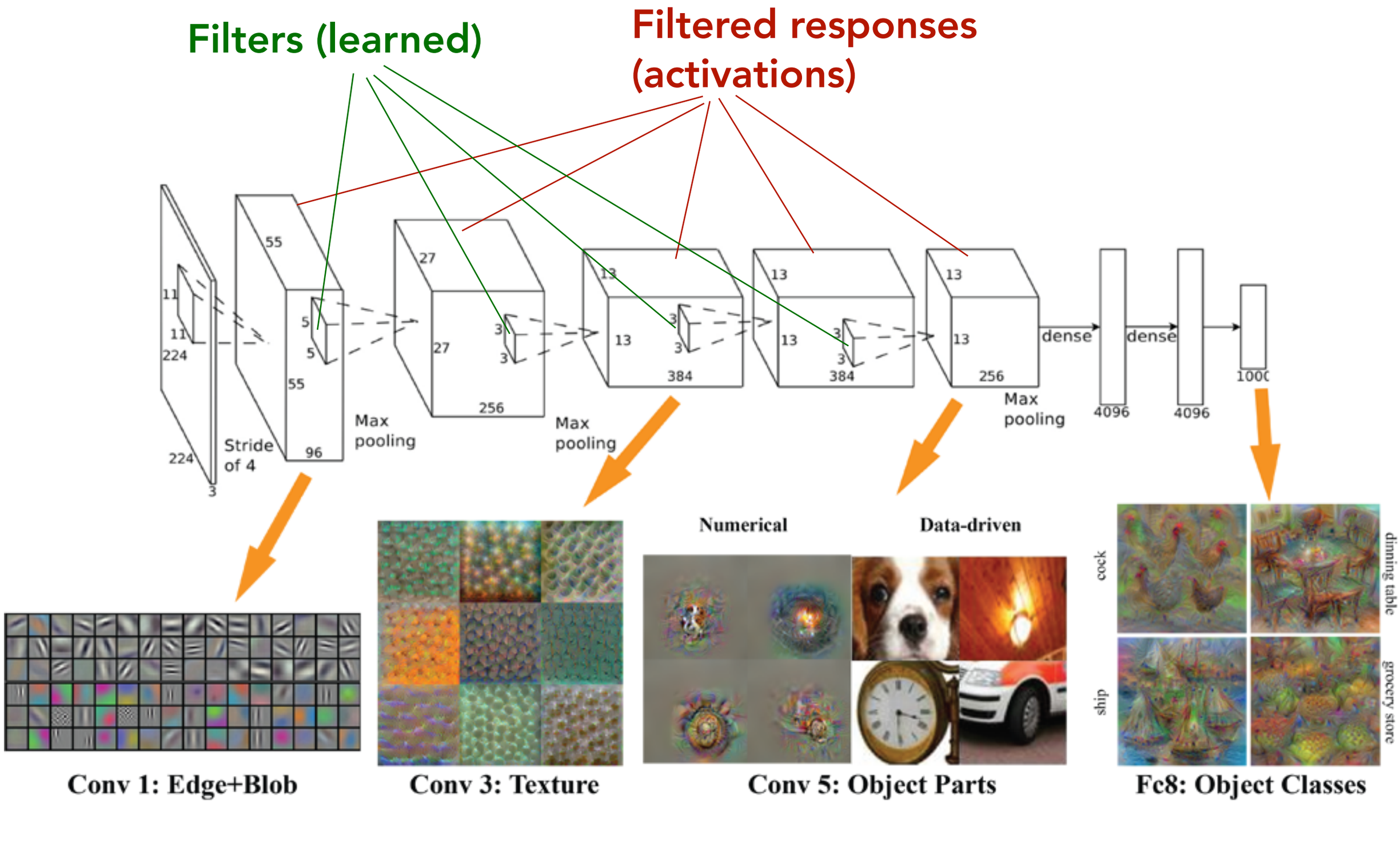
- Image features
  - Histogram of gradients
  - Robust to geometry and lighting changes
  - Used for image recognition, alignment, HDR, denoising, stitching
- Dataset for vision and graphics
  - Generic and subject-specific datasets
  - Data bias
- Deep learning overview
  - Convolutional layers
  - Loss function and gradient descent
  - Learning a probability given the input
  - Output can be a label, image, mesh, point cloud, etc.

Image Synthesis Challenges

- High dimensional output, structured objects
- Uncertainty in mapping; many plausible outputs
- Lack of supervised training data
- Good image quality! (e.g. photo-realism, visually pleasing, etc.)

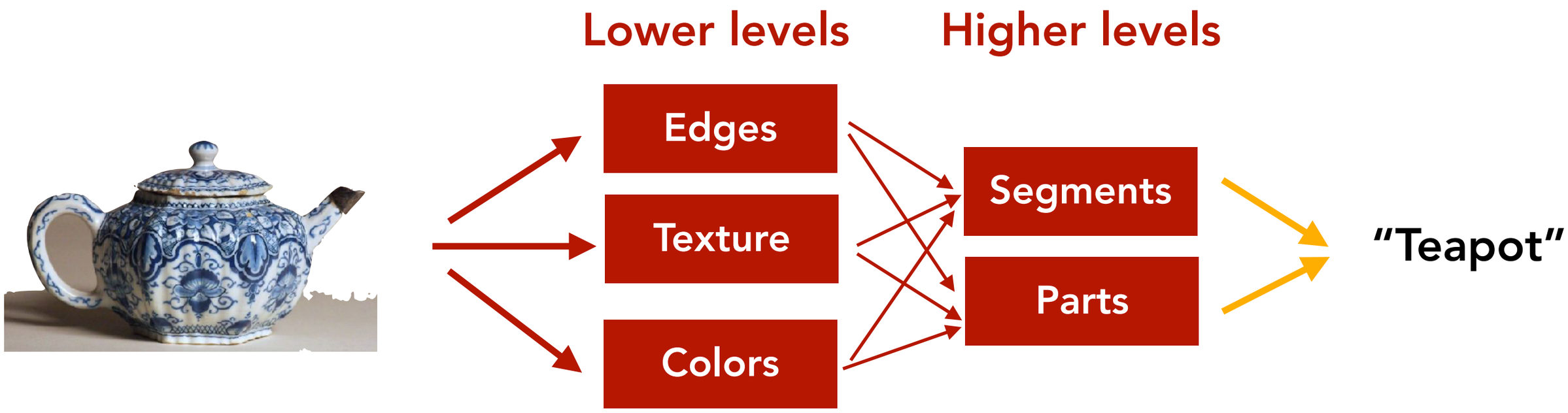
Neural Style Transfer

Visualizing Different CNN Layers



Visualizing Different CNN Layers

- Different information learned at different layers
  - From concrete to abstract
- Low-level features — edges, textures, etc.
- High-level features — semantics, parts
- Why matter? style — low level features; content — high level features



Neural Style Transfer

- Given a content image and a style image, find a new image that
  - Matches the CNN features of the content image (feature reconstruction)
  - Matches the Gram matrices of the style image (texture synthesis)

Content

+

Style

=

Result

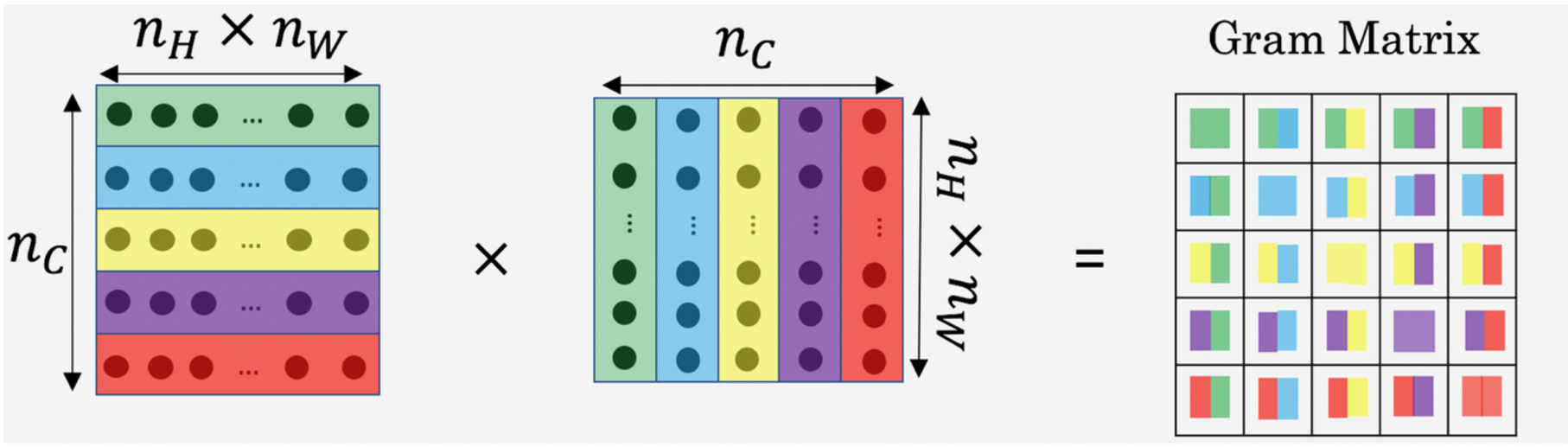
Gram Matrix

$$G_l \in R^{N_l \times N_l}$$

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{jk}^{l}$$

- **Square matrix that contains the dot products between each vectorized feature maps**

- G_{ij}^{l} **: inner product between the vectorized feature map** i **and** j **at layer** l



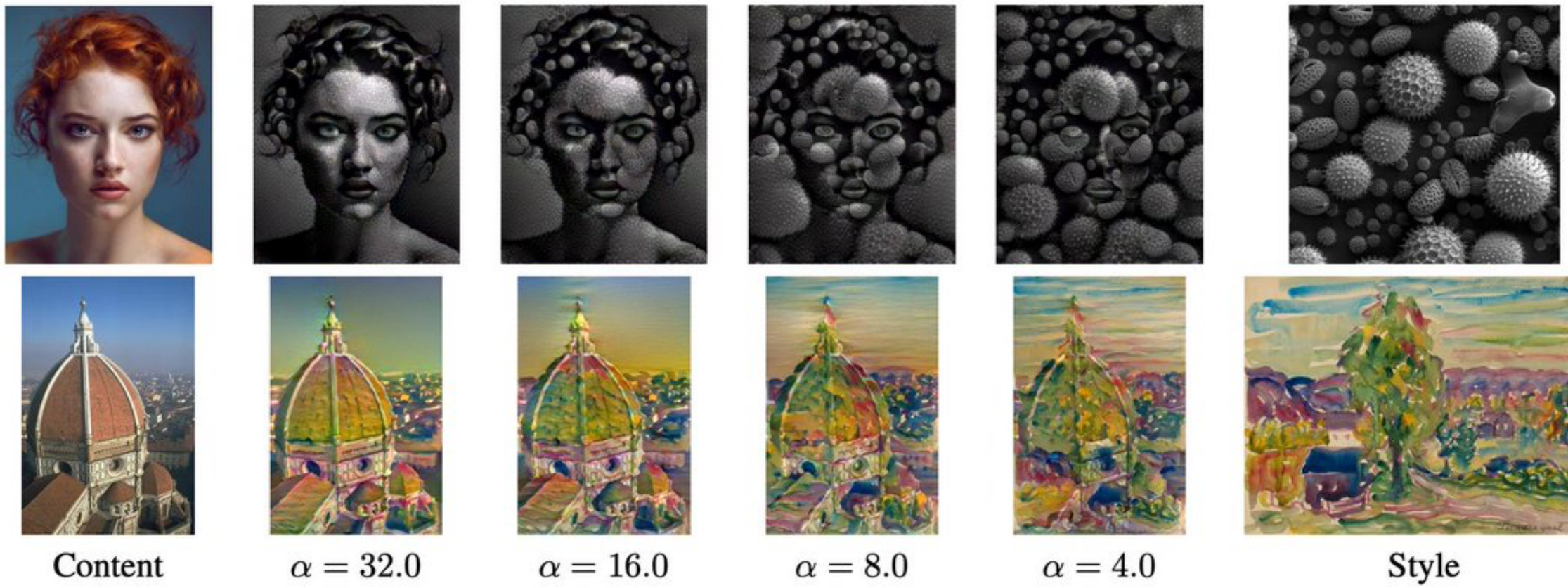
Objectives

$$L = \alpha L_{content}(C, G) + \beta L_{style}(S, G)$$

- G: Generated image
- C: Content image
- S: Style image
- α, β: Weights of content and style

