Advanced Deep Learning 2025
GANs & GAN editing

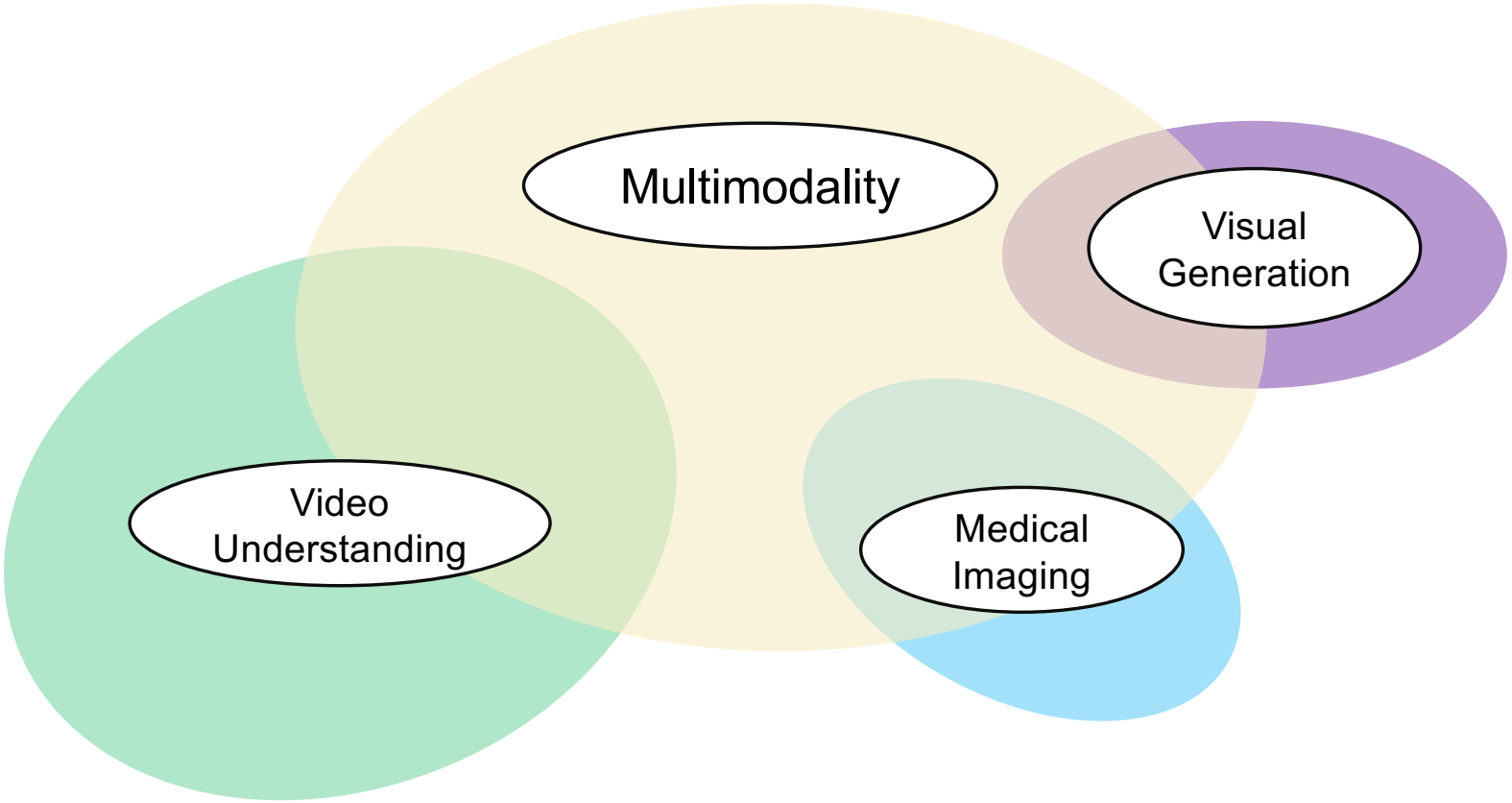
Vicky Kalogeiton

Lecture 1: CSC_52087_EP

About me

- *Assistant Professor*, 2020 –
  - VISTA Group, Ecole Polytechnique, France
  - Main CV and genAI researcher at Polytechnique
- *Research Fellow,* 2019 – 2021
- *Post-doc,* 2018 – 2020
  - Visual Geometry Group, University of Oxford, UK
  - Andrew Zisserman
- *PhD*, 2013 – 2017
  - University of Edinburgh, UK, INRIA, Grenoble, France
  - Vittorio Ferrari, Cordelia Schmid

