




Introduction on Generative Modeling

Bolei Zhou

The Chinese University of Hong Kong

Outline

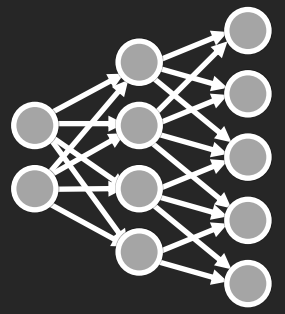
- Introduction: from recognition to recreation
- Overview of the generative models
- Understanding generative models
- Inverting images for real image editing

Outline

- Introduction: from recognition to recreation
- Overview of the generative models
- Understanding generative models
- Inverting images for real image editing

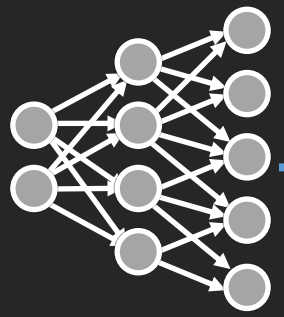
Recognizing the Visual World

Input Image

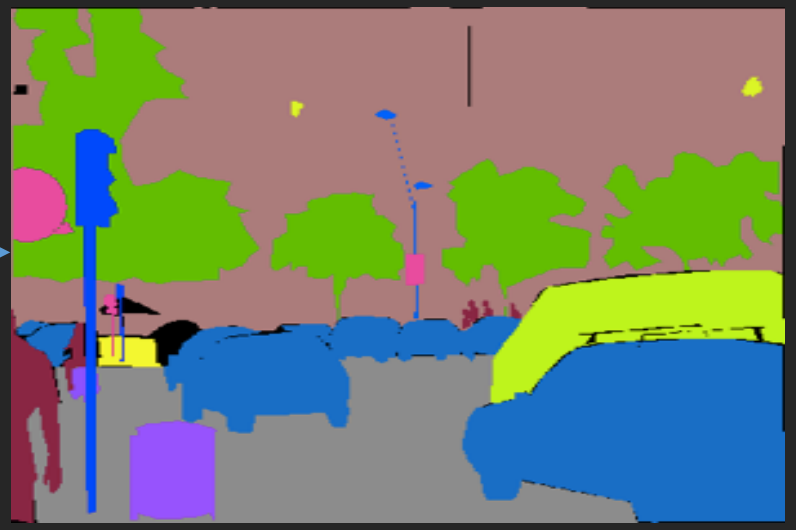


Scene Recognition

Street, Residential neighborhood



Scene Parsing



objects

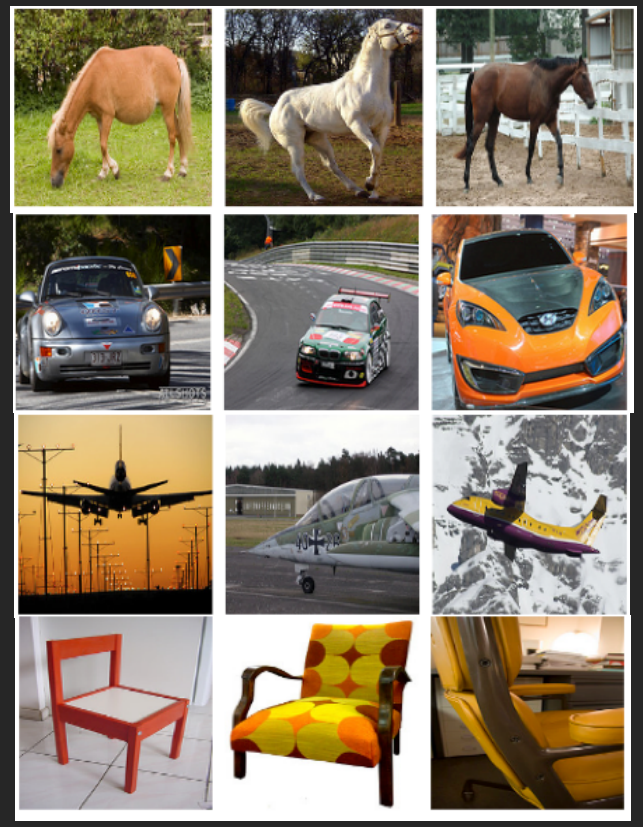
- tree
- car
- van
- ashcan
- person
- streetlight
- signboard
- traffic light

Datasets + Deep Neural Networks

2016: ResNet
2017:DenseNet
>100 layers

IMAGENET

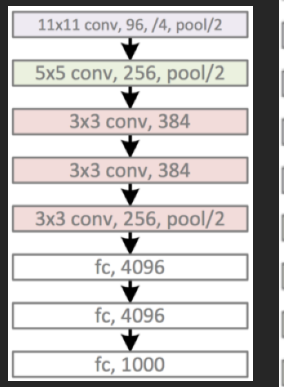
2009: 1.2 million images



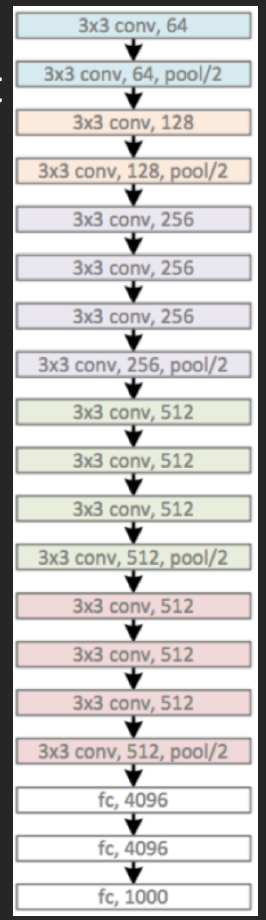
COCO
Common Objects in Context

2014: 120k annotated images

2012: AlexNet
5 layers



2014: VGG
16 layers



2015: GoogLeNet
22 layers

places

Largest Scene Recognition Dataset.

10 million images from 400 categories.

5,000 AMT workers involved

spare bedroom

messy bedroom

teenage bedroom

romantic bedroom

[**Zhou**, Lapedriaza, Xiao, Torralba, Oliva, NIPS 2014],
[**Zhou**, Lapedriaza, Khosla, Oliva, Torralba, IEEE PAMI 2017]

http://places2.csail.mit.edu/

Indoor

bedroom

cafeteria

veterinarians office

elevator door

staircase

bar

conference center

shoe shop

Nature

fishpond

watering hole

field road

rainforest

Urban

windmill

corral

arch

tower

swimming pool

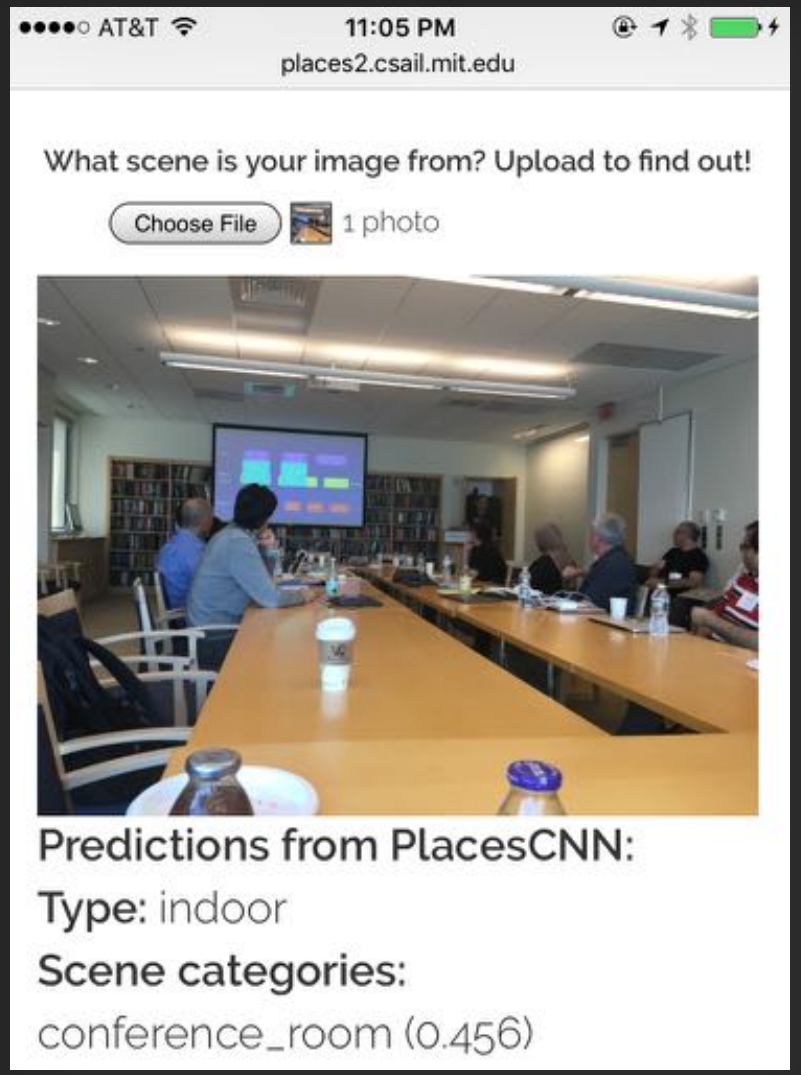
train station platform

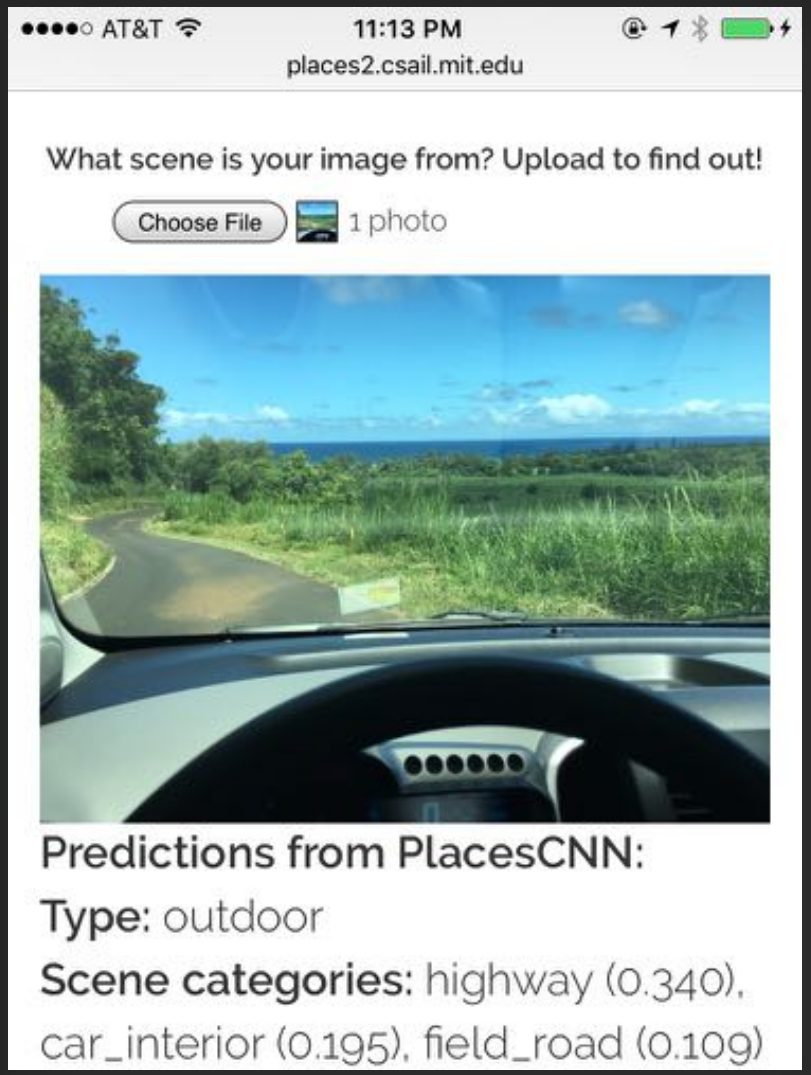
amusement park

soccer field

street

Deep Representation for Classifying Scenes





http://places2.csail.mit.edu/demo.html

places

Live Demo: Scene Recognition based on PlacesCNN

- http://places2.csail.mit.edu/demo.html

Pixel-Annotated ADE20K dataset

- Annotating each object instances in a scene
- Single expert annotator for >5 years of work

Labelme Annotation Tool

Ms. Adela Barriuso

Zhou, et al. Scene parsing through ade20k dataset. CVPR'17

Neural Networks for Scene Parsing

Input | Ground-truth | Prediction

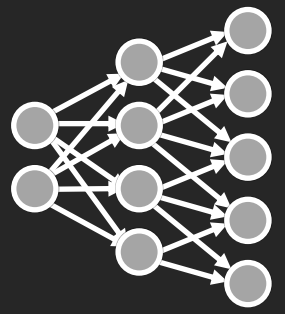
https://github.com/CSAILVision/semantic-segmentation-pytorch

Live Demo: Semantic Segmenation

- https://colab.research.google.com/github/CSAILVision/semantic-segmentation-pytorch/blob/master/notebooks/DemoSegmenter.ipynb

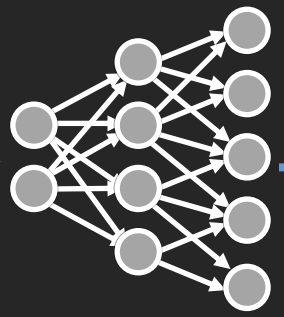
Recognizing the Visual World

Input Image

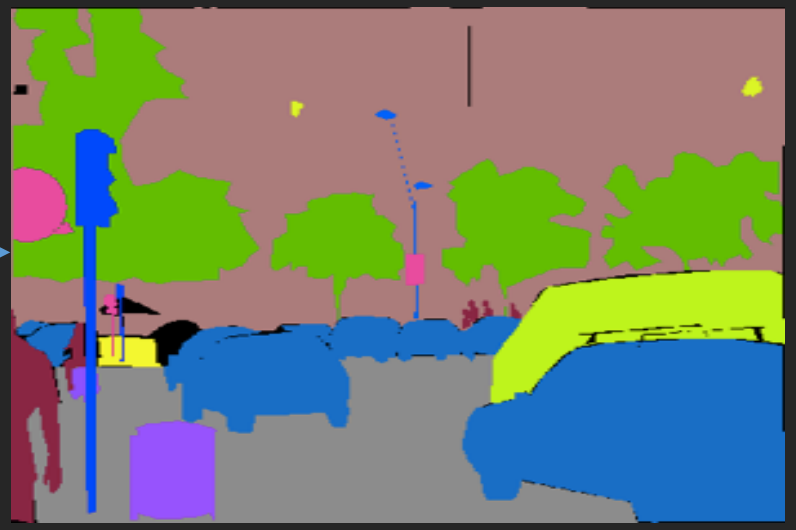


Scene Recognition

Street, Residential neighborhood



Scene Parsing



objects

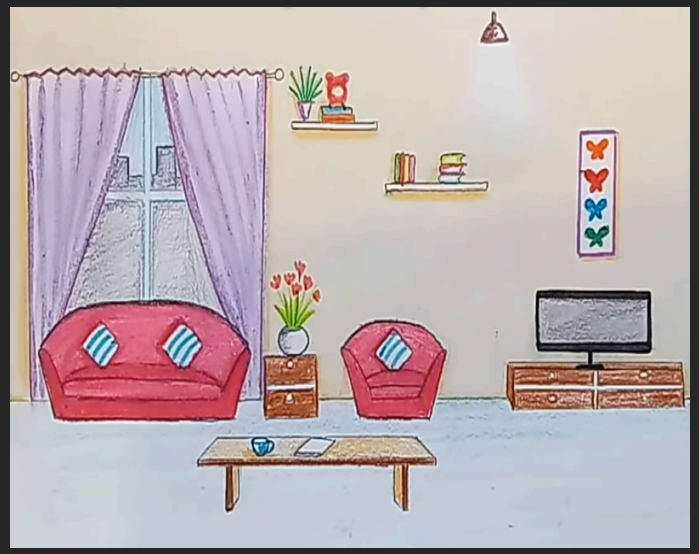
- tree
- car
- van
- ashcan
- person
- streetlight
- signboard
- traffic light

From Recognition to Recreation

Kid

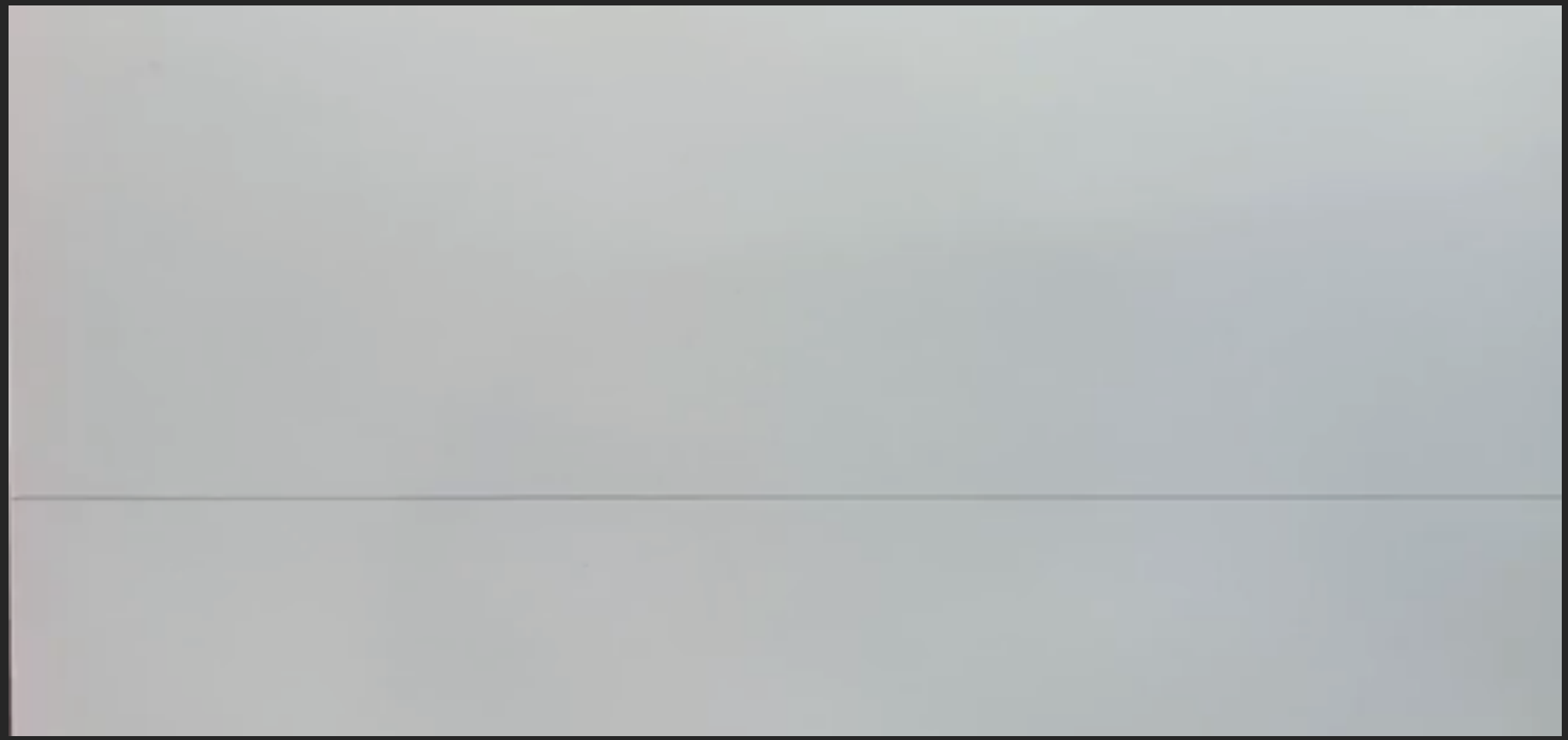
Recognizing the scene

It is a living room

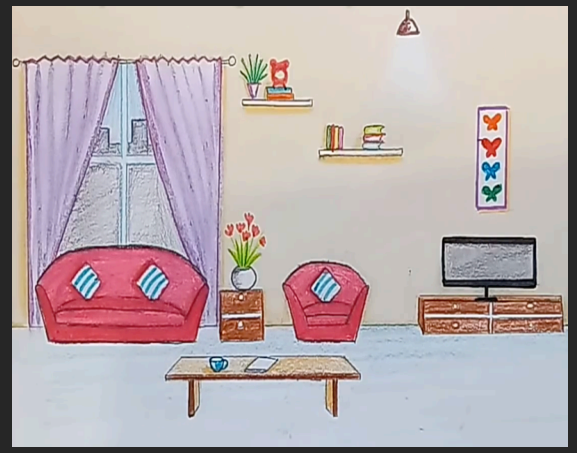


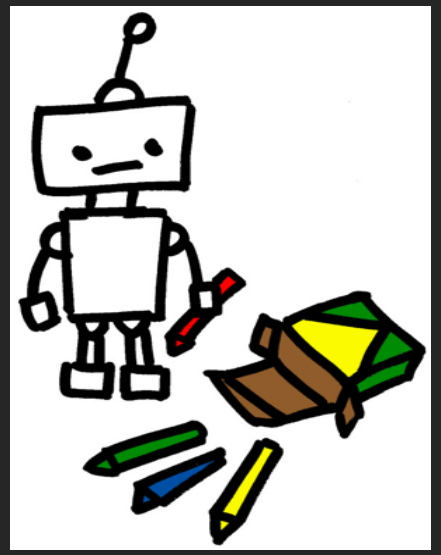
Recreating the scene

Recreating Scene is Challenging

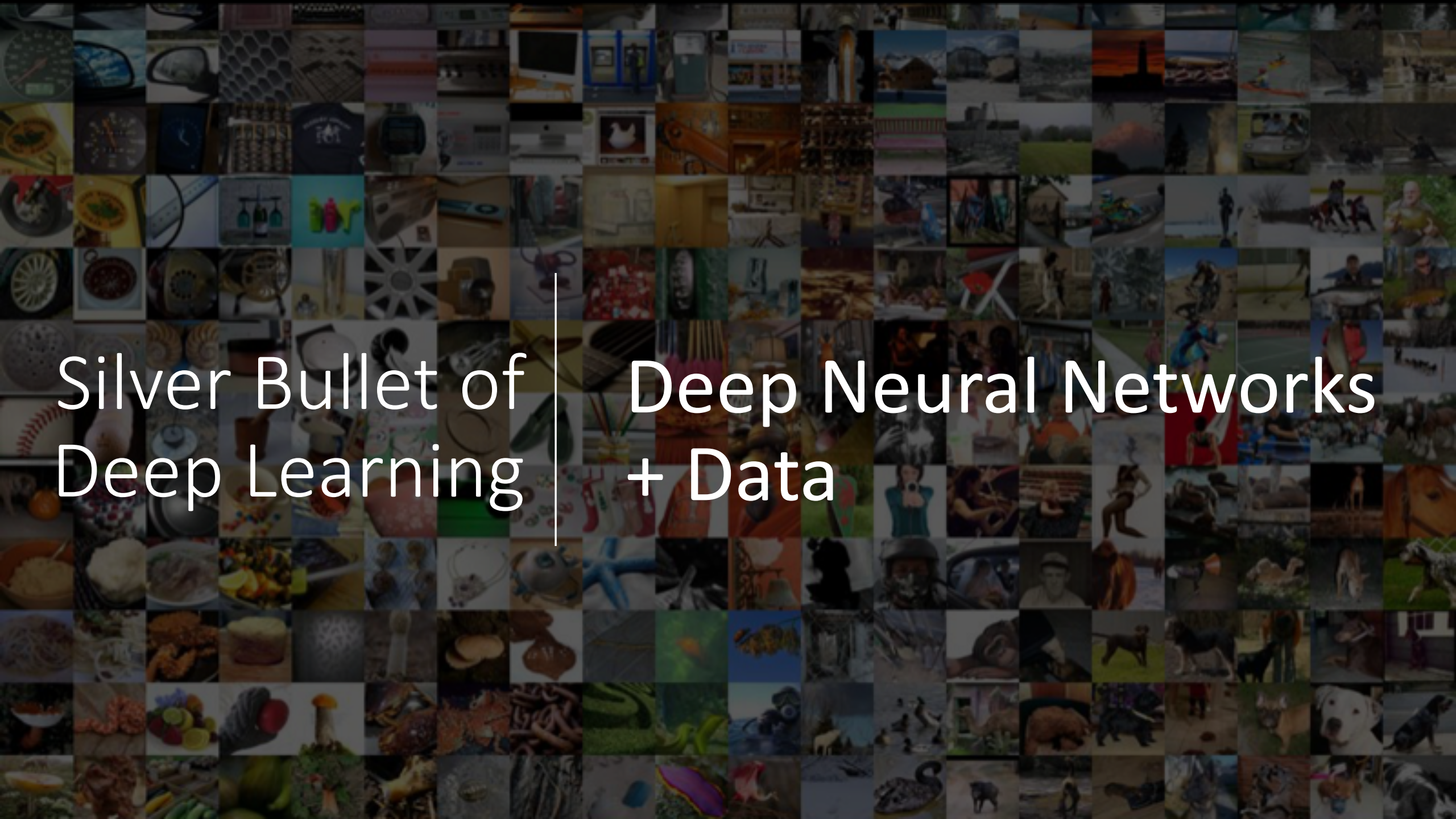


How to Teach Machine to Recreate a Scene?



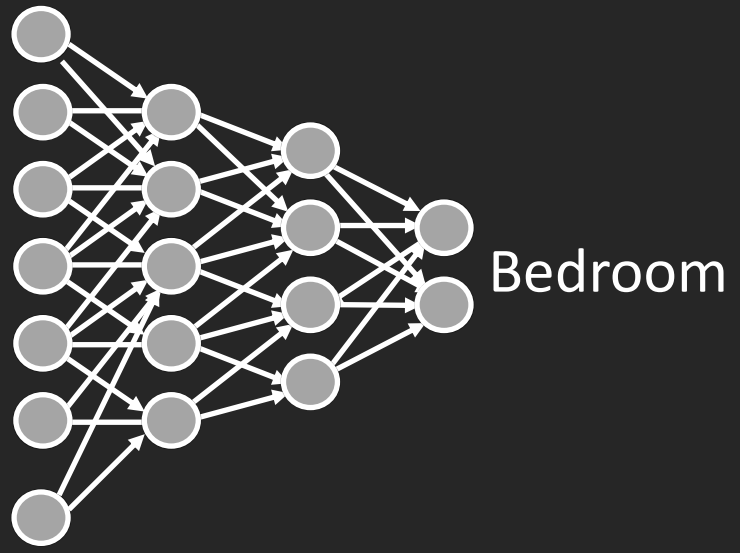


Concept of Perspective?
Concept of Objects?
Textures and Colors of Objects?
Definition of Scene Category?

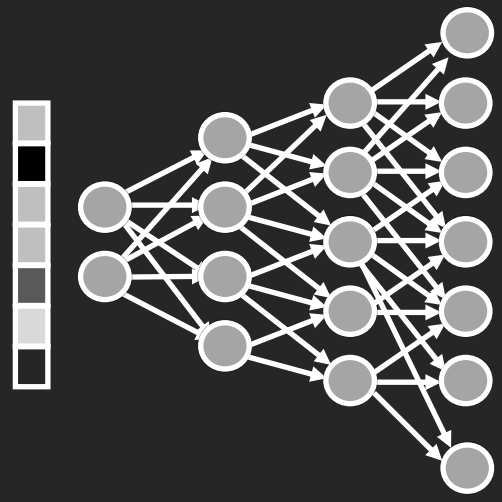

Silver Bullet of Deep Learning
Deep Neural Networks + Data

Classification versus Generation

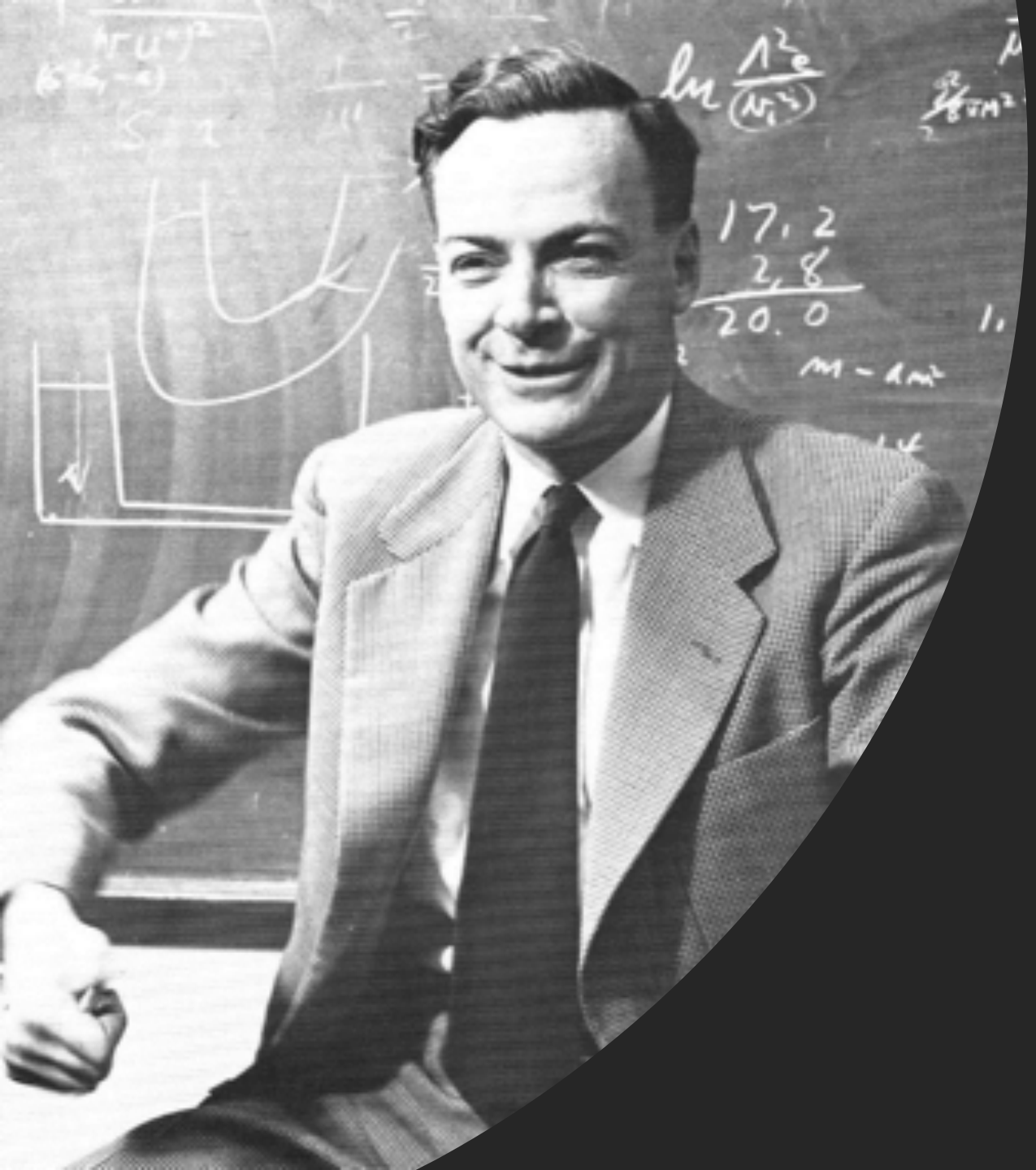
Discriminative Model



Generative Models



Goodfellow, et al. NIPS'14
Radford, et al. ICLR'15
T Karras et al. 2017
A. Brock, et al. 2018



Richard Feynman: “What I cannot create, I do not understand”

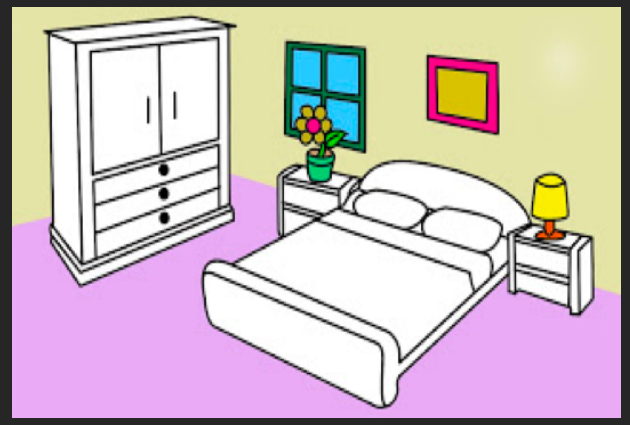
Inference and Generation

Inference

Predictions:
bedroom
wood
foliage

layout

Generation



Statistical Generative Models

- Data: samples such as images of bedrooms
- Prior knowledge: distribution functions (Gaussian, or other non-linear?), loss function (e.g., maximum likelihood, adversarial loss?),

Learn

probability distribution p(x)

Sample

Progress for Image Generation

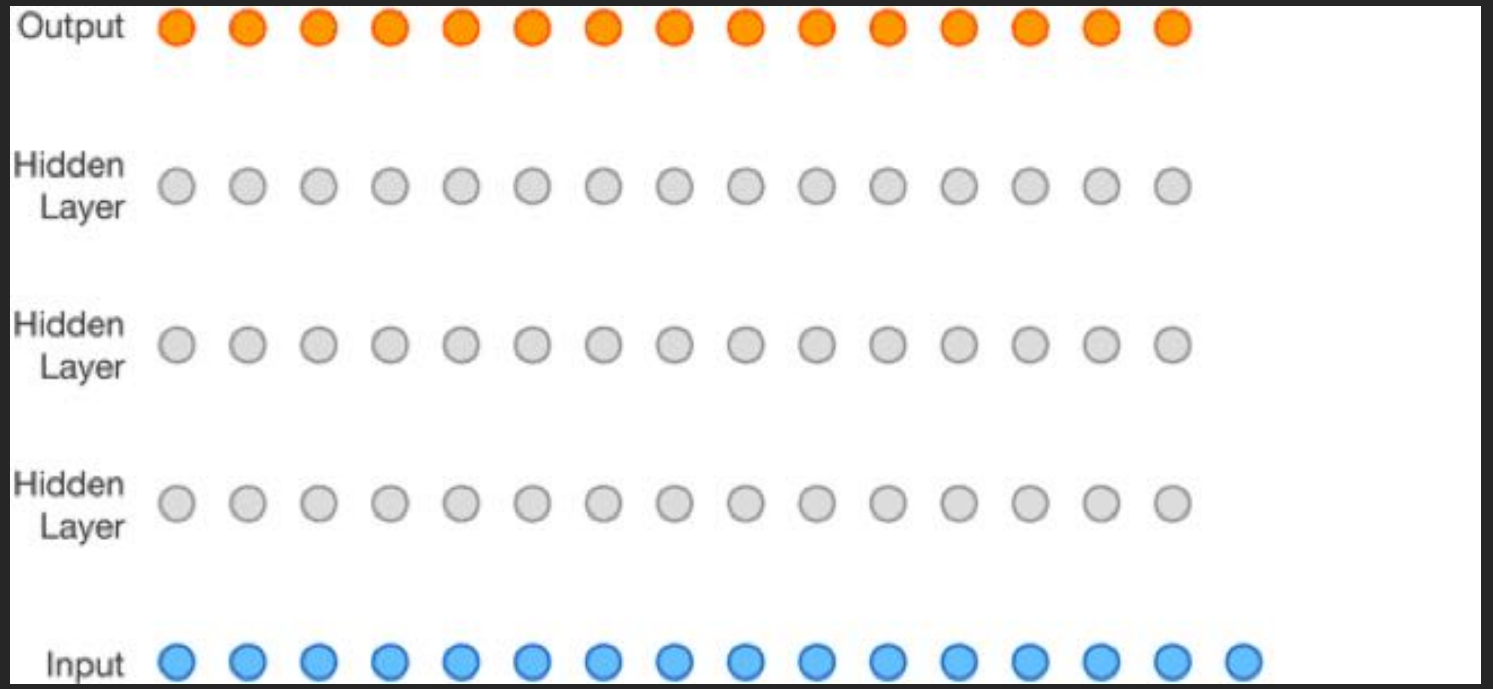
PG-GAN

StyleGAN

BigGAN

Progress for Speech Synthesis

WaveNet



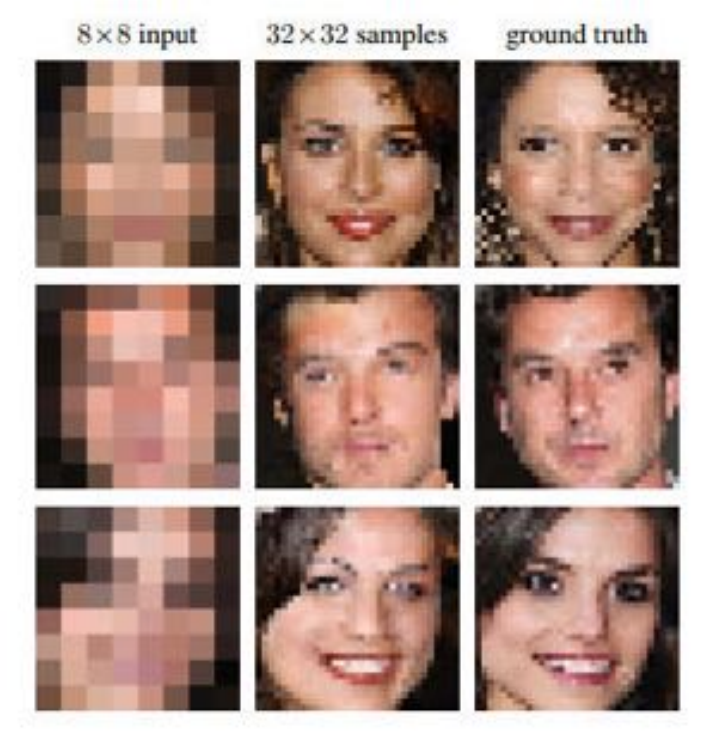
Text to Speech

- Parametric
- Concatenative
- WaveNet
- Unconditional
- Music

van den Oord et al, 2016c

Image Super Resolution

Conditional generative model P(high res image | low res image)



Ledig et al., 2017

Machine Translation

Conditional generative model P(English text| Chinese text)

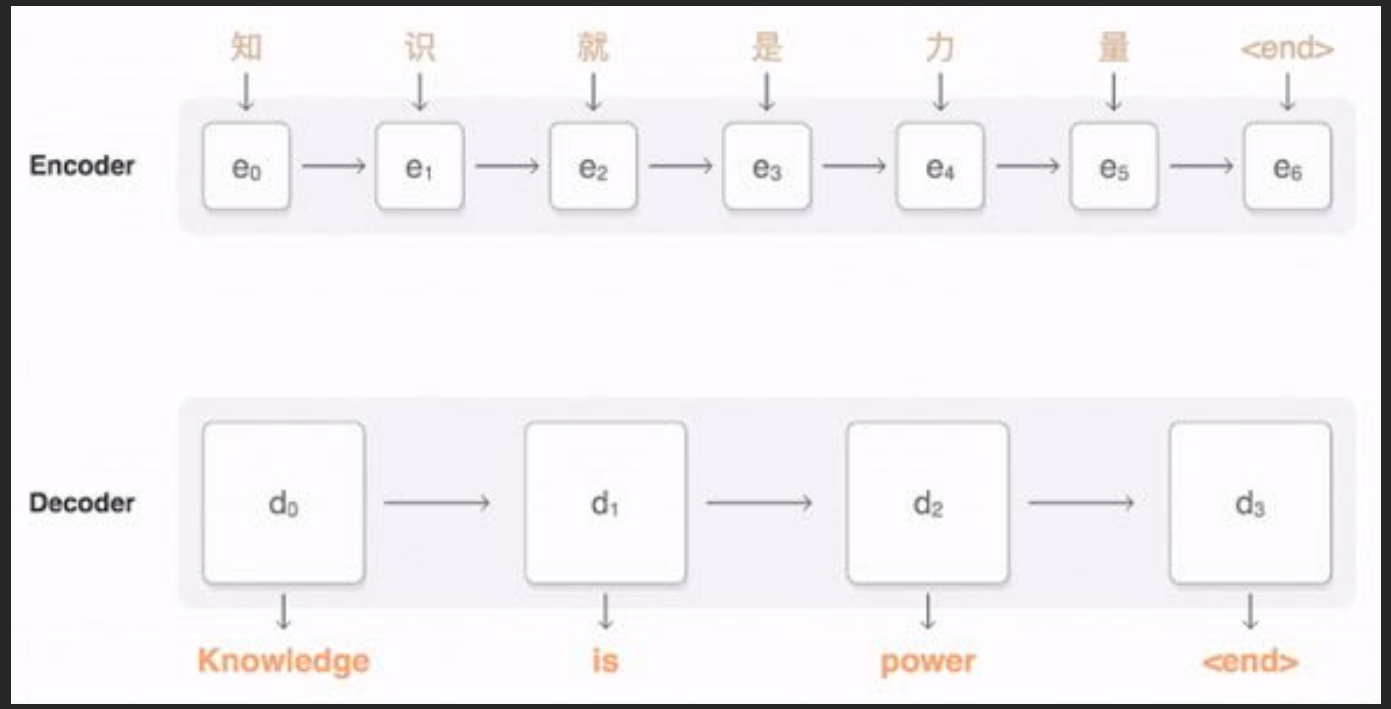
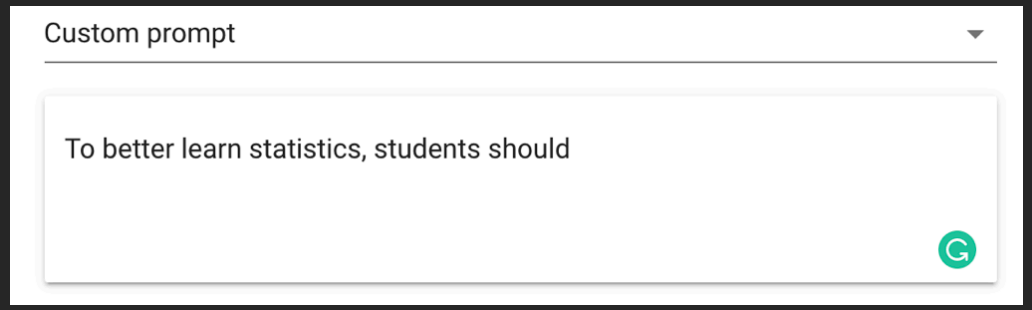


Figure from Google AI research blog.