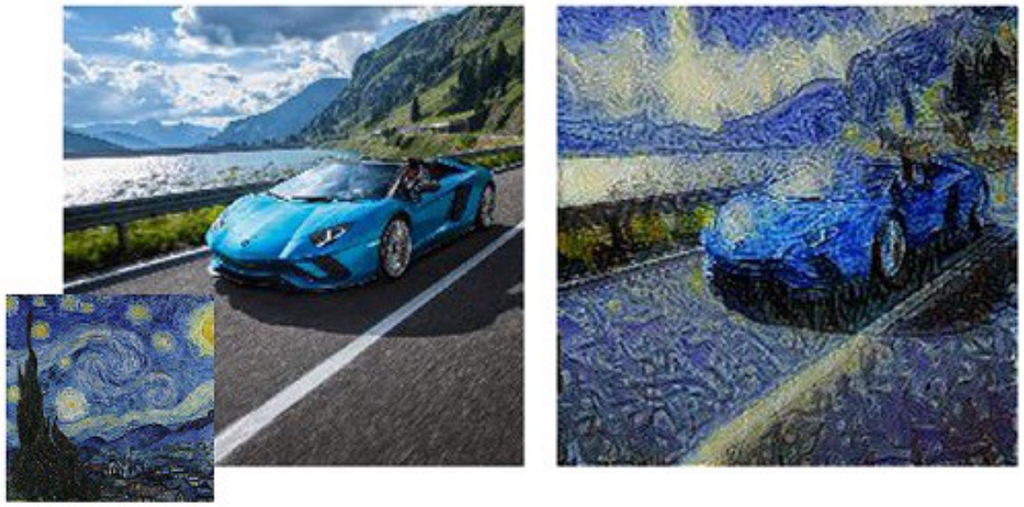
Effects of Varying α and β

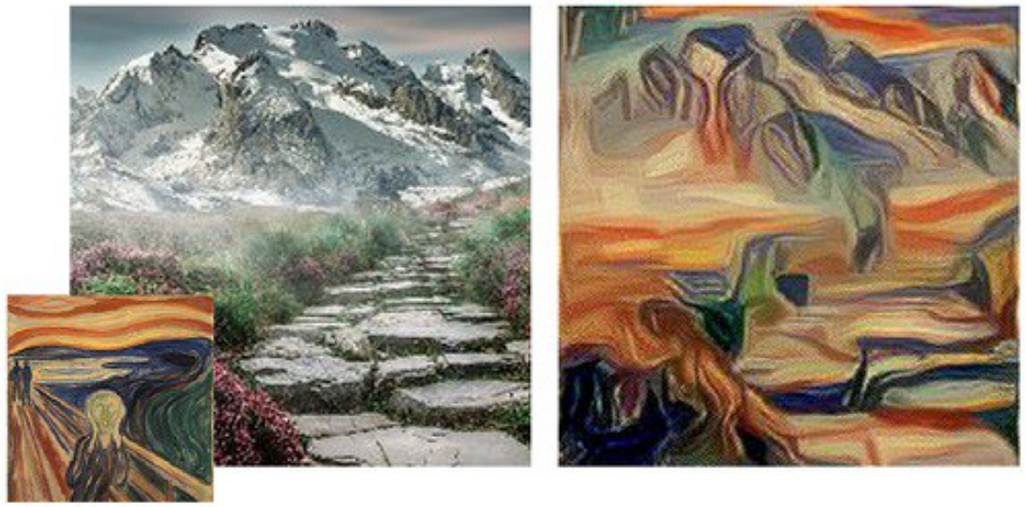
$$L = \alpha L_{content}(C, G) + \beta L_{style}(S, G)$$

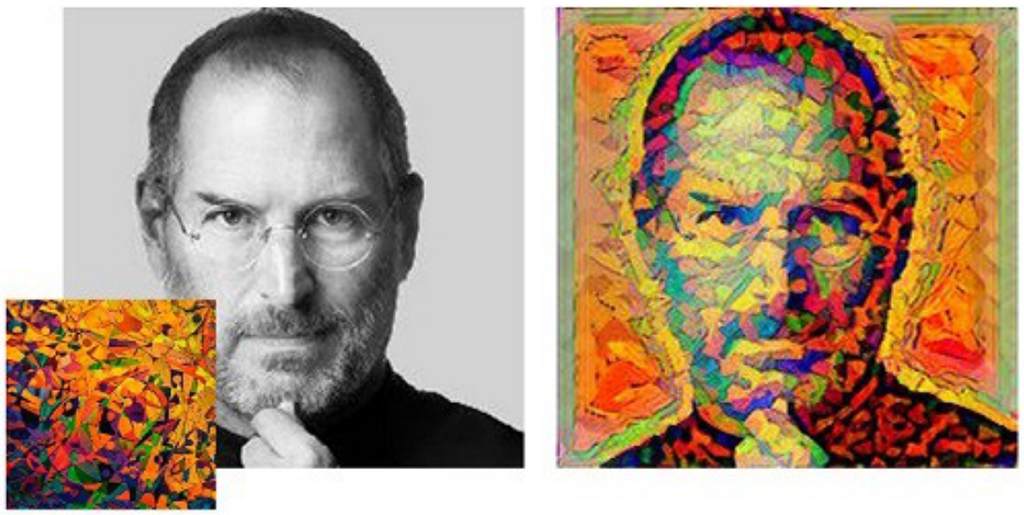


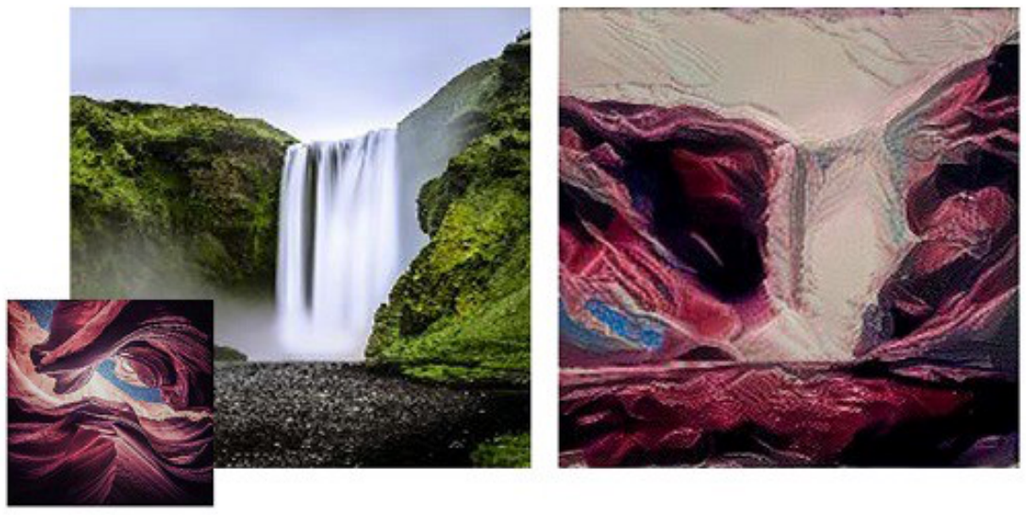
Content $\alpha = 32.0$ $\alpha = 16.0$ $\alpha = 8.0$ $\alpha = 4.0$ Style

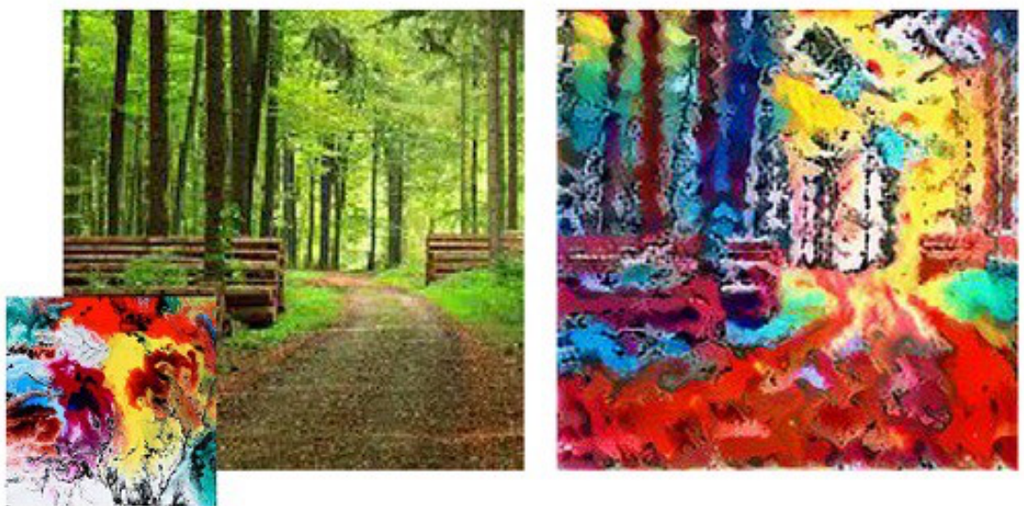
Results











Results on Videos

- Applied frame by frame

Pablo Picasso painting on glass in 1937.

Image-Based Relighting

Graphics Is Really Hard

- Photorealism takes a lot of efforts and time
- Hard to recreate complex geometry and lighting effects found in photos.

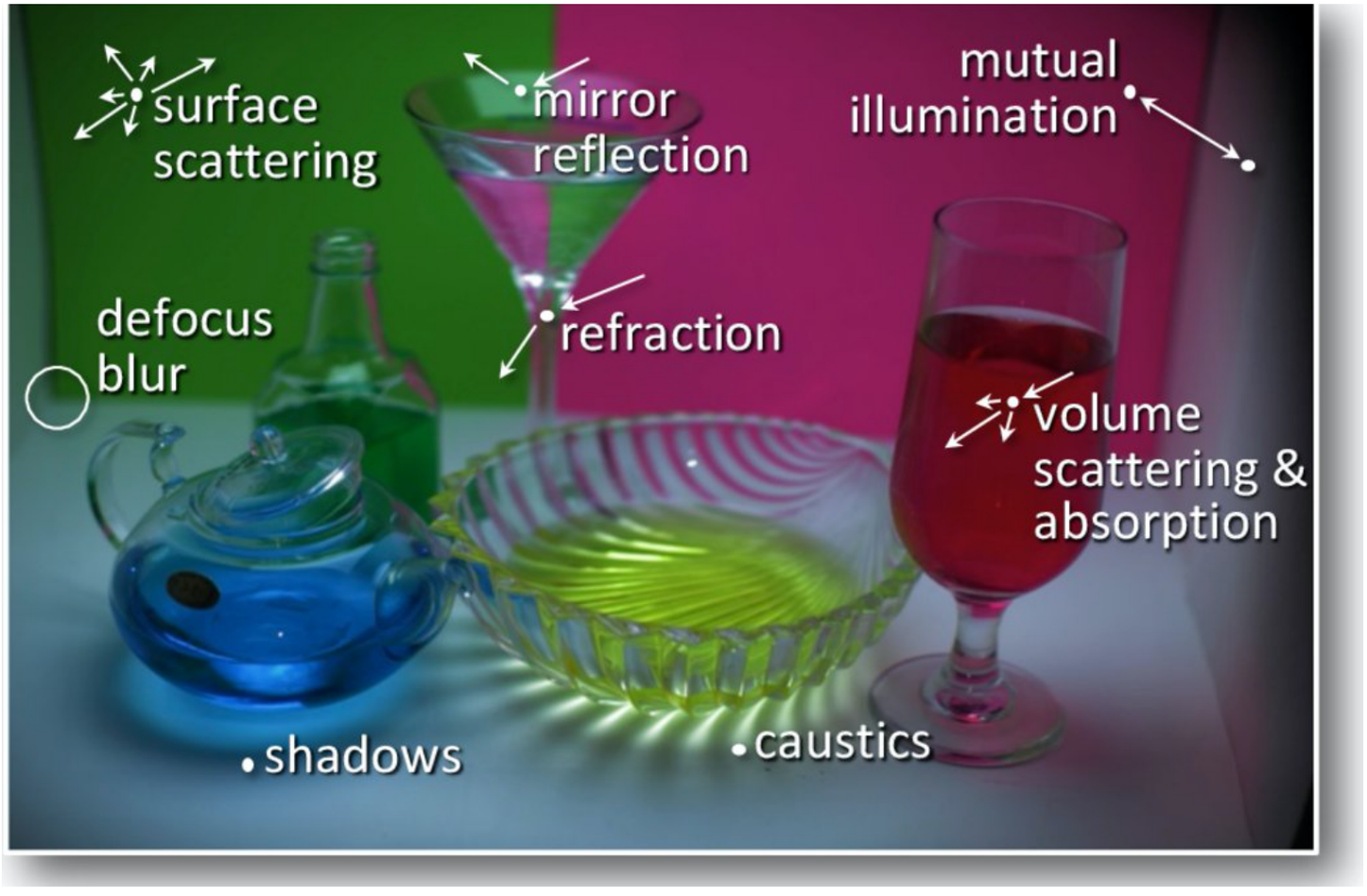


Image-Based Rendering

- Hard to recreate complex geometry and lighting effects found in photos — can we use photos instead?
- In fact, light field is one example
  - Photo-realistically changing viewpoints and other camera parameters (e.g. focus)

Image-Based Relighting

- Photo taken under two light sources = sum of photos taken under each source individually

- **Photo taken with light source emitting** axb **watts =** ax **photo taken with light source emitting** b **watts**

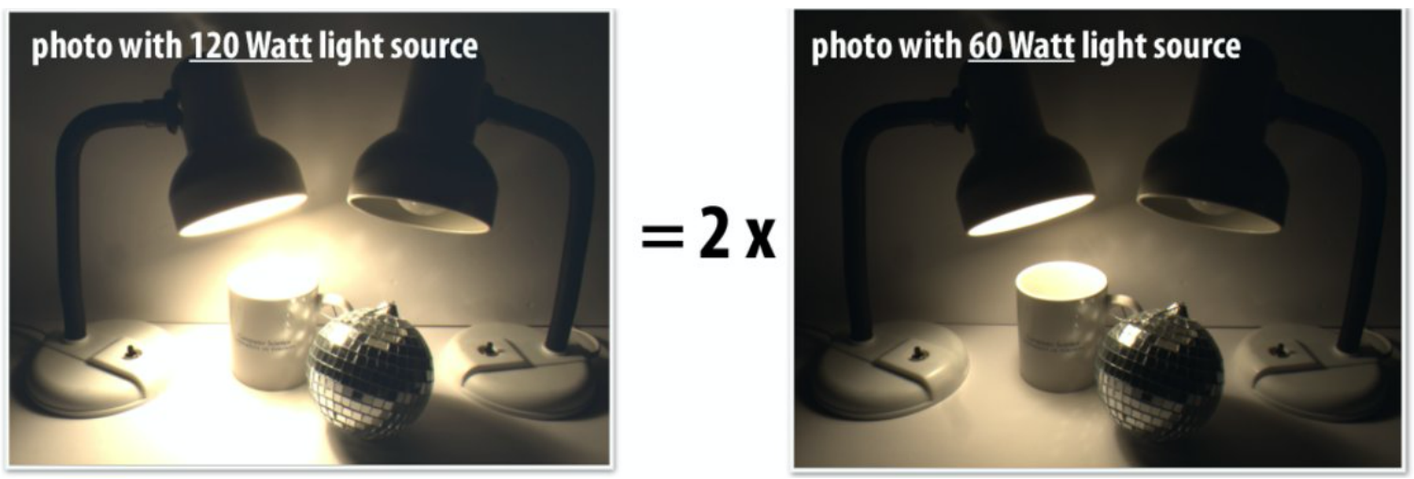


Image-Based Relighting

- Generate new photos by computing weighted combination of images!
  - e.g. adding the red channel of image 1 and the blue channel of image 2