Research agenda



Today's lecture

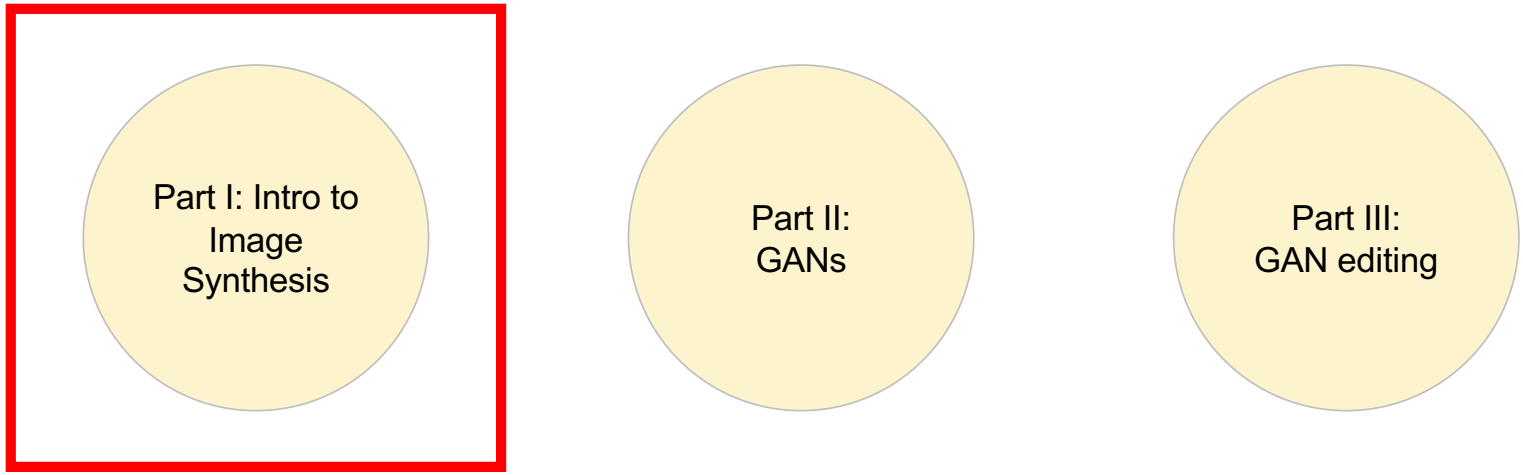
Part I: Intro to Image Synthesis

Part II: GANs

Part III: GAN editing

Slides adapted from many resources:
[Fei-Fei Li & Andrej Karpathy & Justin Johnson & Ross Girshick & VGG]

Today's lecture



Slides adapted from many resources:
[Fei-Fei Li & Andrej Karpathy & Justin Johnson & Ross Girshick & VGG]

How do we understand the world?

- Our brains are like powerful computers, constantly predicting what will happen next based on past experiences

- Just like we use past knowledge to guess future events, generative AI attempts to forecast and create based on what it has learned

The Future of AI with Generative Models

- Generative AI could be a critical part of future AI advancements
- It combines creating new things and understanding causality—how one action leads to another
- Like us, these AI systems can simulate different scenarios and outcomes, helping to make better decisions

Precision Meets Creativity: A new era in AI

- Predict
- Classify
- Recommend

- Imagine
- Augment
- Create

AGI? Artificial General Intelligence

Transforming Business with Generative AI

- Innovating product design
  - Generative Games, word synthesis
  - Shoes, jewelry, clothing, …
  - Rapid Prototyping
  - Material Innovation
- Enhancing customer interaction
  - Personalized recommendations
  - Automated customer support
  - Interactive virtual assistant
  - Sentiment Analysis
- Streamlining content creation
  - 3D house models
  - AI-assisted Video Production, Choreography
  - Dynamic Web Content
  - Marketing Material Creation

Creative palette

- Images
- Audio
- Text
- 3D models
- ....