Paragraph Generation



Completion

To better learn statistics, students should understand the functions of probability, statistics, and probability distributions. They should also understand the basic logic that underlies statistics and find their own ideas and methods for applying the logic to new problems. Their basic understanding of probability and statistics should provide the foundation they will need to think creatively and critically when dealing with critical situations and problems. They should understand the information we can find out by running statistical tests, and they should understand the implications and limitations of such information. They should be able to apply basic math concepts, statistics, probability and mathematics to their own particular circumstances and to teach other students. They should be able to analyze and critique the many issues of statistics that are in our society. They should be able to tell about possible sources

Radford et al., 2019. Demo from talktotransformer.com

DeepFakes (video+audio)

Image Translation

Conditional generative model P(zebra images| horse images)

Zhu et al., 2017

Outline

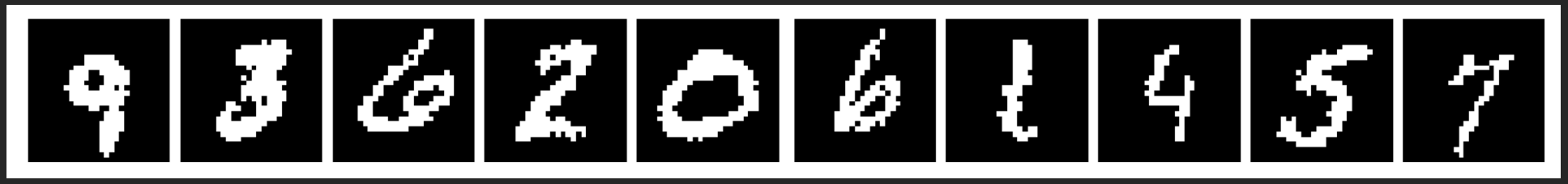
- Introduction: from recognition to recreation
- Overview of the generative models
- Understanding generative models
- Inverting images for real image editing

Overview of Generative Models

- Likelihood-based models: Autoregressive models
- Latent variable models: Variational Autoencoder (VAE)
- Implicit generative models: Generative Adversarial Networks (GANs)

Autoregressive generative model

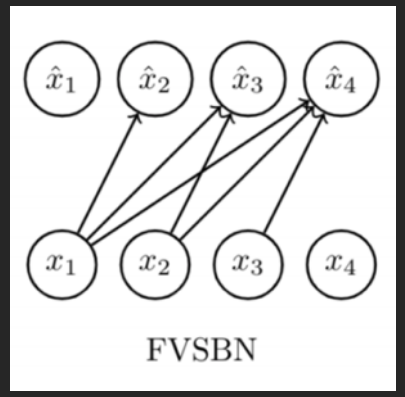
- Given a dataset of handwritten digits (binarized MNIST)



- Each image has n = 28x28 =784 pixels, with each pixel either being black (0) or white (1)
- Goal: Learn a probability distribution $p(x) = p(x_1, \ldots, x_{784})$

Autoregressive generative model

- We can use chain rule for factorization:
  - $p(x_1, \ldots, x_{784}) = p(x_1)p(x_2|x_1)p(x_3|x_1,x_2)\ldots p(x_n|x_1,\ldots,x_{n-1})$
- Then we can have the following simple autoregressive model
  - $P(x_1=1;a_1) = a_1$, $p(x_1=0;a_1) = 1-a_1$
  - $P(x_2=1|x_1;a_2)= \text{logit}(a_{02}+a_{12}x_1)$
  - $P(x_3=1|x_1,x_2;a_3)=\text{logit}(a_{03}+a_{13}x_1+a_{23}x_2)$

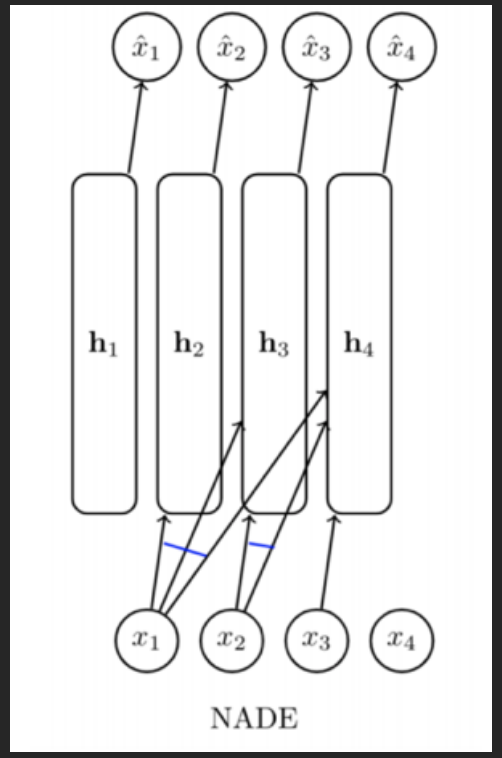


$$\hat{x}_i = p(X_i = 1|x_1, \cdots, x_{i-1}; \boldsymbol{\alpha}^i) = p(X_i = 1|x_{<i}; \boldsymbol{\alpha}^i) = \sigma(\alpha_0^i + \sum_{j=1}^{i-1} \alpha_j^i x_j)$$

Fully visible sigmoid belief network (FVSBN)

NADE: Neural Autoregressive Density Estimation

To improve the factorization model: use one-layer neural network instead of logistic regression



$$\mathbf{h}_i = \sigma(A_i \mathbf{x}_{<i} + \mathbf{c}_i)$$

$$\hat{x}_i = p(x_i | x_1, \cdots, x_{i-1}; \underbrace{A_i, \mathbf{c}_i, \boldsymbol{\alpha}_i, b_i}_{\text{parameters}}) = \sigma(\boldsymbol{\alpha}_i \mathbf{h}_i + b_i)$$

$$\mathbf{h}_2 = \sigma\left(\underbrace{\begin{pmatrix} \vdots \\ w_1 \\ \vdots \end{pmatrix}}_{A_2} x_1\right) \quad \mathbf{h}_3 = \sigma\left(\underbrace{\begin{pmatrix} \vdots & \vdots \\ w_1 & w_2 \\ \vdots & \vdots \end{pmatrix}}_{A_3} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right) \quad \mathbf{h}_4 = \sigma\left(\underbrace{\begin{pmatrix} \vdots & \vdots & \vdots \\ w_1 & w_2 & w_3 \\ \vdots & \vdots & \vdots \end{pmatrix}}_{A_3} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}\right)$$

Training data

Samples from the probability

Figure from The Neural Autoregressive Distribution Estimator, 2011.

RNN: Recurrent Neural Networks

Suppose $x_i \in \{h, e, l, o\}$. Use one-hot encoding:

- h encoded as $[1, 0, 0, 0]$, e encoded as $[0, 1, 0, 0]$, etc.

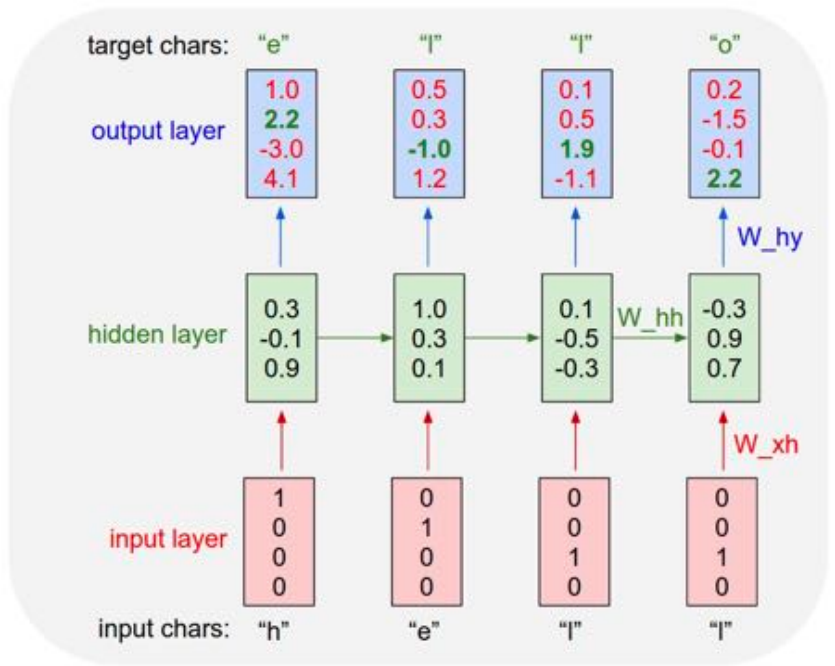
Autoregressive: $p(x = hello) = p(x_1 = h)p(x_2 = e|x_1 = h)p(x_3 = l|x_1 = h, x_2 = e) \cdots p(x_5 = o|x_1 = h, x_2 = e, x_3 = l, x_4 = l)$

For example,

$$p(x_2 = e|x_1 = h) = softmax(o_1) = \frac{\exp(2.2)}{\exp(1.0) + \cdots + \exp(4.1)}$$

$$o_1 = W_{hy} h_1$$

$$h_1 = tanh(W_{hh} h_0 + W_{xh} x_1)$$



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

RNN: Recurrent Neural Networks

RNN model on Wikipedia

```
Naturalism and decision for the majority of Arab countries' capitalide was grounded
by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated
with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal
in the [[Protestant Immineners]], which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired prison, training. The
emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom
of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known
in western [[Scotland]], near Italy to the conquest of India with the conflict.
Copyright was the succession of independence in the slop of Syrian influence that
was a famous German movement based on a more popular servicious, non-doctrinal
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
```

RNN model on Linux source code

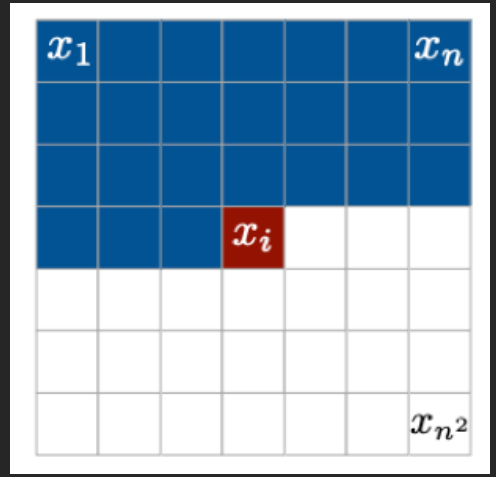
```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
  int error;
  if (fd == MARN_EPT) {
    /*
     * The kernel blank will coeld it to userspace.
     */
    if (ss->segment < mem_total)
      unblock_graph_and_set_blocked();
    else
      ret = 1;
    goto bail;
  }
  segaddr = in_SB(in.addr);
  selector = seg / 16;
  setup_works = true;
  for (i = 0; i < blocks; i++) {
    seq = buf[i++];
    bpf = bd->bd.next + i * search;
    if (fd) {
      current = blocked;
    }
  }
```

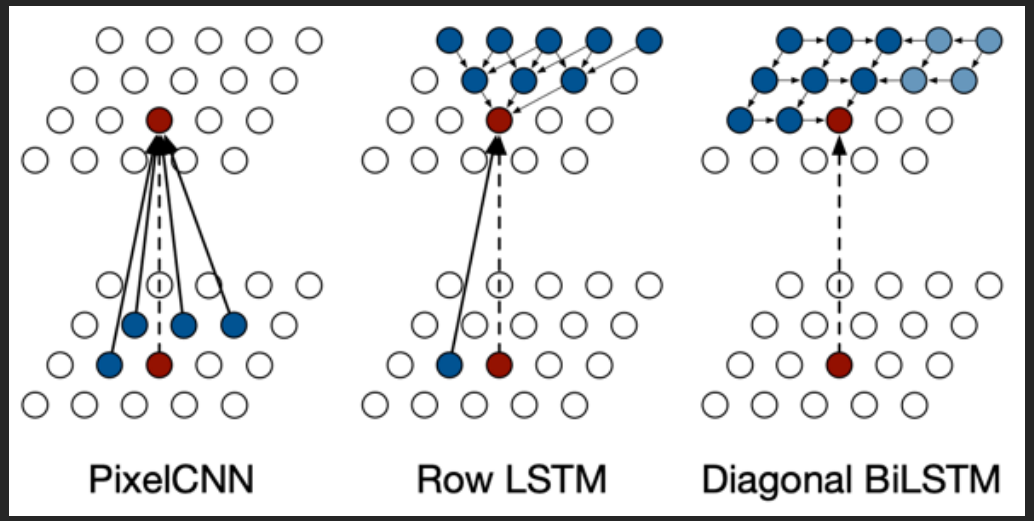
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

PixelRNN and PixelCNN (Oord et al. 2016)

Model image pixel by pixel using raster scan order

$$p(x_t \mid x_{1:t-1}) = p(x_t^{red} \mid x_{1:t-1})p(x_t^{green} \mid x_{1:t-1}, x_t^{red})p(x_t^{blue} \mid x_{1:t-1}, x_t^{red}, x_t^{green})$$





https://arxiv.org/pdf/1601.06759.pdf

PixelCNN and PixelRNN (Oord et al. 2016)

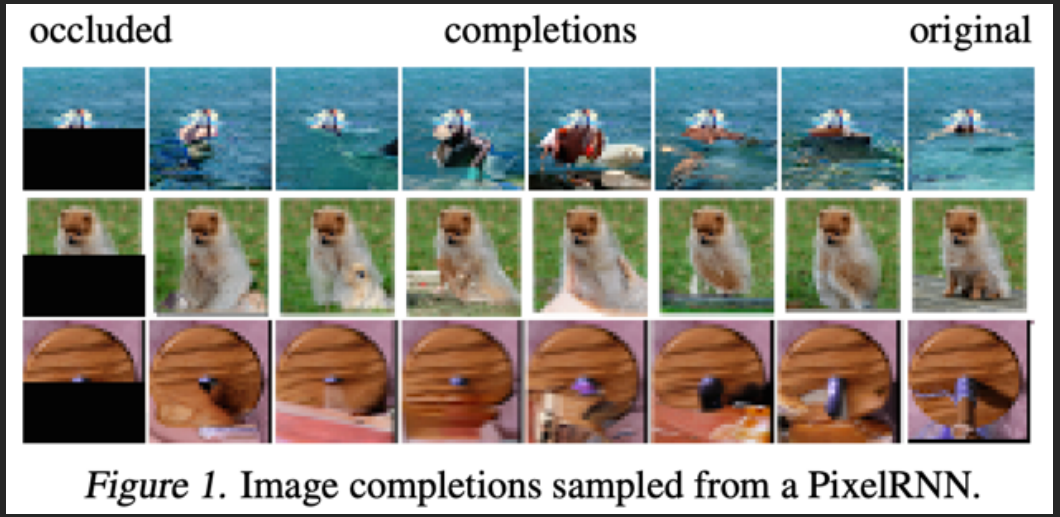
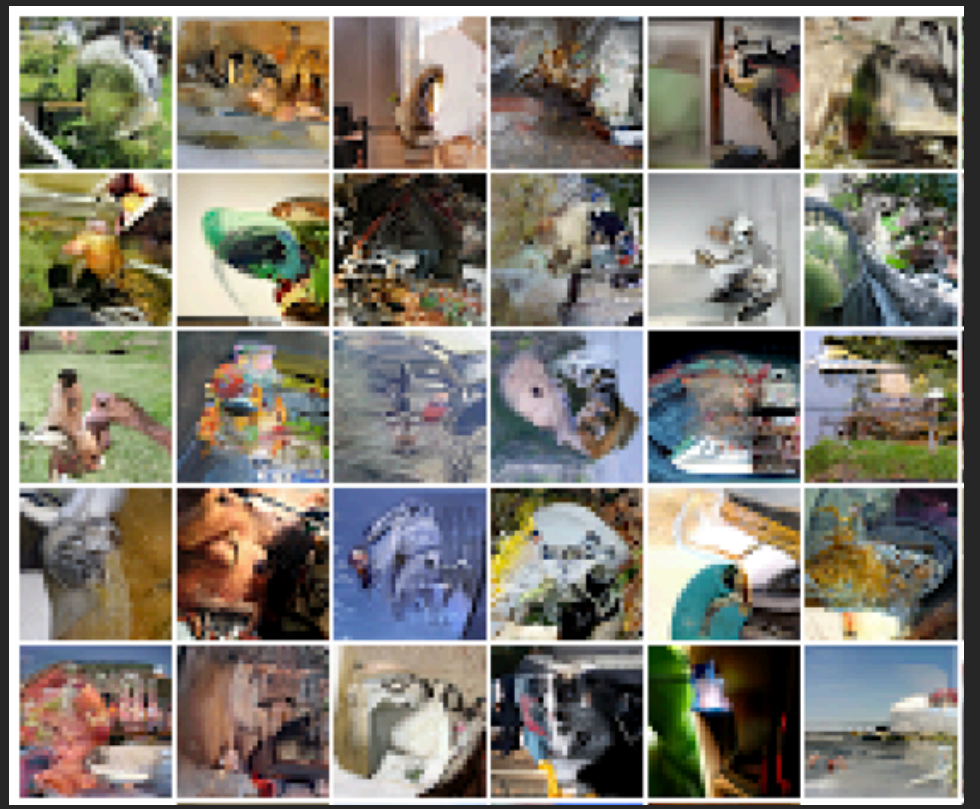


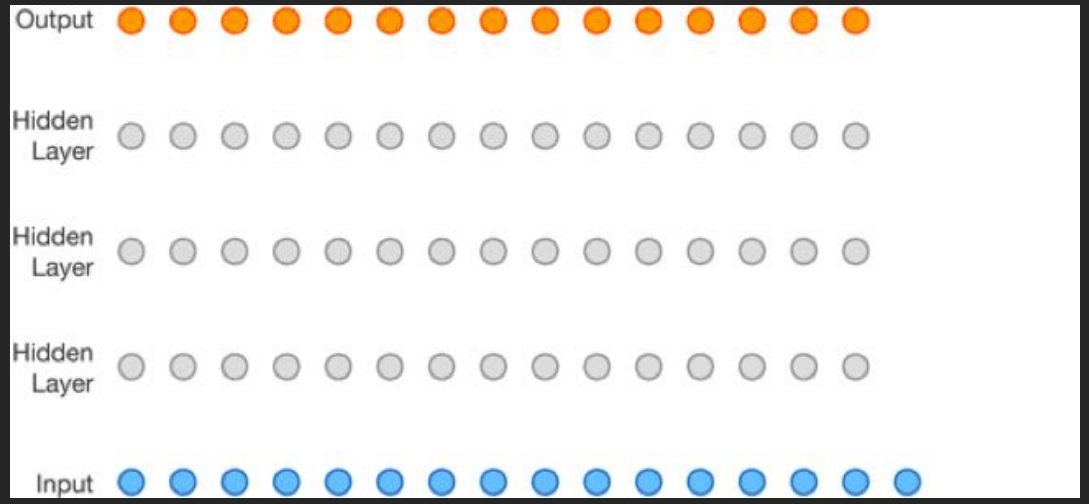
Figure 1. Image completions sampled from a PixelRNN.

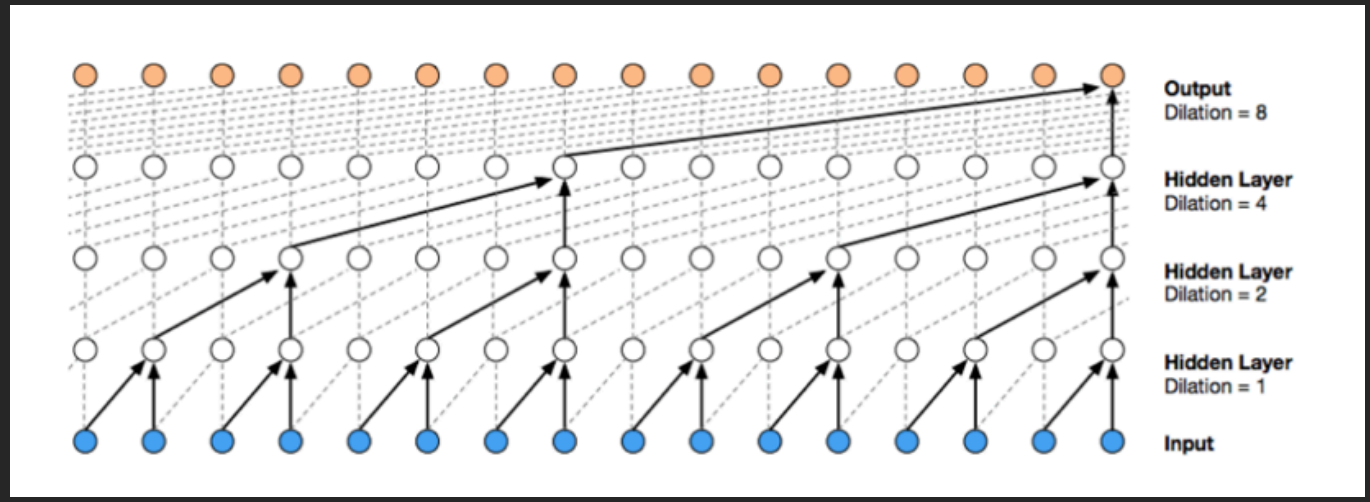
Samples from PixelRNN trained on ImageNet



WaveNet (Oord et al. 2016)

- CNN with 1D dilated convolutions





https://deepmind.com/blog/article/wavenet-generative-model-raw-audio

Summary of Autoregressive Models

- Autoregressive models:
  - Chain-rule factorization is fully general
  - Compact representation via conditional independence and/or neural parameterizations
- Pros:
  - Easy to evaluate likelihoods
  - Easy to train
- Cons:
  - Requires an ordering
  - Generation is sequential
  - Cannot learn features in an unsupervised way

Latent Variable Models

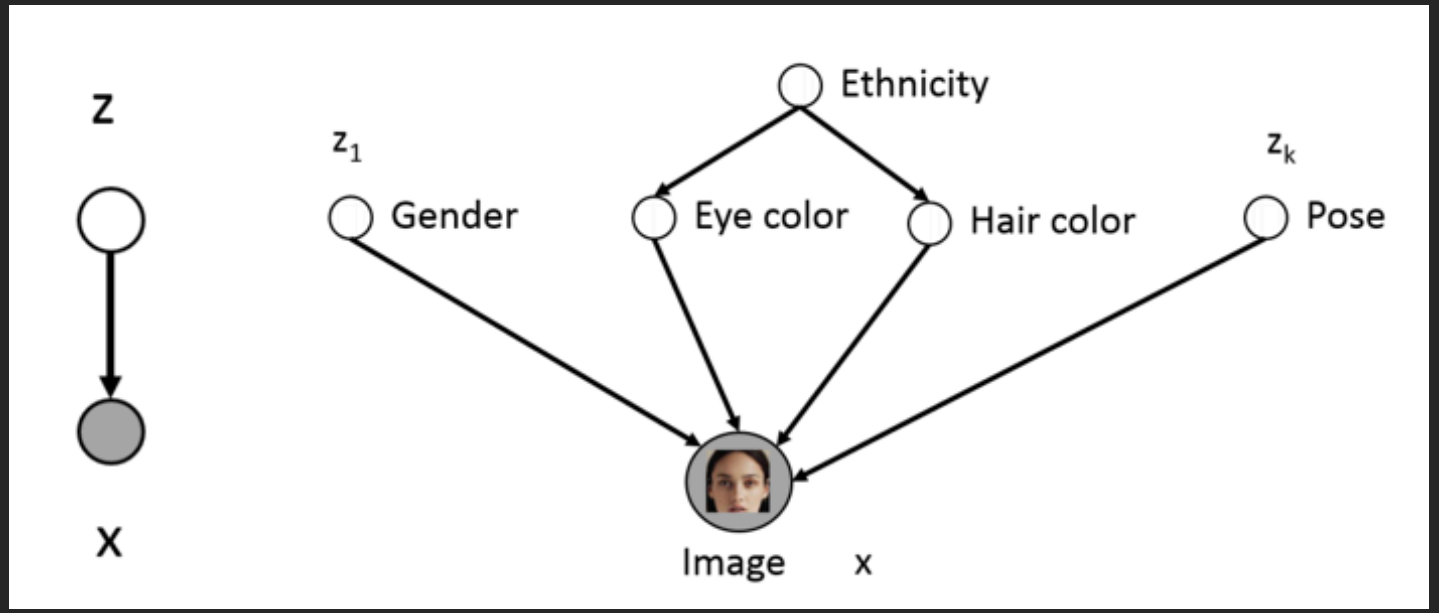
- Gender, eye color, race, pose are the variation factors in images, unless annotated, they are not explicitly available
- Then we can model them as (low-dimensional) **latent** variables

Latent Variable Models

Data generation process could be driven by the latent variables

Latent variables

Observation



Latent Variable Models

- We can define complex models p(x) in terms of simple building blocks p(x|z)
- It can be applied for unsupervised learning tasks easily
- Challenge: much more difficult to learn due to the latent variables.

A mixture of an infinite number of Gaussians:

1. $\mathbf{z} \sim \mathcal{N}(0, I)$
2. $p(\mathbf{x} \mid \mathbf{z}) = \mathcal{N}\left(\mu_\theta(\mathbf{z}), \Sigma_\theta(\mathbf{z})\right)$ where μ_θ,Σ_θ are neural networks
3. Even though $p(\mathbf{x} \mid \mathbf{z})$ is simple, the marginal $p(\mathbf{x})$ is very complex/flexible

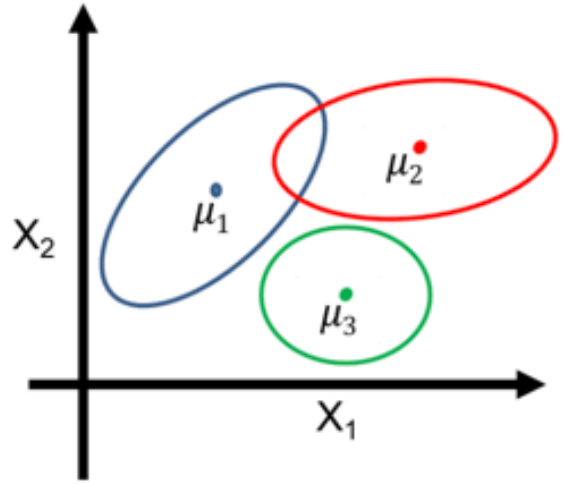
Mixture of Gaussians as a simple latent variable model

Generative process:

- Pick a mixture component k by sampling z
- Generate a data point by sampling from that Gaussian

Mixture of Gaussians. Bayes net: $\mathbf{z} \rightarrow \mathbf{x}$.

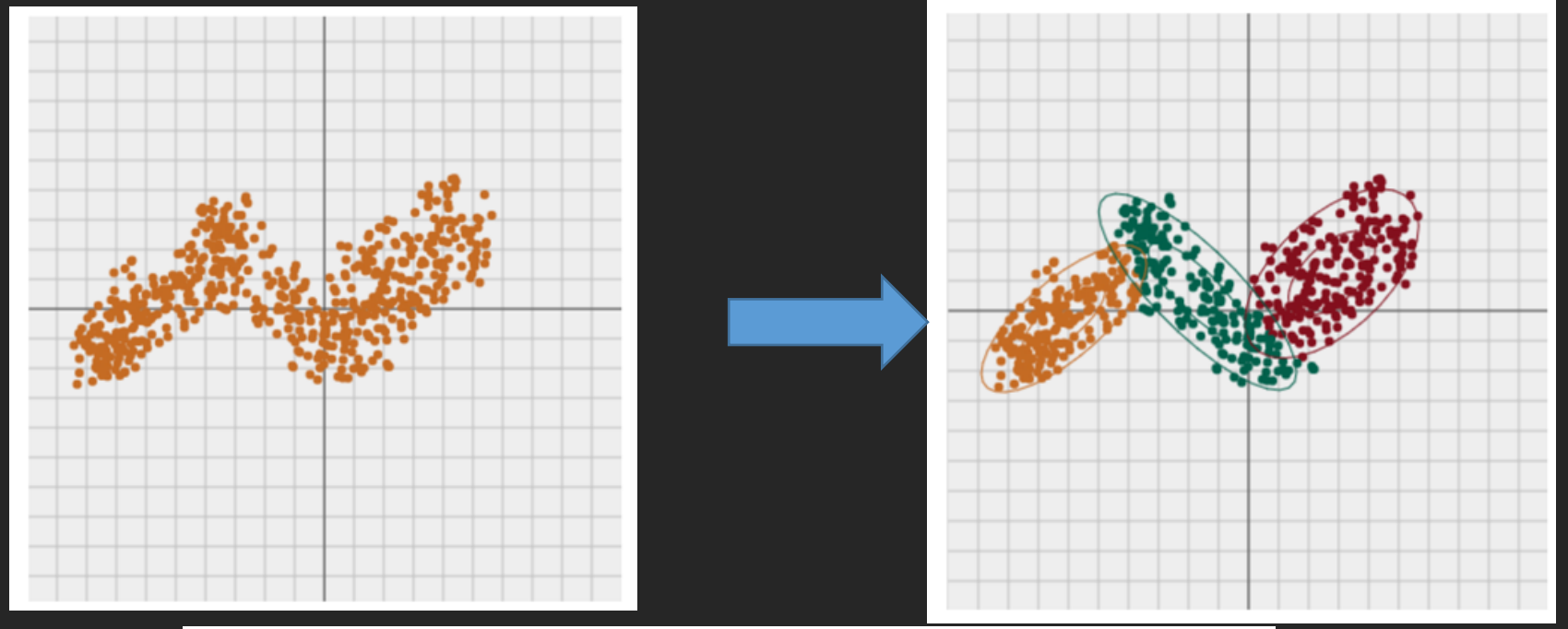
1. $\mathbf{z} \sim \text{Categorical}(1, \cdots, K)$
2. $p(\mathbf{x} \mid \mathbf{z} = k) = \mathcal{N}(\mu_k, \Sigma_k)$



Mixture of Gaussians as a simple latent variable model

- Unsupervised learning: the posterior p(z|x) identifies the mixture component

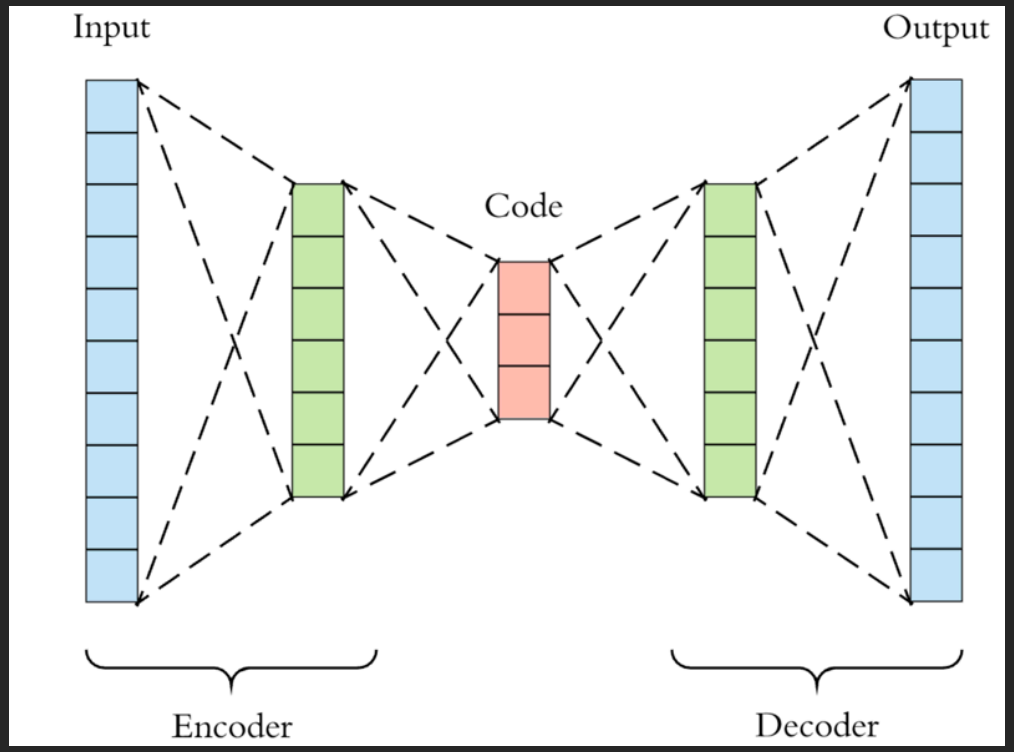
i = argmax p(z=i|x)



$$p(\mathbf{x}) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}) = \sum_{\mathbf{z}} p(\mathbf{z}) p(\mathbf{x} \mid \mathbf{z}) = \sum_{k=1}^{K} p(\mathbf{z} = k) \underbrace{\mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k)}_{\text{component}}$$

Autoencoder

Autoencoder is originally proposed for dimensionality reduction and feature learning



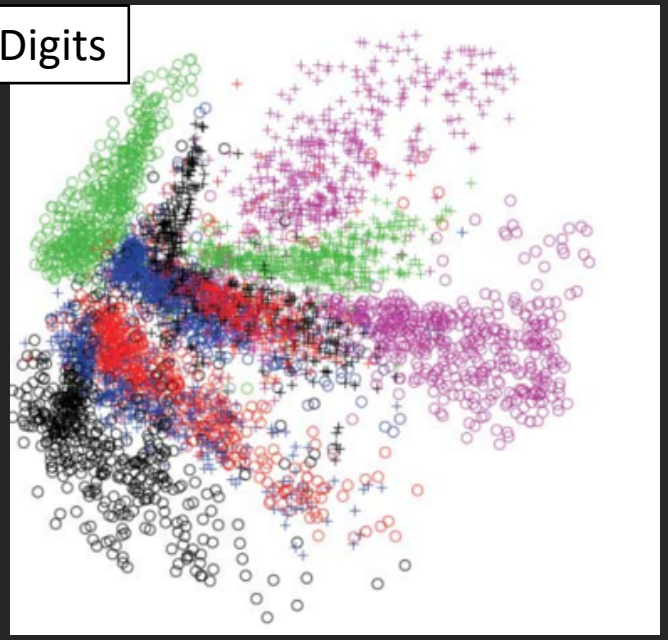
G.E. Hinton et al Science, 2016.