Image-Based Relighting

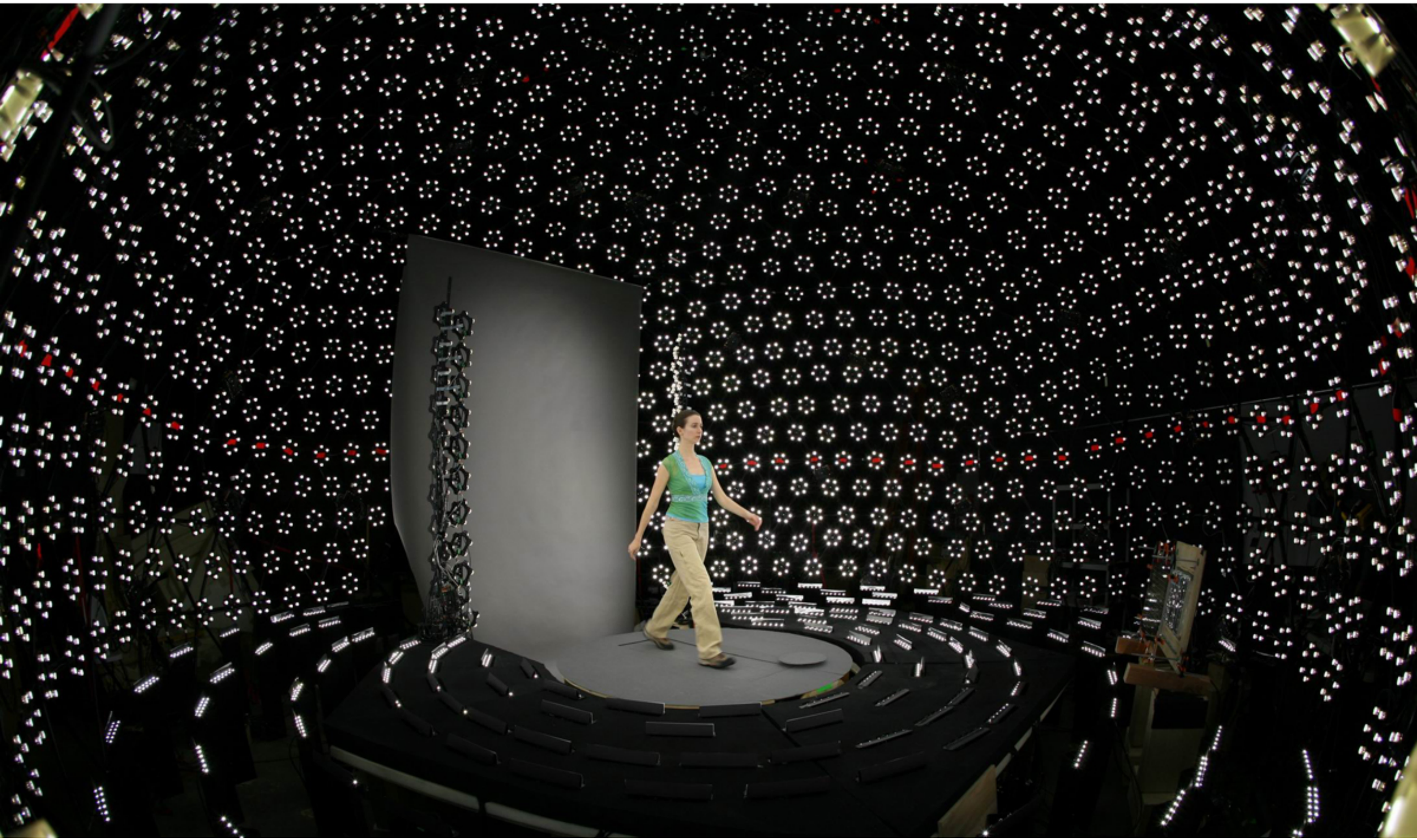
- We can come up with a complete set of rules for working with images u, v and w (all lit by different light sources)

For all vectors $\mathbf{u}, \mathbf{v}, \mathbf{w}$ and scalars a, b:

- $\mathbf{u} + \mathbf{v} = \mathbf{v} + \mathbf{u}$
- $\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}$
- There exists a *zero vector* "$\mathbf{0}$" such that $\mathbf{v} + \mathbf{0} = \mathbf{0} + \mathbf{v} = \mathbf{v}$
- For every $\mathbf{v}$ there is a vector "$-\mathbf{v}$" such that $\mathbf{v} + (-\mathbf{v}) = \mathbf{0}$
- $1\mathbf{v} = \mathbf{v}$
- $a(b\mathbf{v}) = (ab)\mathbf{v}$
- $a(\mathbf{u} + \mathbf{v}) = a\mathbf{u} + a\mathbf{v}$
- $(a + b)\mathbf{v} = a\mathbf{v} + b\mathbf{v}$

(Looks familiar!)

The Light Stage 6

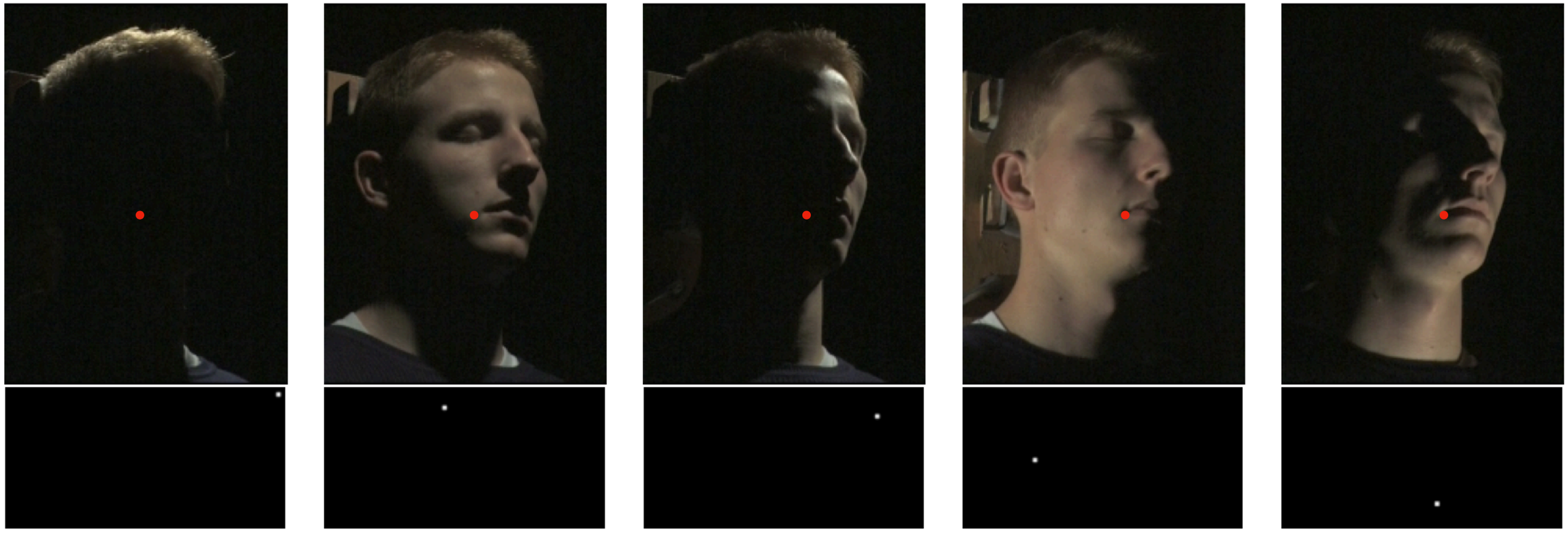


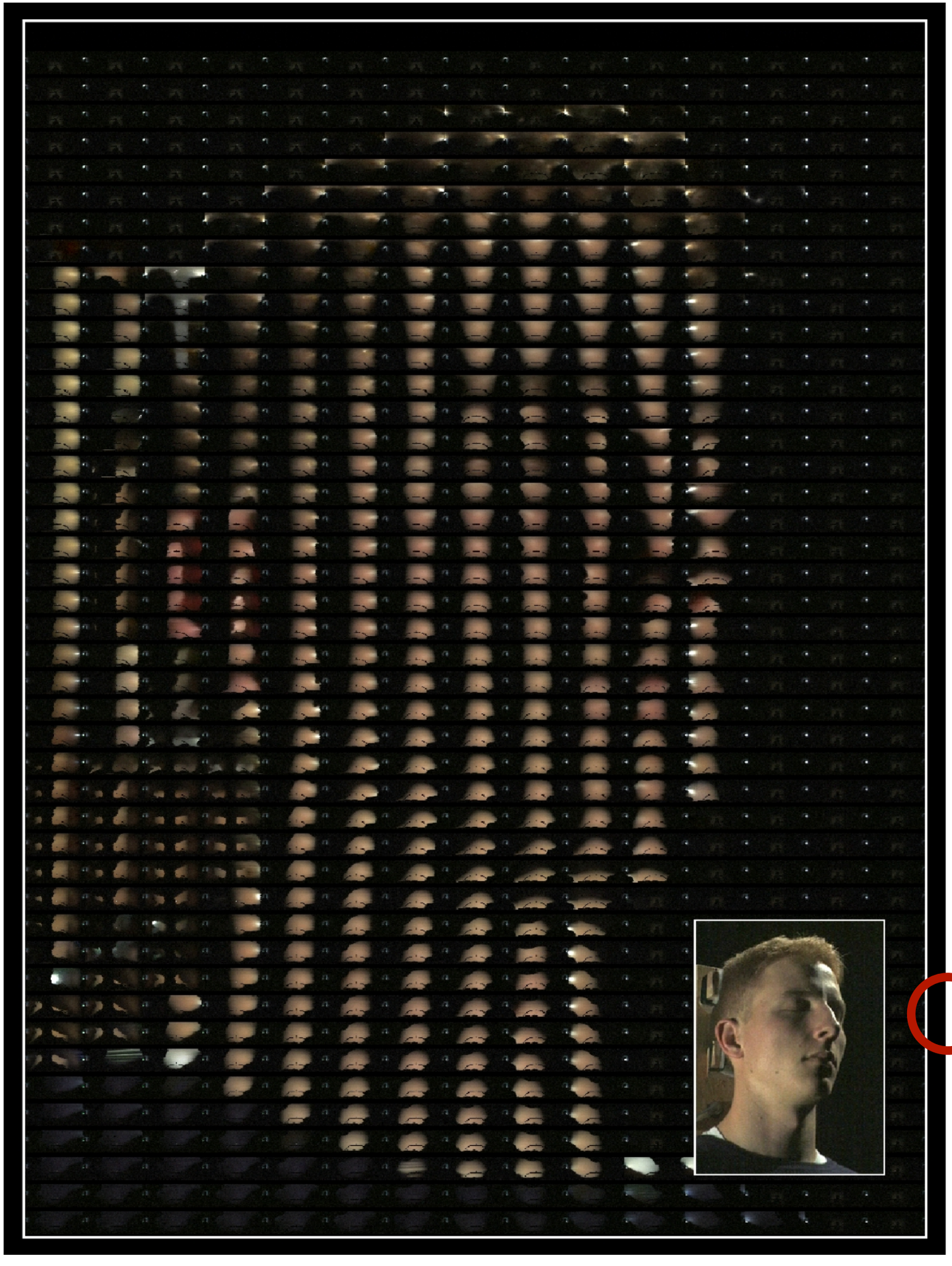
The Light Stage 1 from USC:ICT

Light Stage 1 movie from SIGGRAPH 2000 Electronic Theater

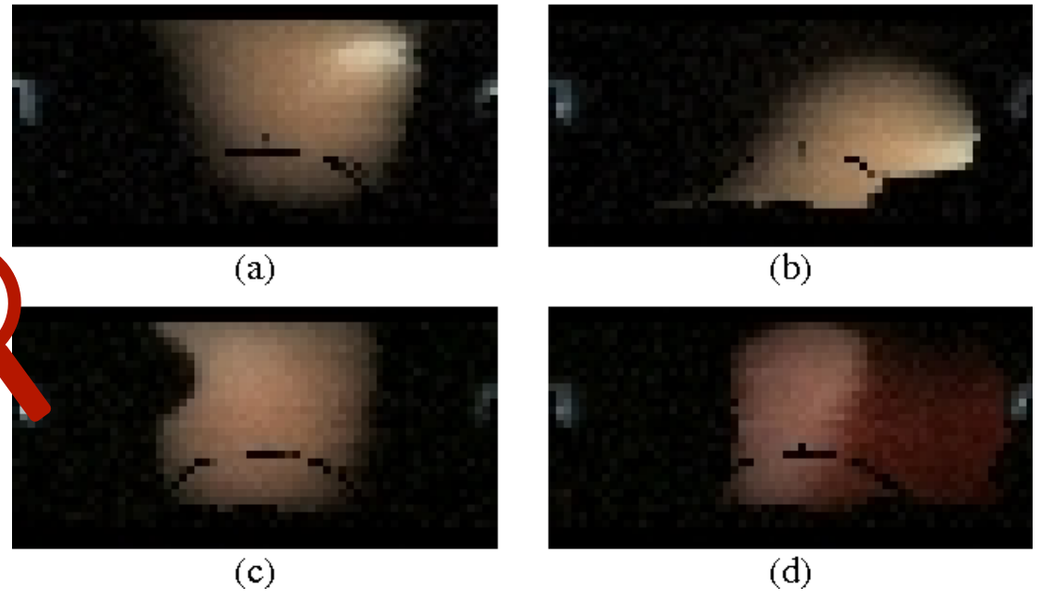
Light Stage Images

- Images captured by the Light Stage
- Images illuminated by 5 directional lights, in total there are 64 x 32 lights over the sphere