Information

Introduction

Introduction

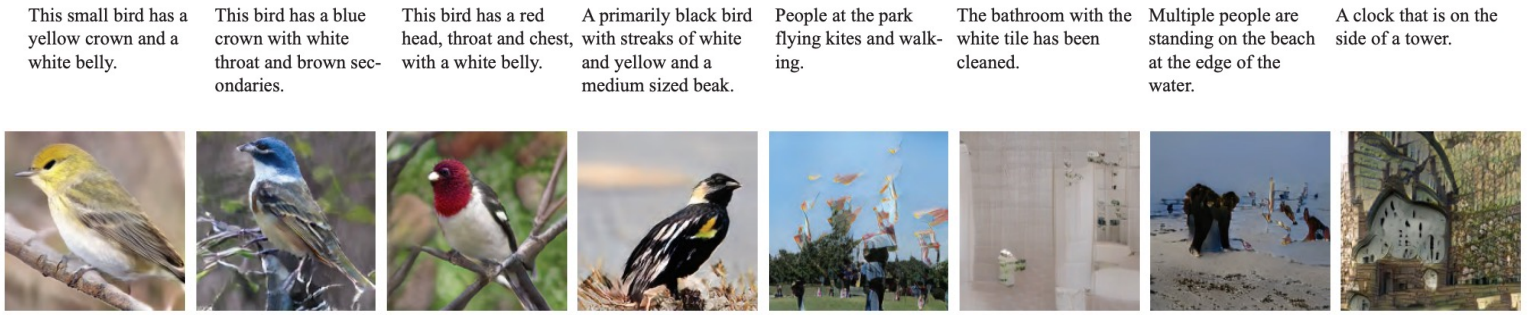
A stop sign is flying in blue skies.

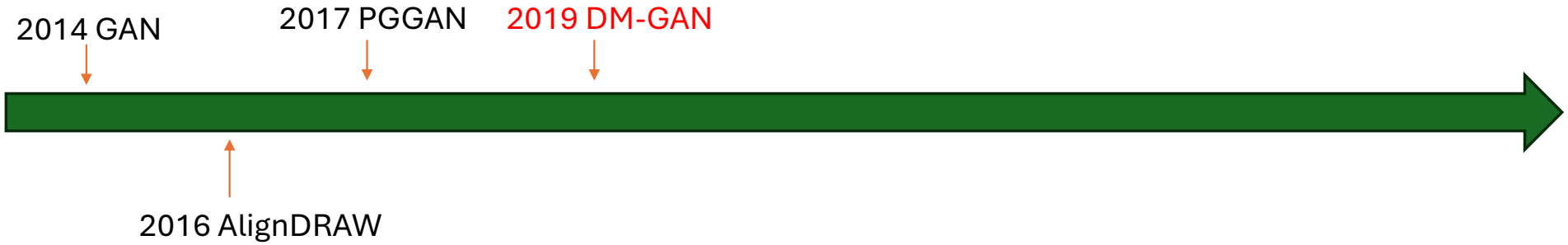
A herd of elephants flying in the blue skies.

A toilet seat sits open in the grass field.

A person skiing on sand clad vast desert.

Introduction





Introduction

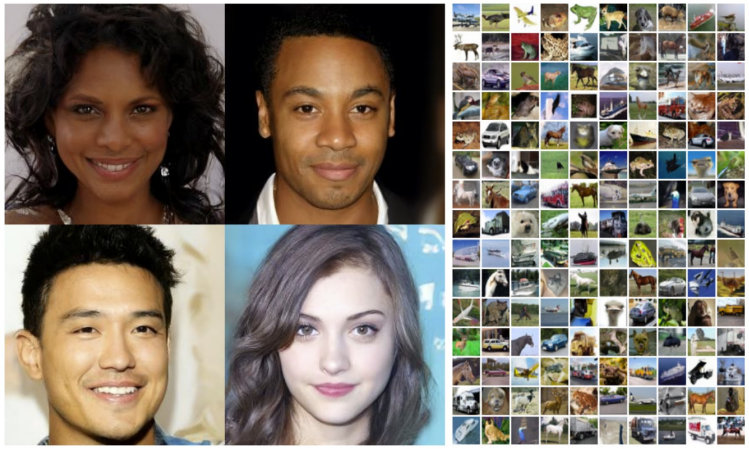
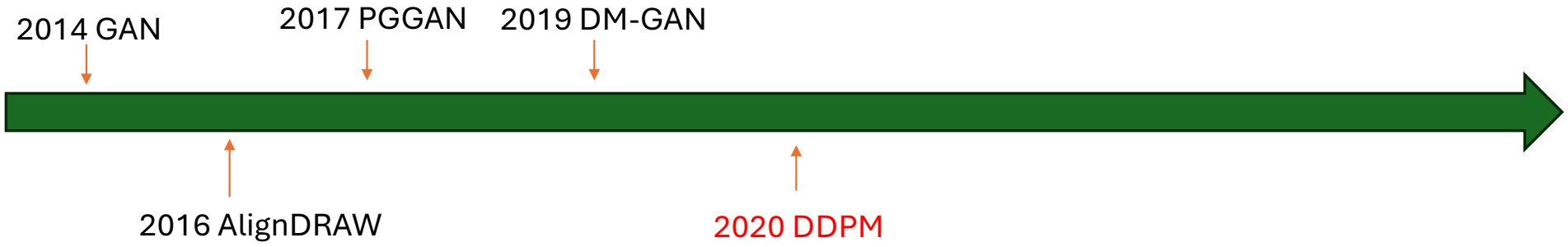
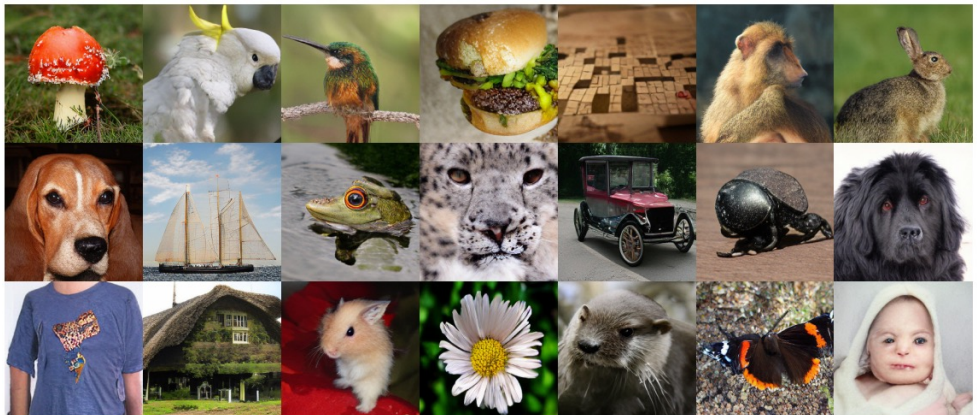