Autoencoder

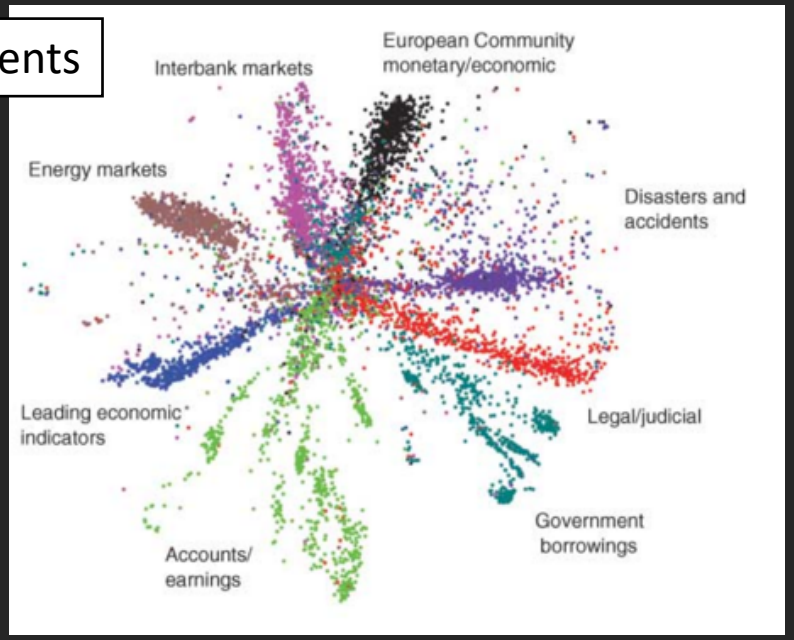
Autoencoder is originally proposed for dimensionality reduction and feature learning

Clustering using the latent features

MNIST Digits



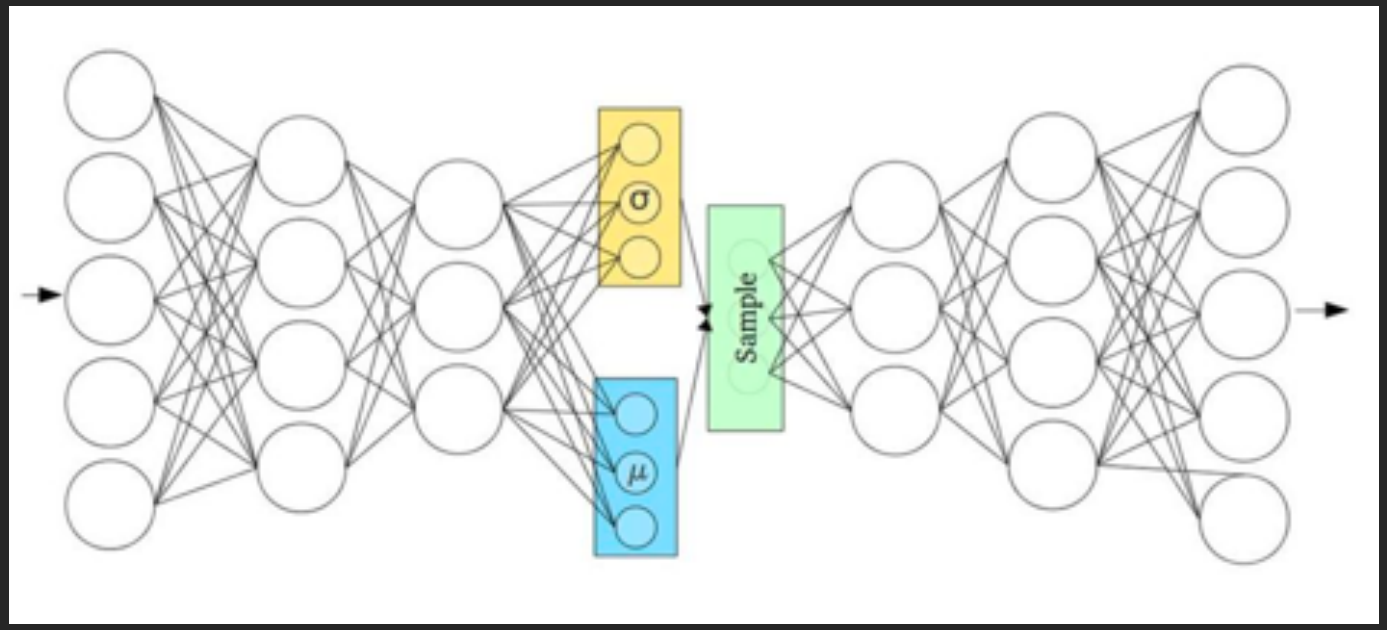
Documents



G.E. Hinton et al Science, 2016.

Variational Autoencoder

- Problem with the vanilla autoencoder is that we cannot sample the latent code to generate new image
- Thus variational autoencoder is introduced



https://github.com/pytorch/examples/blob/master/vae/main.py

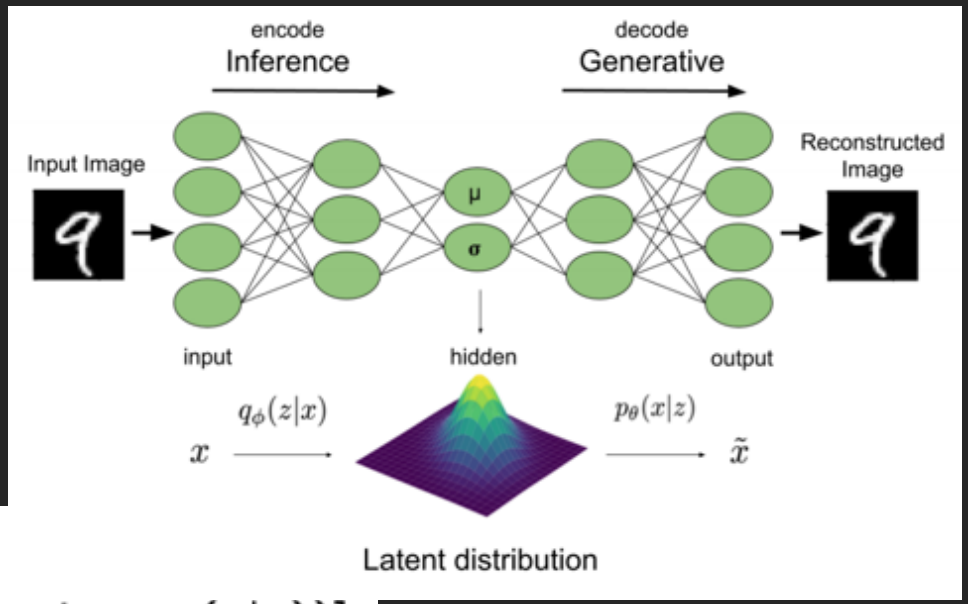
Variational Inference

Evidence Lower Bound (ELBO) holds for any q

$$
\begin{aligned}
\log p(\mathbf{x};\theta) &\geq \sum_{\mathbf{z}} q(\mathbf{z}) \log \left(\frac{p_\theta(\mathbf{x},\mathbf{z})}{q(\mathbf{z})} \right) \\
&= \sum_{\mathbf{z}} q(\mathbf{z}) \log p_\theta(\mathbf{x},\mathbf{z}) \underbrace{- \sum_{\mathbf{z}} q(\mathbf{z}) \log q(\mathbf{z})}_{\text{Entropy } H(q) \text{ of } q} \\
&= \sum q(\mathbf{z}) \log p_\theta(\mathbf{x},\mathbf{z}) + H(q) \\
&= \mathcal{L}(\mathbf{x};\theta,\phi) + D_{KL}(q(\mathbf{z};\phi) \| p(\mathbf{z}|\mathbf{x};\theta))
\end{aligned}
$$

The better $q(\mathbf{z};\phi)$ can approximate the posterior $p(\mathbf{z}|\mathbf{x};\theta)$, the smaller $D_{KL}(q(\mathbf{z};\phi)\|p(\mathbf{z}|\mathbf{x};\theta))$ we can achieve, the closer ELBO will be to $\log p(\mathbf{x};\theta)$. Next: jointly optimize over θ and ϕ to maximize the ELBO over a dataset

Variational Inference



$$\begin{aligned}\mathcal{L}(\mathbf{x};\theta,\phi) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log q_\phi(\mathbf{z}|\mathbf{x}))] \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{z},\mathbf{x};\theta) - \log p(\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}))] \\ &= E_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p(\mathbf{x}|\mathbf{z};\theta)] - D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))\end{aligned}$$

1. Take a data point $\mathbf{x}^i$
2. Map it to $\hat{\mathbf{z}}$ by sampling from $q_\phi(\mathbf{z}|\mathbf{x}^i)$ (*encoder*)
3. Reconstruct $\hat{\mathbf{x}}$ by sampling from $p(\mathbf{x}|\hat{\mathbf{z}};\theta)$ (*decoder*)

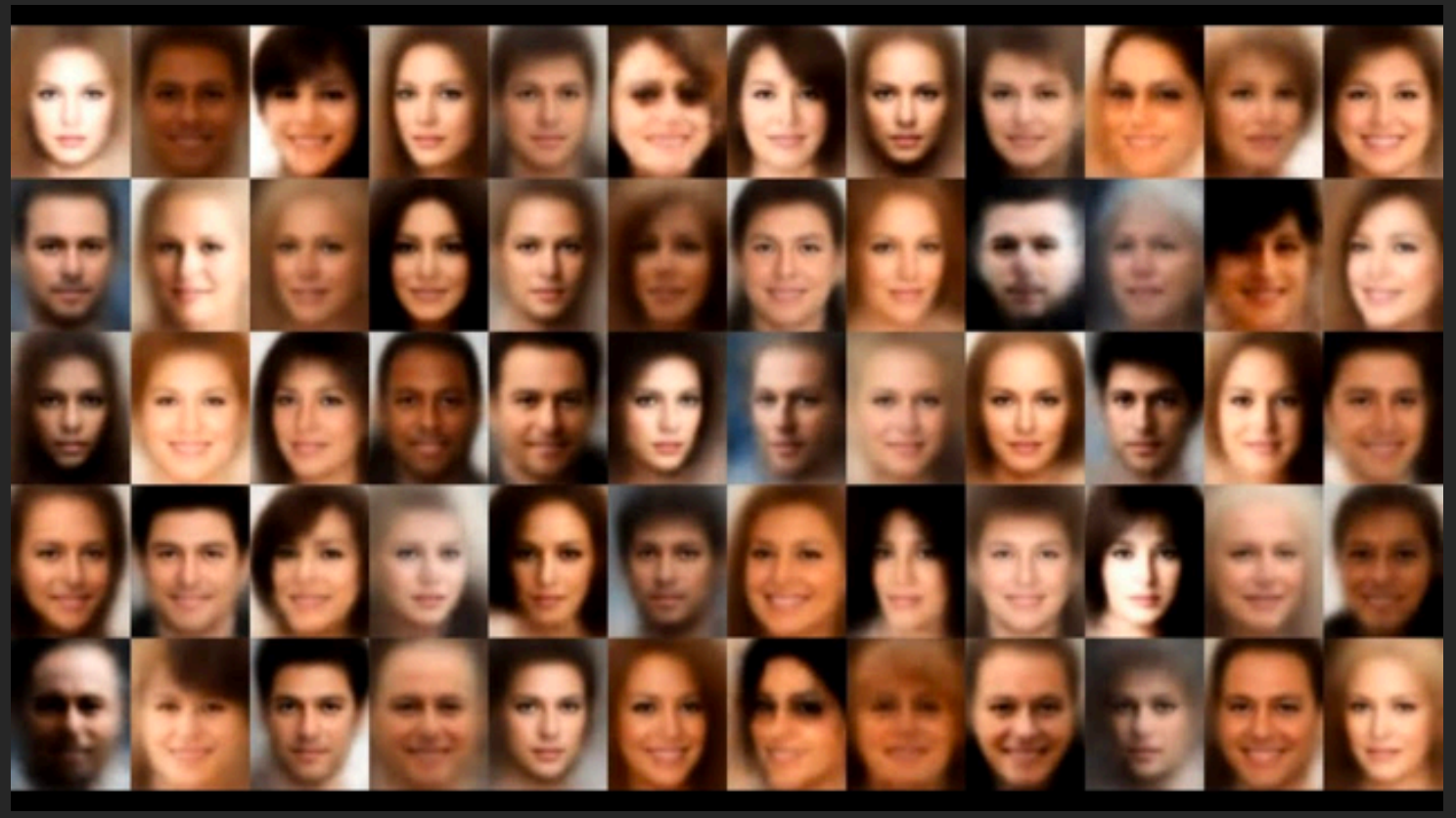
What does the training objective $\mathcal{L}(\mathbf{x};\theta,\phi)$ do?

- First term encourages $\hat{\mathbf{x}} \approx \mathbf{x}^i$ ($\mathbf{x}^i$ likely under $p(\mathbf{x}|\hat{\mathbf{z}};\theta)$)
- Second term encourages $\hat{\mathbf{z}}$ to be likely under the prior $p(\mathbf{z})$

https://deepgenerativemodels.github.io/assets/slides/cs236_lecture5.pdf
https://deepgenerativemodels.github.io/assets/slides/cs236_lecture6.pdf

Variational Autoencoder

Synthesis quality looked good, but not good for recent two-year standard



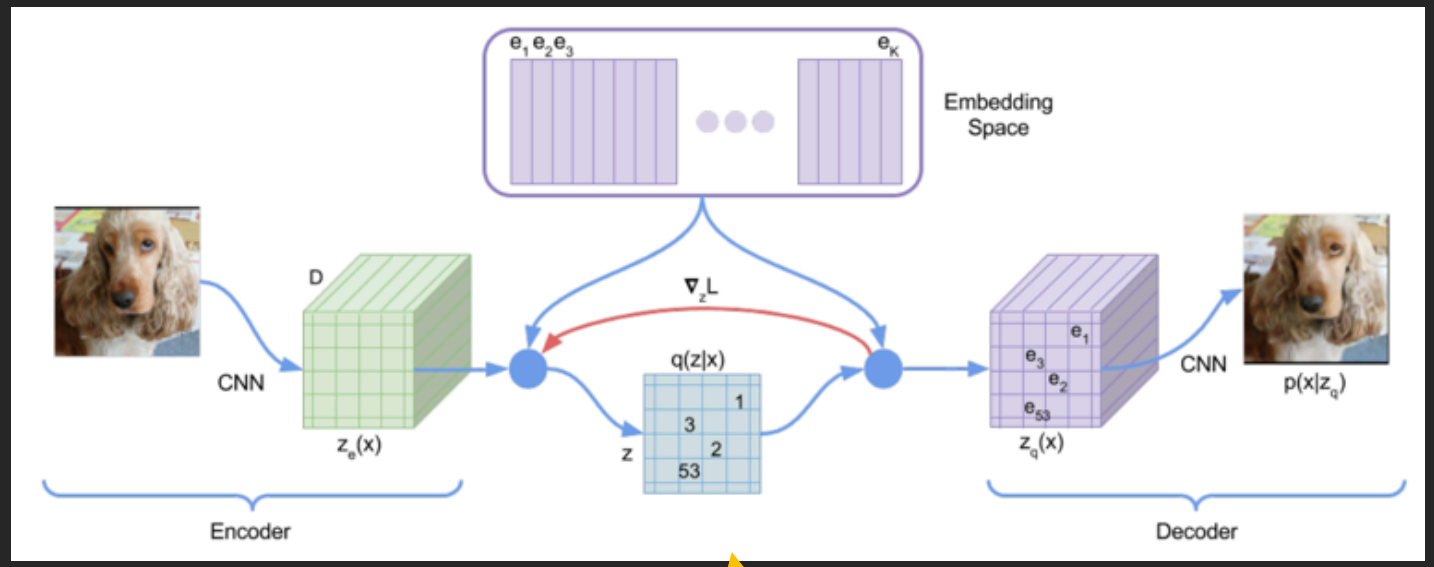
VQ-VAE and VQ-VAE-2

- Vector Quantized Variational AutoEncoder (VQ-VAE)
- Discrete latent representation + PixelCNN for handling posterior collapse

Figure 1: Class-conditional 256x256 image samples from a two-level model trained on ImageNet.

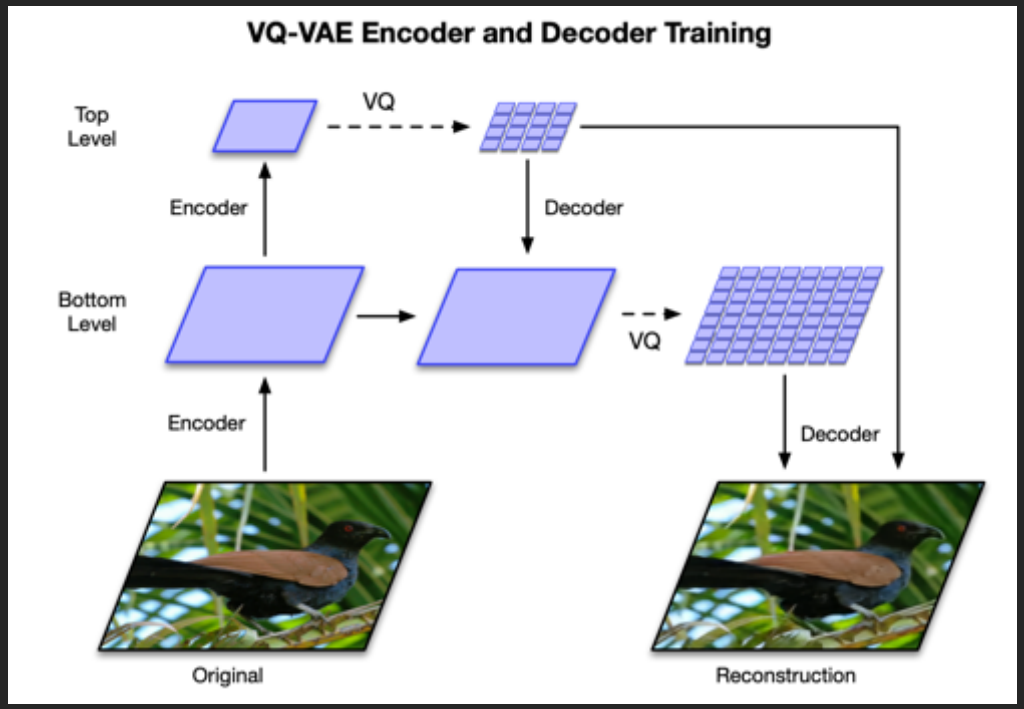
https://arxiv.org/pdf/1711.00937.pdf https://arxiv.org/pdf/1906.00446.pdf

VQ-VAE-1



PixeCNN is learned on the discrete latent space

VQ-VAE-2



https://arxiv.org/pdf/1711.00937.pdf https://arxiv.org/pdf/1906.00446.pdf

VQ-VAE and VQ-VAE-2

Very well handle mode-collapse issue in generative models

VQ-VAE on ImageNet's Tinca class

BigGAN

https://arxiv.org/pdf/1711.00937.pdf https://arxiv.org/pdf/1906.00446.pdf

Summary of Latent Variable Models

Pros:

- Easy to build flexible models
- Suitable for unsupervised learning

Cons:

- Hard to evaluate likelihood
- Hard to train via maximum-likelihood, as it is difficult to compute the posterior inference $p(z|x)$ which requires variational approximations

Autoregressive models and Variational Autoencoders

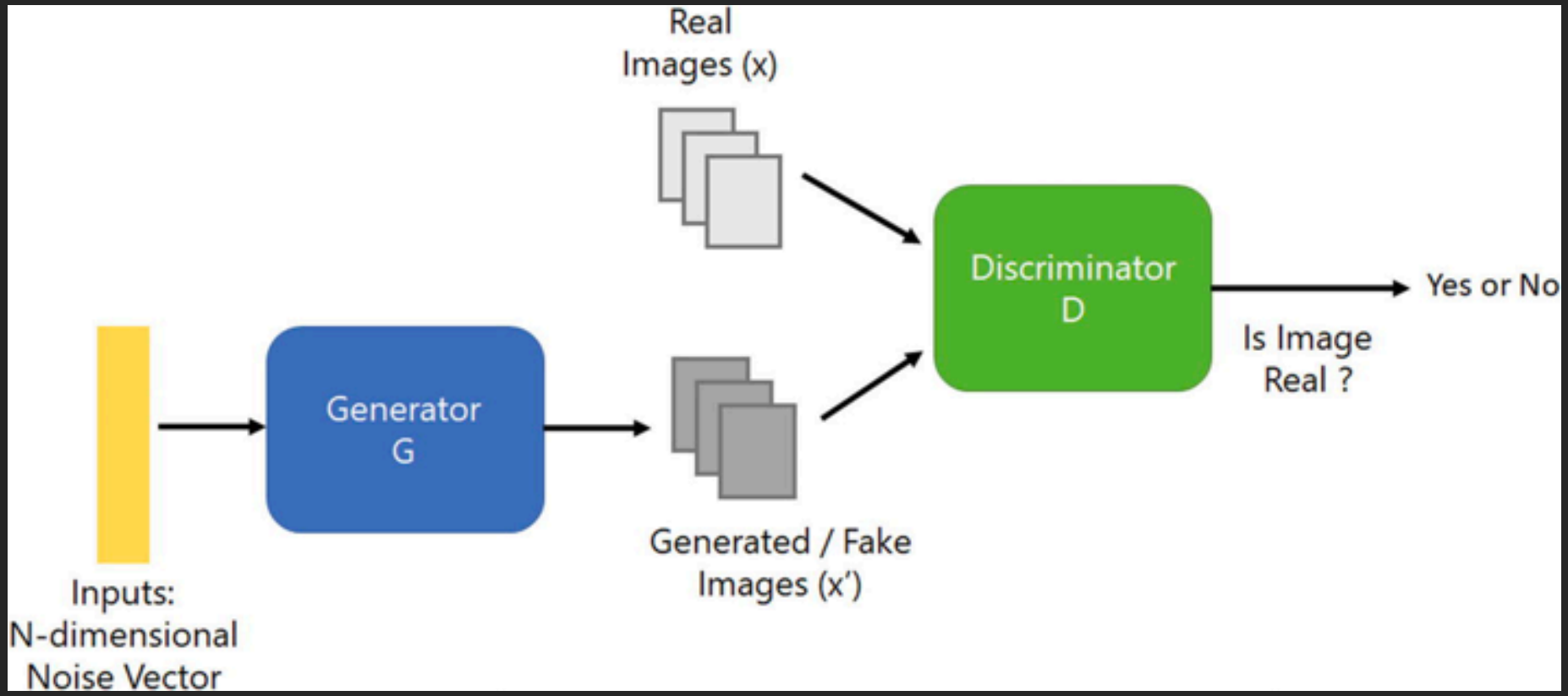
- Both model families are based on maximizing likelihoods (or approximation)

$$\text{Autoregressive Models: } p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i|\mathbf{x}_{<i})$$
$$\text{Variational Autoencoders: } p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z}$$

- Is that likelihood a good indicator? Any other likelihood-free generative modeling?
- Can we throw away KL-divergence and likelihood???

Generative Adversarial Networks

A two-player minimax game between a generator and a discriminator



Goodfellow et al. NIPS'14

Training objective for discriminator:

$$\max_D V(G, D) = E_{\mathbf{x} \sim p_{\text{data}}}[\log D(\mathbf{x})] + E_{\mathbf{x} \sim p_G}[\log(1 - D(\mathbf{x}))]$$

For a fixed generator G, the discriminator is performing binary classification with the cross entropy objective

- Assign probability 1 to true data points $\mathbf{x} \sim p_{\text{data}}$
- Assign probability 0 to fake samples $\mathbf{x} \sim p_G$

Optimal discriminator

$$D_G^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x})}$$

Training objective for generator:

$$\min_G V(G,D) = E_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + E_{\mathbf{x}\sim p_G}[\log(1-D(\mathbf{x}))]$$

For the optimal discriminator $D_G^*(\cdot)$, we have

$$V(G, D_G^*(\mathbf{x}))$$

$$= E_{\mathbf{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_G(\mathbf{x})}\right] + E_{\mathbf{x}\sim p_G}\left[\log \frac{p_G(\mathbf{x})}{p_{\text{data}}(\mathbf{x})+p_G(\mathbf{x})}\right]$$

$$= E_{\mathbf{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x})+p_G(\mathbf{x})}{2}}\right] + E_{\mathbf{x}\sim p_G}\left[\log \frac{p_G(\mathbf{x})}{\frac{p_{\text{data}}(\mathbf{x})+p_G(\mathbf{x})}{2}}\right] - \log 4$$

$$= \underbrace{D_{KL}\left[p_{\text{data}}, \frac{p_{\text{data}}+p_G}{2}\right] + D_{KL}\left[p_G, \frac{p_{\text{data}}+p_G}{2}\right]}_{2\times\text{Jenson-Shannon Divergence (JSD)}} - \log 4$$

$$= 2D_{JSD}[p_{\text{data}}, p_G] - \log 4 \qquad p_G = p_{\text{data}}$$

Symmetric KL divergence

Optimal generator

Joint loss:

$$\min_{\theta} \max_{\phi} V(G_\theta, D_\phi) = E_{\mathbf{x}\sim p_{\text{data}}}[\log D_\phi(\mathbf{x})] + E_{\mathbf{z}\sim p(\mathbf{z})}[\log(1 - D_\phi(G_\theta(\mathbf{z})))]$$

GAN training algorithm

Repeat:

- Sample minibatch of m training points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}$ from $\mathcal{D}$
- Sample minibatch of m noise vectors $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \ldots, \mathbf{z}^{(m)}$ from p_z
- Update the generator parameters θ by stochastic gradient **descent**

$$\nabla_\theta V(G_\theta, D_\phi) = \frac{1}{m} \nabla_\theta \sum_{i=1}^{m} \log(1 - D_\phi(G_\theta(\mathbf{z}^{(i)})))$$

- Update the discriminator parameters ϕ by stochastic gradient **ascent**

$$\nabla_\phi V(G_\theta, D_\phi) = \frac{1}{m} \nabla_\phi \sum_{i=1}^{m} [\log D_\phi(\mathbf{x}^{(i)}) + \log(1 - D_\phi(G_\theta(\mathbf{z}^{(i)})))]$$

Example code of training GAN

```
for i, data in enumerate(dataloader, 0):
    ############################
    # (1) Update D network: maximize log(D(x)) + log(1 - D(G(z)))
    ###########################
    # train with real
    netD.zero_grad()
    real_cpu = data[0].to(device)
    batch_size = real_cpu.size(0)
    label = torch.full((batch_size,), real_label, device=device)

    output = netD(real_cpu)
    errD_real = criterion(output, label)
    errD_real.backward()
    D_x = output.mean().item()

    # train with fake
    noise = torch.randn(batch_size, nz, 1, 1, device=device)
    fake = netG(noise)
    label.fill_(fake_label)
    output = netD(fake.detach())
    errD_fake = criterion(output, label)
    errD_fake.backward()
    D_G_z1 = output.mean().item()
    errD = errD_real + errD_fake
    optimizerD.step()
```

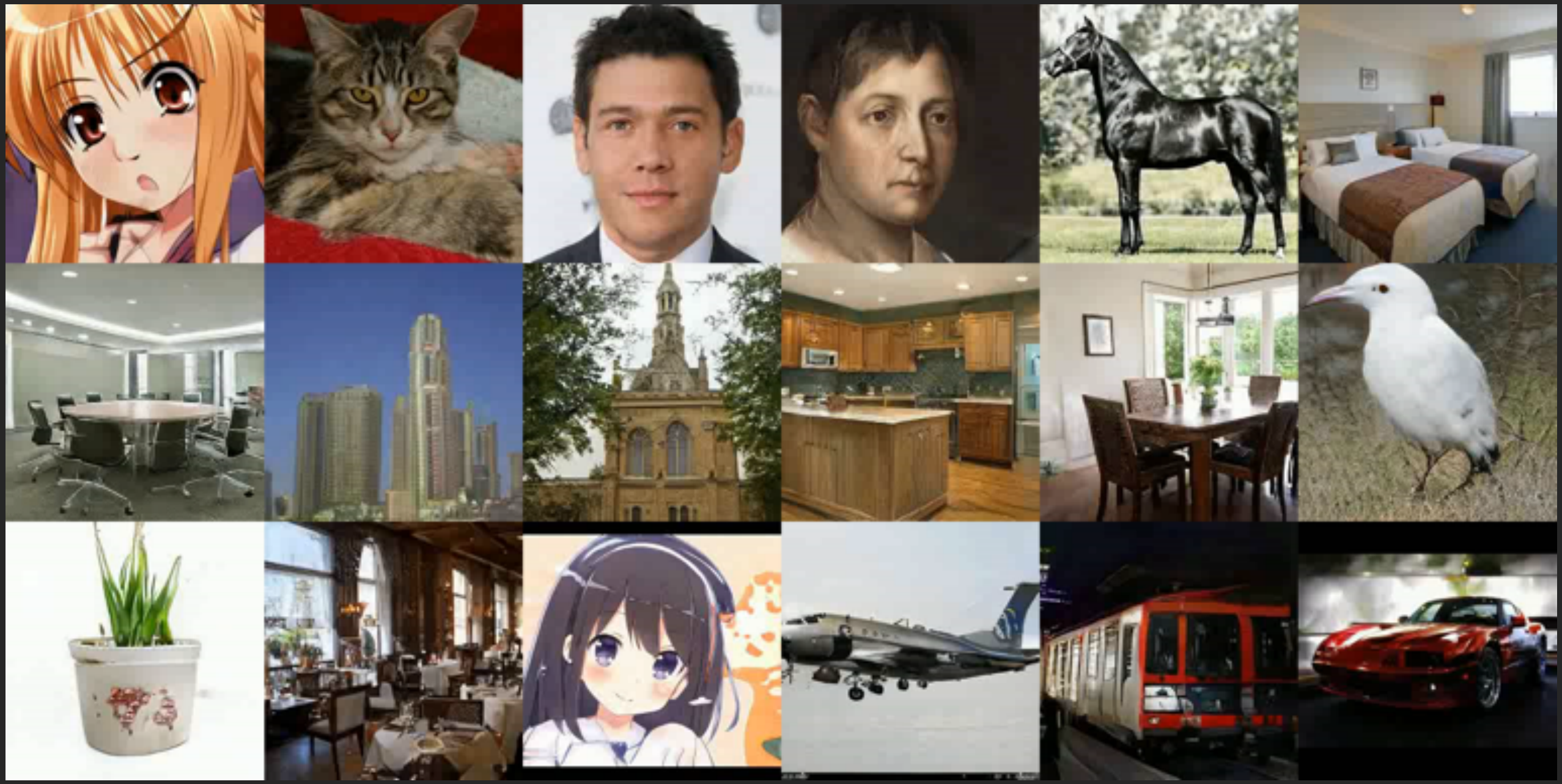
```
    ############################
    # (2) Update G network: maximize log(D(G(z)))
    ###########################
    netG.zero_grad()
    label.fill_(real_label)  # fake labels are real for generator cost
    output = netD(fake)
    errG = criterion(output, label)
    errG.backward()
    D_G_z2 = output.mean().item()
    optimizerG.step()
```

https://github.com/pytorch/examples/blob/master/dcgan/main.py

GenForce: Generative Modeling for Everyone

- https://github.com/genforce/genforce
- Open-source PyTorch library for generative modeling
- 60+ pretrained GAN models with Colab live demo

https://colab.research.google.com/github/genforce/genforce/blob/master/docs/synthesize_demo.ipynb



Live Demo: Try Pretrained Generative Models

- Try the Google Colab for BigGAN:

https://colab.research.google.com/github/tensorflow/hub/blob/master/examples/colab/biggan_generation_with_tf_hub.ipynb

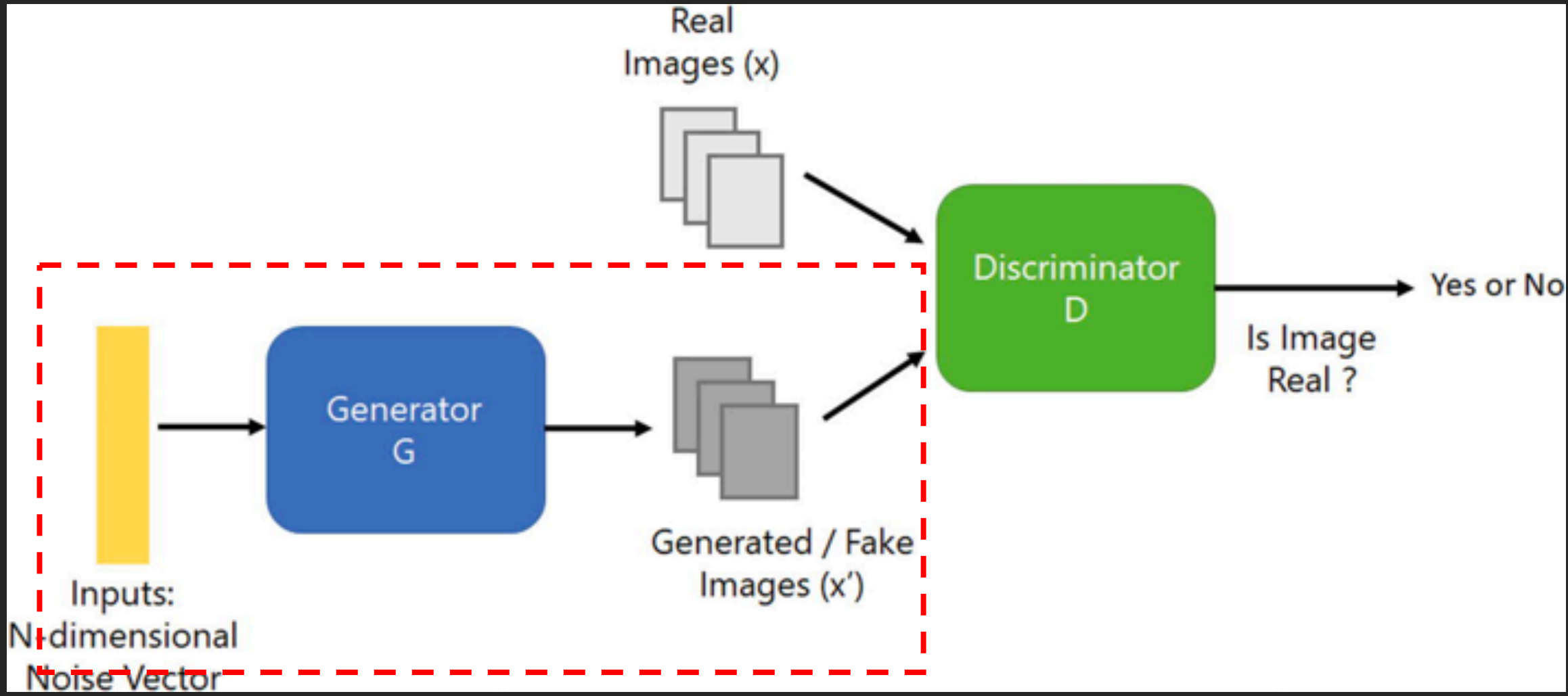
- Try the Google Colab for StyleGAN from GenForce:

https://colab.research.google.com/github/genforce/genforce/blob/master/docs/synthesize_demo.ipynb

Outline

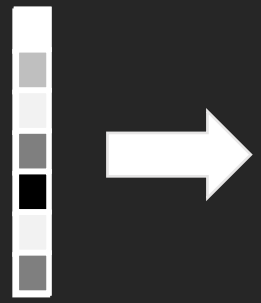
- Introduction: from recognition to recreation
- Overview of the generative models
- Understanding generative models
- Inverting images for real image editing

What have been learned inside the GANs?



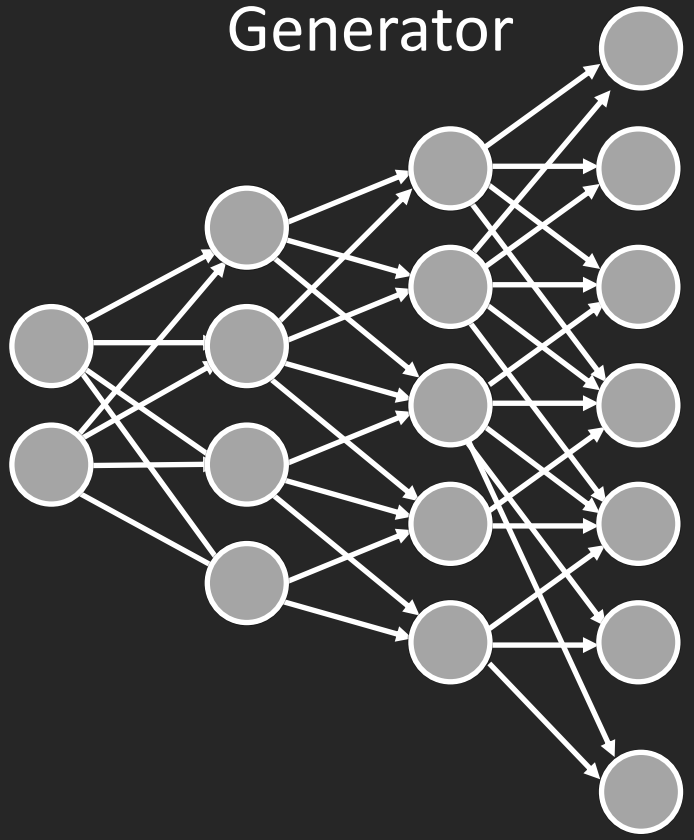
Deep Generative Representation

Random vector



512 dimensions

Generator



Randomly generated image

Some slides are from David Bau

Progressive GANs [Karras et al., ICLR 2018]

Slide from David Bau

Progressive GANs [Karras et al., ICLR 2018]

Slide from David Bau



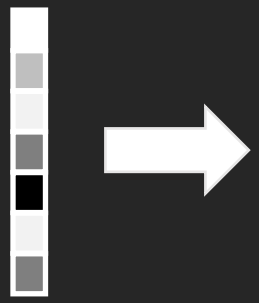




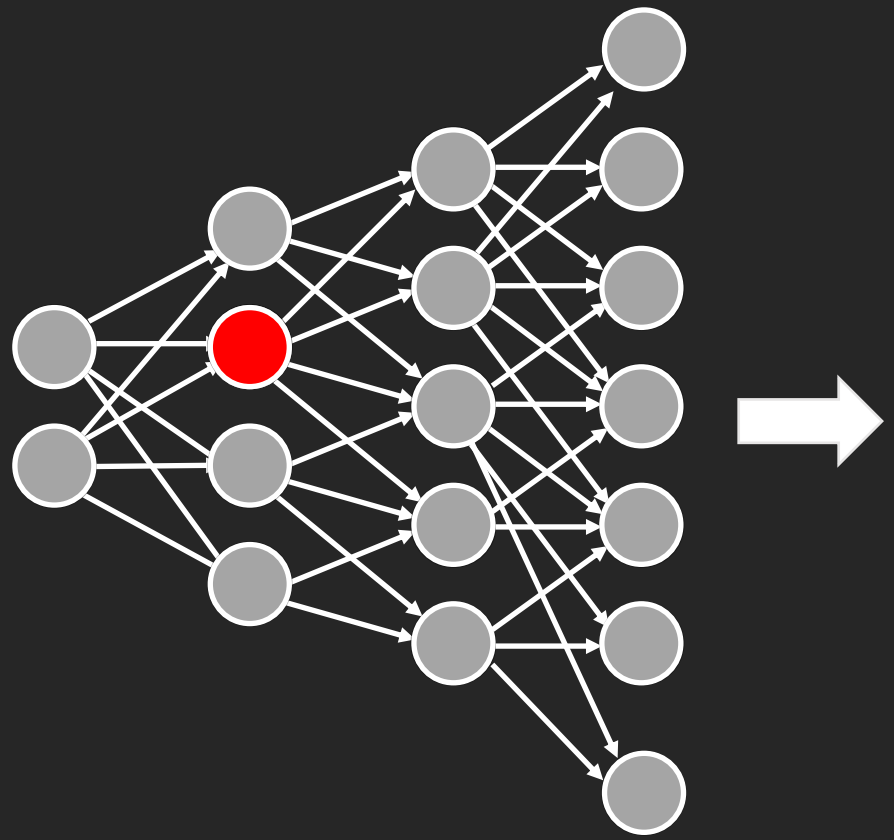

Slide from David Bau

Are there watermark units?