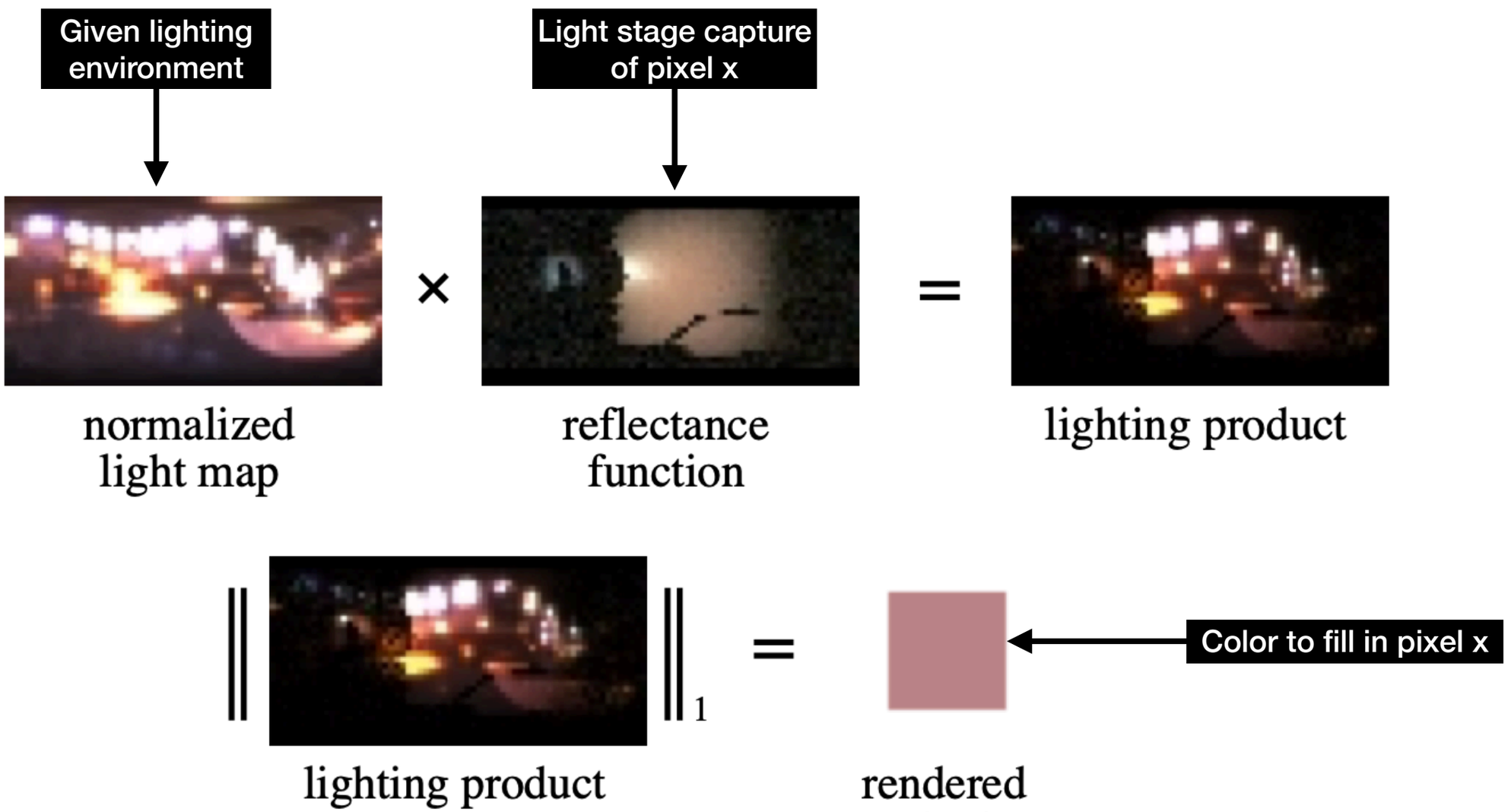


- Each 64 32 reflectance function consists of the corresponding pixel location's appearance under two thousand lighting directions distributed throughout the sphere. The inset shows the same view of the face under a combination of three lighting directions. The functions have been brightened by a factor of four from the original data.



How to Relight

- Given an environment map, for a pixel x:

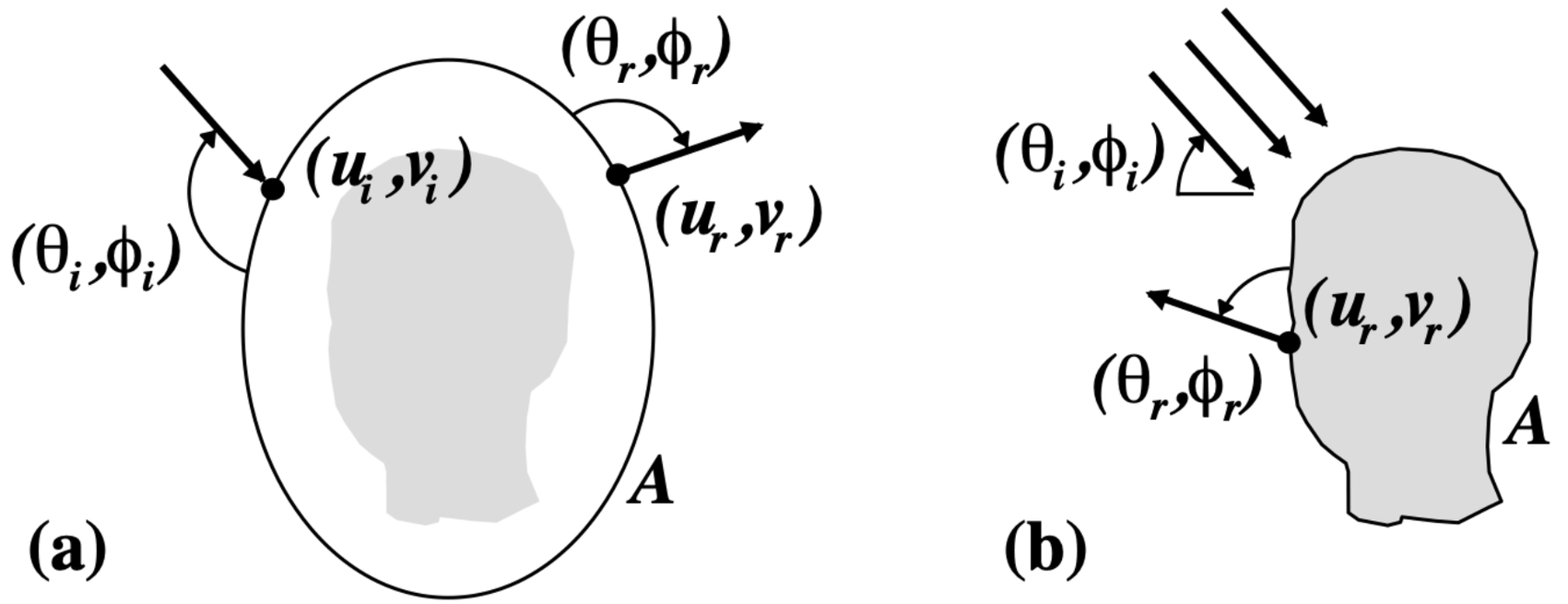


Light Stage Images and Relit Images

Mobile Light Stage From USC

3D Portrait of the President

Capturing the Reflectance Field



- Transform from an incident field of illumination $R_i(u_i, v_i, \theta_i, \phi_i)$ into a radiant field of illumination $R_r(u_r, v_r, \theta_r, \phi_r)$.

- The light stage uses specific R_i and records its R_r (the images)

Sampling the Reflectance Field

- 64 x 32 — no aliasing if light map is downsampled to 64 x 32 with proper filtering
  - Aliasing happens for high frequency lighting (e.g. stair stepped harsh shadows
- Not able to capture local effects
  - Consider rendering a face with a shaft of light hitting below the eye. The light below the eye would throw indirect illumination on the underside of the brow and the side of the nose — not able to render using the Light Stage 1

Combine CNN and the Light Stage

- Can we use the light stage images for training and apply to arbitrary image in the wild?

Input

3 examples of relit images under different lighting environment

Sun et al. “Single Image Portrait Relighting” SIGGRAPH 2019

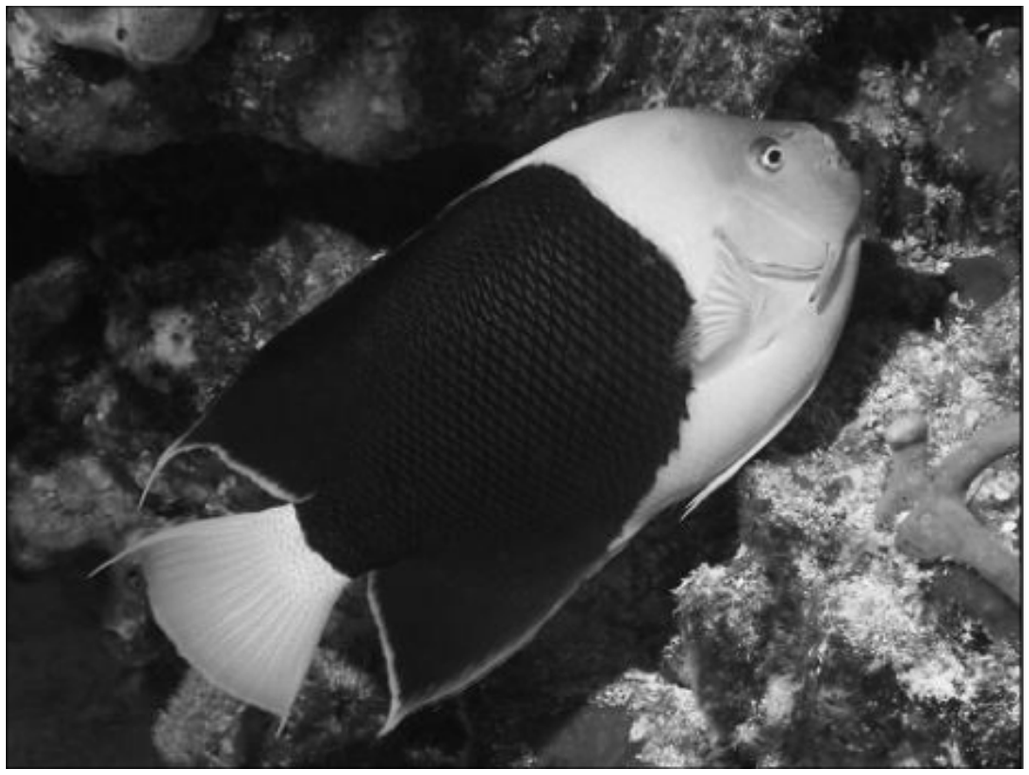
Image Colorization

Common 2D Loss Functions for CNN

- **L1 (MAE):** $L_1 = \sum_{i=1}^{n} |y_{predict} - y_{true}|$
- **L2 (MSE):** $L_2 = \sum_{i=1}^{n} (y_{predict} - y_{true})^2$
- Can operate in different image space (e.g. RGB, LAB, Image gradients, etc.)
  - Depend on the task

Ansel Adams. ***Yosemite Valley Bridge.***

Chroma From Luminance



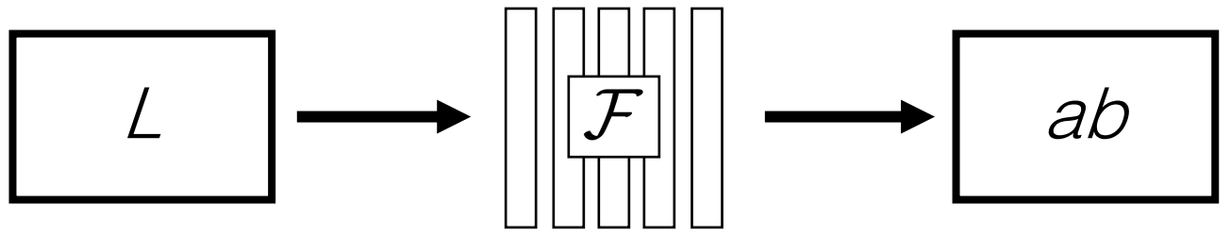
Grayscale image: *L* channel

$$\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$$

$\mathcal{F}$

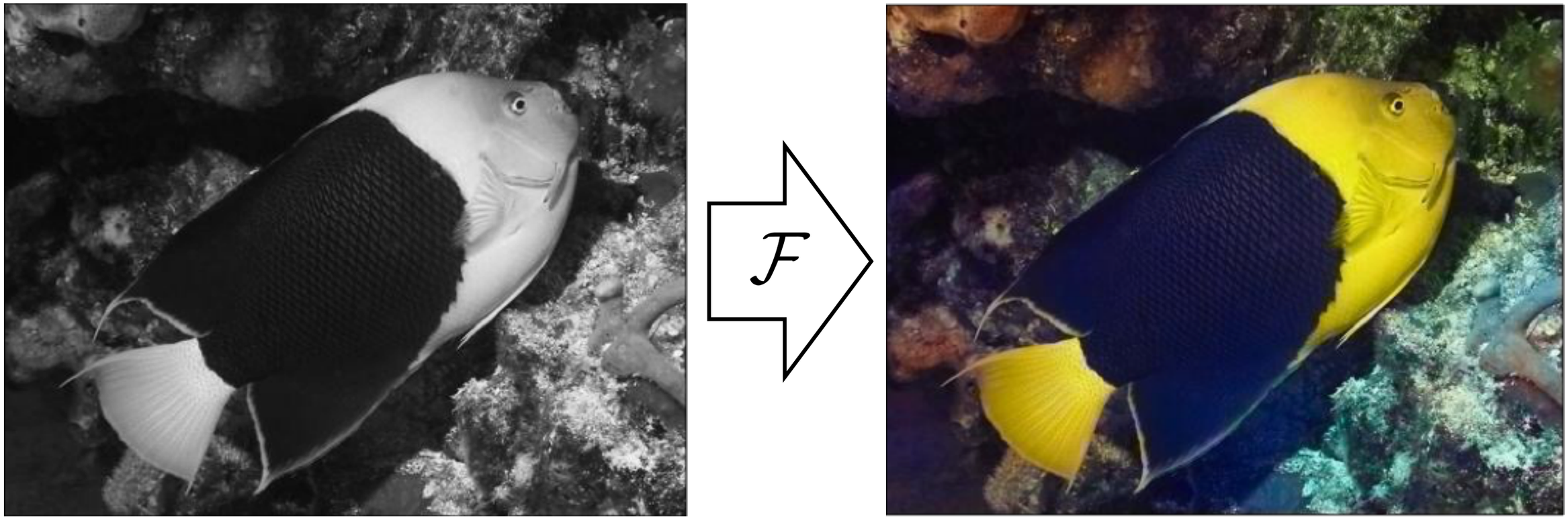
Color information: *ab* channels

$$\hat{\mathbf{Y}} \in \mathbb{R}^{H \times W \times 2}$$



Zhang, Isola, Efros. ***Colorful Image Colorization.*** In *ECCV*, 2016.