Figure 1: Generated samples on CelebA-HQ 256 × 256 (left) and unconditional CIFAR10 (right)



Introduction



2014 GAN

2017 PGGAN

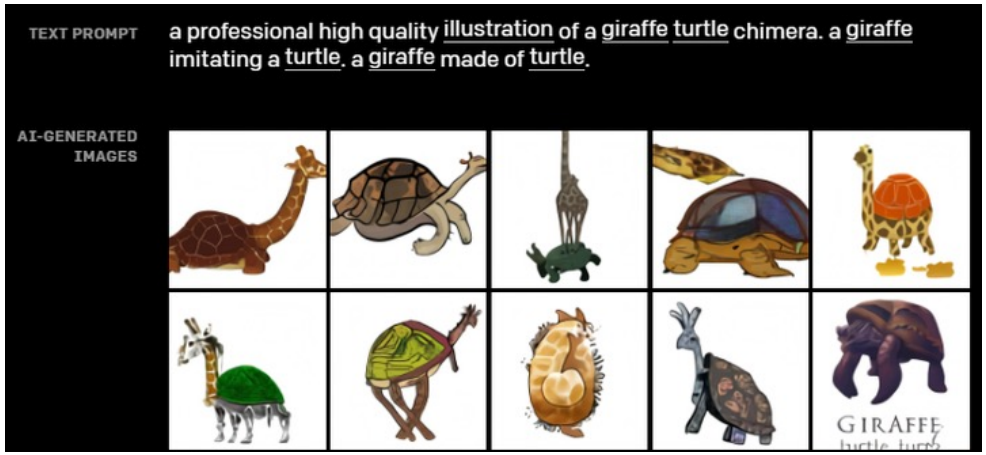
2019 DM-GAN(T2I)