Random vector



512 dimensions



Randomly generated image

Slide from David Bau








Slide from David Bau

Deactivating banner units in layer 4








Slide from David Bau

Deactivating watermark units in layer 4





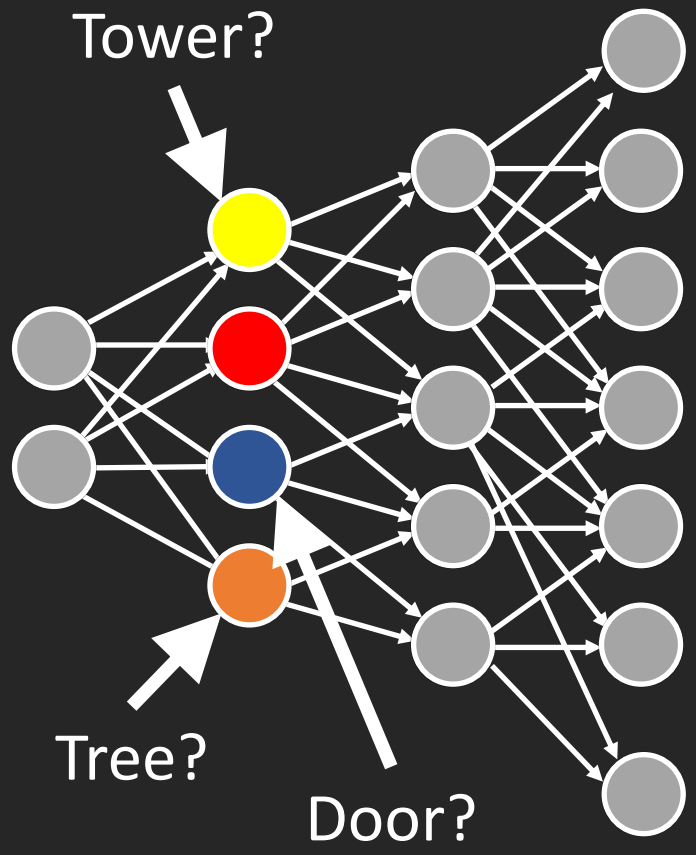



Slide from David Bau

Are there other objects?

Random vector

512 dimensions



Randomly generated image

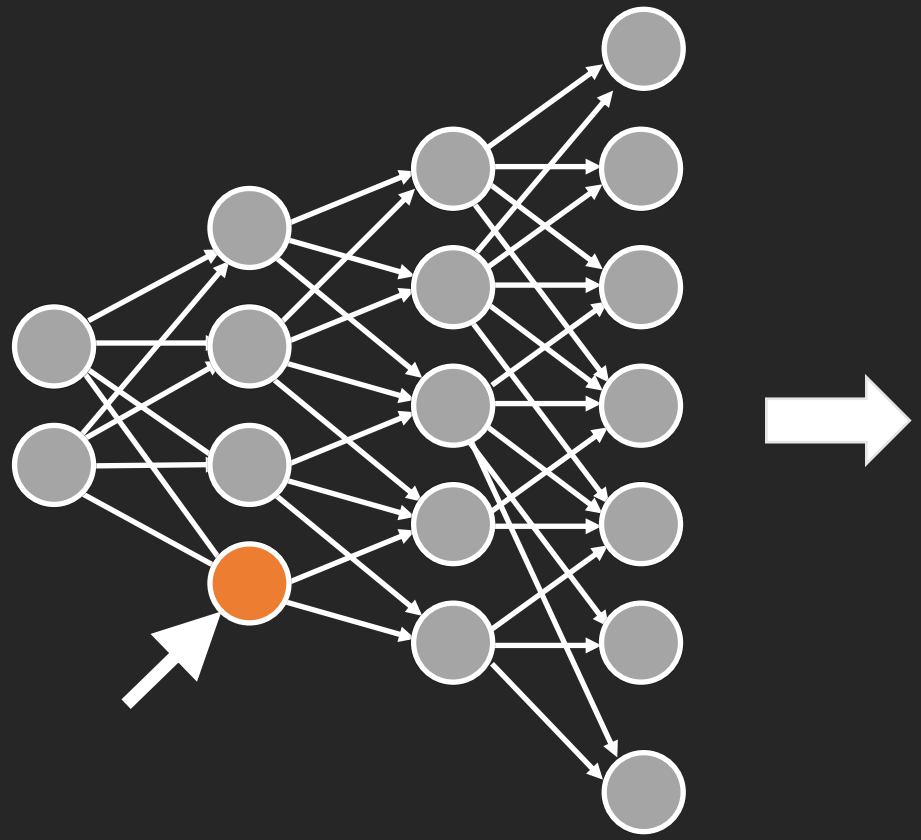
Slide from David Bau

GAN Dissection [Bau et al., ICLR 2019]

Are there other objects?

Random vector

512 dimensions



Randomly generated image

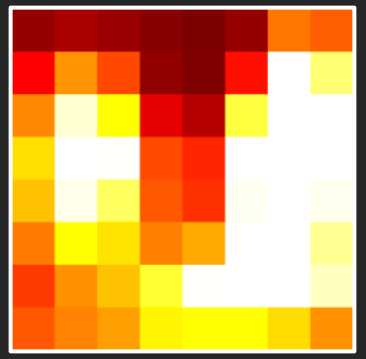
Slide from David Bau

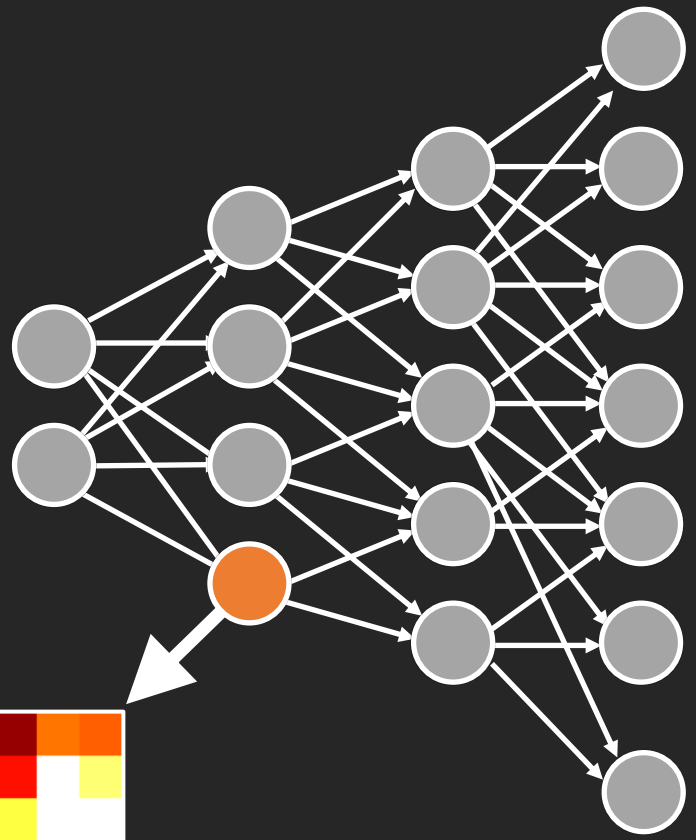
GAN Dissection [Bau et al., ICLR 2019]

Are there other objects?

Random vector

512 dimensions



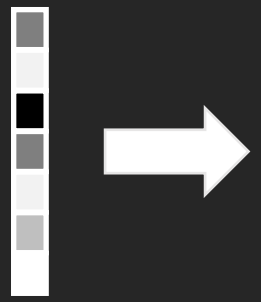


Randomly generated image

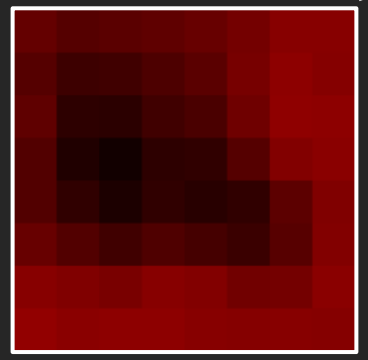
GAN Dissection [Bau et al., ICLR 2019]

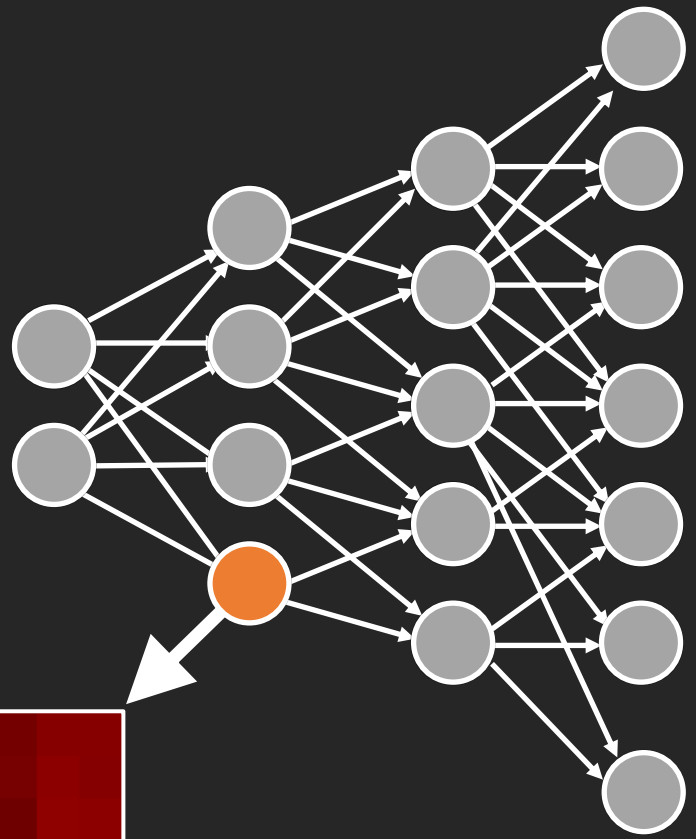
Are there other objects?

Random vector



512 dimensions



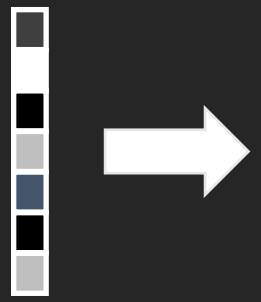


Randomly generated image

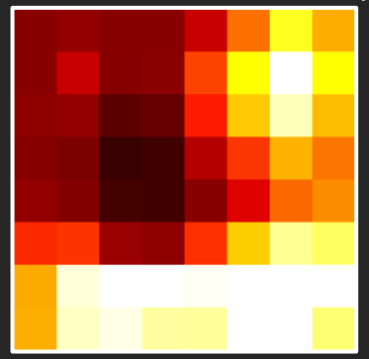
GAN Dissection [Bau et al., ICLR 2019]

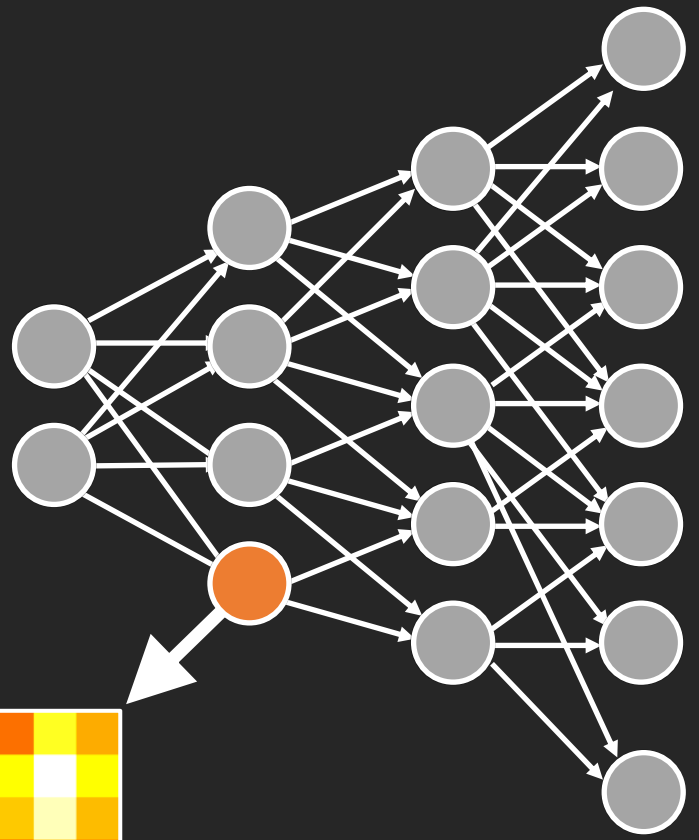
Are there other objects?

Random vector



512 dimensions





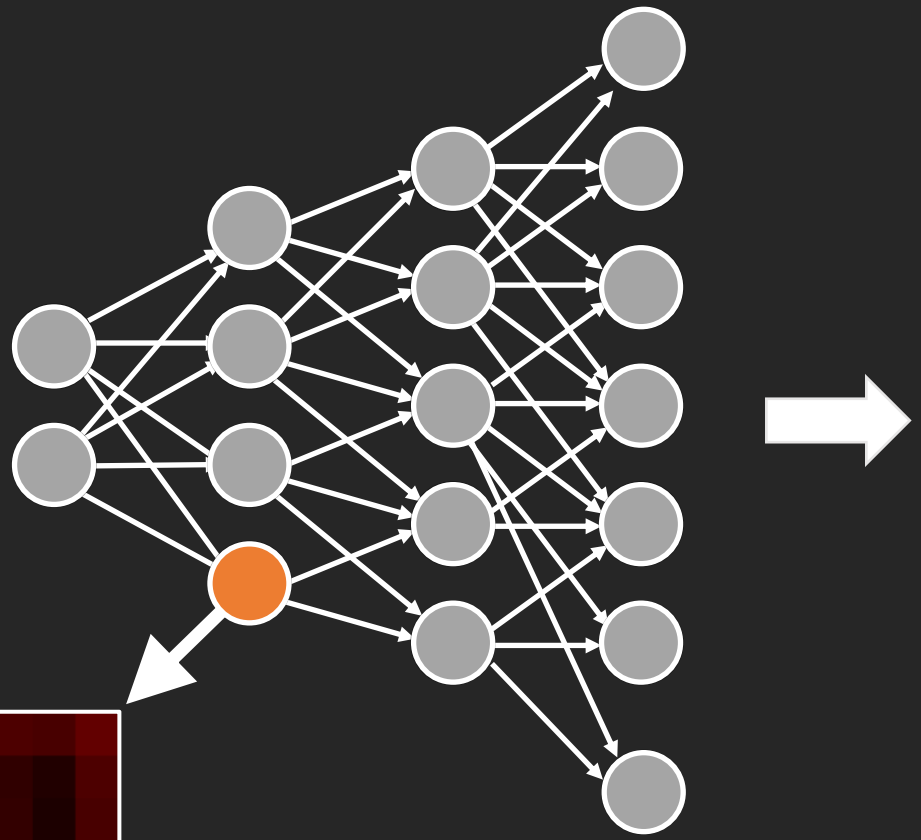
Randomly generated image

GAN Dissection [Bau et al., ICLR 2019]

Are there other objects?

Random vector

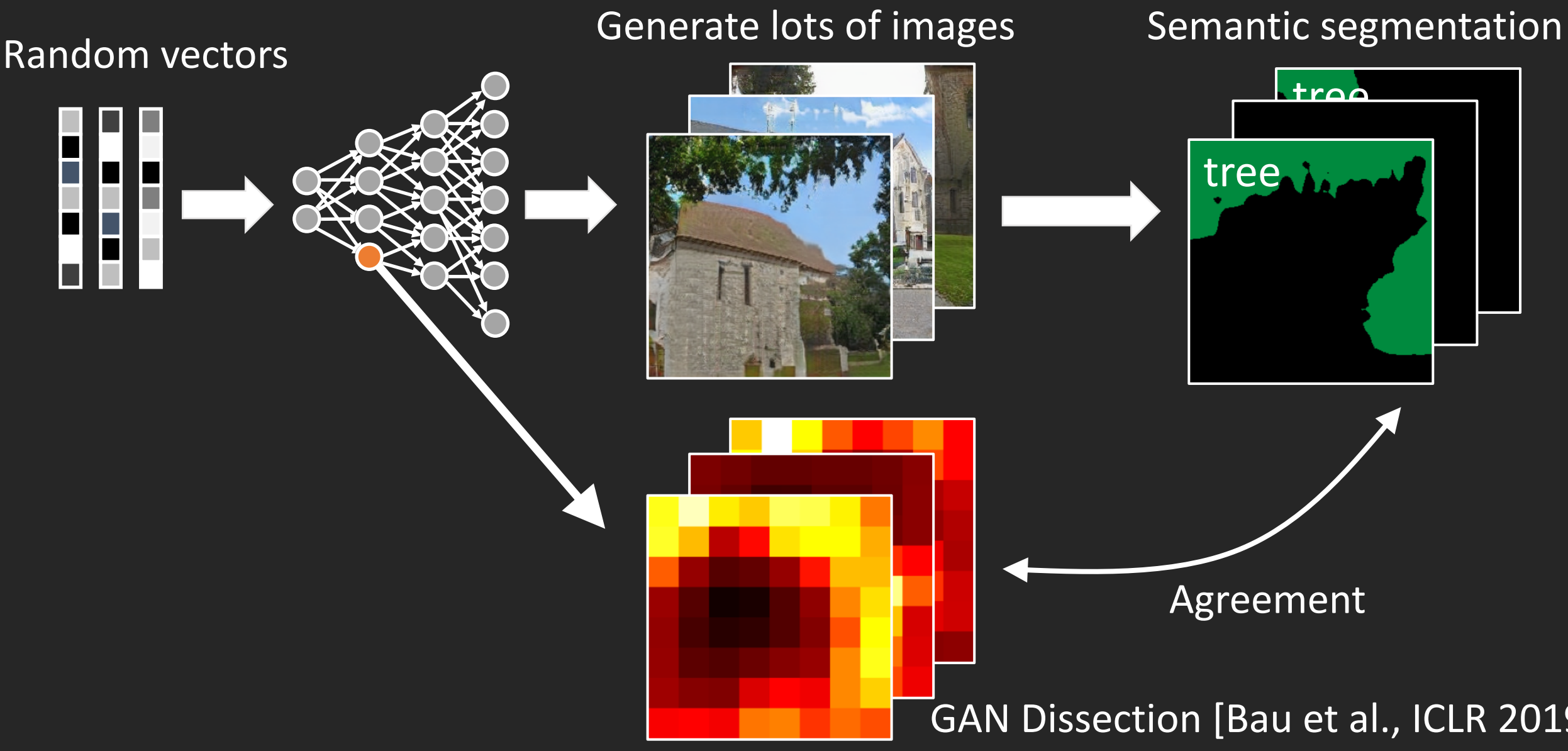
512 dimensions



Randomly generated image

GAN Dissection [Bau et al., ICLR 2019]

Dissecting a GAN

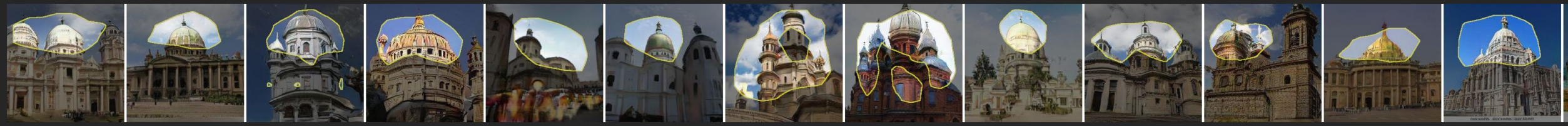


GAN Dissection [Bau et al., ICLR 2019]

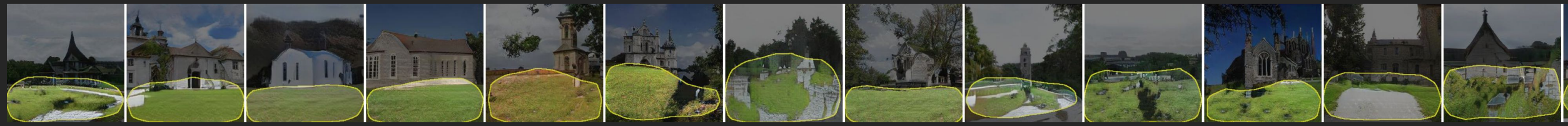
Units Emerge as Drawing Objects

Unit 365 draws trees

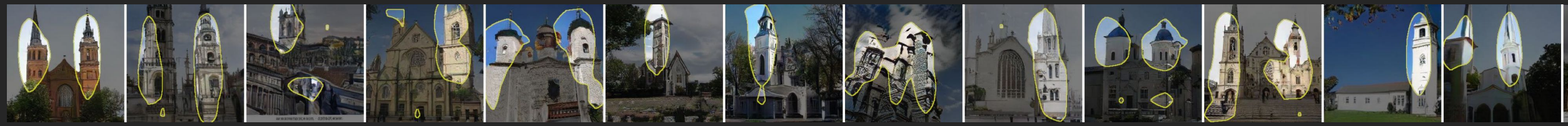
Unit 43 draws domes



Unit 14 draws grass

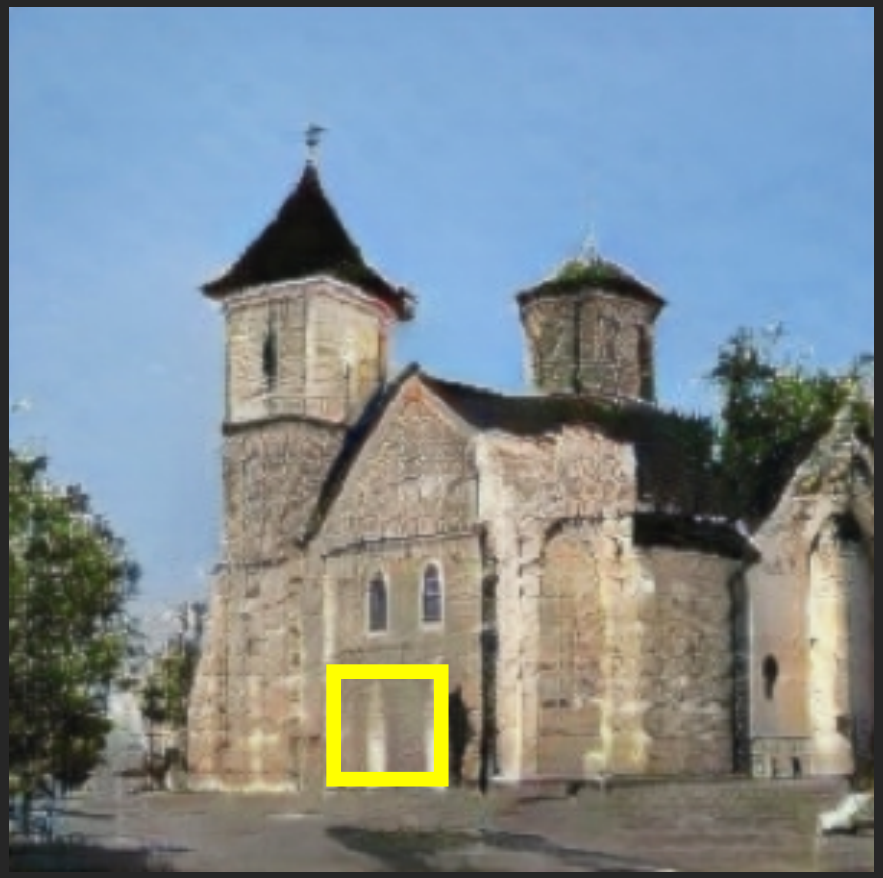


Unit 276 draws towers

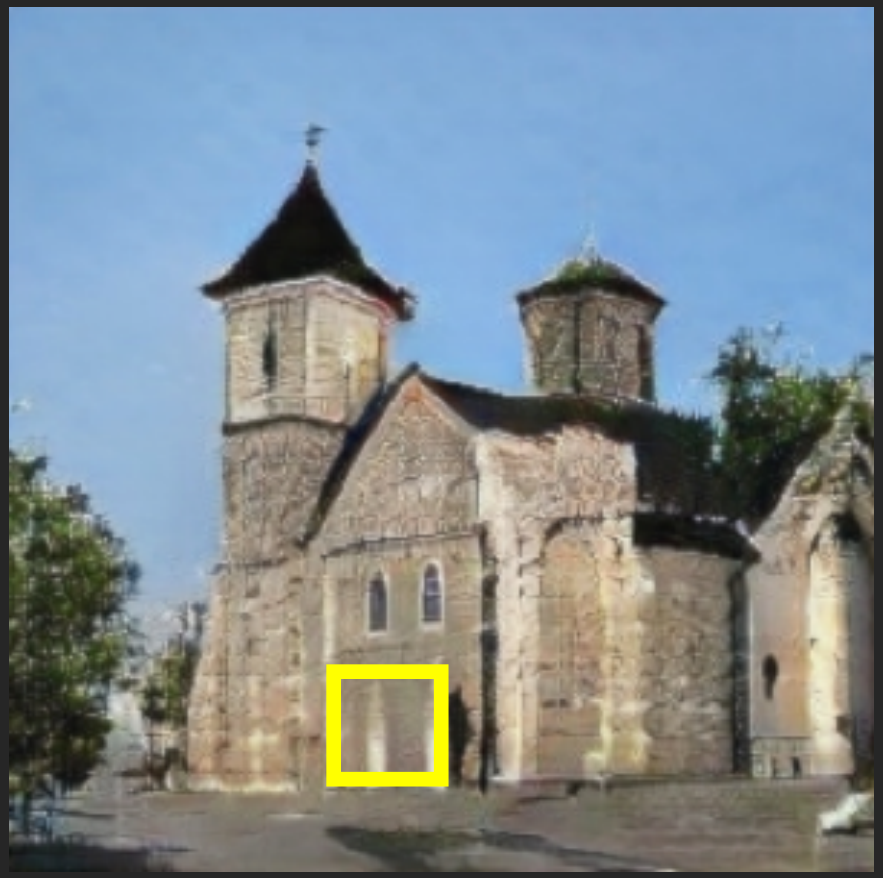


Turning off window units

Turning on door units



Turning on door units



Turning on door units

Turning on door units

Turning on door units

Turning on door units

Interactive Object Editing

Online demo:
ganpaint.io

Semantic Photo Manipulation with a Generative Image Prior. SIGGRAPH 2019

Editing Objects vs. Global Attributes

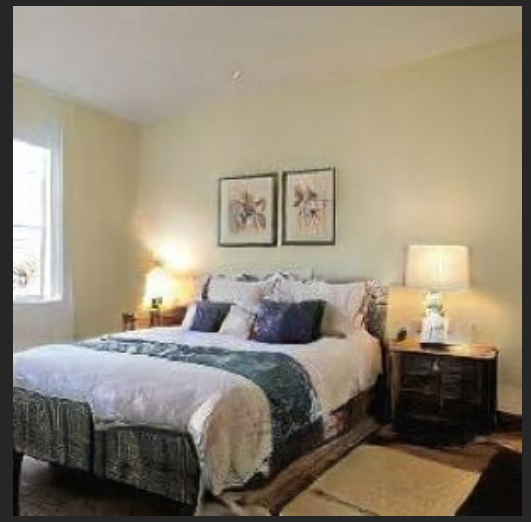
Manipulating Global Attributes

Indoor lighting

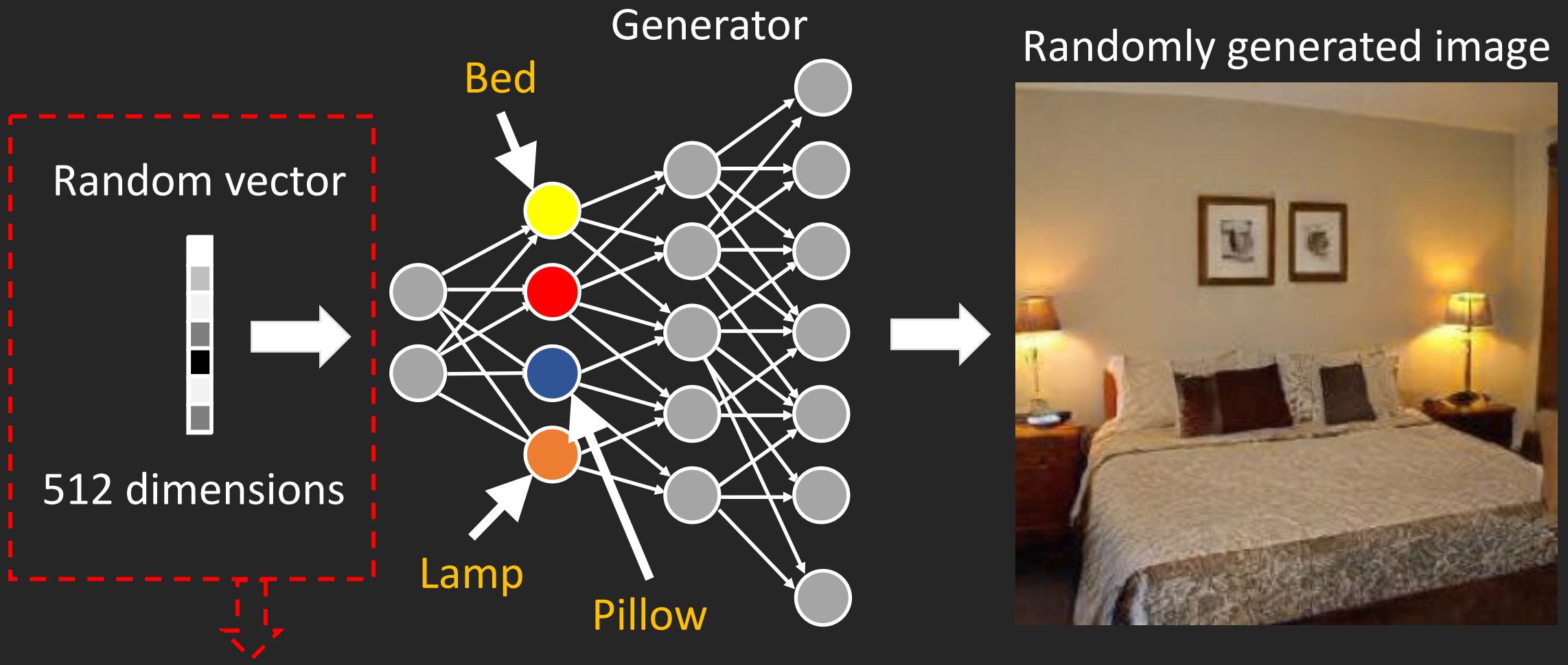
Wooden style

Different layout

Living room



Deep Generative Representation



How latent code affects the output?

Random Walk in Latent Space of Bedroom

Multiple Levels of Abstractions for Scenes

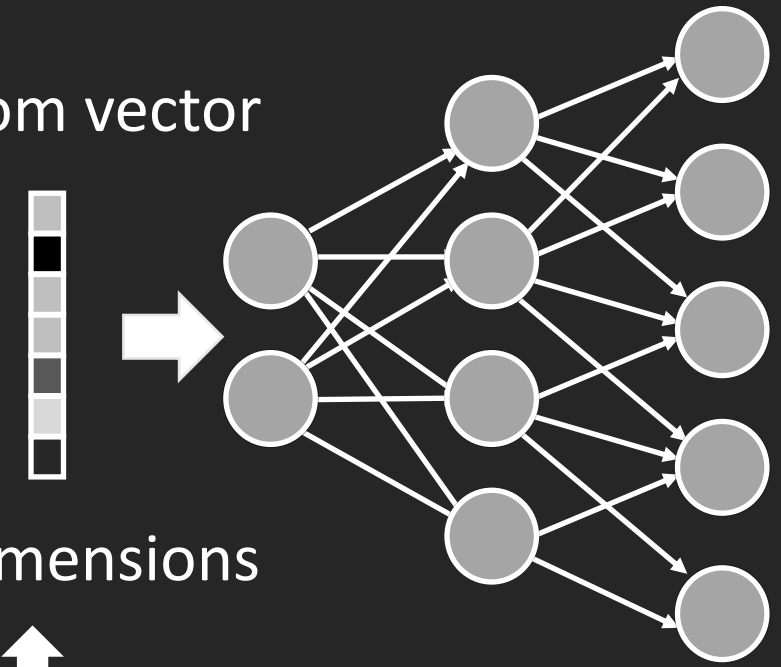
Scene category:
bedroom
Scene attributes:
nature lighting
wood
foliage
...