Chroma From Luminance

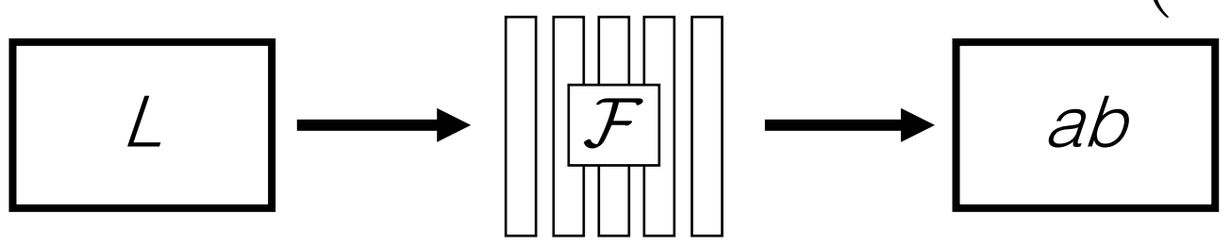


Grayscale image: *L* channel

$\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$

Concatenate (*L*,*ab*) channels

$(\mathbf{X}, \hat{\mathbf{Y}})$



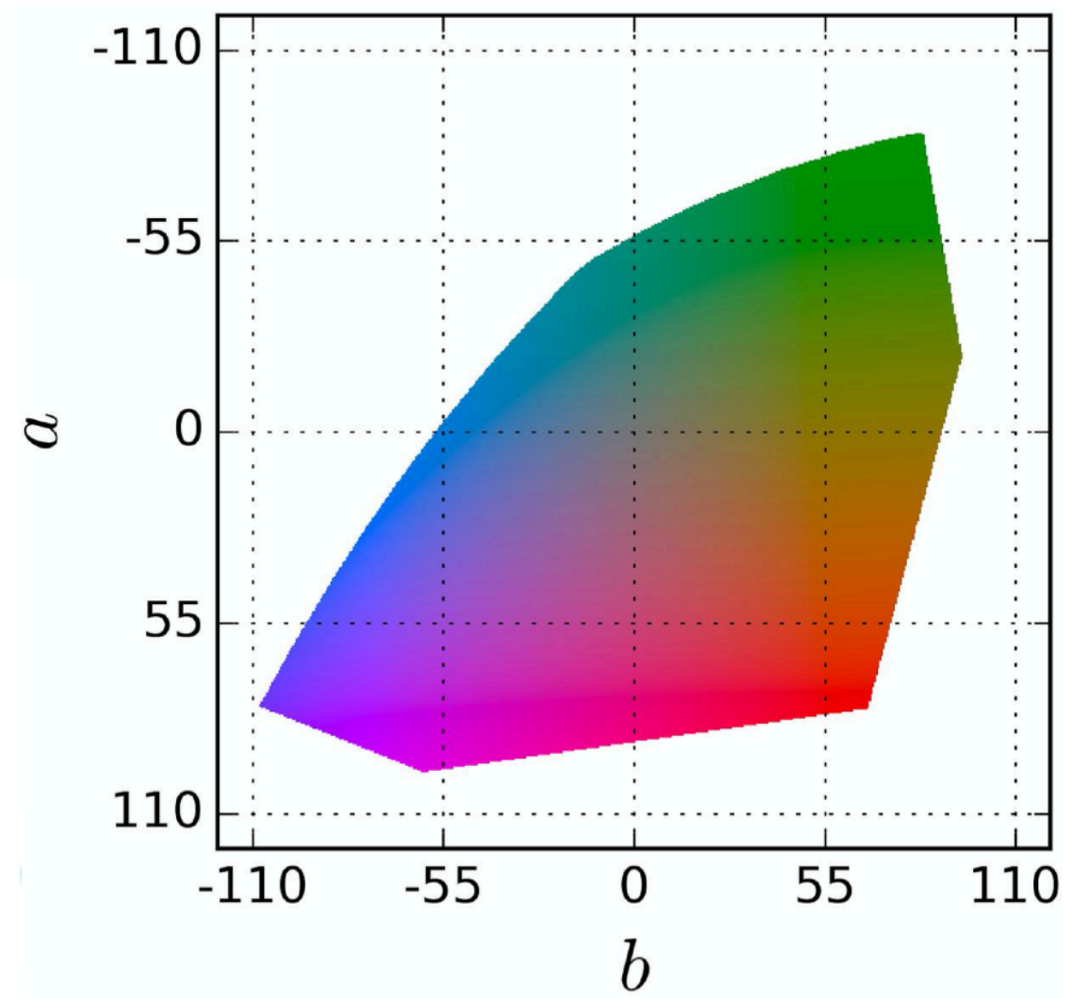
Zhang, Isola, Efros. ***Colorful Image Colorization.*** In *ECCV*, 2016.

Inherent Ambiguity

- One luminance image has Many colored versions
- Since the objective is also “ambiguous”

Better Loss Function

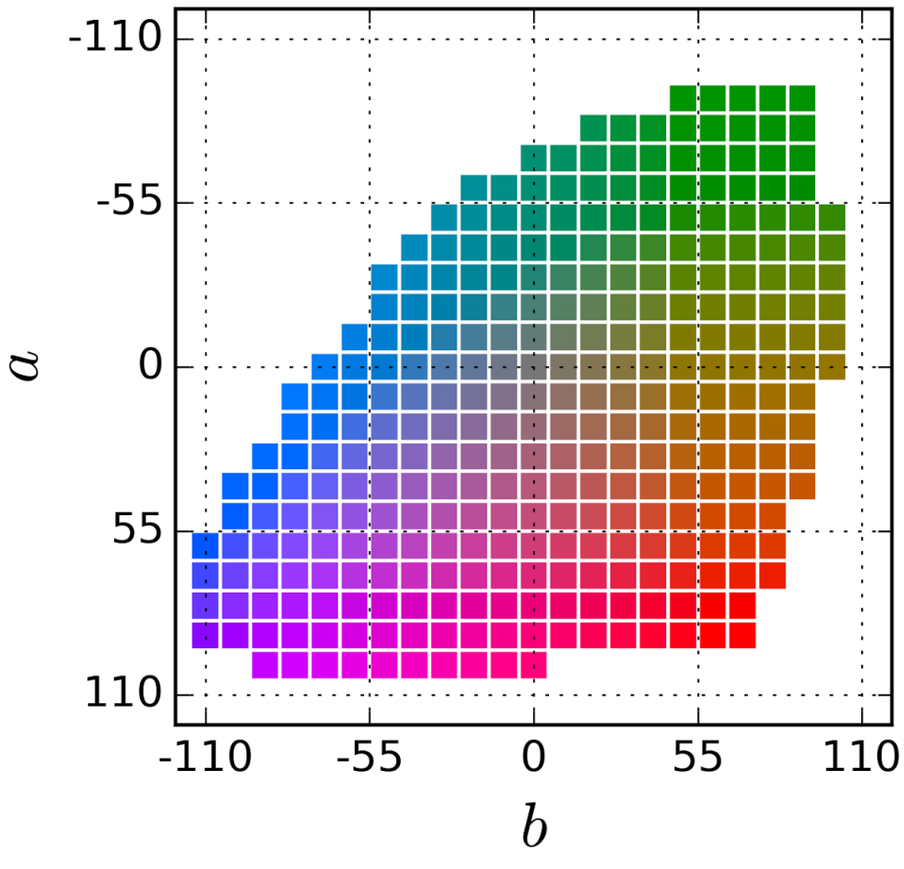
- Regression with L2 loss inadequate
- “Forcing” one right answer is not ideal



Color in AB Space

Learn a Probability in the Chromaticity Diagram

- Can predict a distribution of color options for each pixel!
- For a given image X, learn a mapping to a probability distribution over possible colors $\hat{Z} \in [0,1]^{H \times W \times Q}$
- Q is the the number of quantized ab values



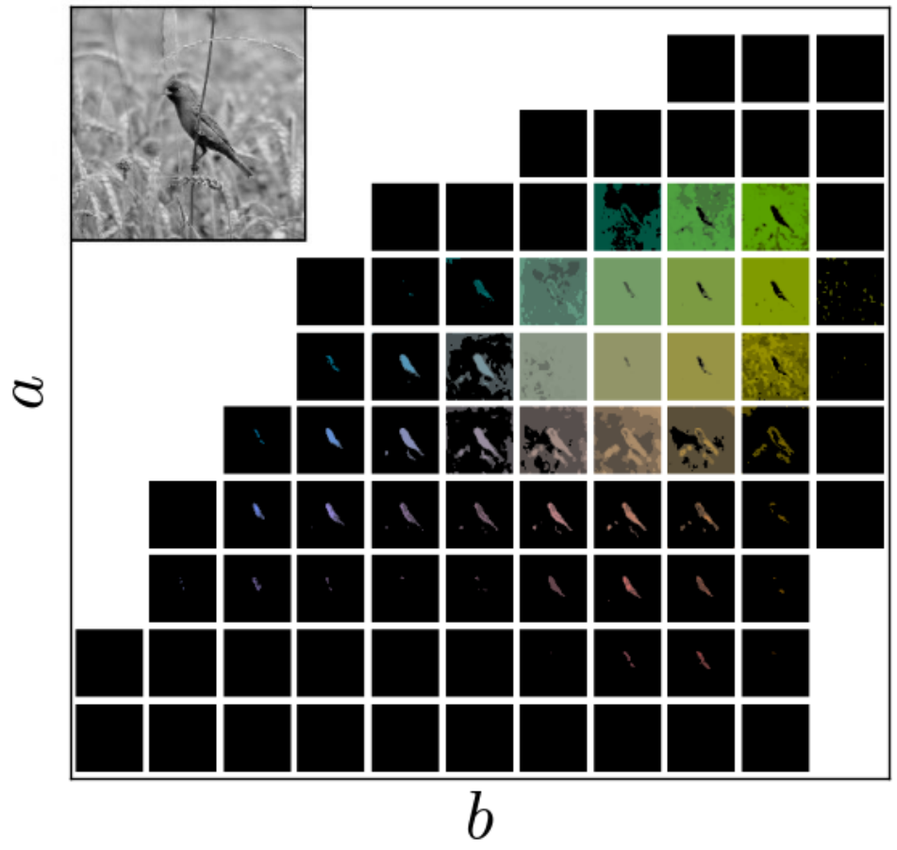
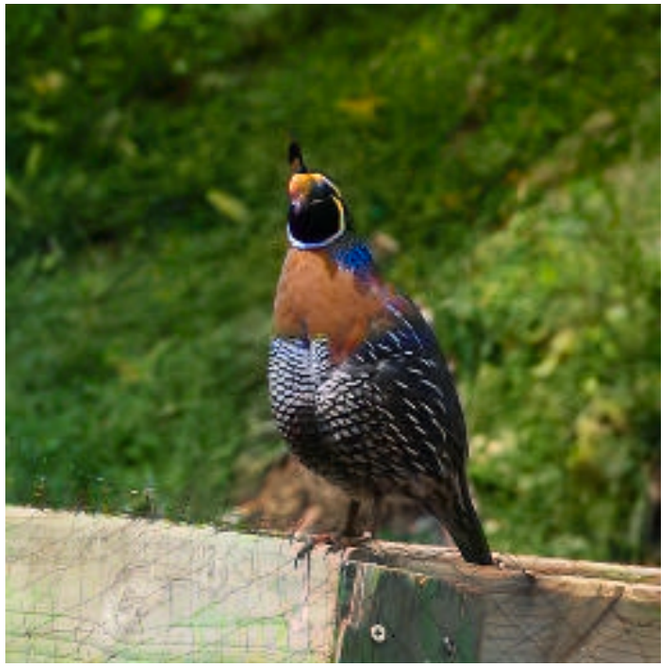


Image Colorization

Image Colorization



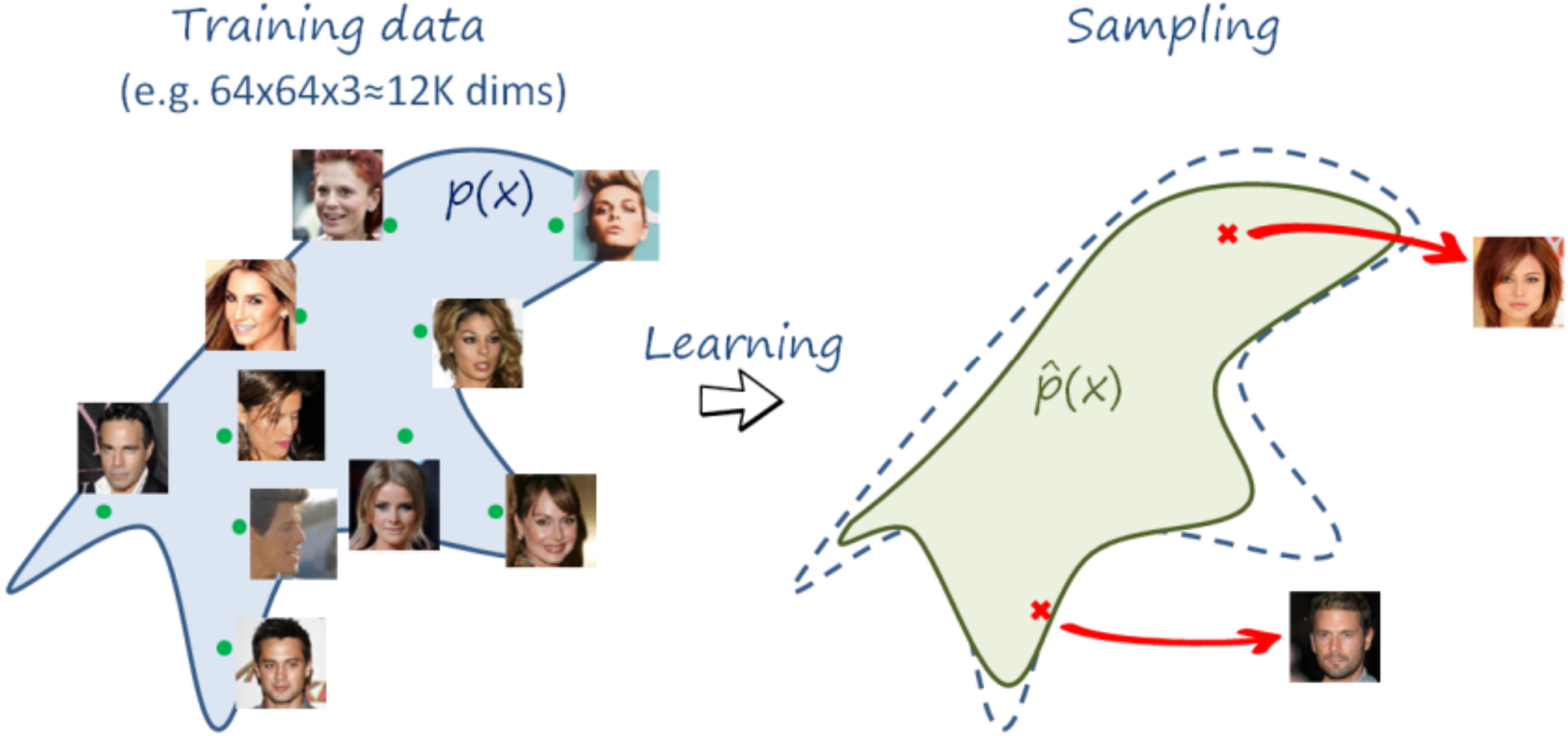
Generative Model

Generative and Discriminative Models

- We've only seen discriminative models so far
  - Given an image X, predict a label Y
  - Estimates $P(Y|X)$
- Discriminative models have several key limitations
  - Can't model P(X), i.e. the probability of seeing a certain image
  - Can't generate new images!
- Generative models (in general) cope with all of above
  - Can model P(X)
  - Can generate new images

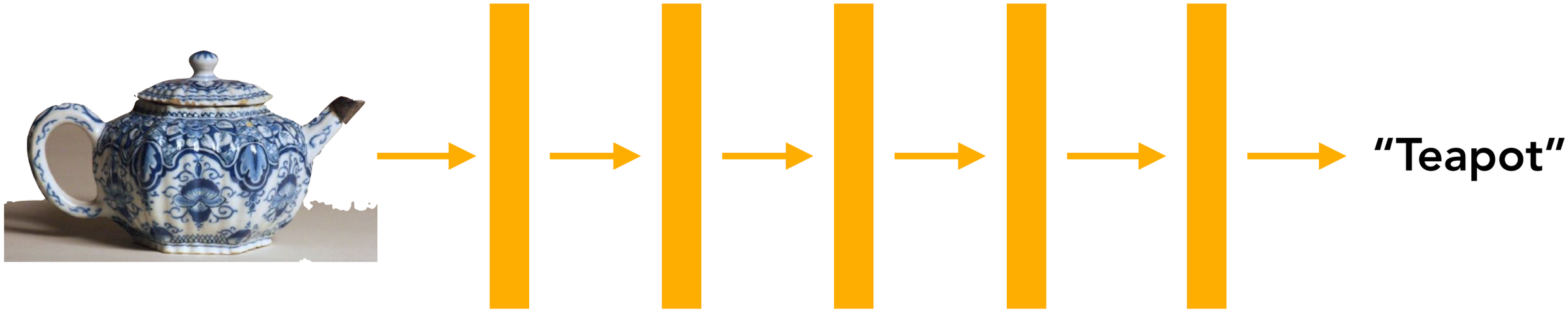
Generate New Images

- Generative model let us learn a distribution
- With that distribution we can generate new images (images not seen during training)

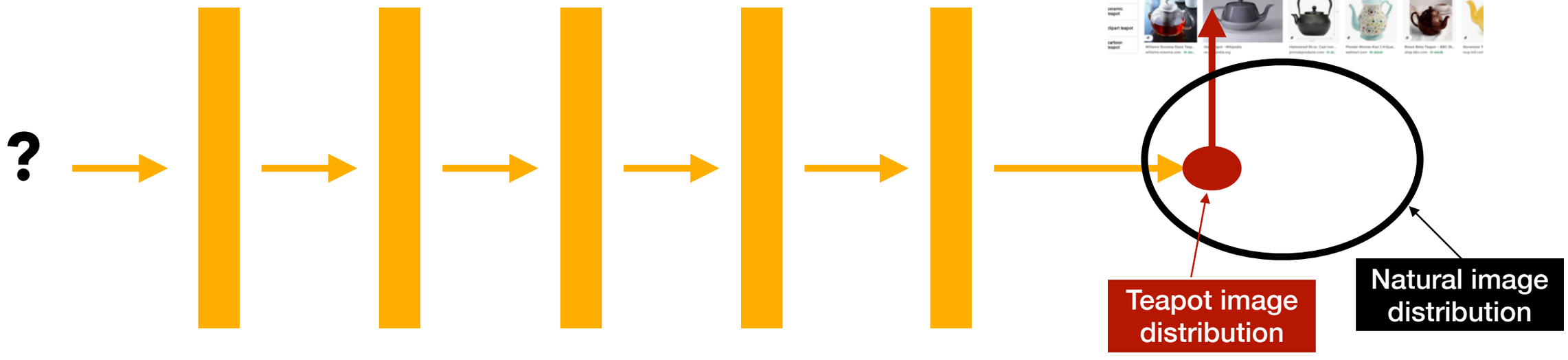


Generative Models

- Discriminative Model:



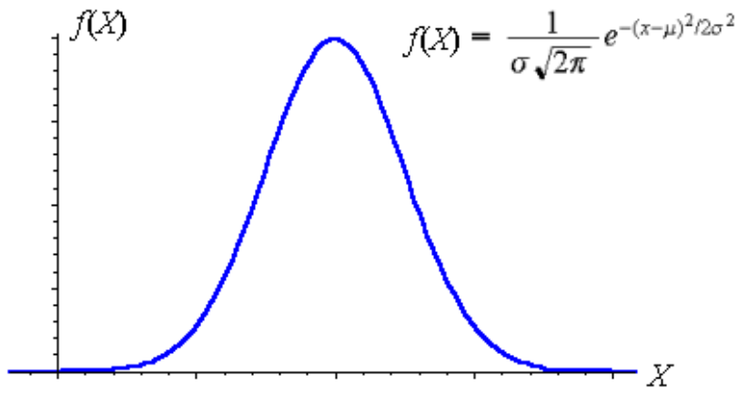
- Generative Model:



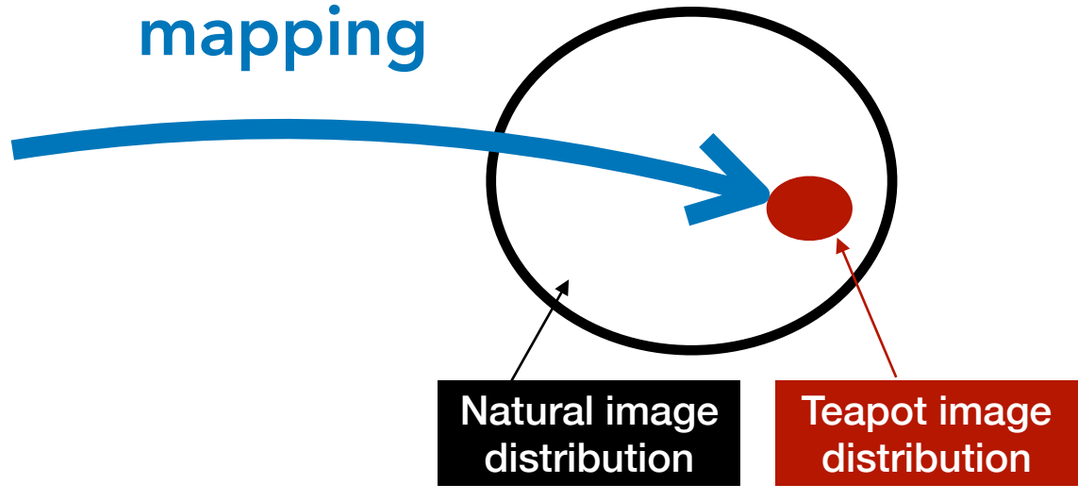
How To Learn the Distribution

- If we want to sample, we need to know the distribution — but that is also what we want to learn

  - So how? — we know how to sample from a normal distribution!

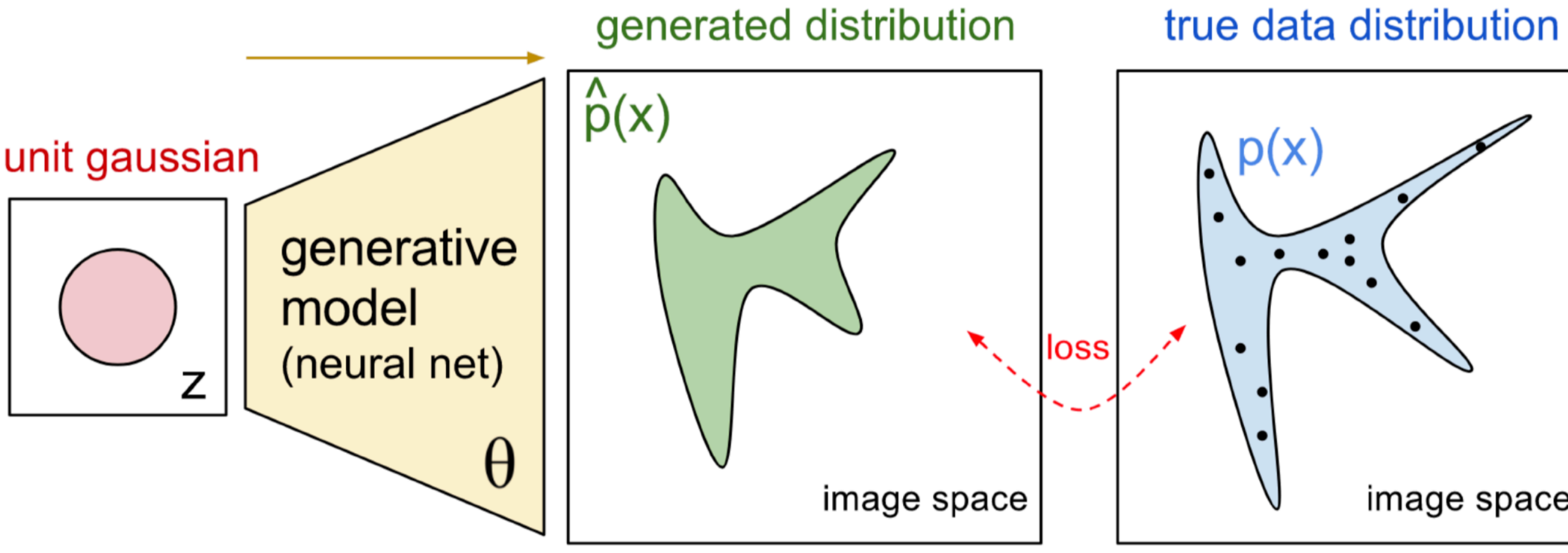


Goal: Learn the mapping



A Generative Model

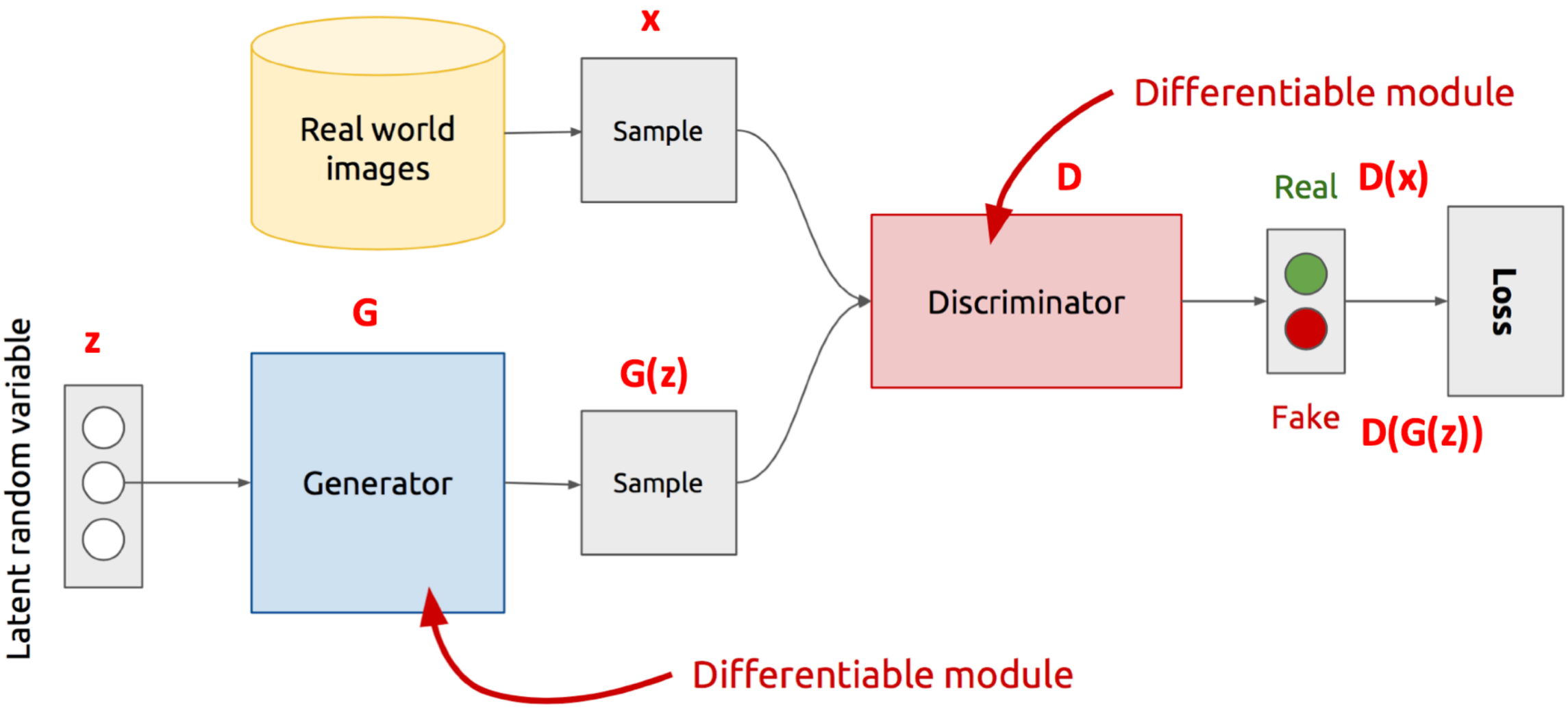
- Generate an image from a randomly sampled vector
- Learn the mapping parameters to target image space