Layout

Segmentation

Identifying the Causality in Latent Space

Random vector



512 dimensions

Predicted semantics

Scene category: bedroom

Scene attributes: nature lighting, wood, foliage

To identify the cause-effect relations

Ceyuan Yang*, Yujun Shen*, Bolei Zhou. Semantic Hierarchy Emerges in Deep Generative Representations for Scene Synthesis. https://arxiv.org/pdf/1911.09267.pdf

Identifying Causality in Latent Space

Latent Space

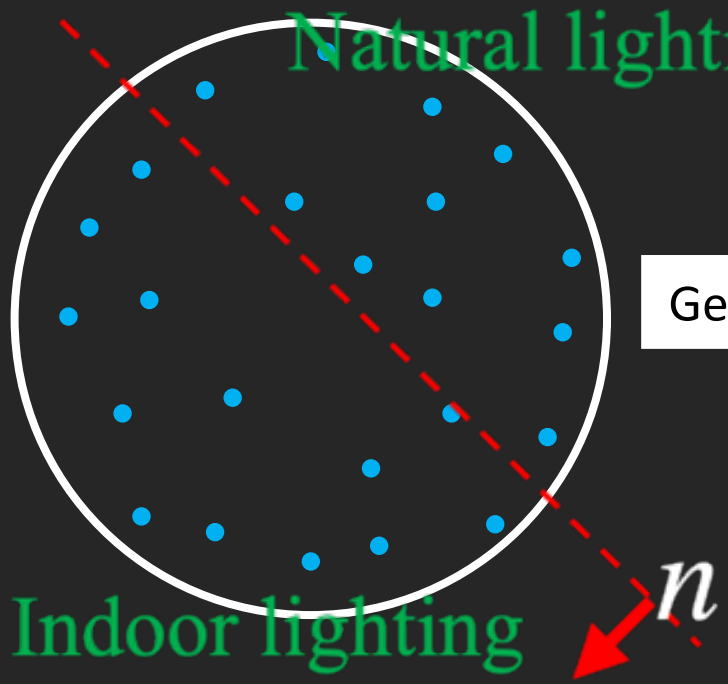
$z_k \sim N(0, I)$

Image Space

$x_k = G(z_k)$

Attribute Space

$a_k = F(x_k)$



Generator

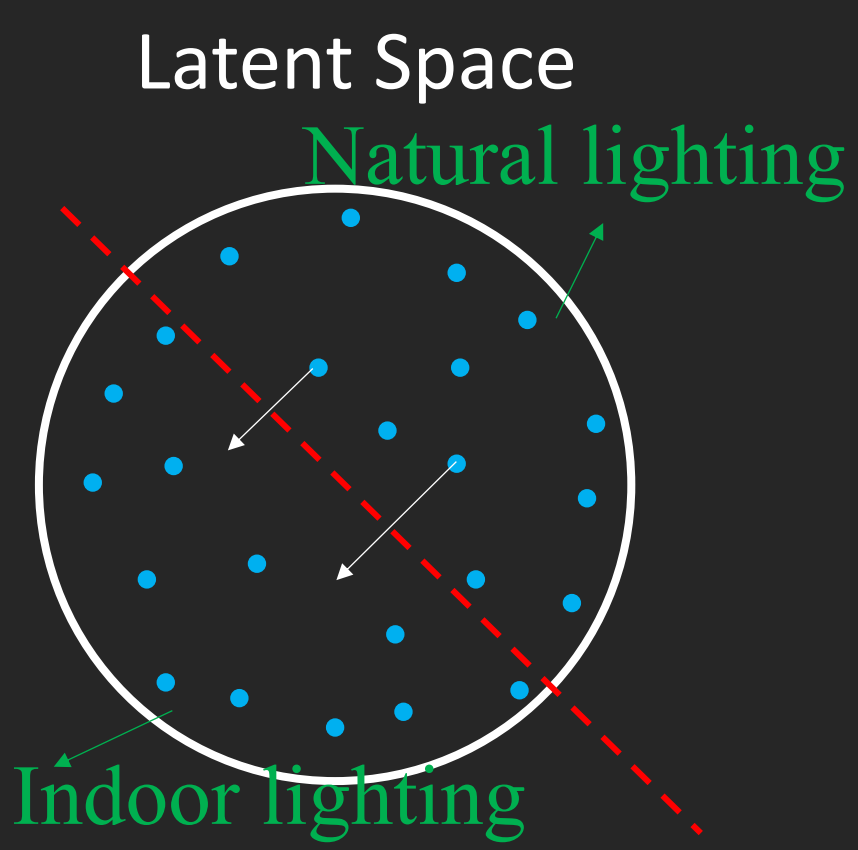
Off-shelf Classifier

Train linear discriminative boundary

Counterfactual Verification:

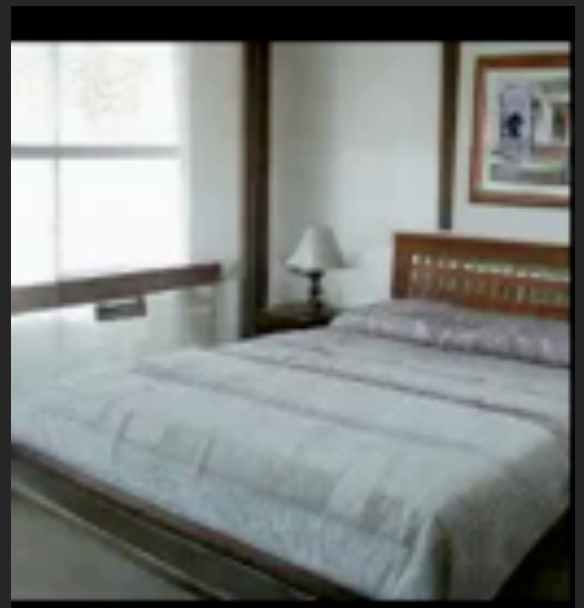
$$\Delta s = \frac{1}{K}\sum_{k=1}^{K} \max(F(G(z_k + \lambda n)) - F(G(z_k)), 0)$$

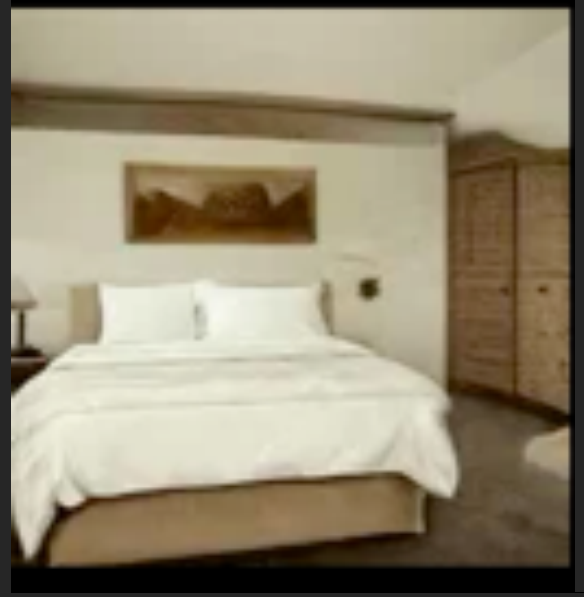
Pushing Latent Code through Boundary



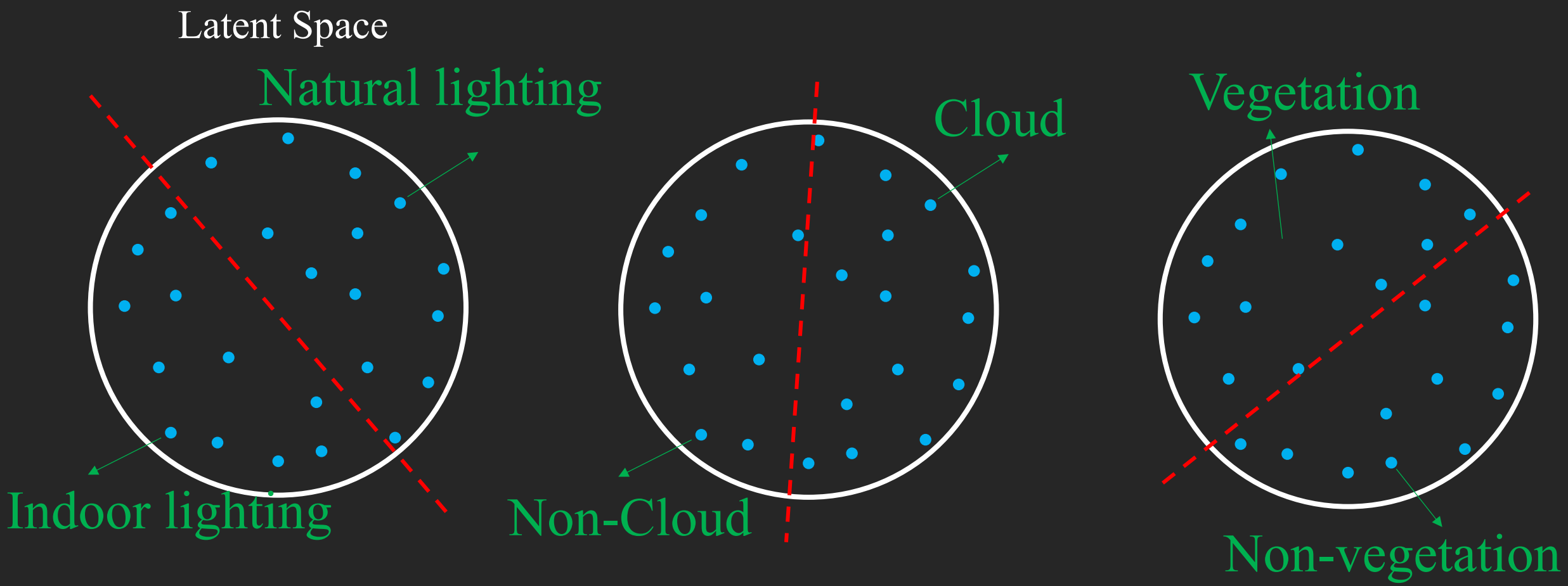
$G(\mathbf{z})$ → More indoor lighting → $G(\mathbf{z} + \lambda\mathbf{b})$

Turning up the lights

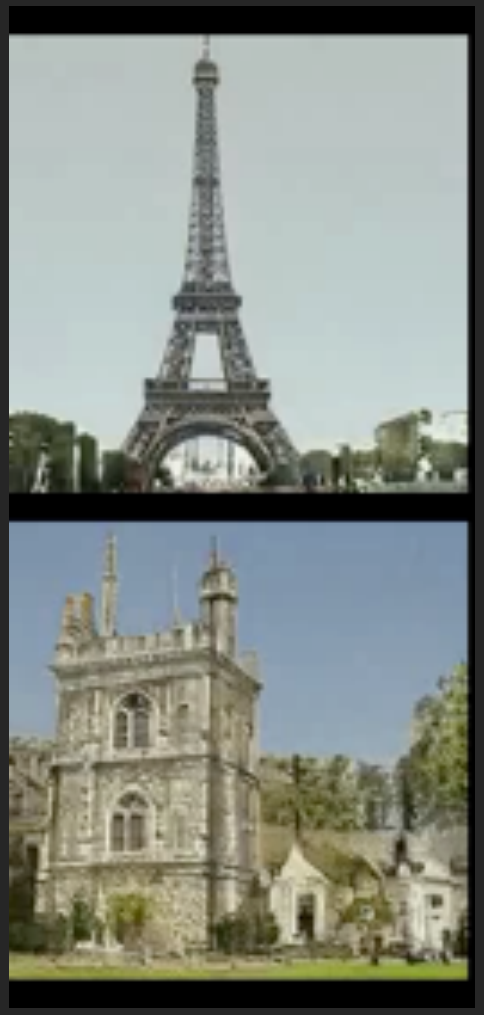




Various Attribute Boundaries in Latent Space

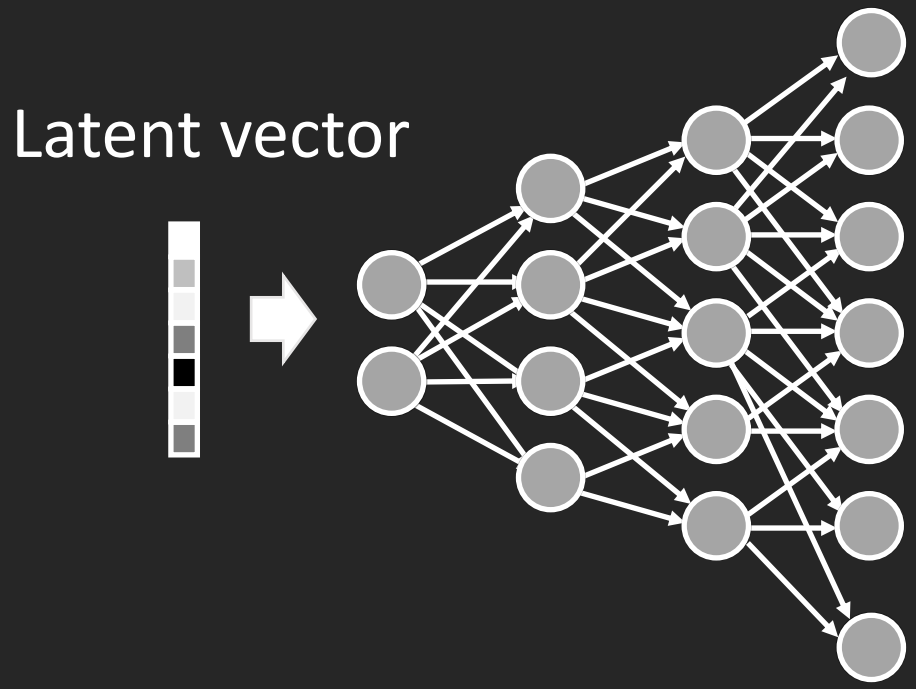


Adding clouds

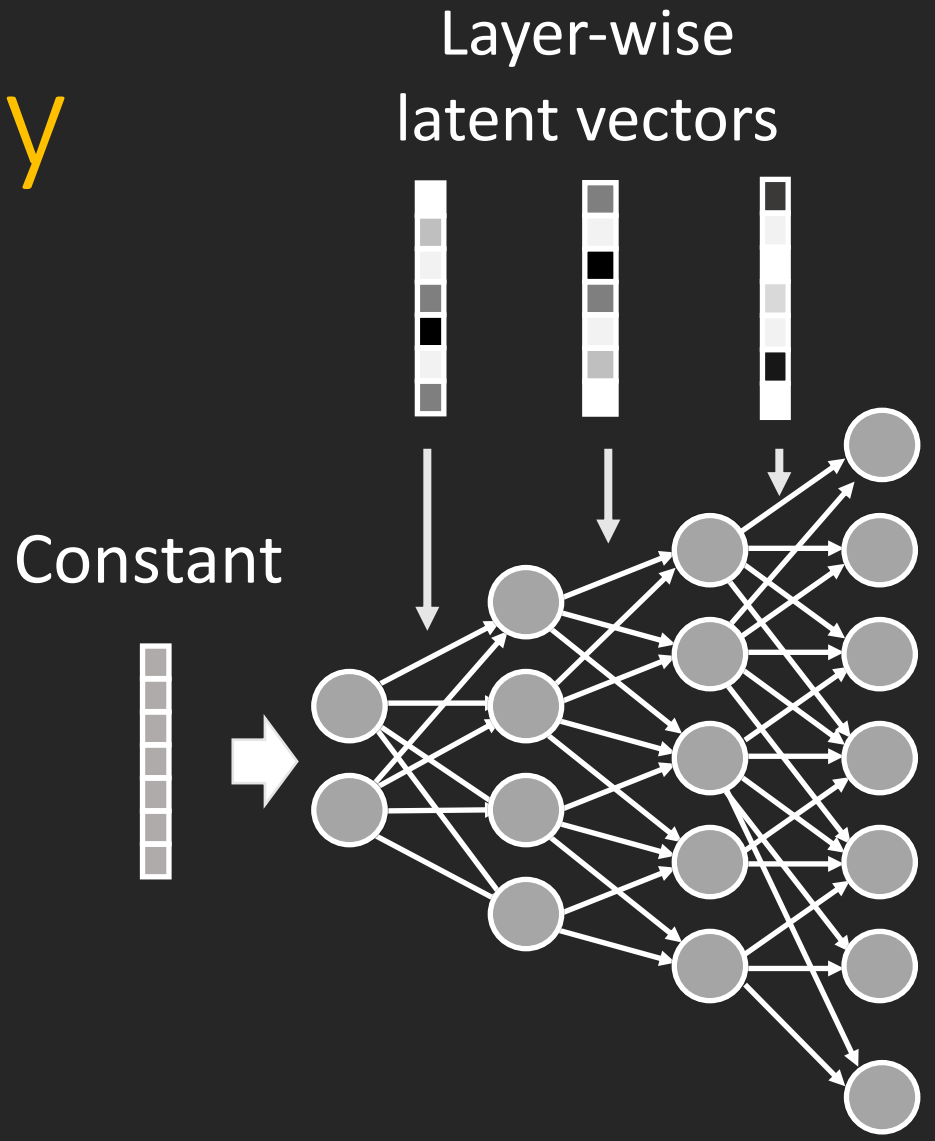


Adding vegetation

Layer-wise Stochasity

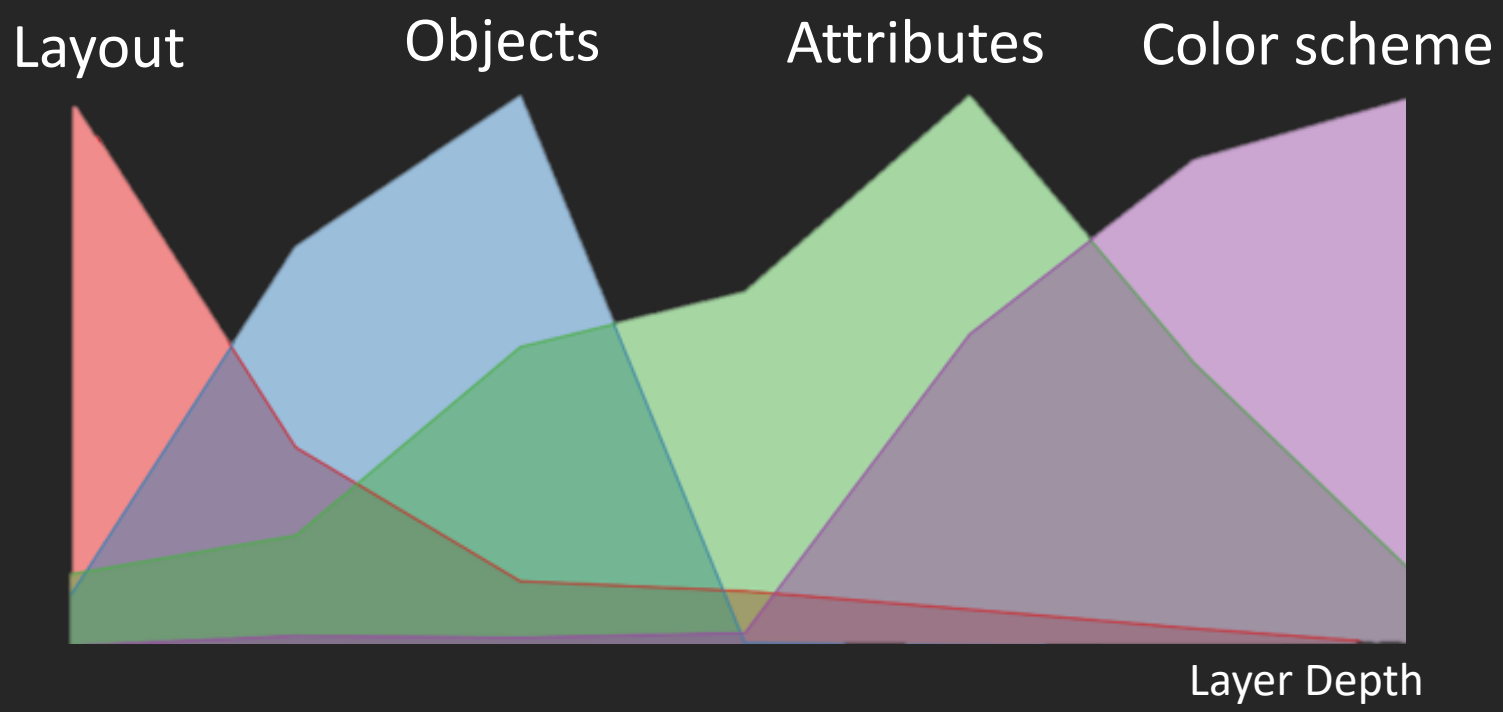


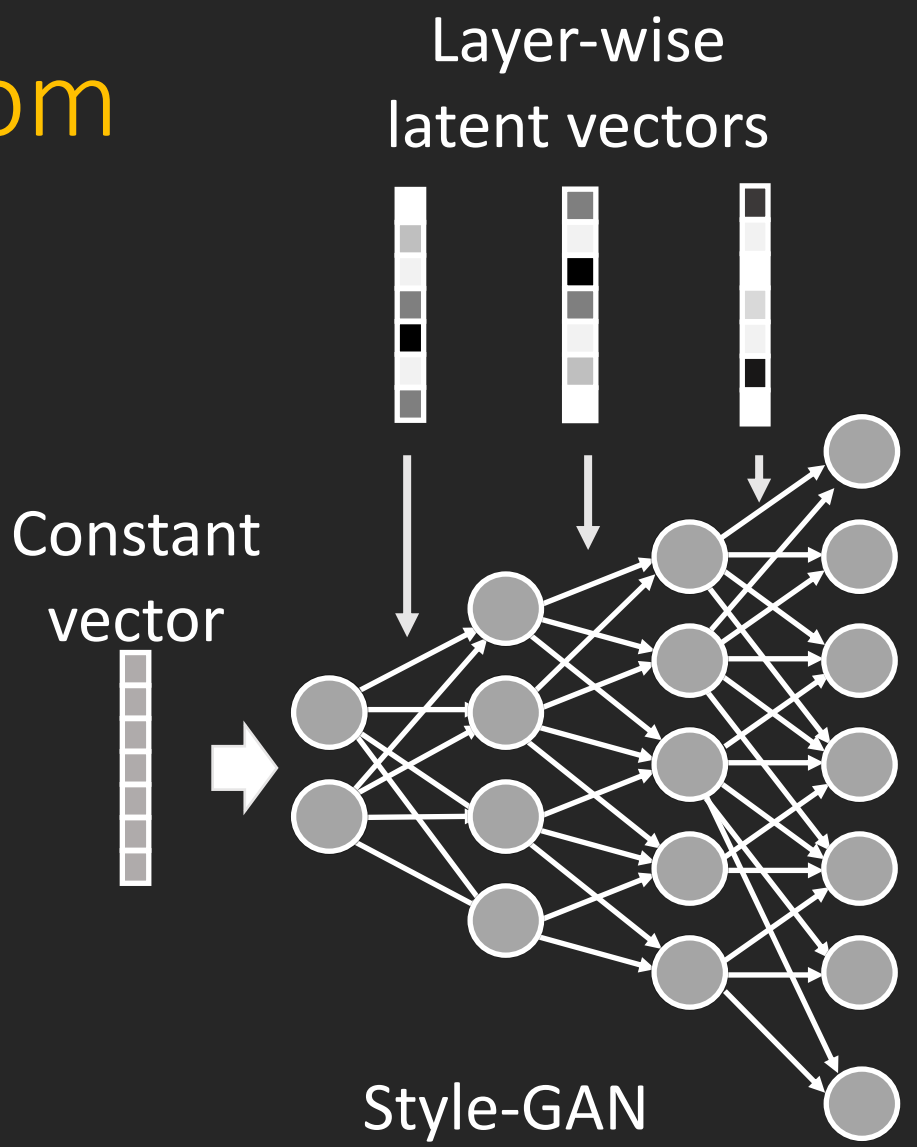
DC-GAN, PG-GAN



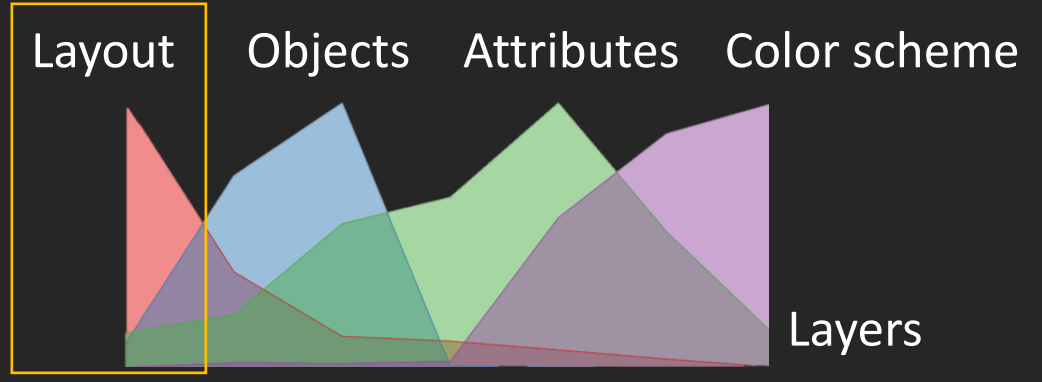
StyleGAN, StyleGANv2 [Karras et al]

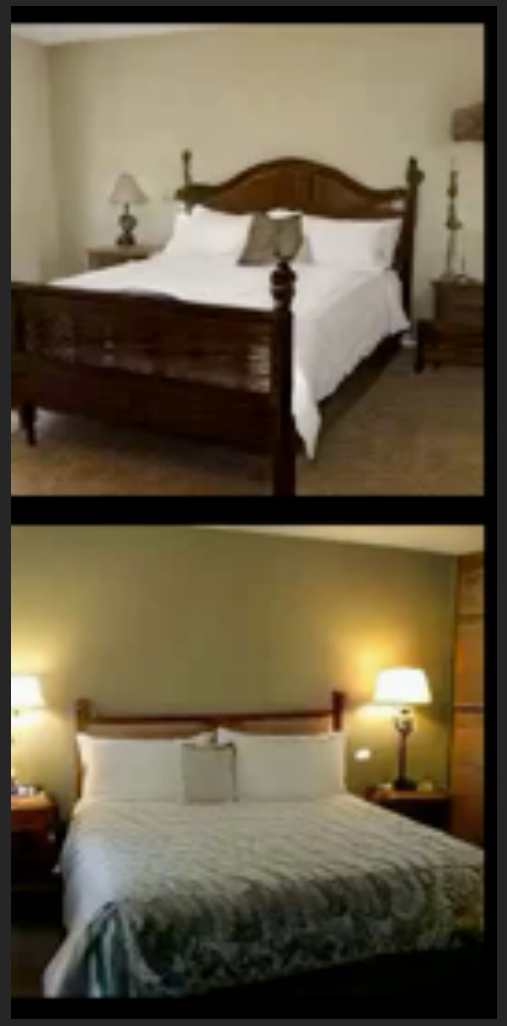
Semantic hierarchy emerges from synthesizing scenes

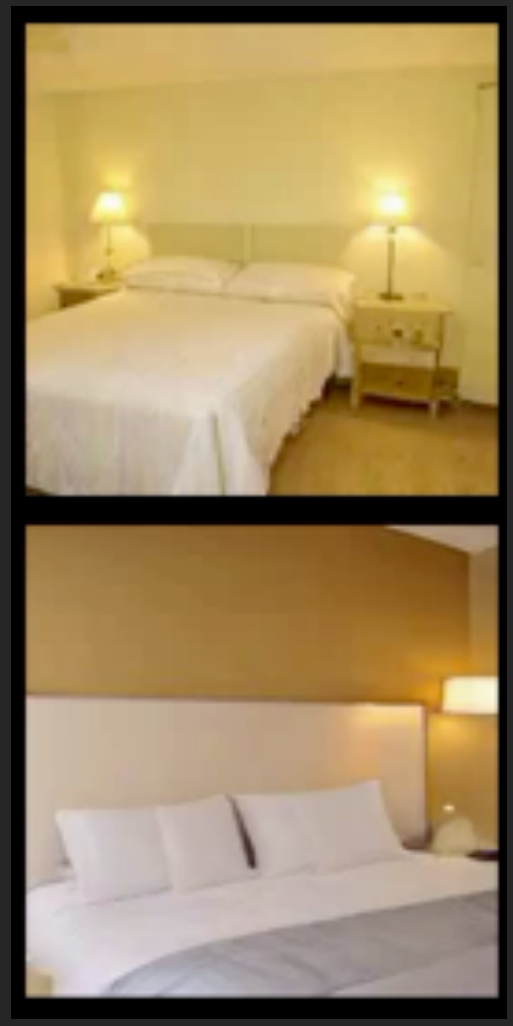




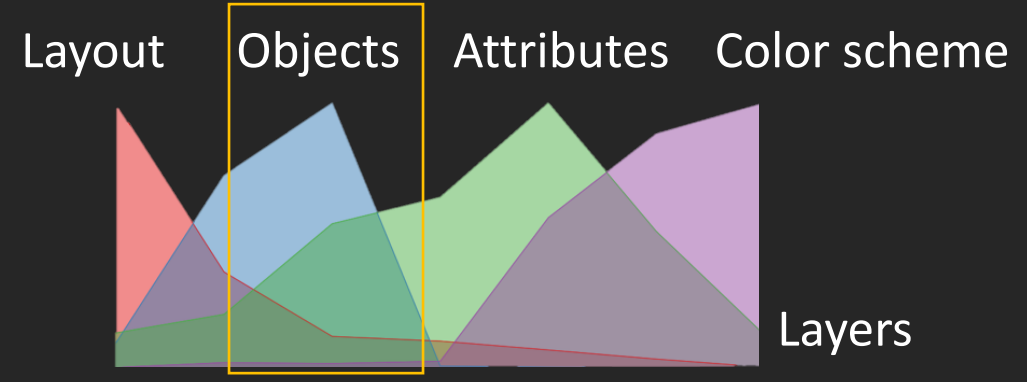
Edit layout at layers 0-1







Edit category (Bedroom to Dining Room) at layers 3-6

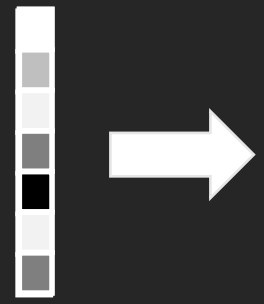


Live Demo: Code of HiGAN

- https://github.com/genforce/higan
- Colab live demo: https://colab.research.google.com/github/genforce/higan/blob/master/docs/HiGAN_Bedroom.ipynb

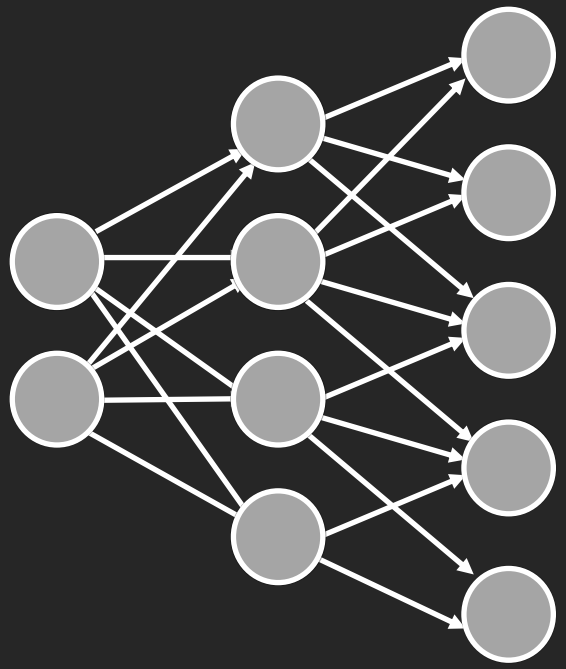
Deep Generative Model for Faces

Random vector



512 dimensions

Generator



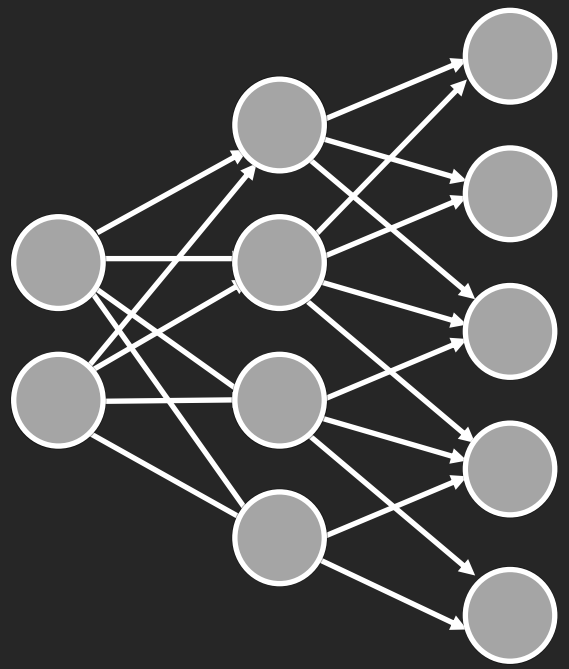
Randomly generated image

How to Customize the Output?

Latent space

512 dimensions

Generator



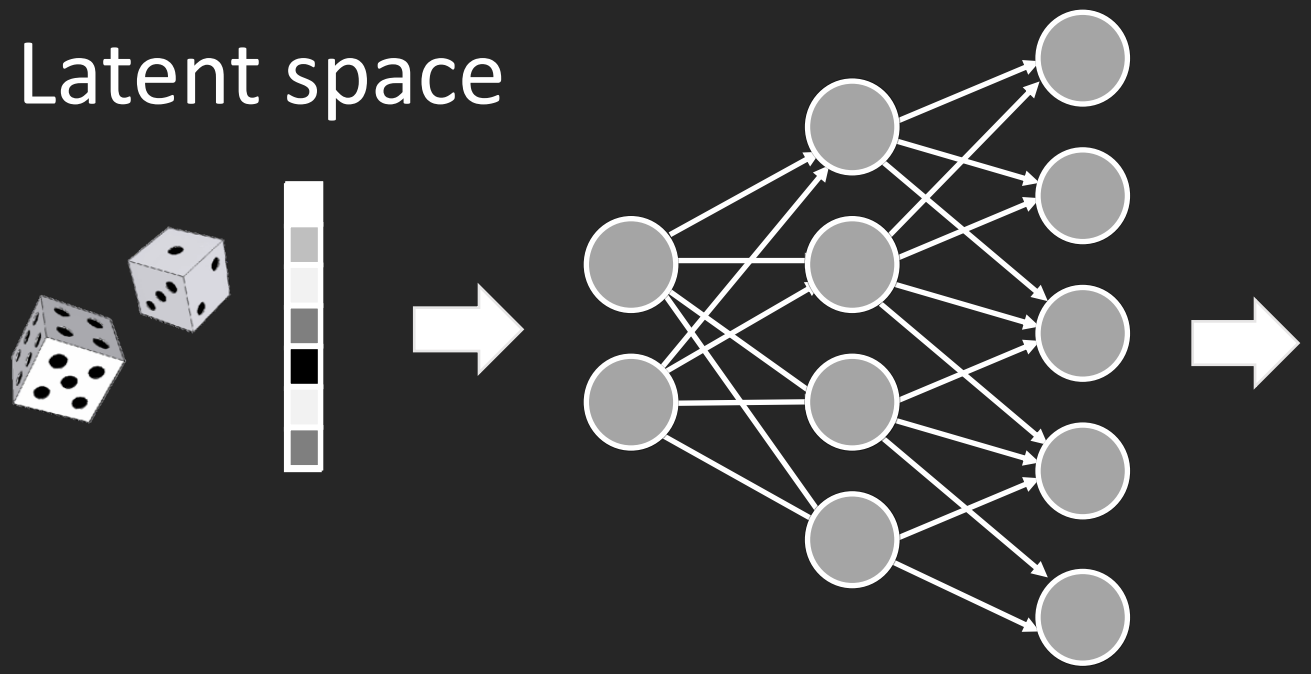
Output

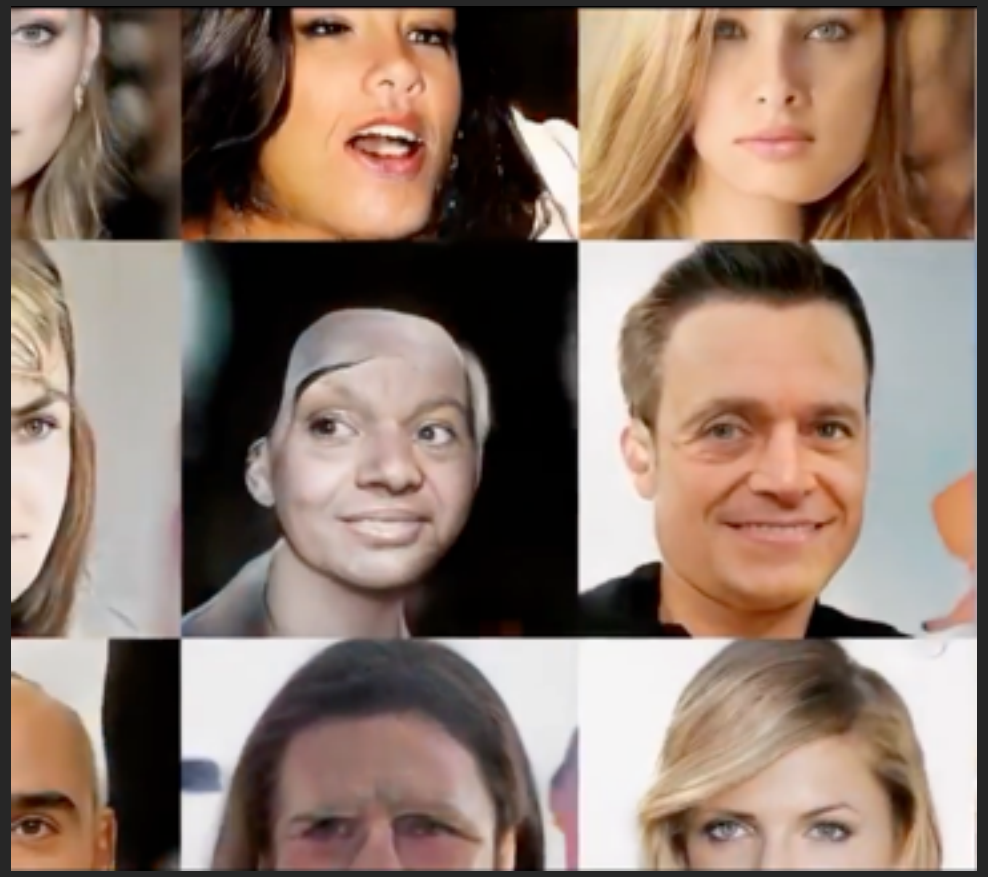
Different angle

Different age

Random Walk in the Latent Space

Latent space

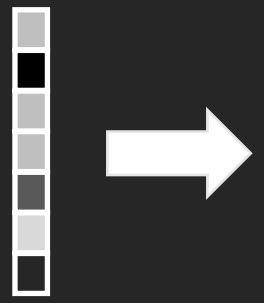


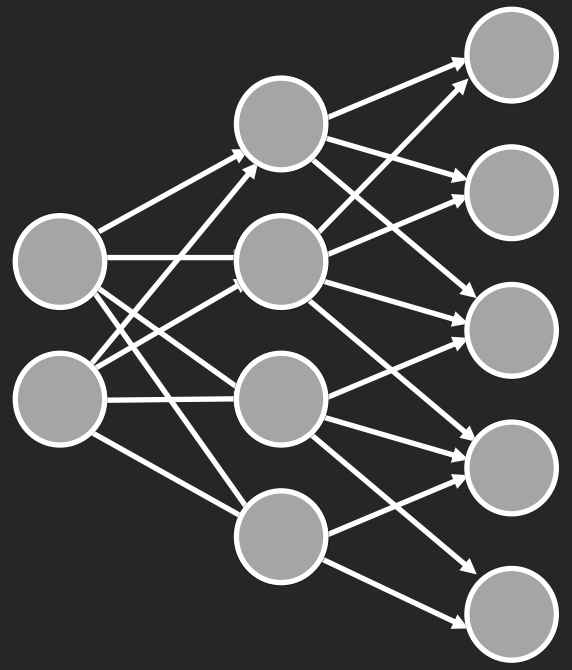


Causal Relations in the Latent Space

Generator

Latent space

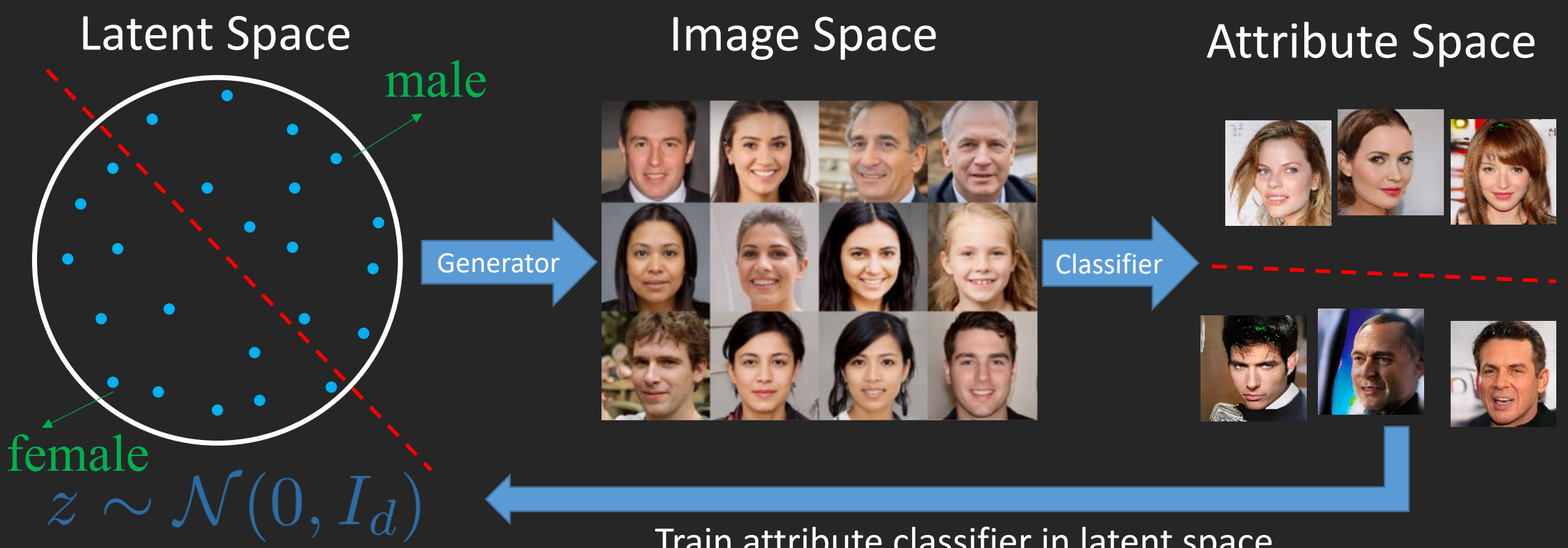




Predicted Attributes:

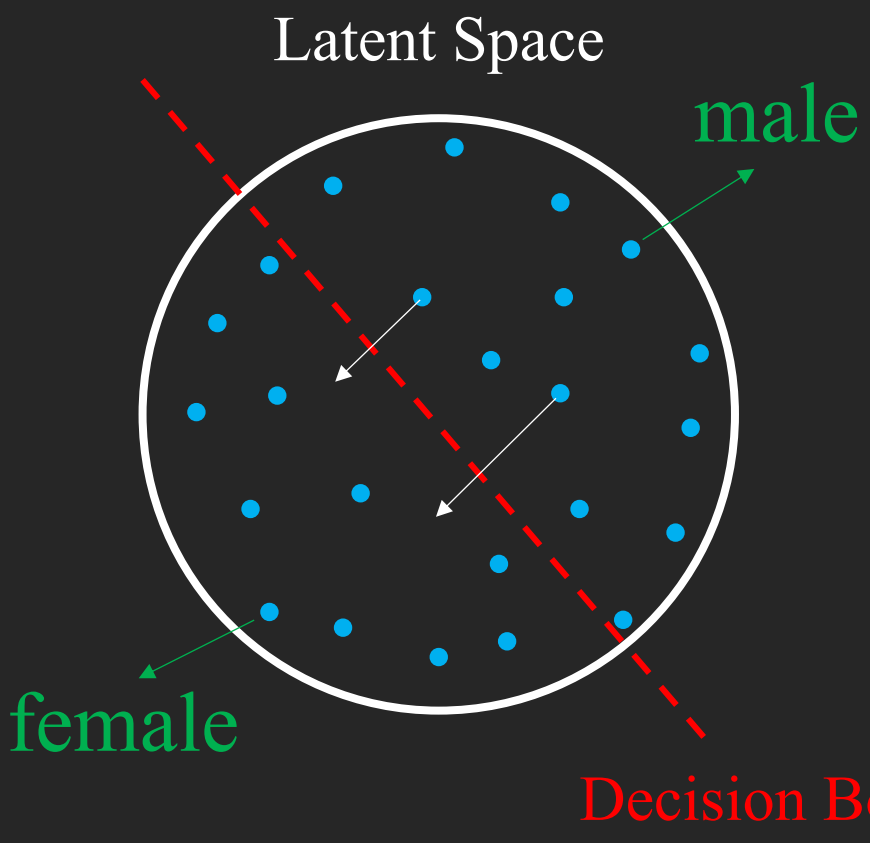
Male, middle age, front face, no glasses

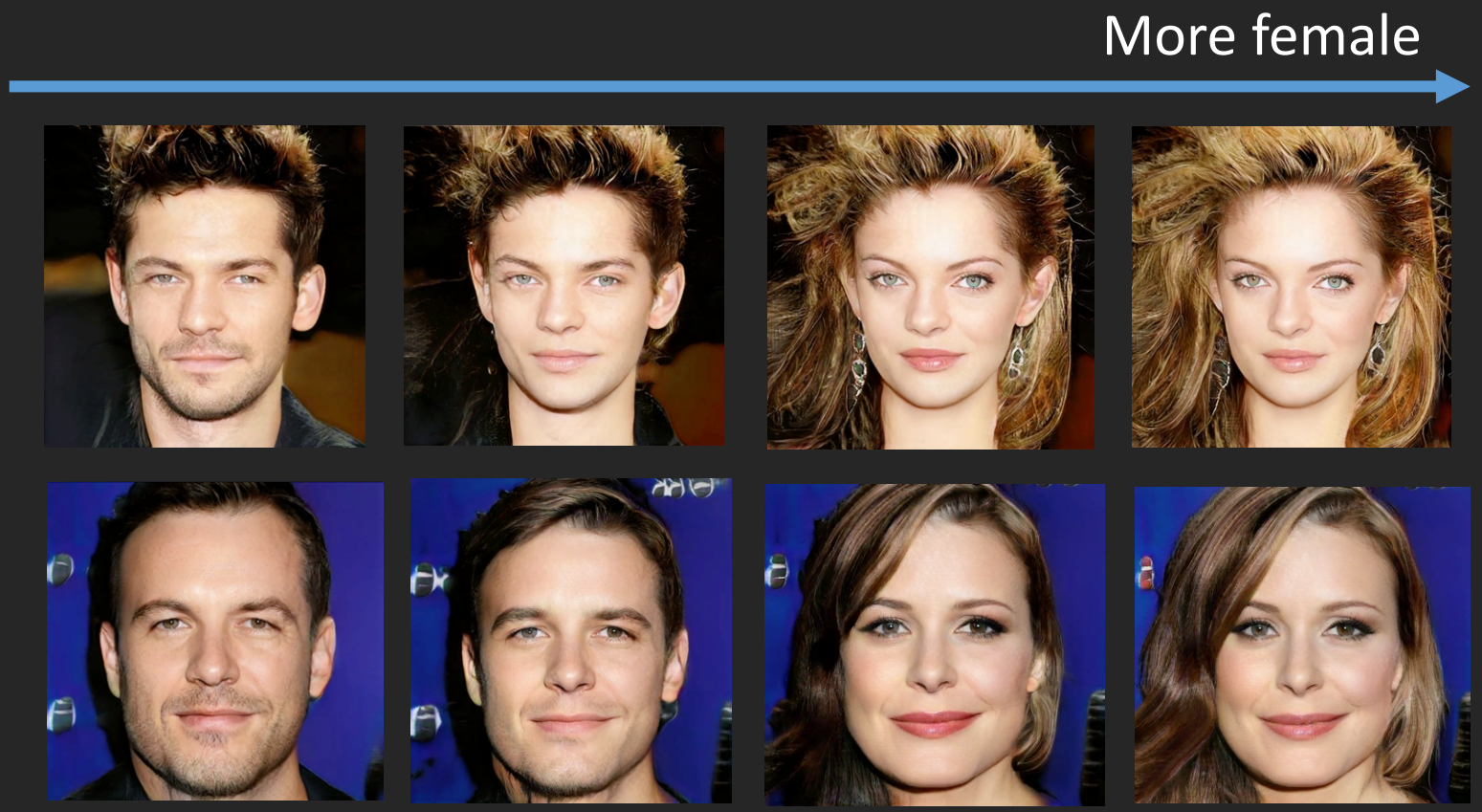
InterFaceGAN: Interpreting Semantics in Face GANs



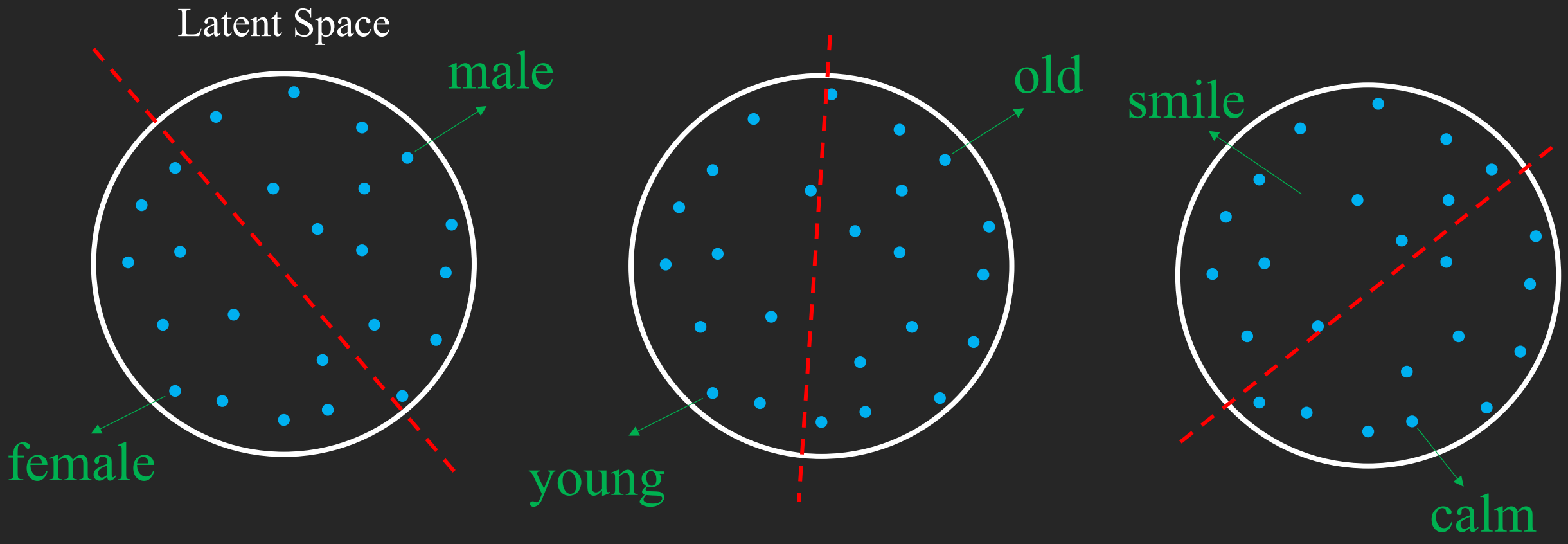
Yujun Shen, Jinjin Gu, Xiaoou Tang, Bolei Zhou. CVPR2020

Varying the Latent Code through Boundary





Various Attribute Boundaries to Divide the Latent Space



Make me cooler

Make me younger

Make me front faced

Make me more man

Correcting the Mistakes Made by GAN


Demo Video for
Semantic Face Editing
with InterFaceGAN

Code for InterfaceGAN

- https://github.com/genforce/interfacegan
- Colab live demo: https://colab.research.google.com/github/genforce/interfacegan/blob/master/docs/InterFaceGAN.ipynb

Outline

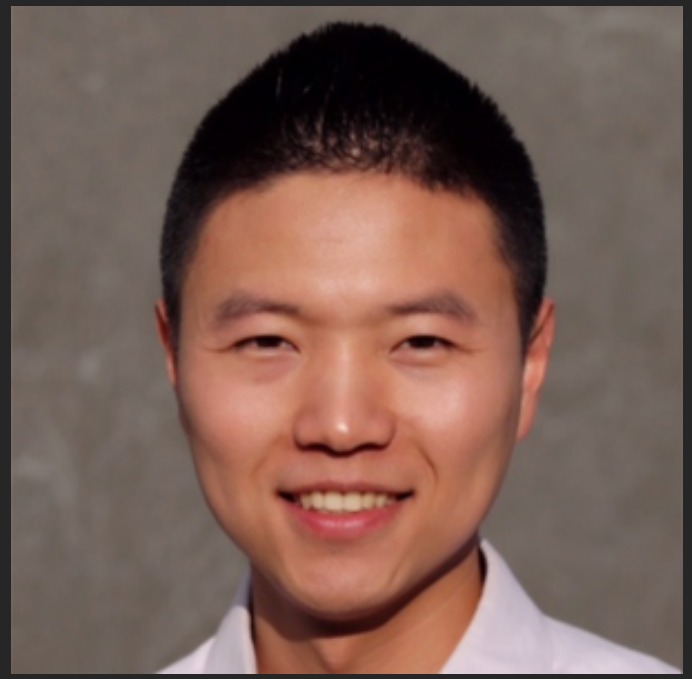
- Introduction: from recognition to recreation
- Overview of the generative models
- Understanding generative models
- Inverting images for real image editings

How to edit my own face?

Make me cooler

Make me younger

GAN-Synthesized Image

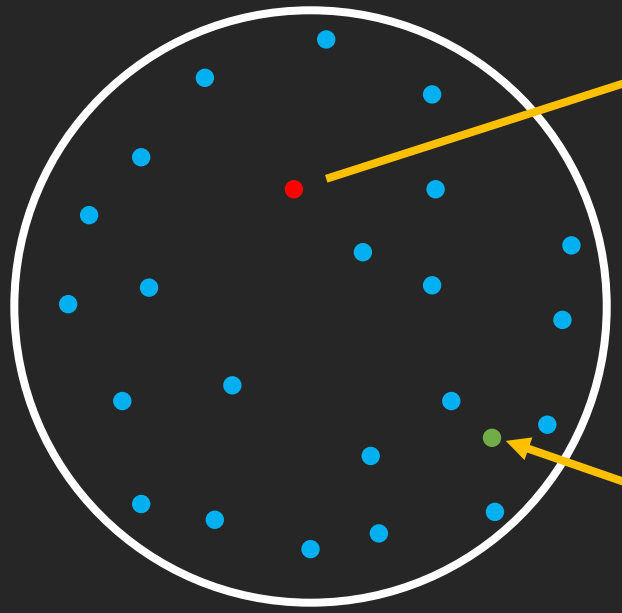


My Profile photo

GAN Inversion: Inverting Real Face to Latent Code

$$\mathbf{x} = G(\mathbf{z})$$

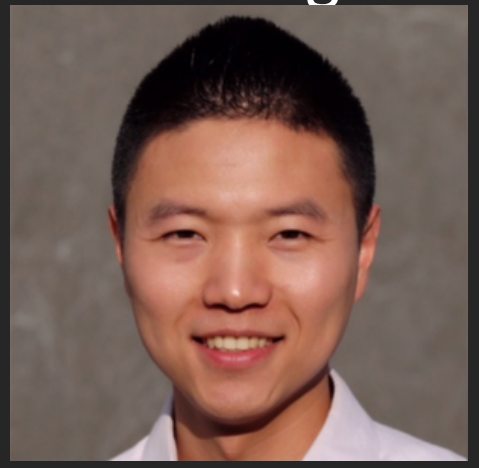
Latent Space **z**



Synthesis

Inversion

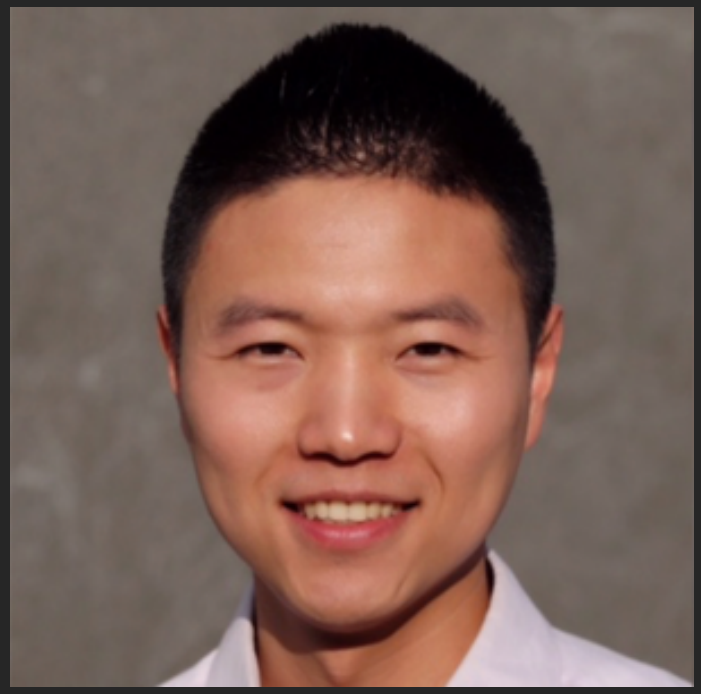
Real Image $\mathbf{x}$



$$\mathbf{z}^* = \arg\min_{\mathbf{z}} ||G(\mathbf{z}) - \mathbf{x}||^2$$

GAN Inversion

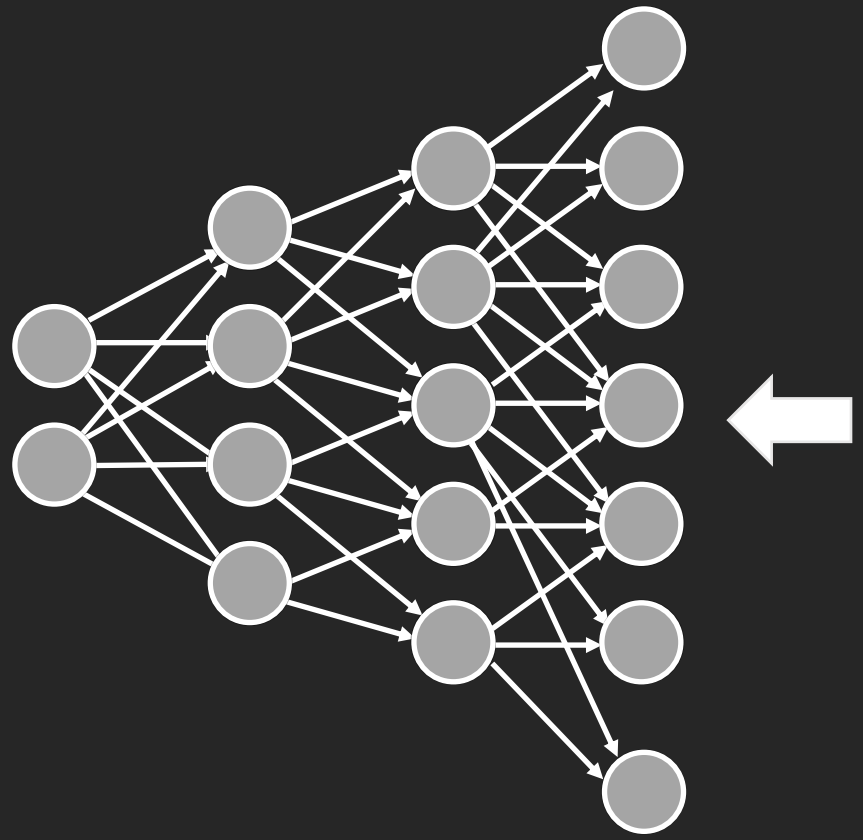
Real face x



Optimized code

z^*

512 dimensions



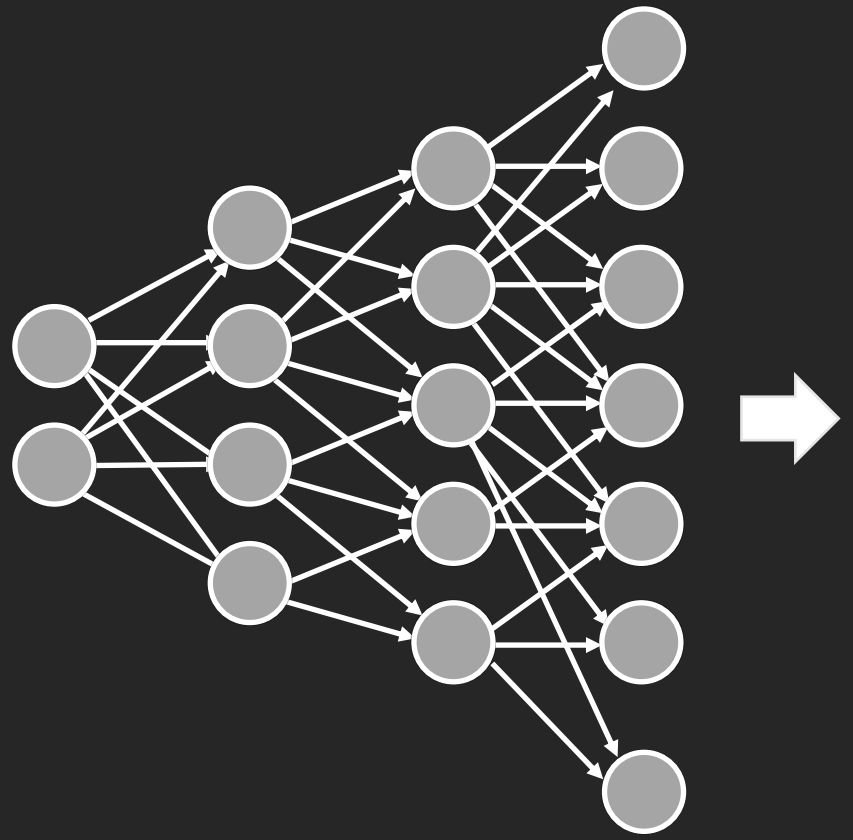
$$\mathbf{z}^* = \arg\min_{\mathbf{z}} ||G(\mathbf{z}) - \mathbf{x}||^2$$

GAN Inversion

Optimized code

z^*

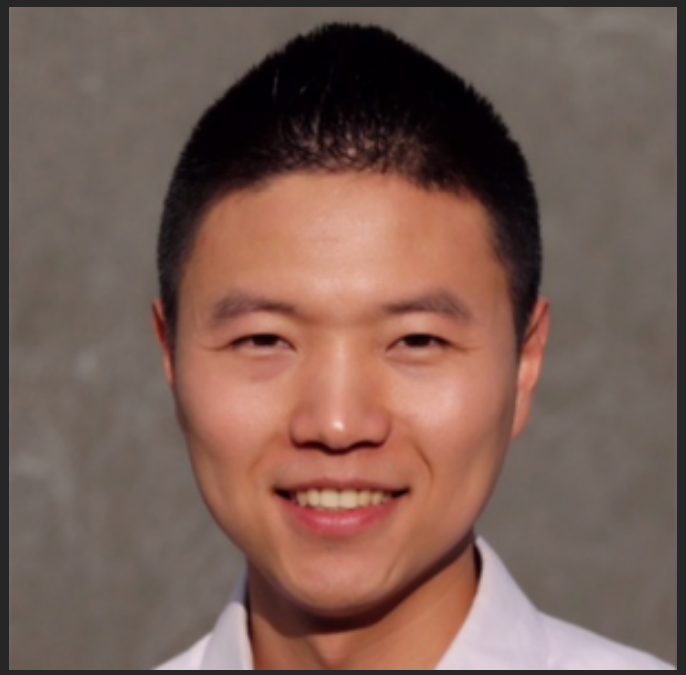
512 dimensions



Reconstructed face

GAN inversion is challenging!

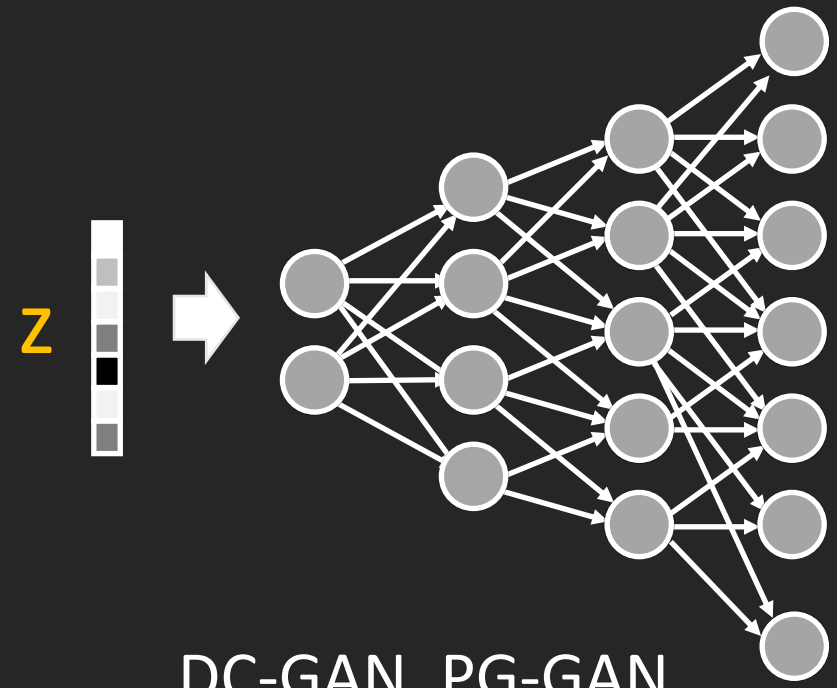
$$\mathbf{z}^* = \arg\min_{\mathbf{z}} ||G(\mathbf{z}) - \mathbf{x}||^2$$



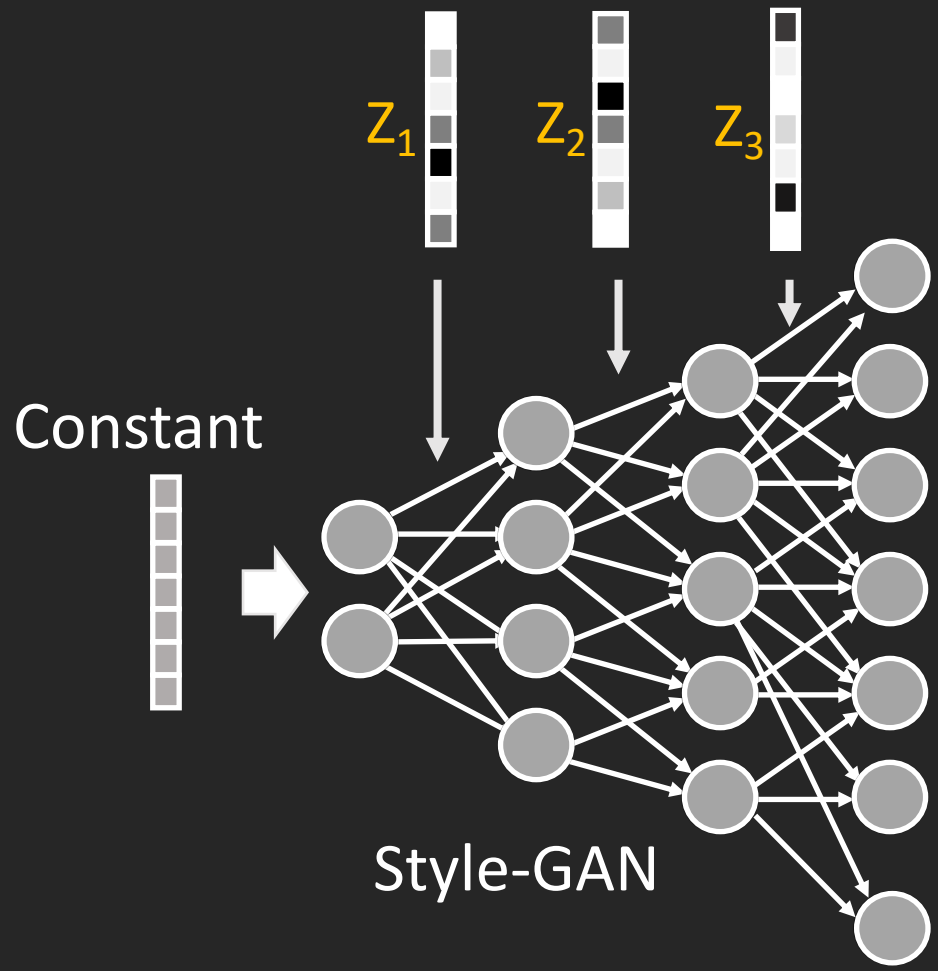
Inversion

Different initialization

Extended latent codes



DC-GAN, PG-GAN



Style-GAN

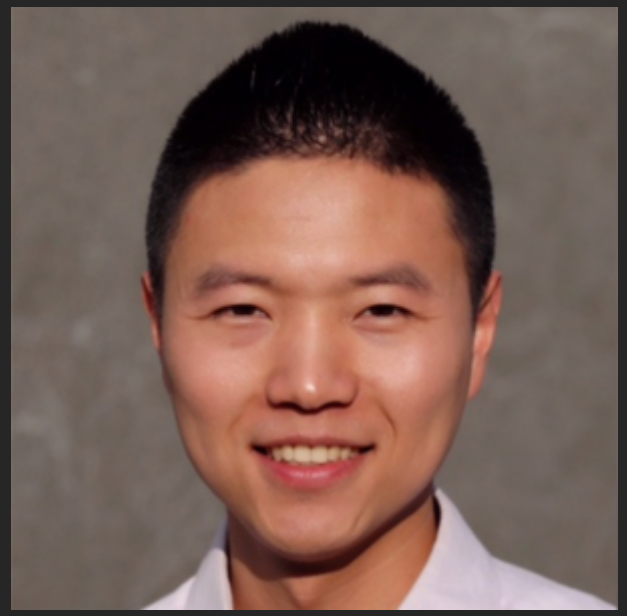
$$\mathbf{z}^* = \arg\min_{\mathbf{z}} ||G(\mathbf{z}) - \mathbf{x}||^2$$

$$\mathbf{z}_1^*, \mathbf{z}_2^*, \ldots = \arg\min_{\mathbf{z}_1, \mathbf{z}_2, \ldots} ||G(\mathbf{z}_1, \mathbf{z}_2, \ldots) - \mathbf{x}||^2$$

Abdal, et al. Image2StyleGAN: How to Embed Images Into the StyleGAN. ICCV'19

Image2StyleGAN: it works to some degree

Target

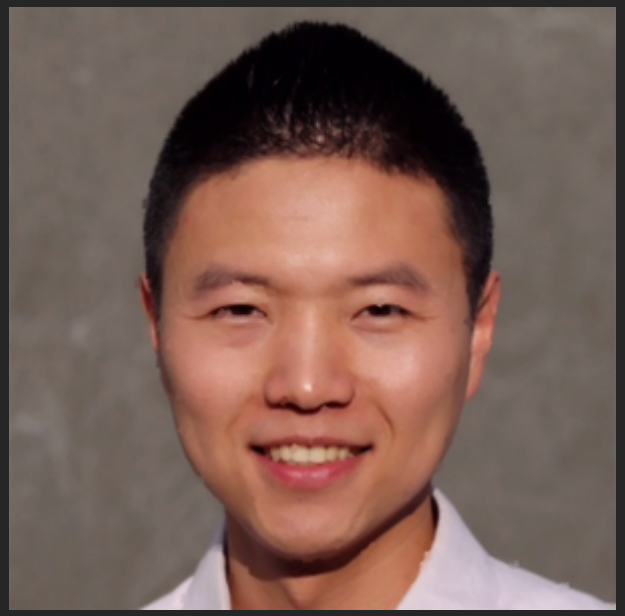


Inversion from single code

512 dimensions

Inversion from Image2StyleGAN

14x512 dimensions



But it seems overfitting the target image

Target

Image2StyleGAN

Target

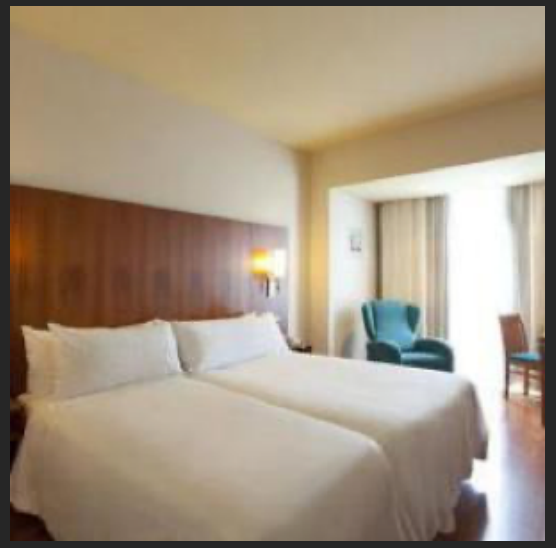


Image2StyleGAN

* Generator is trained on human faces only

But it seems overfitting the target image

Inverted codes don’t very well support the manipulation

Target

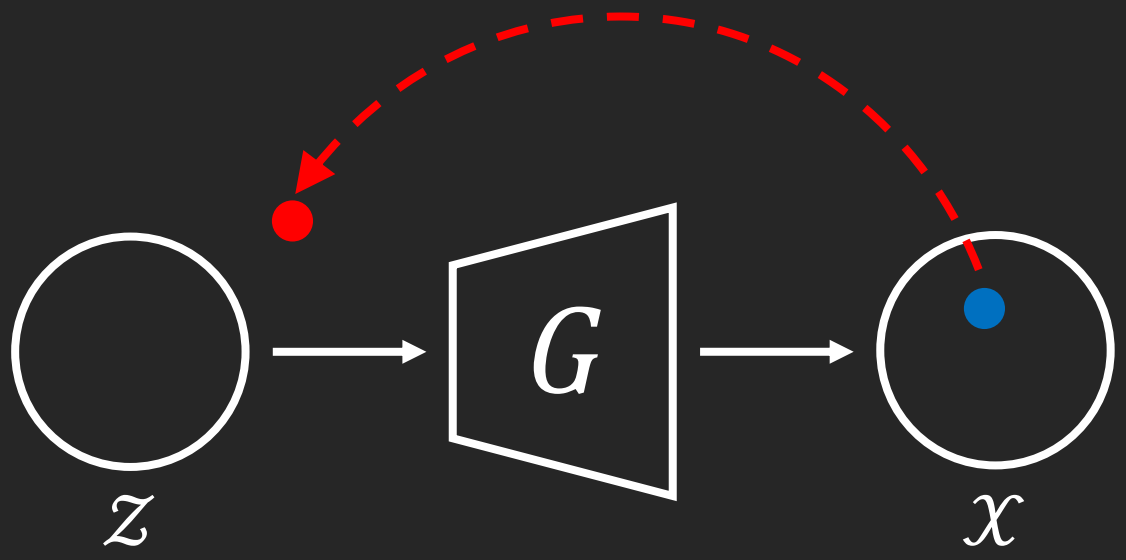
Image2StyleGAN

Add smile

Add glasses

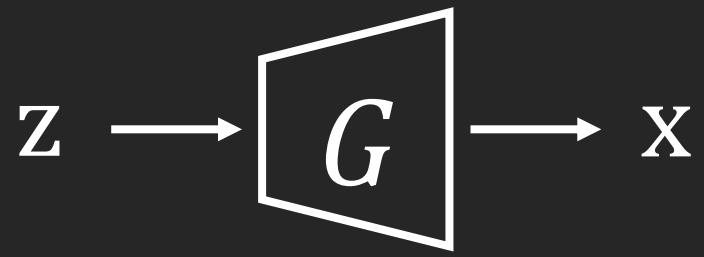
Issue: resulting code might be out of the original latent domain

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} ||G(\mathbf{z}) - \mathbf{x}||^2$$

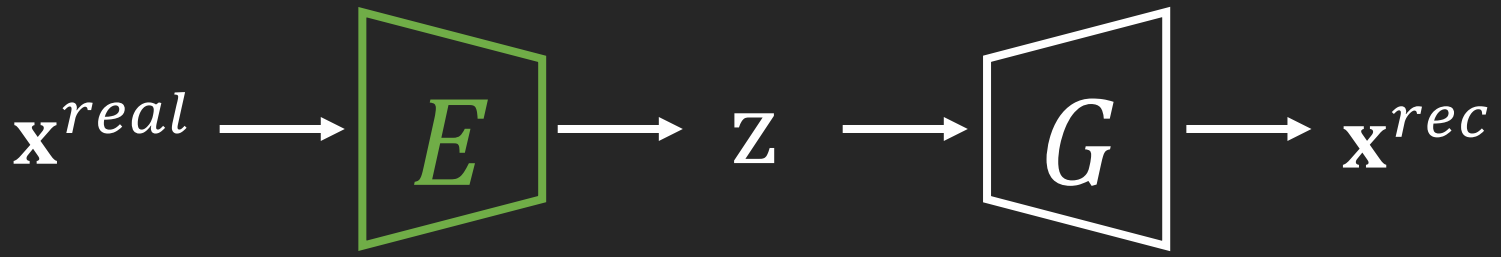


Out-of-Domain Inversion

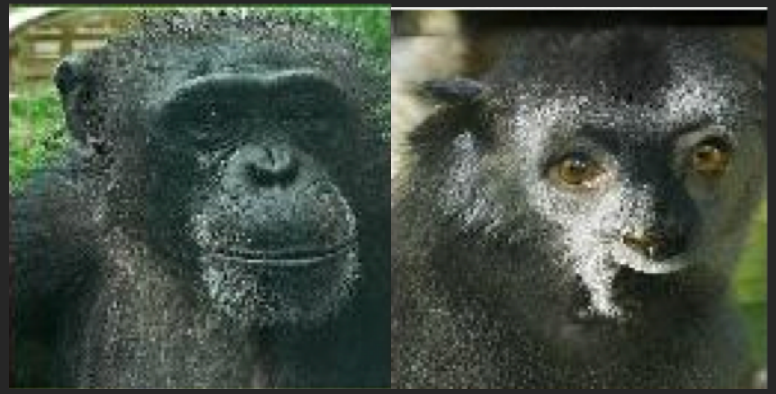
GAN lacks the inference ability:



Many recent works on adding encoder to GAN generator:



BigBiGAN (NeurIPS'19)



Input Reconstruction

Adversarial Latent Autoencoders (CVPR'20)