From Generative Model to GAN

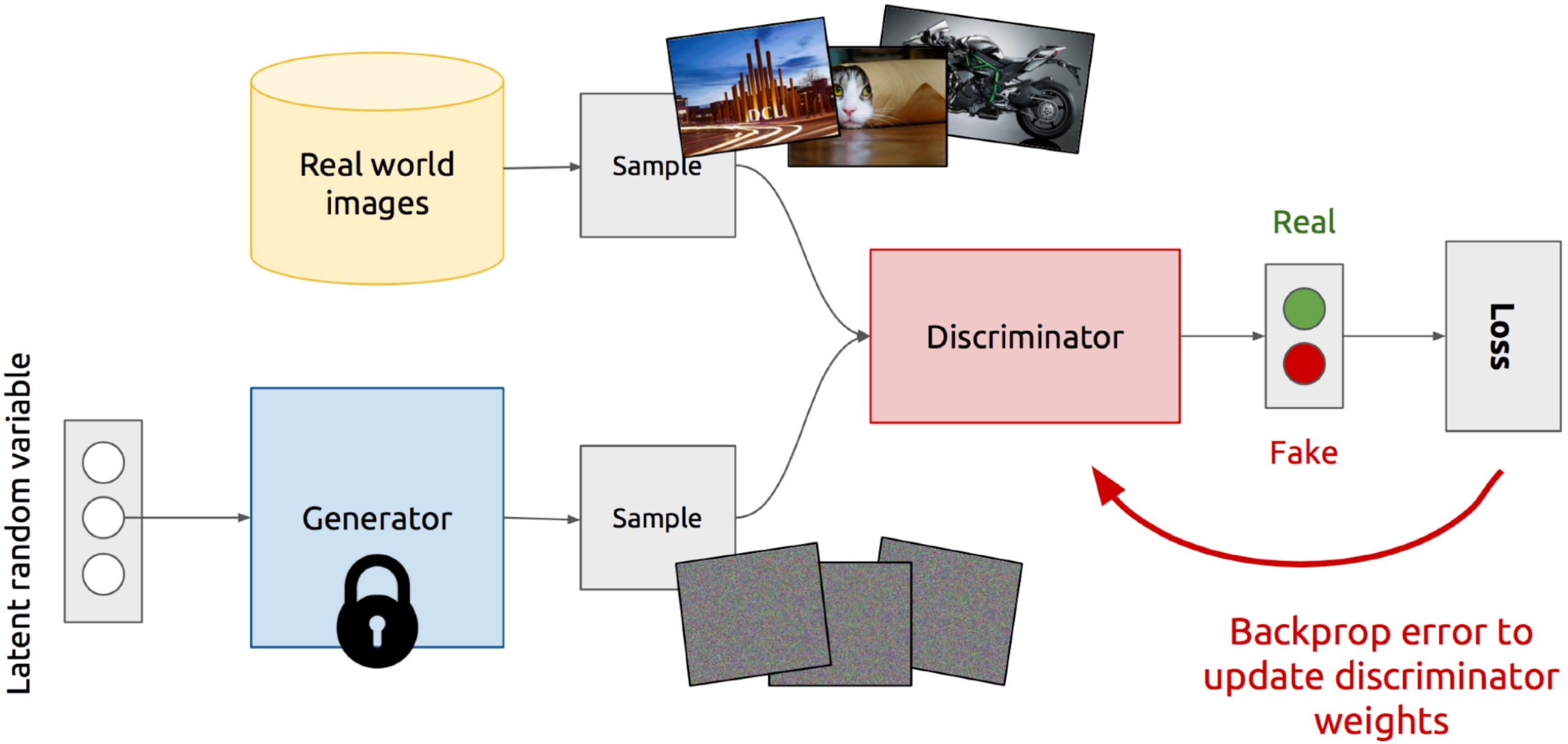
- Most of the time, we can tell whether the image has been manipulated or not
- What if a machine can generate images and do a self-check on its realism? — an “adversarial” module

GAN Architecture



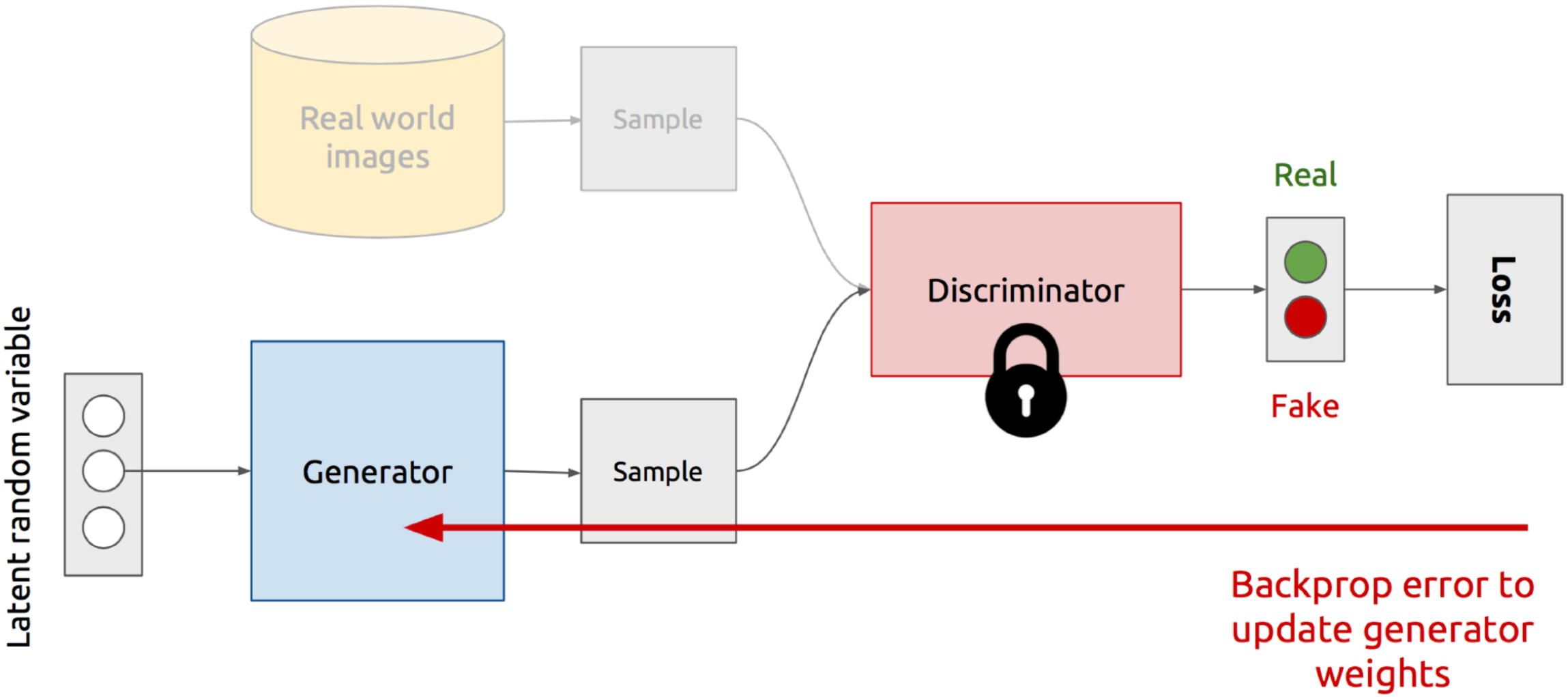
- Z is some random noise (Gaussian/Uniform)
- Z can be thought as the latent representation of the image

Training a GAN



- Loss — gradients descent
- Update weights in discriminator *only*
- Generator weights are fixed at this iteration

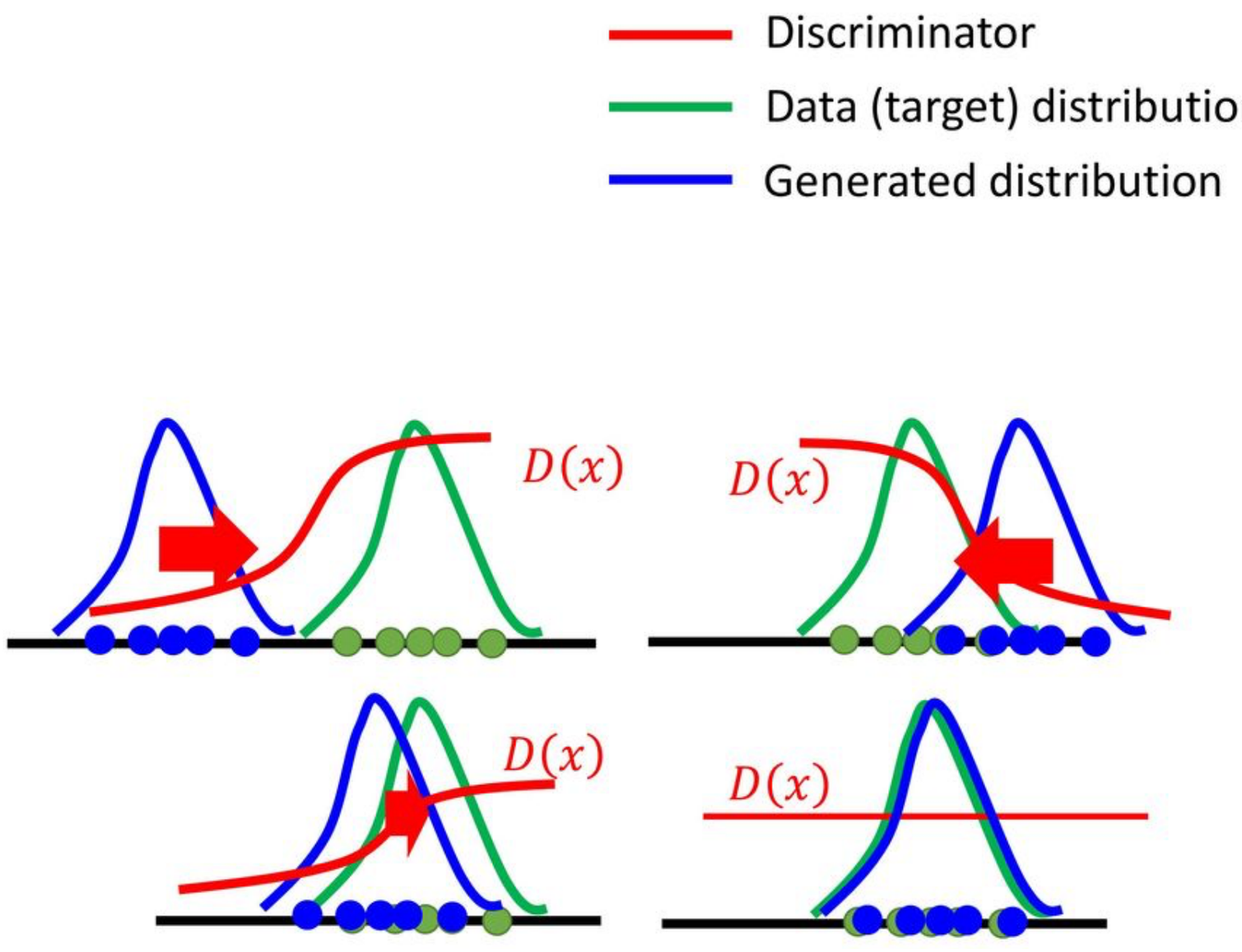
Training a GAN



- Update weights in the generator *only*
- Discriminator weights are fixed at this iteration
- An alternating training scheme

GAN Intuition

- Goal: bring target and generated distribution together
- Discriminator leads the generator



More (Fun) GAN Intuition

Generator

Discriminator

Real Money

Fake Money

Counterfeiter prints fake money. It is labelled as fake for police training. Sometimes, the counterfeiter attempts to fool the police by labelling the fake money as real.

The police are trained to spot real from fake money. Sometimes, the police give feedback to the counterfeiter why the money is fake.