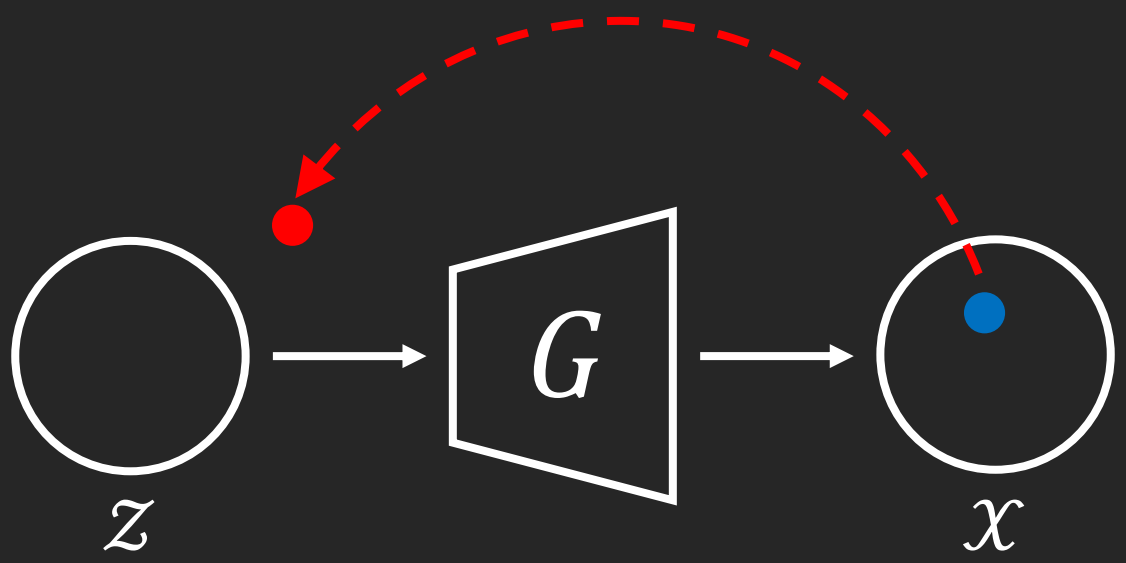
Input Reconstruction

In-Domain Inversion (ECCV'20)

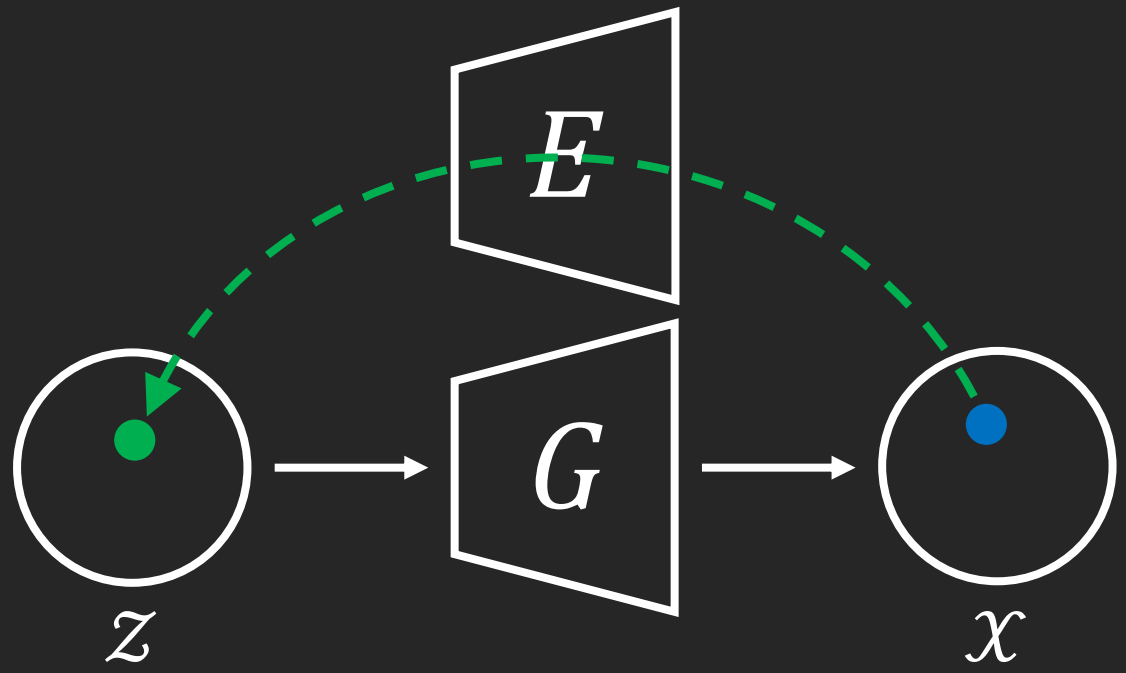
Input Reconstruction

Encoder-constrained Optimization

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} ||G(\mathbf{z}) - \mathbf{x}||^2$$



Unconstrained Inversion



In-Domain Inversion

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} ||G(\mathbf{z}) - \mathbf{x}||^2 + ||\mathbf{z} - E(G(\mathbf{z}))||^2$$

Jiapeng Zhu, Yujun Shen, Deli Zhao, Bolei Zhou. In-Domain GAN Inversion for Real Image Editing. ECCV 2020

Comparison with Image2StyleGAN

Image2StyleGAN





In-Domain Inversion





Target | Reconstruction | Decrease age | Add smile | Add glasses

Demo of image manipulation

Demo of image interpolation

Semantic Diffusion

Background

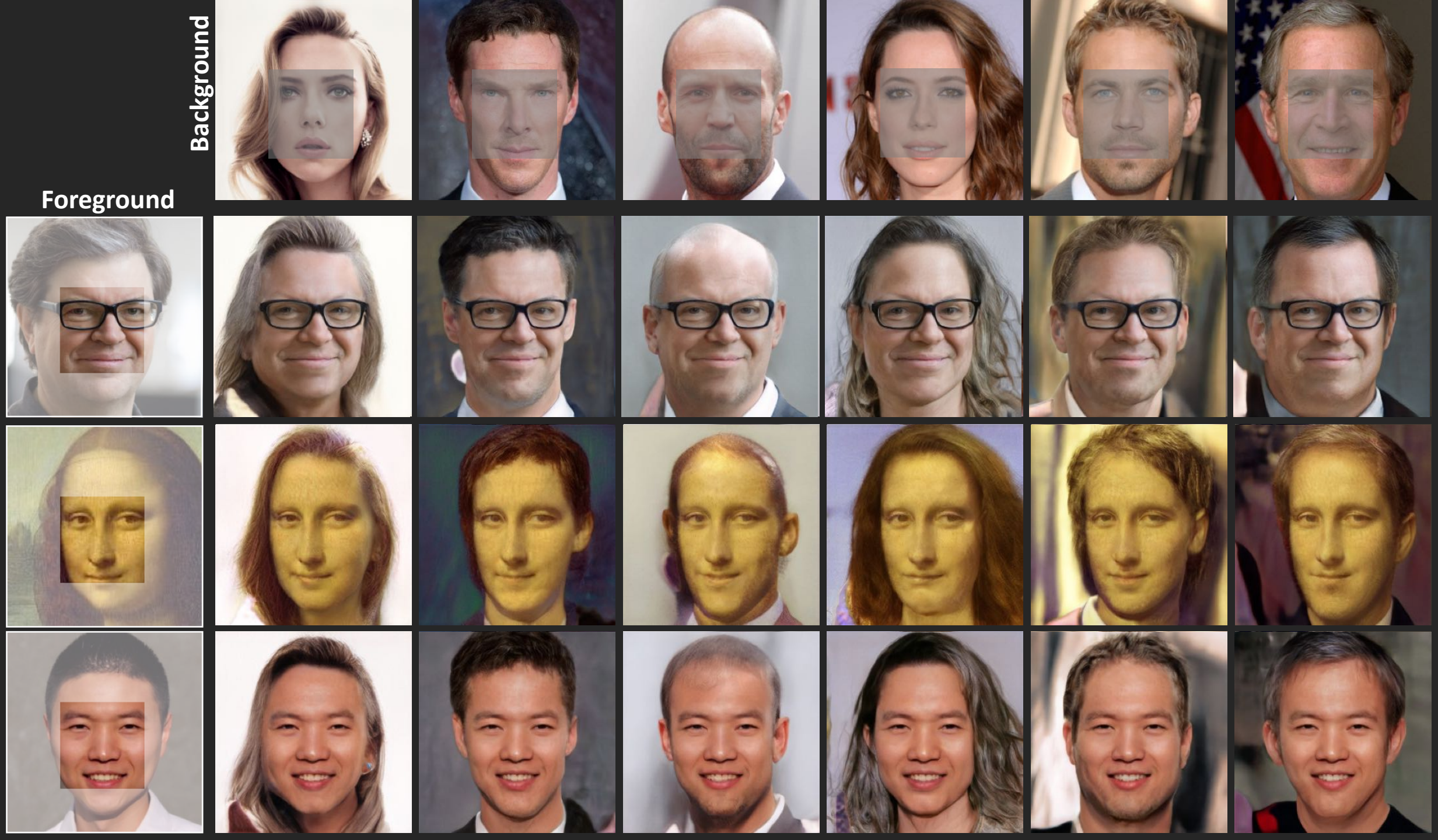
Foreground

Copy and paste

$$\mathbf{z}^* = \text{argmax}_{\mathbf{z}} ||\mathbf{m} \bullet G(\mathbf{z}) - \mathbf{m} \bullet \mathbf{x}||^2 + ||\mathbf{z} - E(G(\mathbf{z}))||^2$$

Background

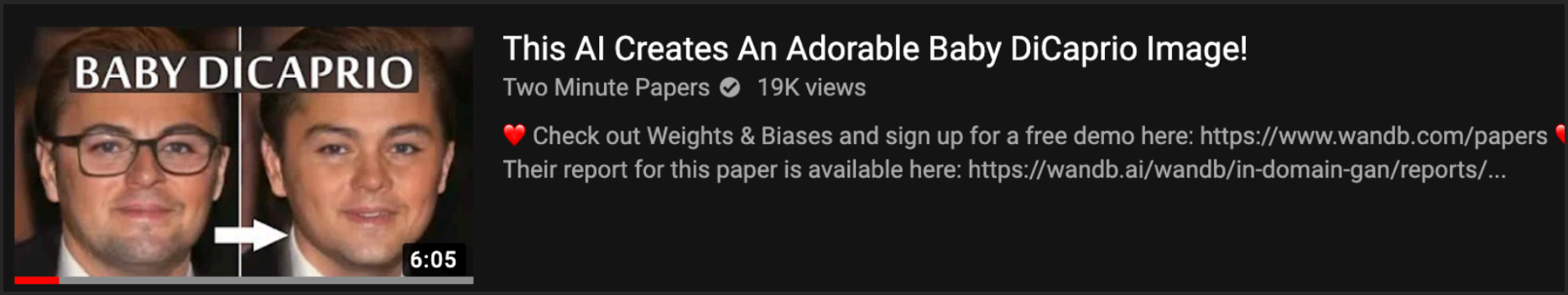
Foreground



Live Demo: Try your own image!

- Colab: https://colab.research.google.com/github/genforce/idinvert_pytorch/blob/master/docs/Idinvert.ipynb

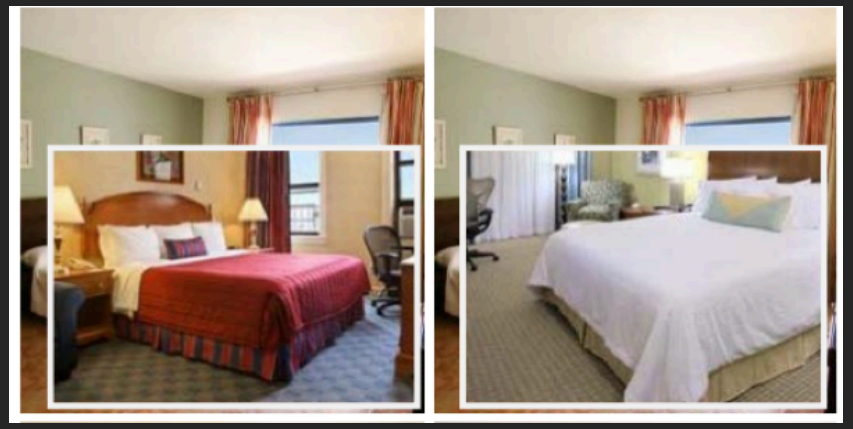
Two Minute Papers Channel's report



https://www.youtube.com/watch?v=2qMw8sOsNg0

Scene Harmonization

Copy and paste a new bed

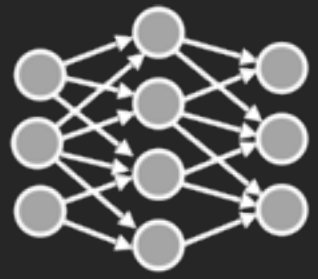


Masked optimization

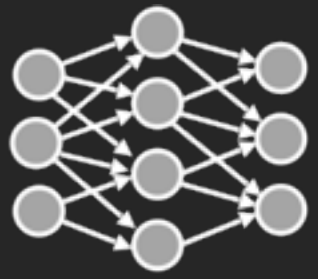
Copy and paste a new window

Masked optimization

Unsupervised Discovery of Steerable Dimensions



Generative model for cartoons



Generative model for cars

Unsupervised Discovery of Steerable Dimensions

Affine transform in the first layer of generator:

$$G_1(z) \triangleq y = Az + b$$

Linear manipulation:

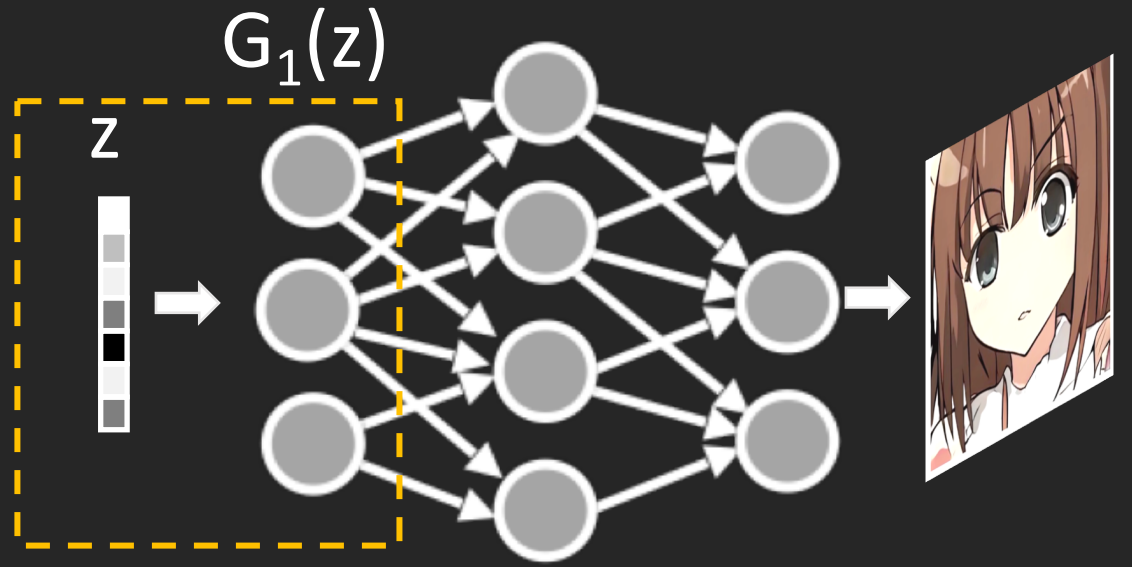
$$\Delta y = G_1(z + n) - G_1(z) = An$$

Unsupervised learning objective:
maximize the feature variations:

$$n^* = \text{argmax}_{\{n \in R^d:\ n^T n = 1\}} ||An||_2^2$$

Solving the eigenvectors:

$$A^T A n = \lambda n$$



Shen and Zhou. Closed-Form Factorization of Latent Semantics in GANs. CVPR'21

https://genforce.github.io/sefa/

Steerable Dimensions in Generative Model of Cats

Head pose

Body pose

Zoom-in and out

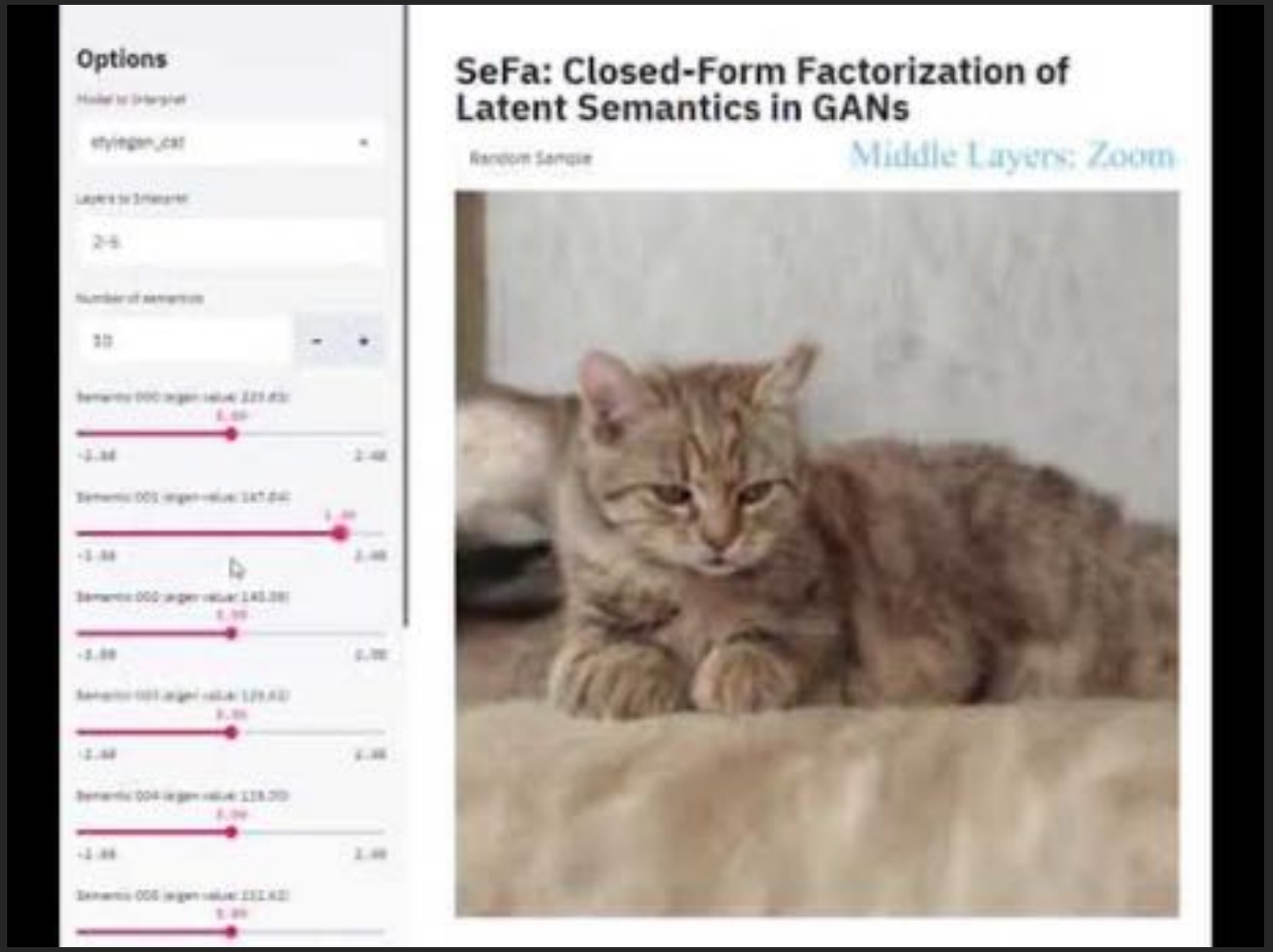
Steerable Dimensions in Generative Model of Animes

Head pose

Eye size

Mouth

demo video

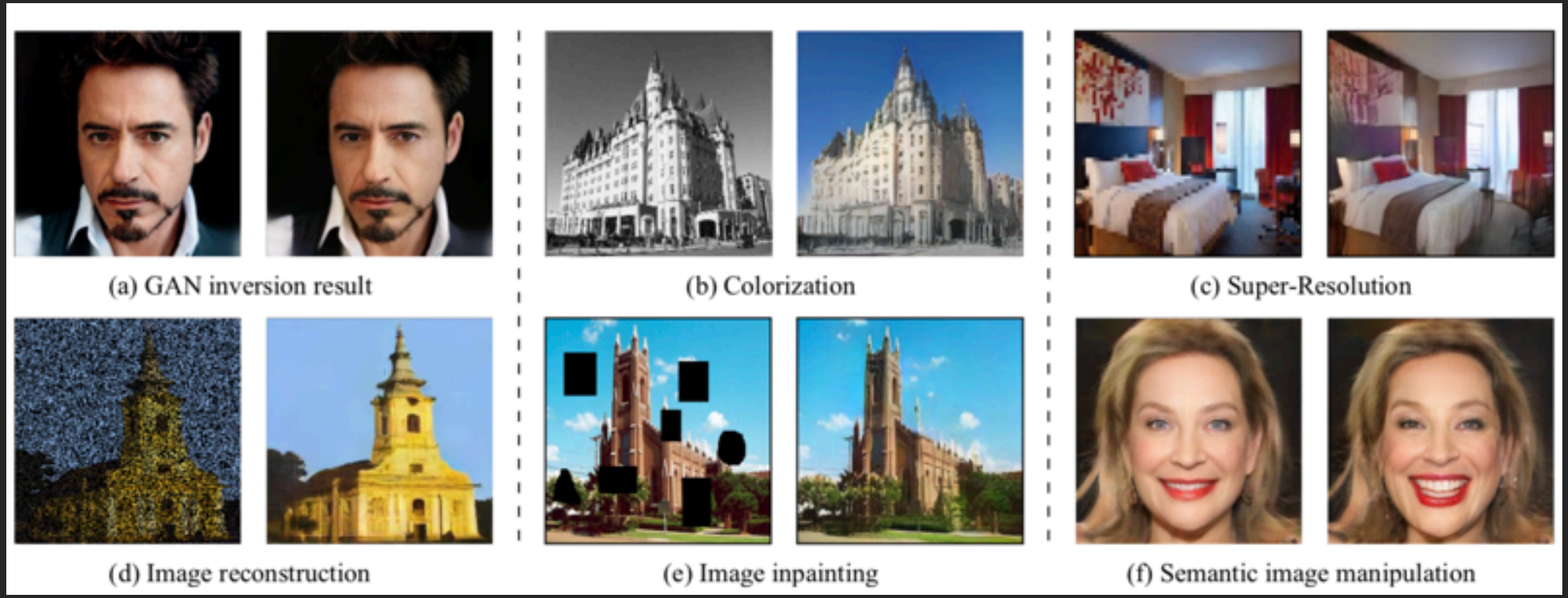


https://genforce.github.io/sefa/

Live Demo: SeFa

- Interface: 192.168.72.172:1234 (to be open by TA)
- Google Colab: https://colab.research.google.com/github/genforce/sefa/blob/master/docs/SeFa.ipynb
- Open-source code: https://github.com/genforce/sefa

Deep Generative Prior for Image Processing



Jinjin Gu, Yujun Shen, Bolei Zhou. CVPR'20

Deep Generative Prior for Image Processing

Inversion loss

$$\{z_n^*\}_{n=1}^N, \{\alpha_n^*\}_{n=1}^N = \underset{\{z_n\}_{n=1}^N, \{\alpha_n\}_{n=1}^N}{\arg\min} L(x^{inv}, x)$$

Super-resolution loss: $L_{SR} = L(down(x^{inv}), I_{LR})$

Colorization loss: $L_{color} = L(gray(x^{inv}), I_{gray})$

Inpainting loss: $L_{inp} = L(x^{inv} \circ \mathbf{m}, I_{ori} \circ \mathbf{m})$

Jinjin Gu, Yujun Shen, Bolei Zhou. CVPR'20

Image inpainting

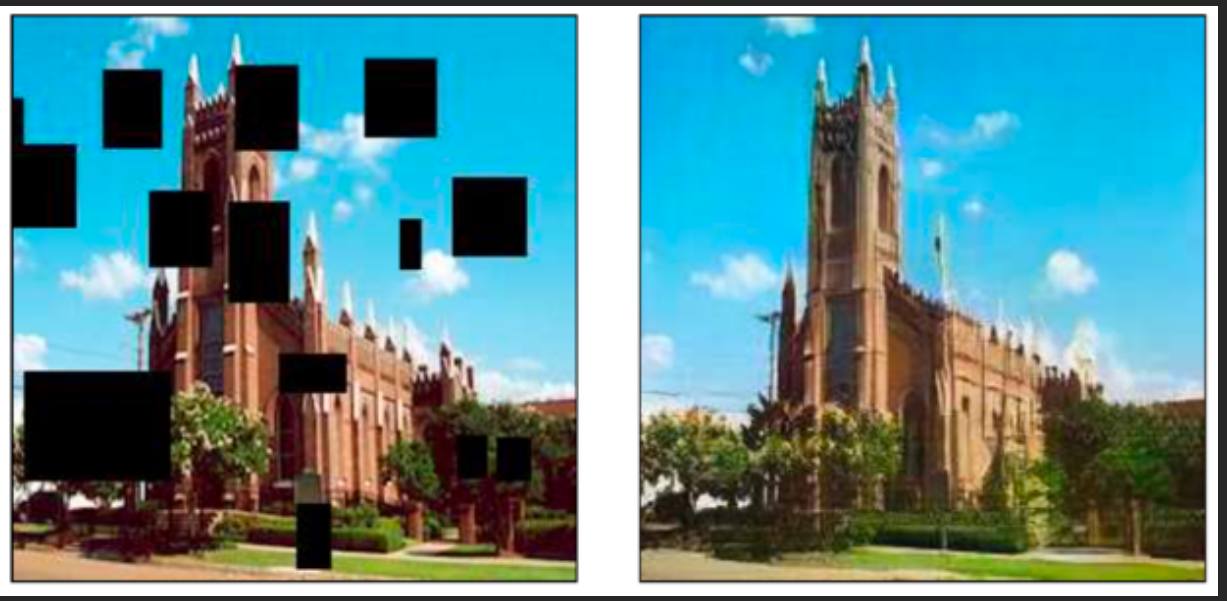
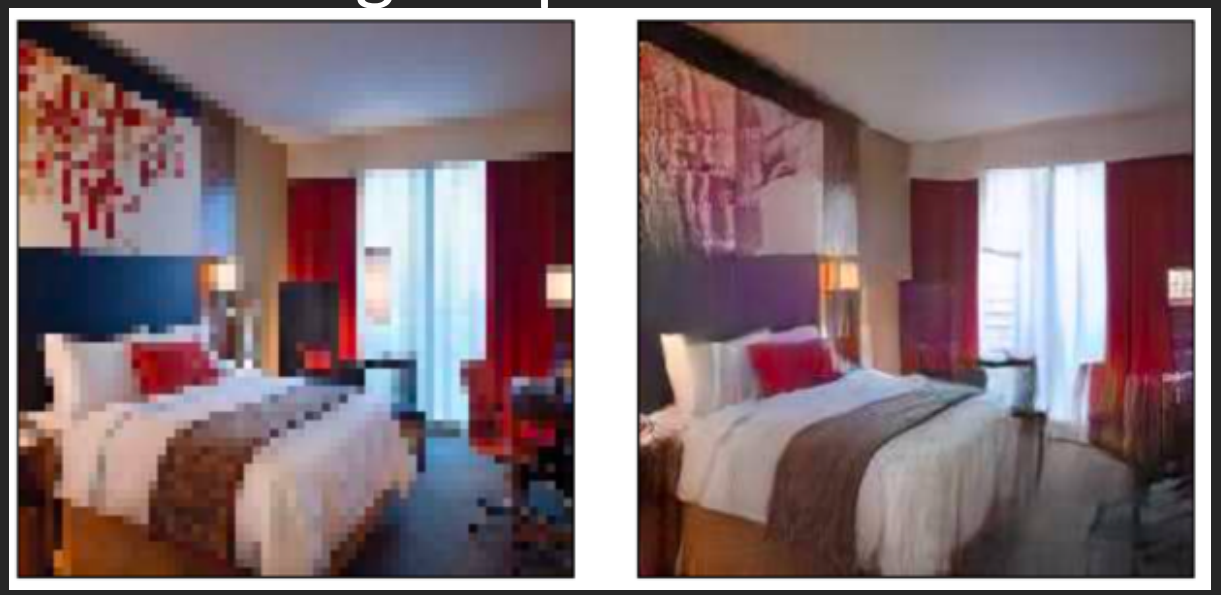


Image colorization

Image denoising

Image super-resolution



A recent survey paper on GAN inversion

- https://arxiv.org/pdf/2101.05278.pdf

GAN Inversion: A Survey

Weihao Xia, Yulun Zhang, Yujiu Yang*, Jing-Hao Xue, Bolei Zhou*, Ming-Hsuan Yang*

Abstract—GAN inversion aims to invert a given image back into the latent space of a pretrained GAN model, for the image to be faithfully reconstructed from the inverted code by the generator. As an emerging technique to bridge the real and fake image domains, GAN inversion plays an essential role in enabling the pretrained GAN models such as StyleGAN and BigGAN to be used for real image editing applications. Meanwhile, GAN inversion also provides insights on the interpretation of GAN's latent space and how the realistic images can be generated. In this paper, we provide an overview of GAN inversion with a focus on its recent algorithms and applications. We cover important techniques of GAN inversion and their applications to image restoration and image manipulation. We further elaborate on some trends and challenges for future directions. A curated list of GAN inversion methods, datasets, and other related information can be found at `github.com/weihaox/awesome-gan-inversion`.

Index Terms—Generative Adversarial Networks, Interpretable Machine Learning, Image Reconstruction, Image Manipulation

Other generative models

- Normalizing flows
- Energy-based models
- Image2Image translation

Image to Image Translation

noise2image

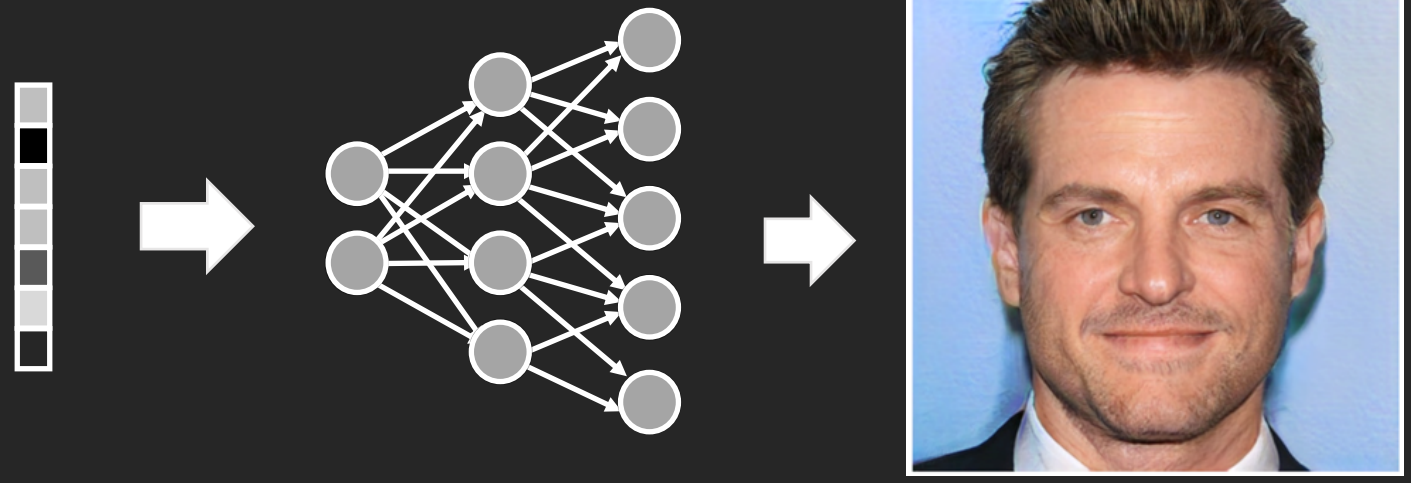


image2image

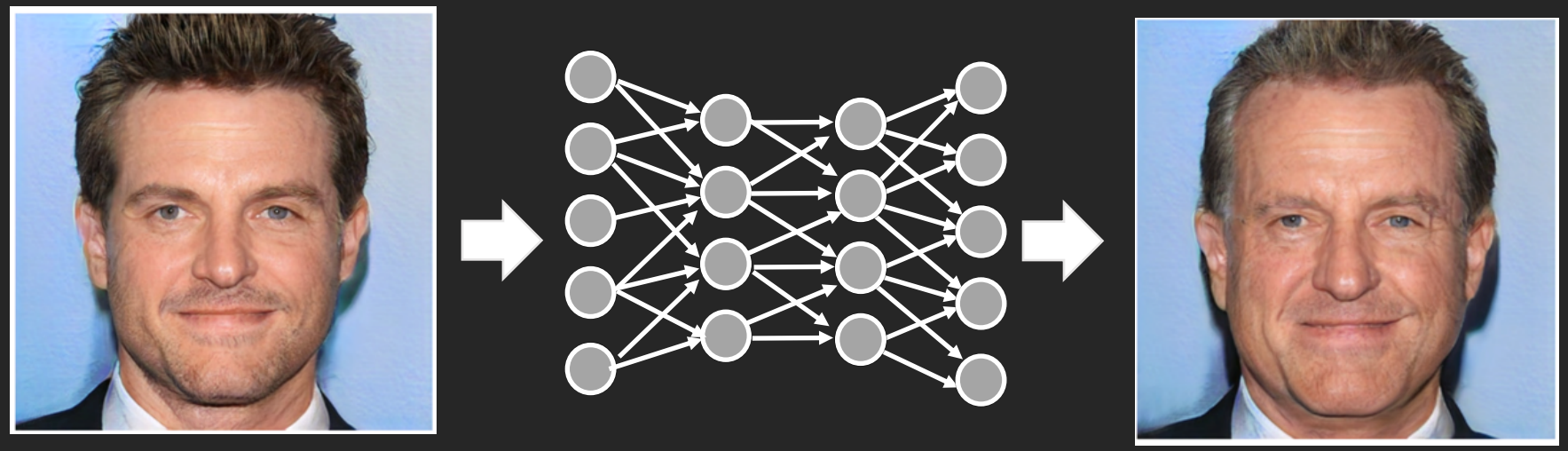
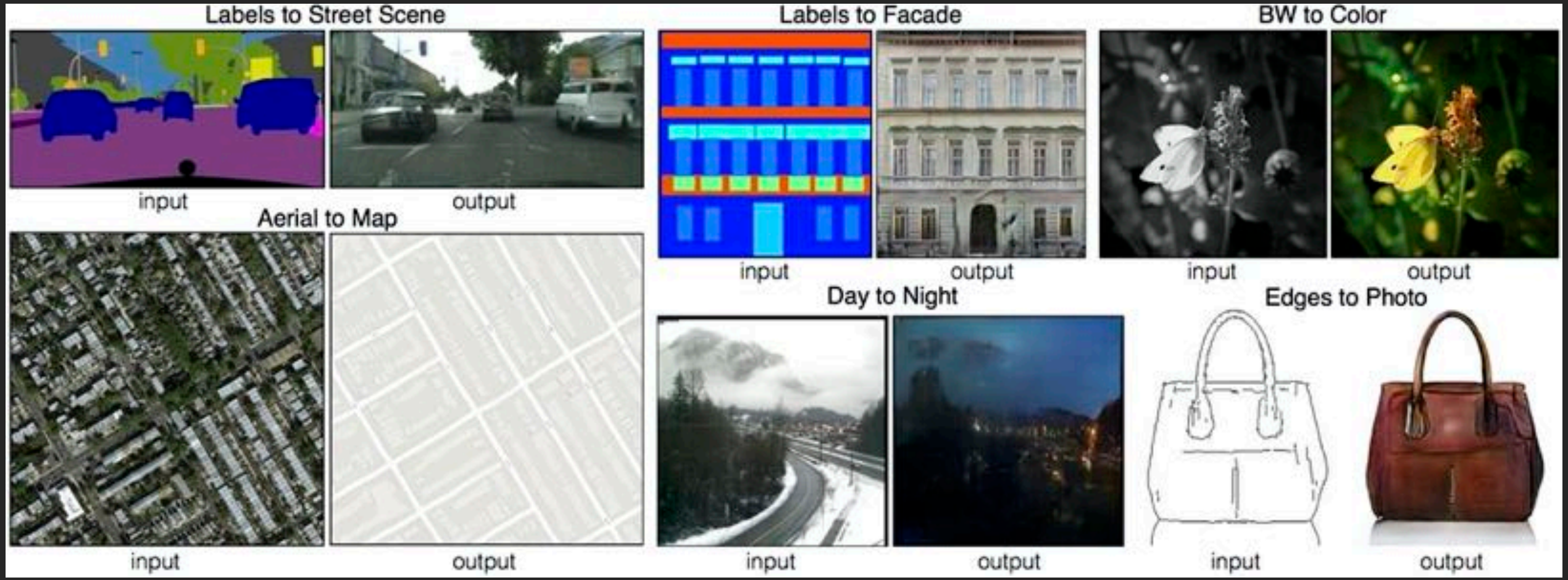
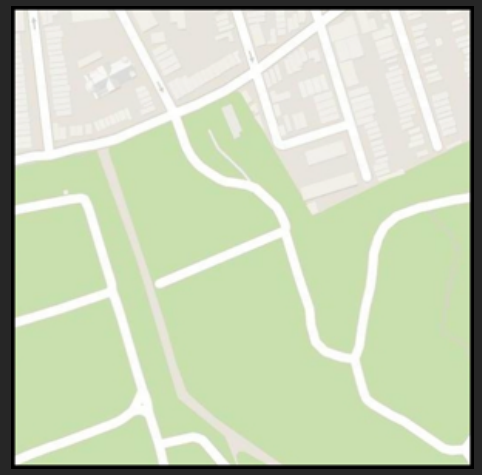
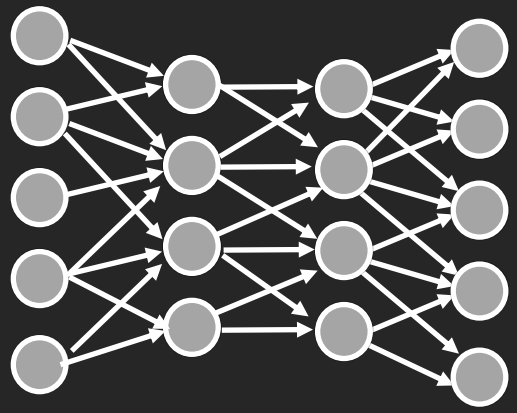


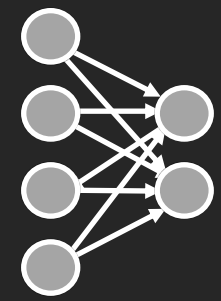
Image to Image Translation



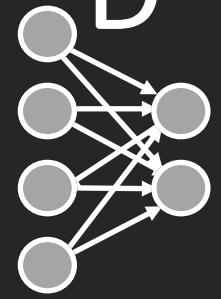
Isola et al. Image-to-Image Translation with Conditional Adversarial Nets. CVPR'17







fake



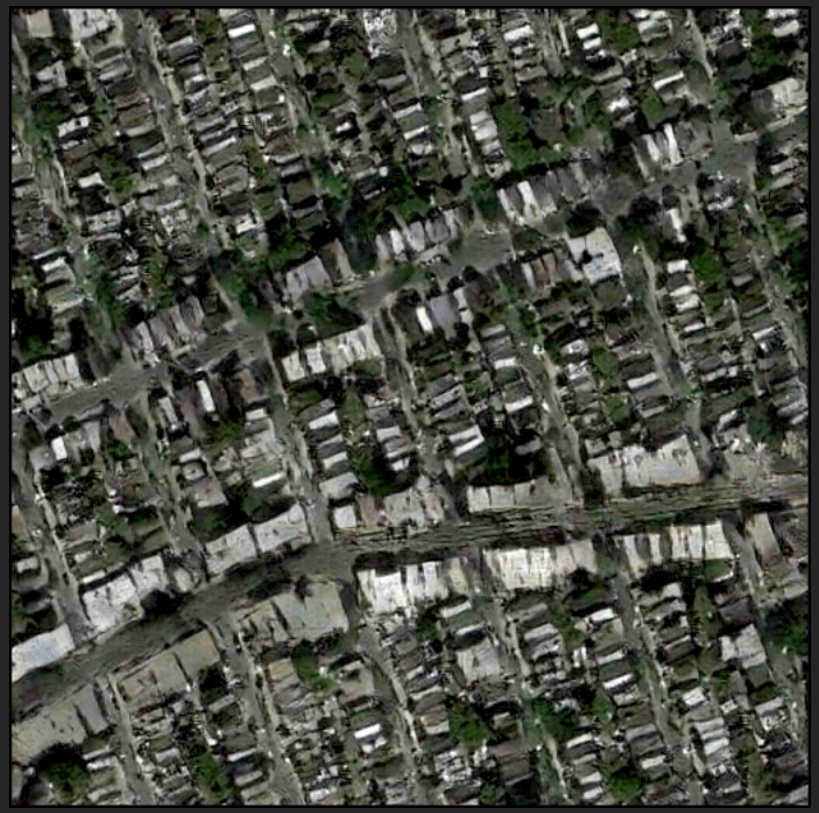
real

loss function:

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

Input

Output

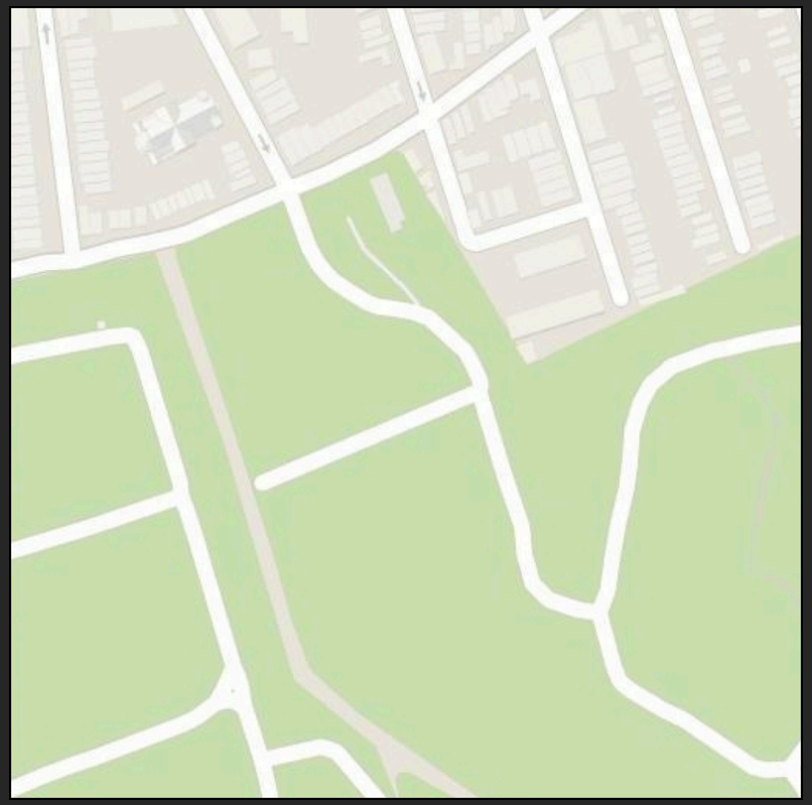


Ground-truth

data downloaded from the Google Map

Isola et al. Image-to-Image Translation with Conditional Adversarial Nets. CVPR'17

Input

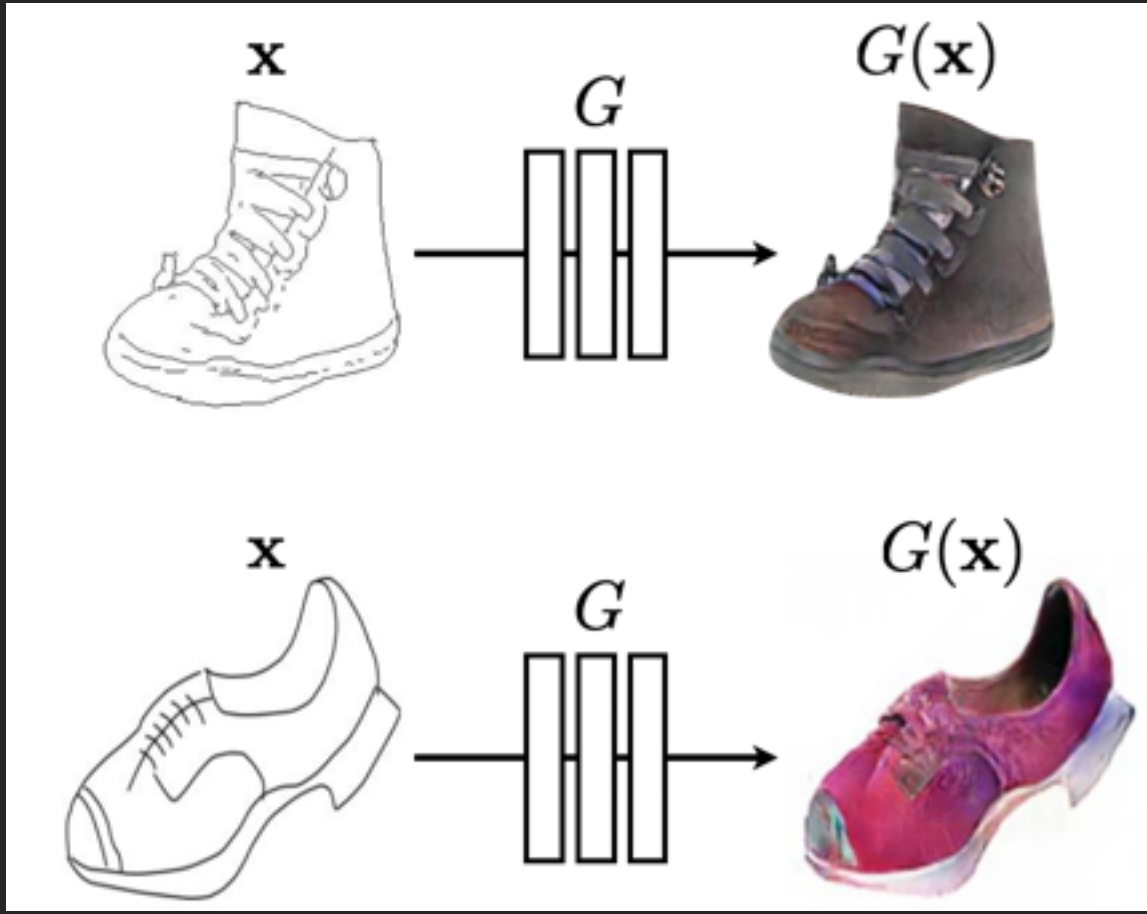


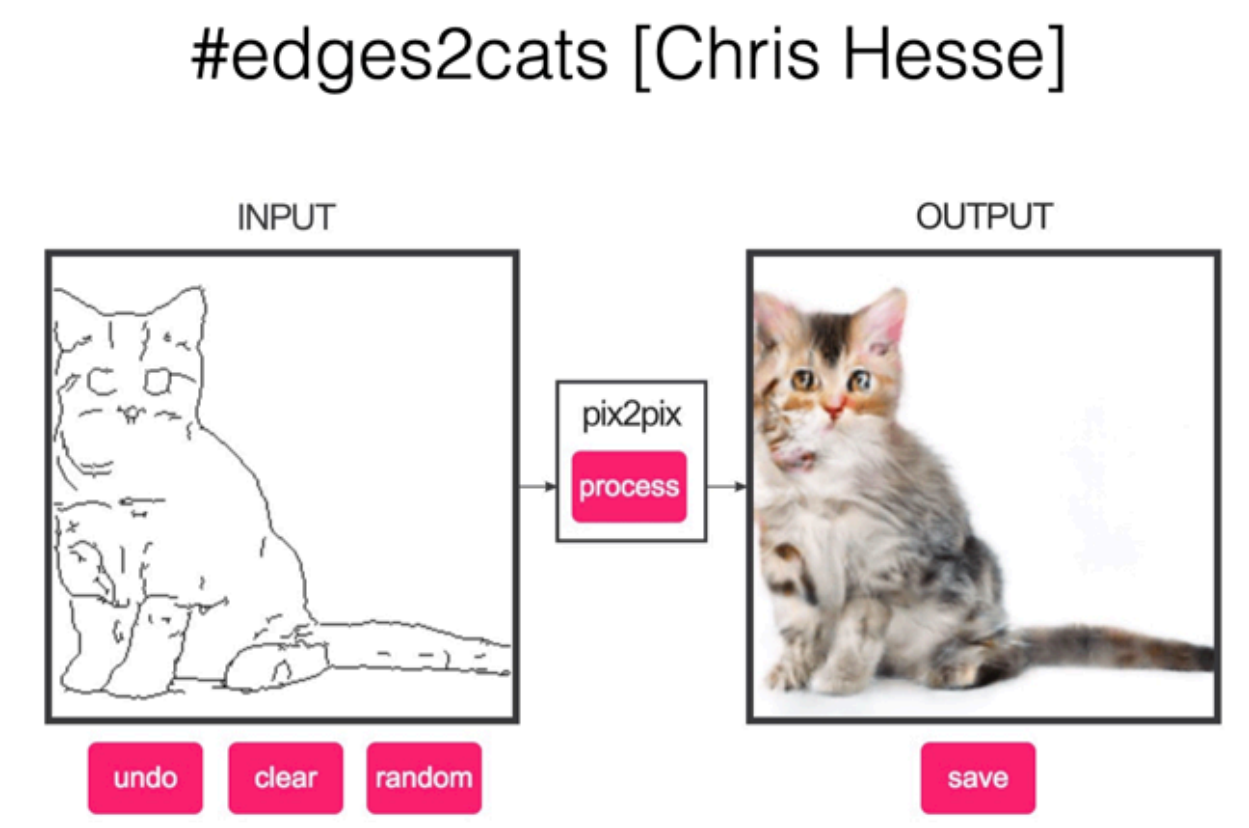
L1 loss

GAN loss

More structured loss

Amazing applications based on Pix2Pix





https://phillipi.github.io/pix2pix/

Amazing applications based on Pix2Pix

#edges2cats

Christopher Hesse trained our model on converting edge maps to photos of cats, and included this in his interactive demo. Apparently, this is what the Internet wanted most, and #edges2cats briefly went viral. The above cats were designed by Vitaly Vidmirov (@vvid).

Alternative Face

Mario Klingemann used our code to translate the appearance of French singer Francoise Hardy onto Kellyanne Conway's infamous "alternative facts" interview. Interesting articles about it can be read here and here.

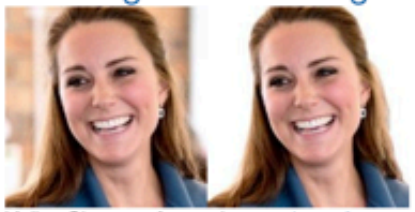
Person-to-Person

Brannon Dorsey recorded himself mimicking frames from a video of Ray Kurzweil giving a talk. He then used this data to train a Dorsey→Kurzweil translator, allowing him to become a kind of puppeter in control of Kurzweil's appearance.

Interactive Anime

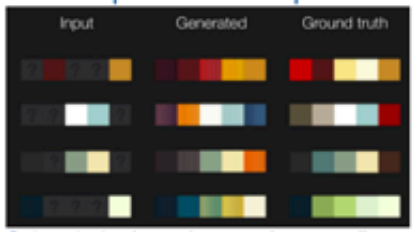
Bertrand Gondouin trained our method to translate sketches→Pokemon, resulting in an interactive drawing tool.

Background masking



Kaihu Chen performed a number of interesting experiments using our method, including getting it to mask out the background of a portrait as shown above.

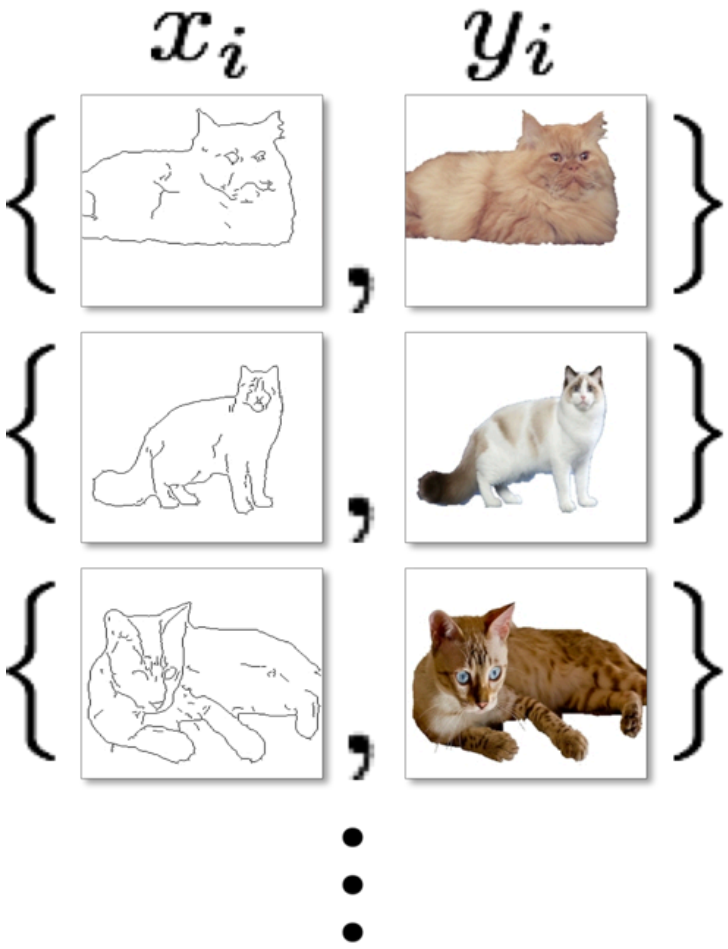
Color palette completion



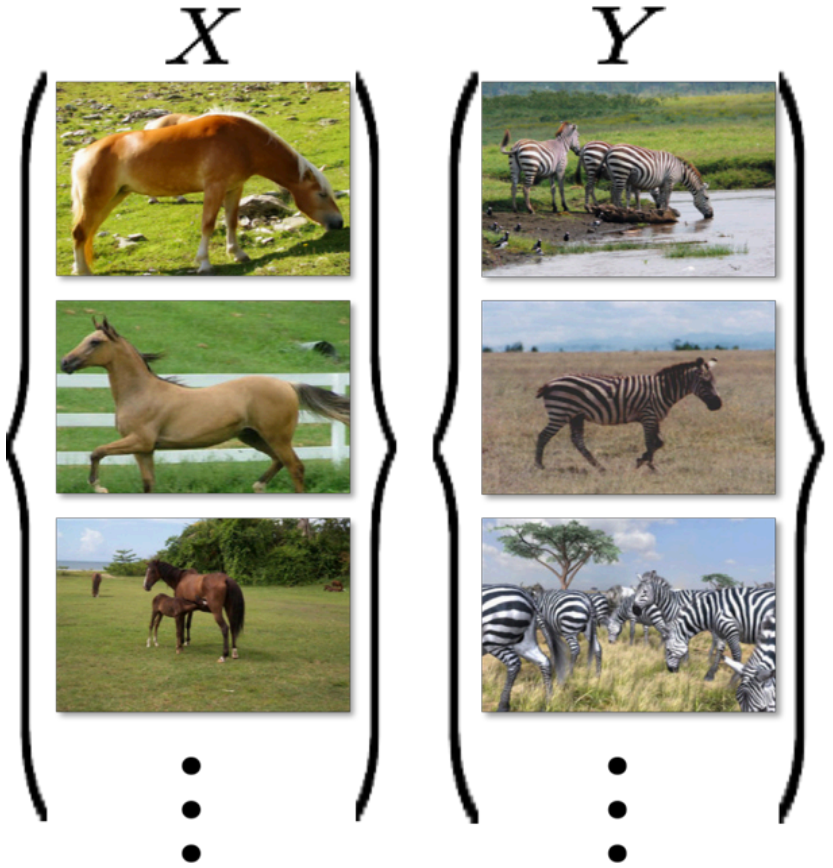
Colormind adapted our code to predict a complete 5-color palette given a subset of the palette as input. This application stretches the definition of what counts as "image-to-image translation" in an exciting way: if you can visualize your input/output data as images, then image-to-image methods are applicable! (not that this is necessarily the best choice of representation, just one to think about.)

https://phillipi.github.io/pix2pix/

Paired data



Unpaired data

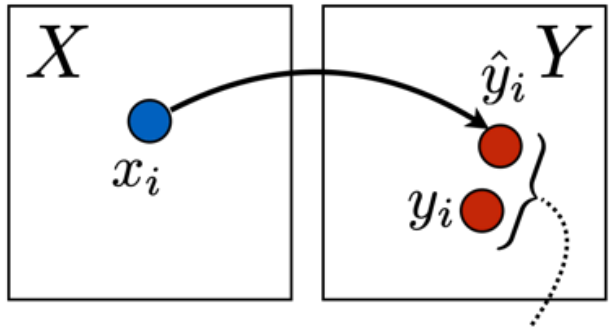


slide from Phillip Isola

Paired translation

Training data

Objective



regression error

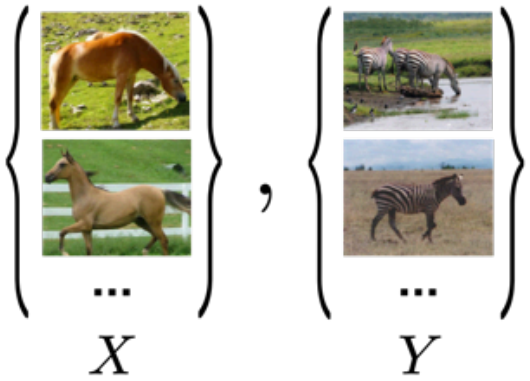
Input

Result

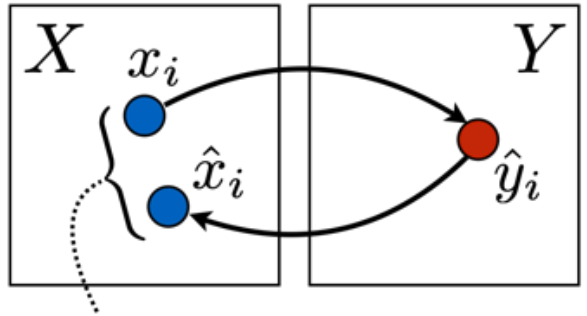
["pix2pix", Isola, Zhu, Zhou, Efros, 2017]

Unpaired translation

Training data



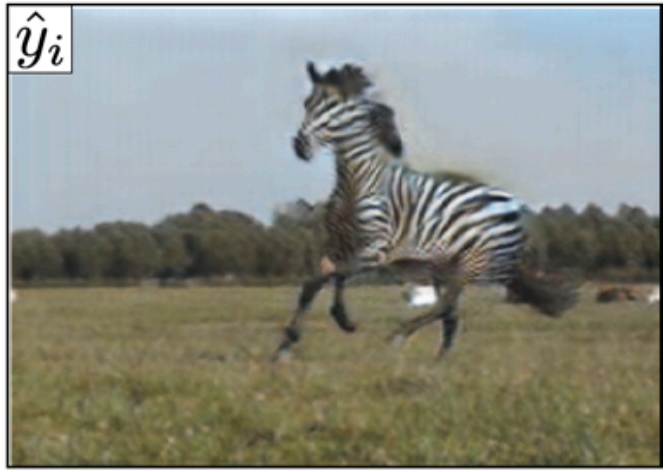
Objective



cycle-consistency error

Input

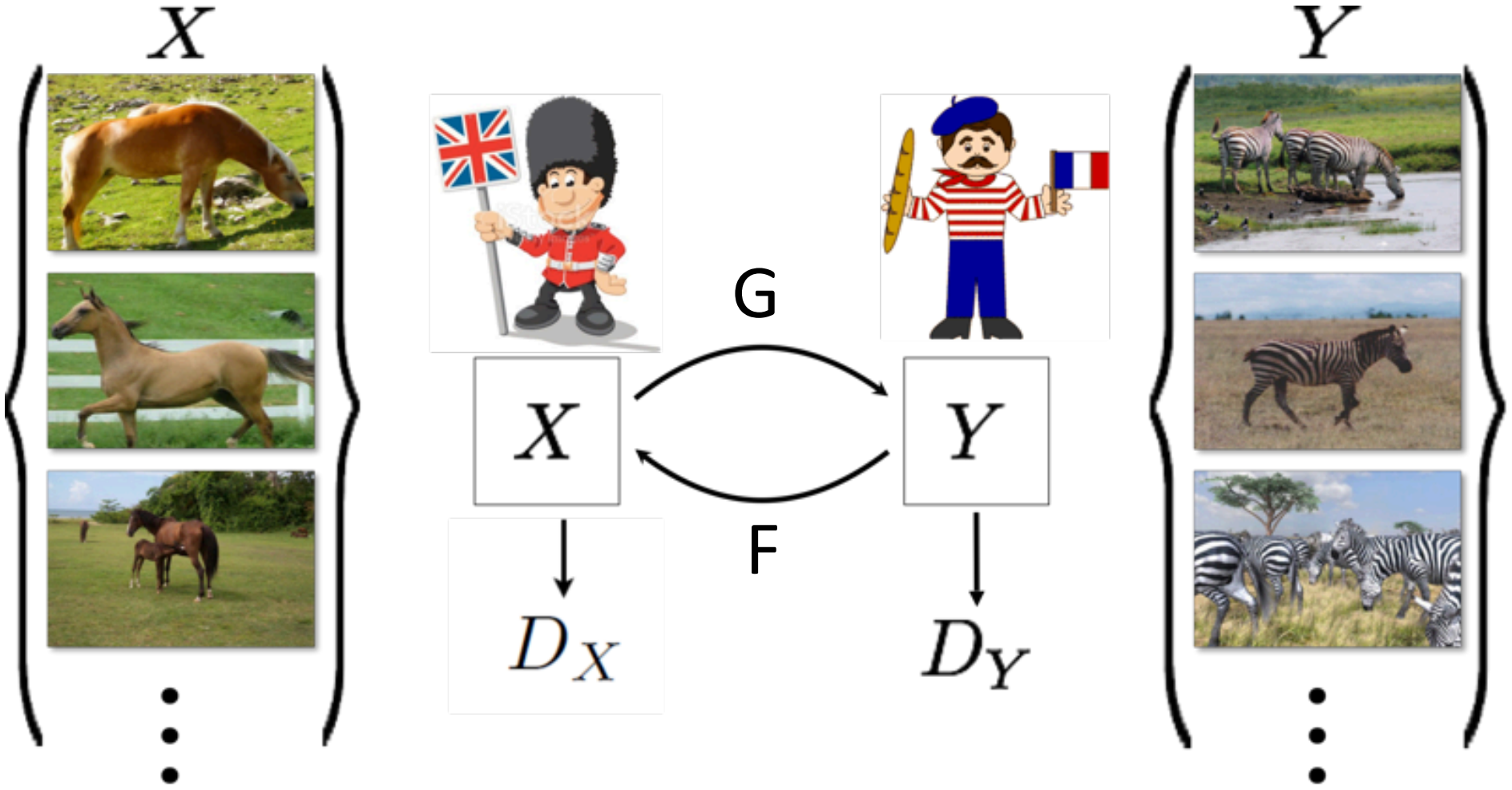
Result



["CycleGAN", Zhu*, Park*, Isola, Efros, 2017]

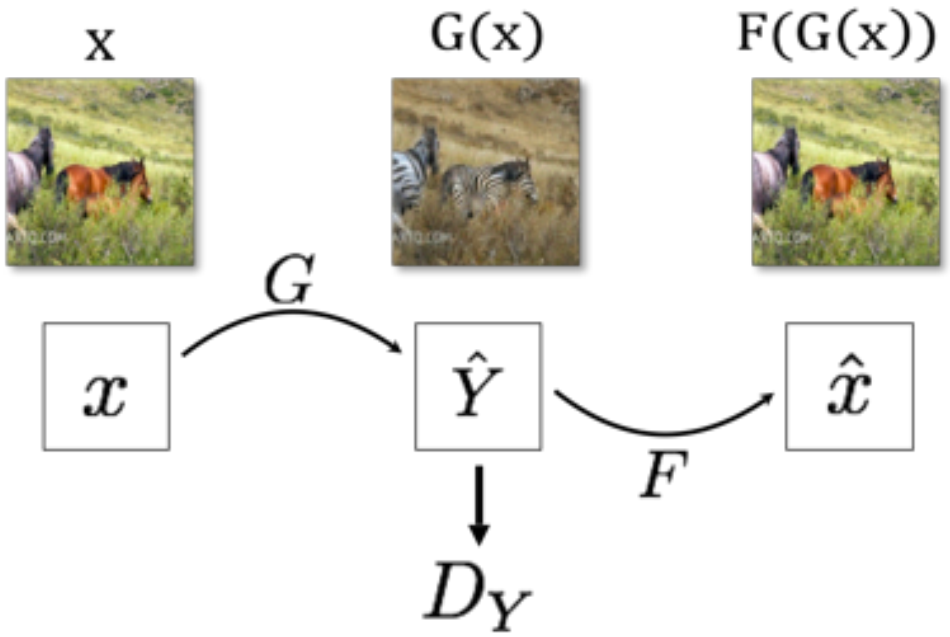
slide from Phillip Isola

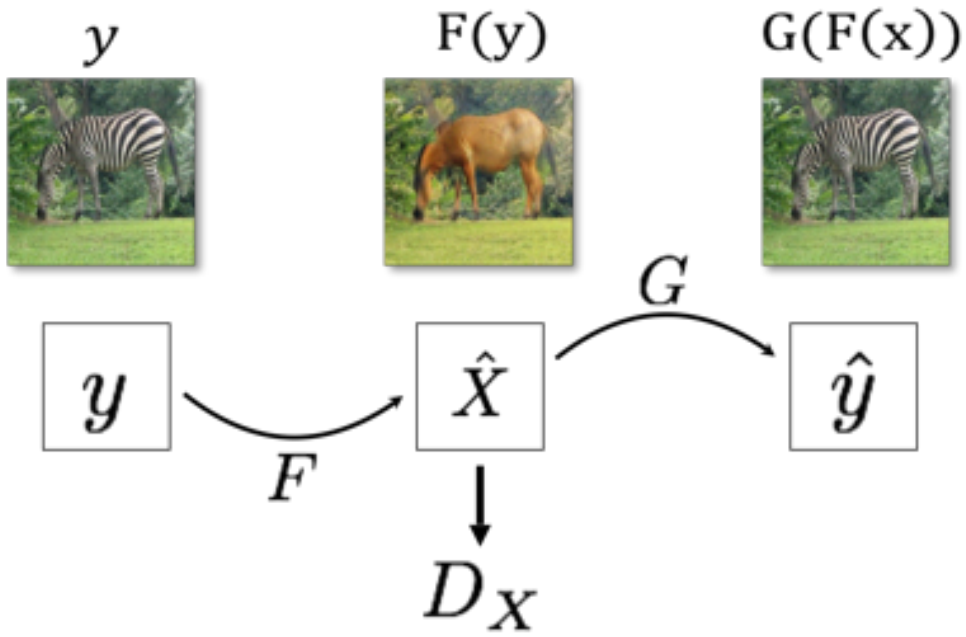
Cycle Consistency

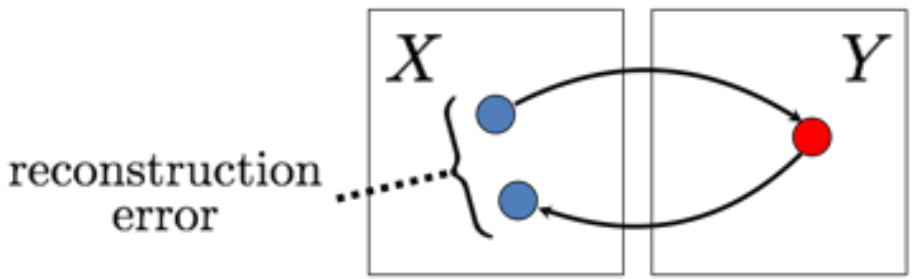


slide from Phillip Isola

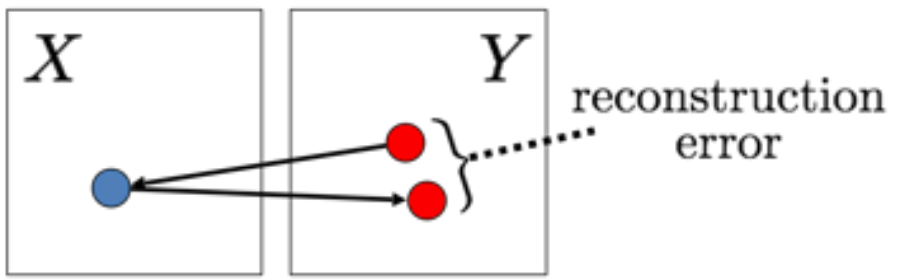
Cycle Consistency







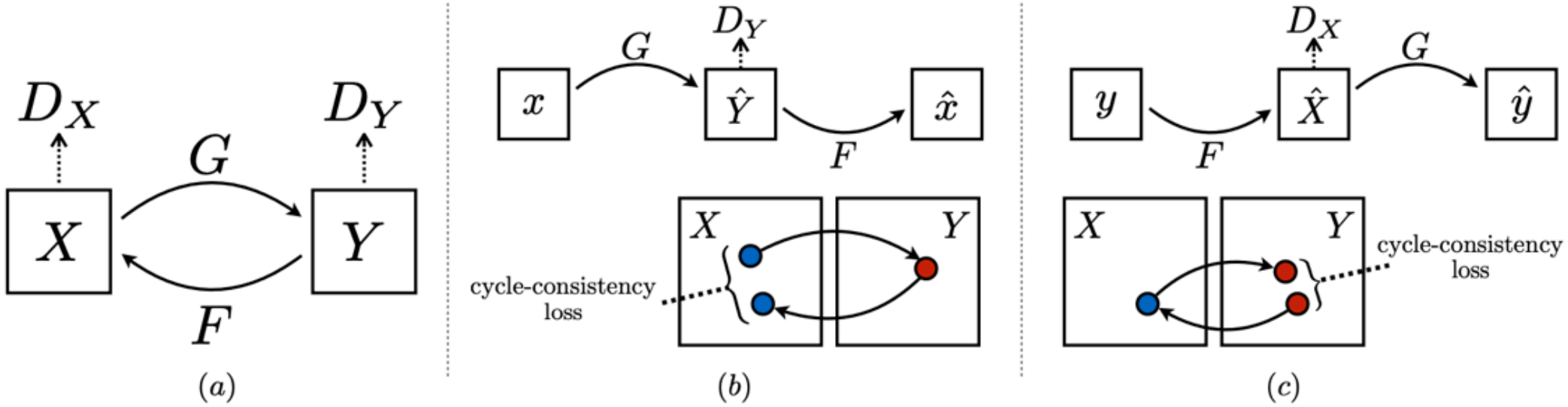
$$\|F(G(x)) - x\|_1$$



$$\|G(F(y)) - y\|_1$$

slide from Phillip Isola

Cycle Consistency



$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],$$

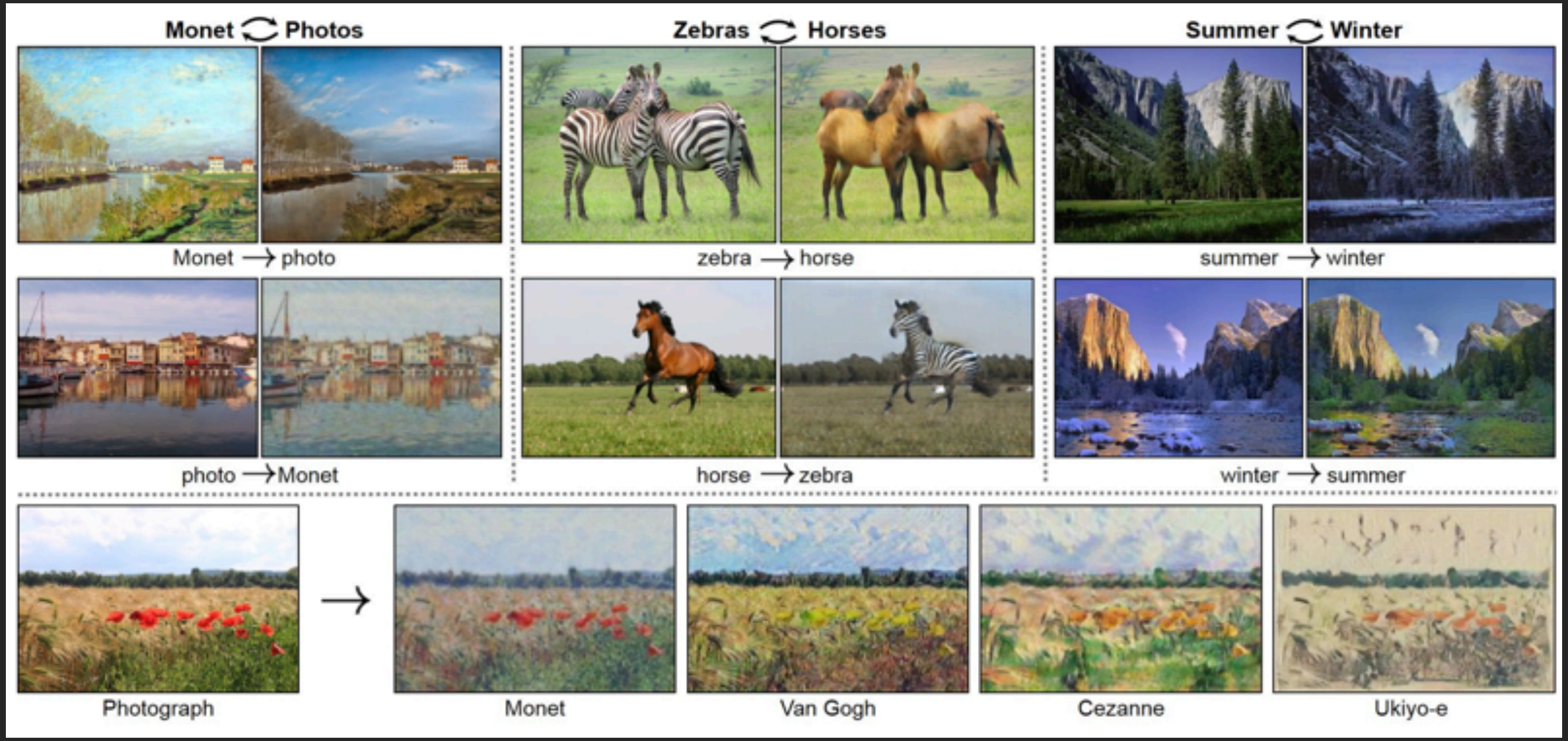
$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

Our full objective is:

$$\mathcal{L}(G, F, D_X, D_Y) = \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) + \lambda \mathcal{L}_{\text{cyc}}(G, F),$$

https://arxiv.org/pdf/1703.10593.pdf

Amazing applications based on CycleGAN



Check this out: https://junyanz.github.io/CycleGAN/

Summary

- Introduction: from recognition to recreation
- Overview of the generative models
- Understanding generative models
- Inverting images for real image editing
- Image2Image translation

Acknowledgement

GenForce Team at CUHK

Yujun Shen

Ceyuan Yang

Jiapeng Zhu

Yinghao Xu

Code and papers are at
https://genforce.github.io/

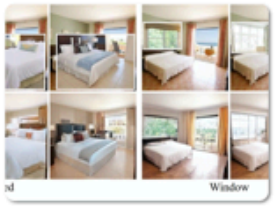
GenForce

Research Initiative on Generative Modeling

May Generative Force Be with You

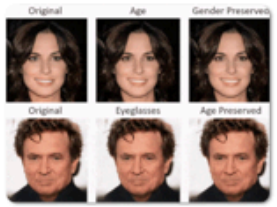
Projects

GenForce Lib
Yujun Shen, Yinghao Xu, Ceyuan Yang, Jiapeng Zhu, Bolei Zhou
*This is an **efficient PyTorch library** for deep generative modeling.*
[Code] [Colab]



Generative Hierarchical Features from Synthesizing Images
Yinghao Xu*, Yujun Shen*, Jiapeng Zhu, Ceyuan Yang, Bolei Zhou
arXiv.2007.10379 preprint
[Paper] [Project Page] [Code]

Closed-Form Factorization of Latent Semantics in GANs
Yujun Shen, Bolei Zhou
arXiv.2007.06600 preprint
[Paper] [Project Page] [Code] [Demo Video]



InterFaceGAN: Interpreting the Disentangled Face Representation Learned by GANs
Yujun Shen, Ceyuan Yang, Xiaoou Tang, Bolei Zhou
IEEE Transactions on Pattern Recognition (TPAMI), Oct 2020
[Paper] [Project Page] [Code] [Demo Video]

In-Domain GAN Inversion for Real Image Editing
Jiapeng Zhu*, Yujun Shen*, Deli Zhao, Bolei Zhou
European Conference on Computer Vision (ECCV), 2020
[Paper] [Project Page] [Code] [Demo Video]