GAN Formulation

- It is formulated as a minimax game, where:
  - The Discriminator is trying to maximize its reward
  - The Generator is trying to minimize Discriminator's reward (or maximize the Discriminator's loss)

$$V(D) = \mathbb{E}_{x \sim p(x)}[\log D(x)] + \mathbb{E}_{z \sim q(z)}[\log(1 - D(G(z)))]$$

log prob of D predicting that real-world data is real

log prob of D predicting that G's generated data is not real

$$V(G) = \mathbb{E}_{z \sim q(z)}[\log(D(G(z)))]$$

log prob of G predicting that G's generated data is real

Training a GAN

- One example of GAN training pseudocode
  - Stochastic gradient descent training of GAN
  - The number of steps to apply to the discriminator k is a hyperparameter

for number of training iterations **do**

  for k steps **do**

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

GAN Formulation

- The Nash equilibrium of this particular game is achieved at:
  - A Nash Equilibrium is one, where each player does not want to change their actions, given the other players actions
  - $D(x) = \frac{1}{2} \forall x$

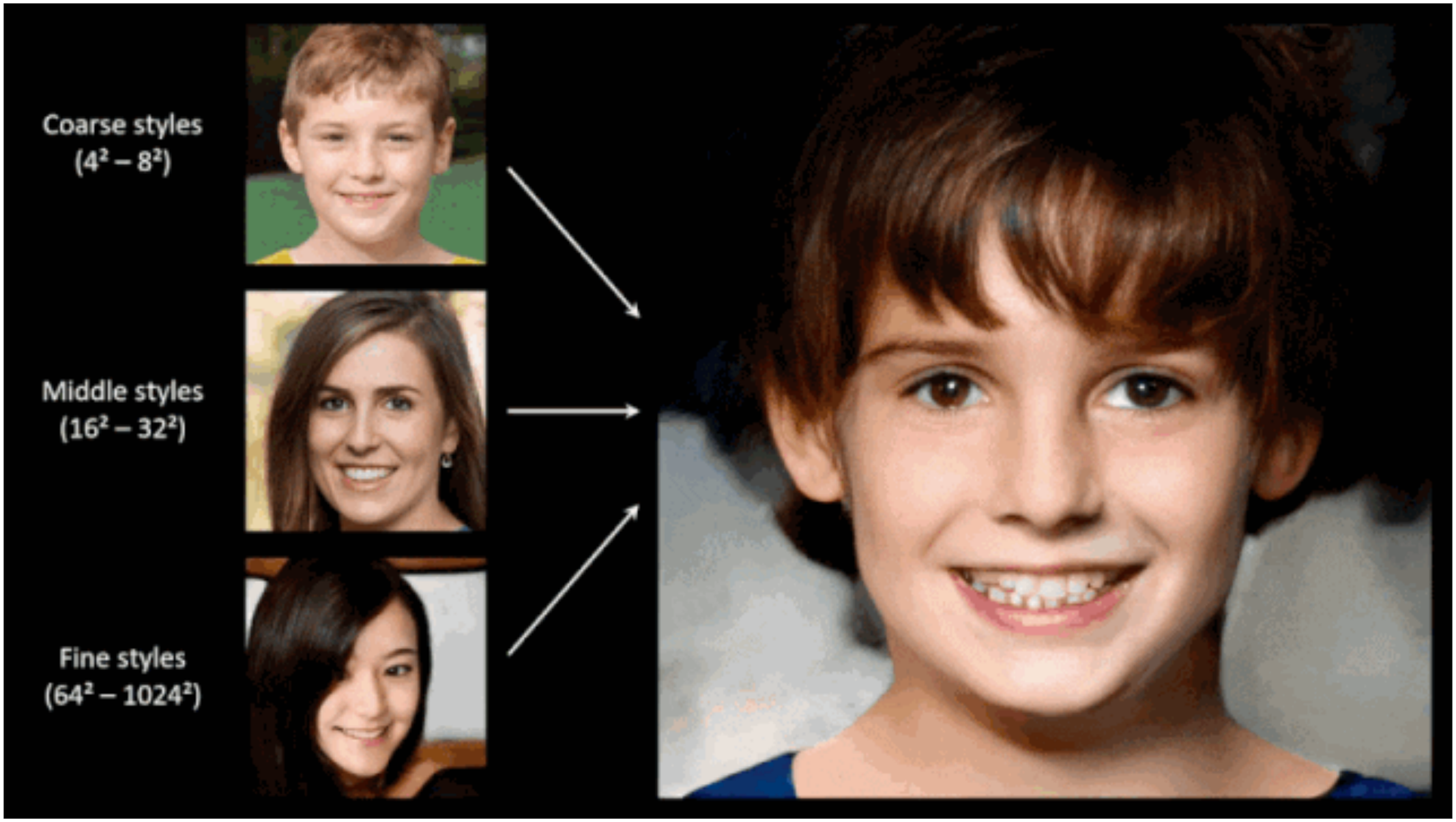
- Challenges in training GAN
  - How much should we train G before going back to D? If we train too much we won't converge (overfitting)
  - Avoid saturating gradients early on when G is terrible

Results — Generate Bedrooms (DCGAN)

Radford et al. "Unsupervised representation learning with deep convolutional generative adversarial networks."

StyleGAN

- Hyper-realistic face generator
  - These images are not real people



GAN Input Can Also Be an Image

- Conditional GAN

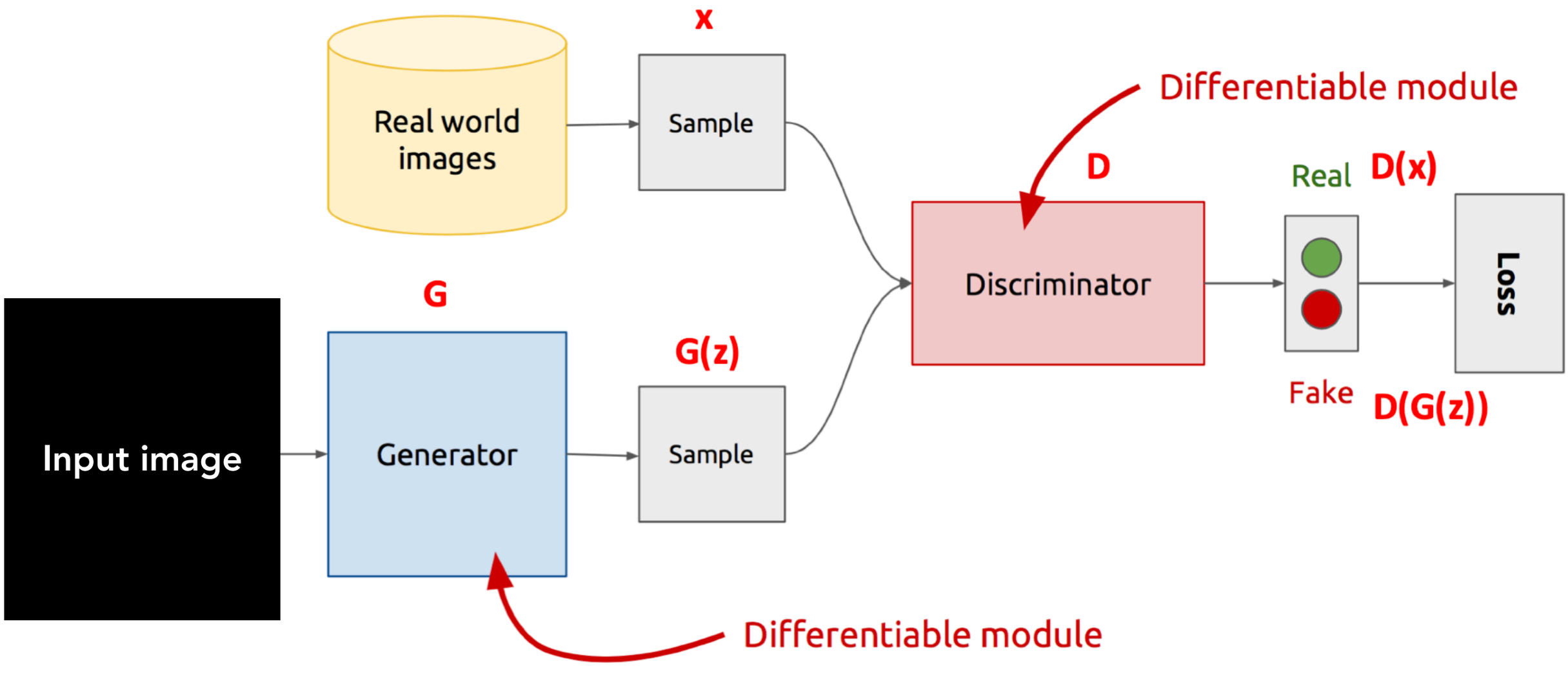
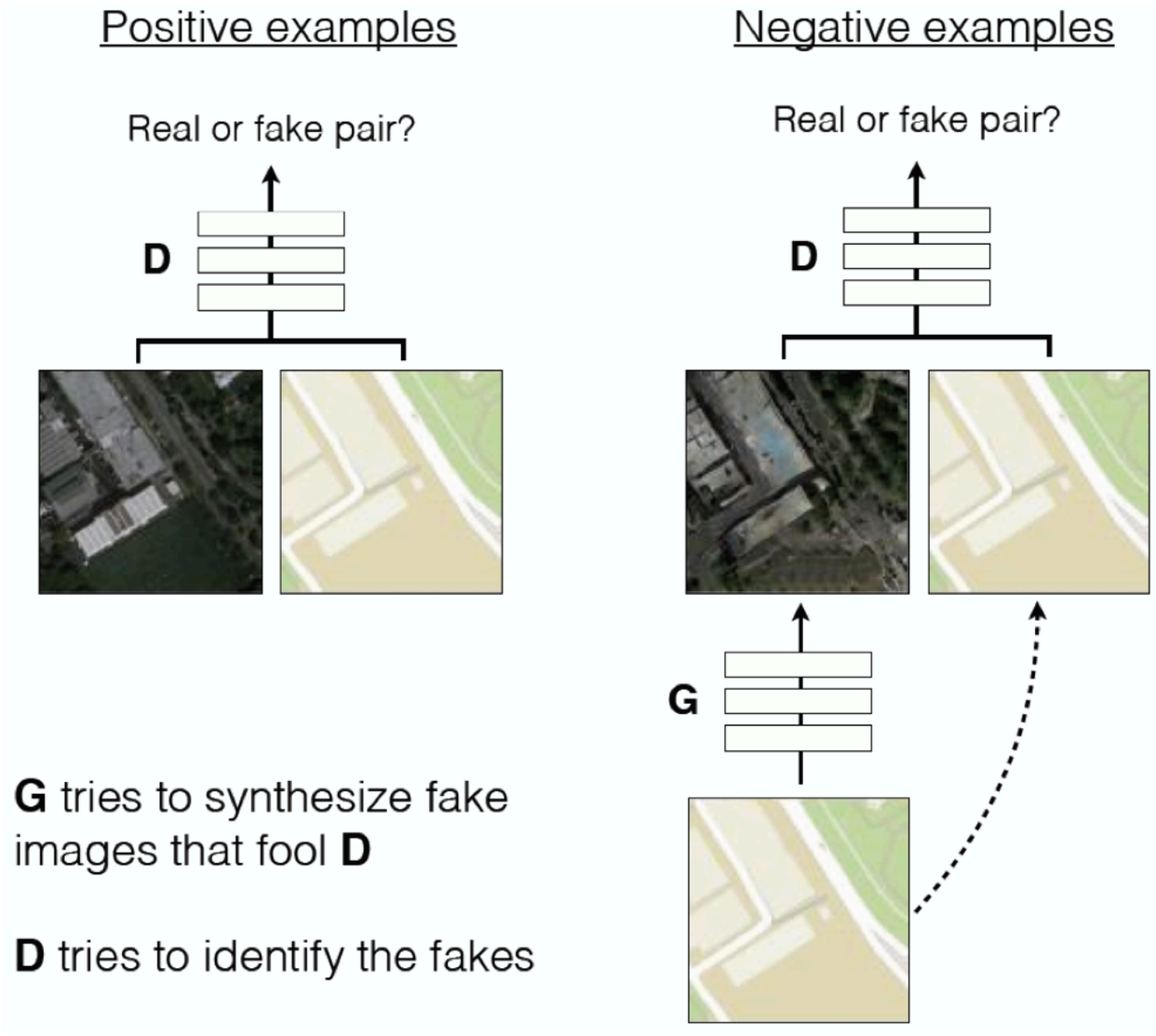


Image-to-image Translation

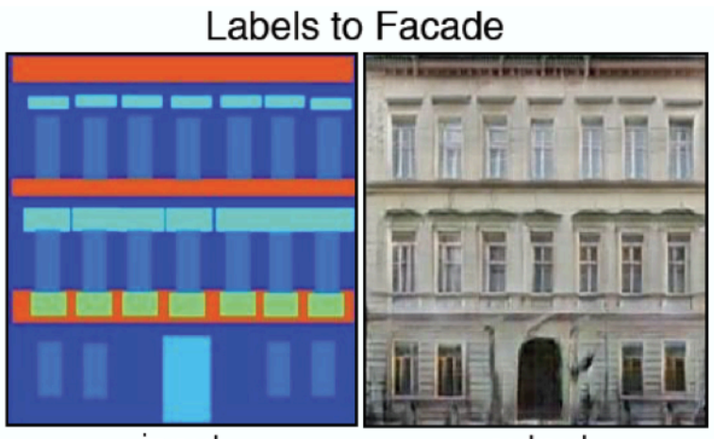
- Training is conditioned on the images from the source domain.
- Conditional GANs provide an effective way to handle many complex domains without worrying about designing structured loss functions explicitly.

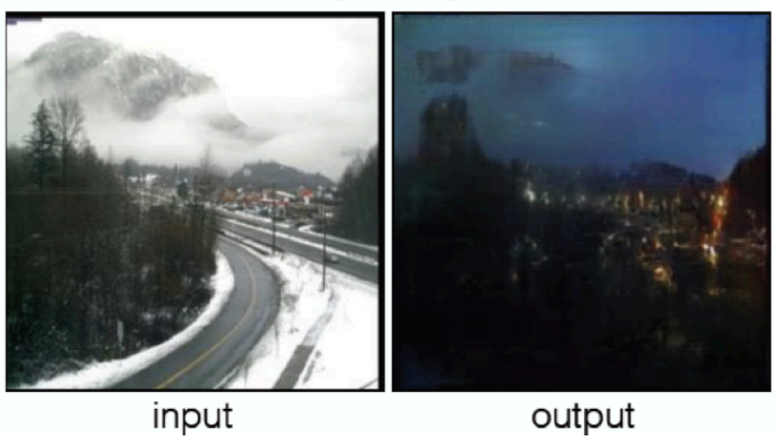


Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. "Image-to-image translation with conditional adversarial networks"

Pix2pix

- Same framework for different applications





Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. "Image-to-image translation with conditional adversarial networks"

Moving from 2D to 3D

- Of course we are not limited to (or satisfied with) generating 2D images
- 2 images of the same scene tell us a lot about its 3D geometry — just like stereo perception of our eyes
- What are good representations for 3D data?
  - Point cloud? volume? mesh? image sequences?
- More on 3D Tomorrow!