2021 VQ-GAN

2016 AlignDRAW

2020 DDPM

Introduction



DALLE-1

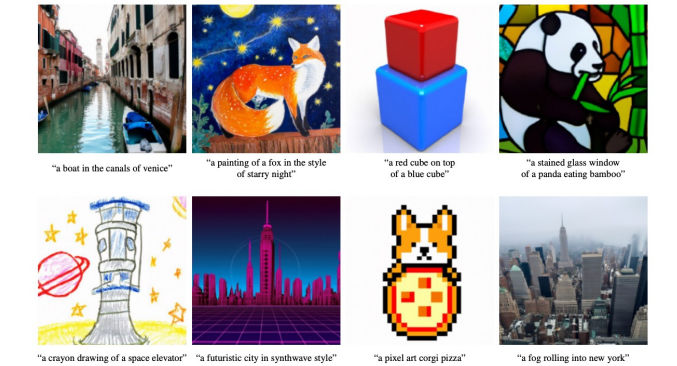
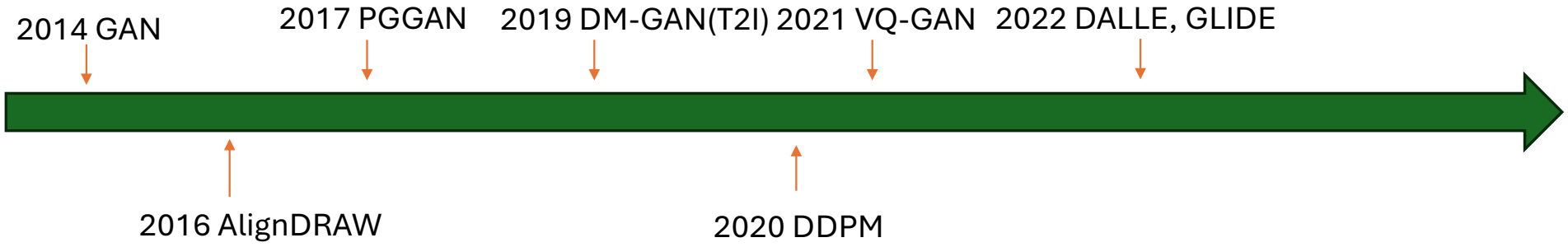


Figure 1. Selected samples from GLIDE using classifier-free guidance. We observe that our model can produce photorealistic images with shadows and reflections, can compose multiple concepts in the correct way, and can produce artistic renderings of novel concepts. For random sample grids, see Figure 17 and 18.

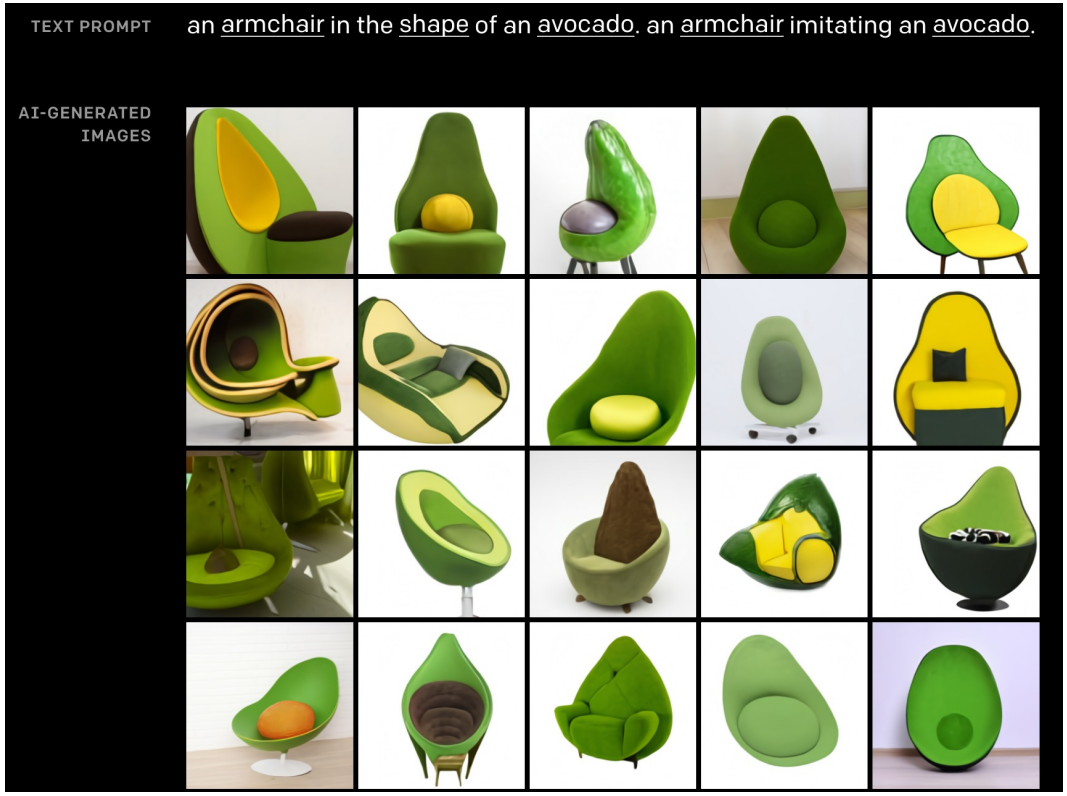
GLIDE



Examples

http://www.youtube.com/watch?v=_x9AwxfjxvE&t=1m45s

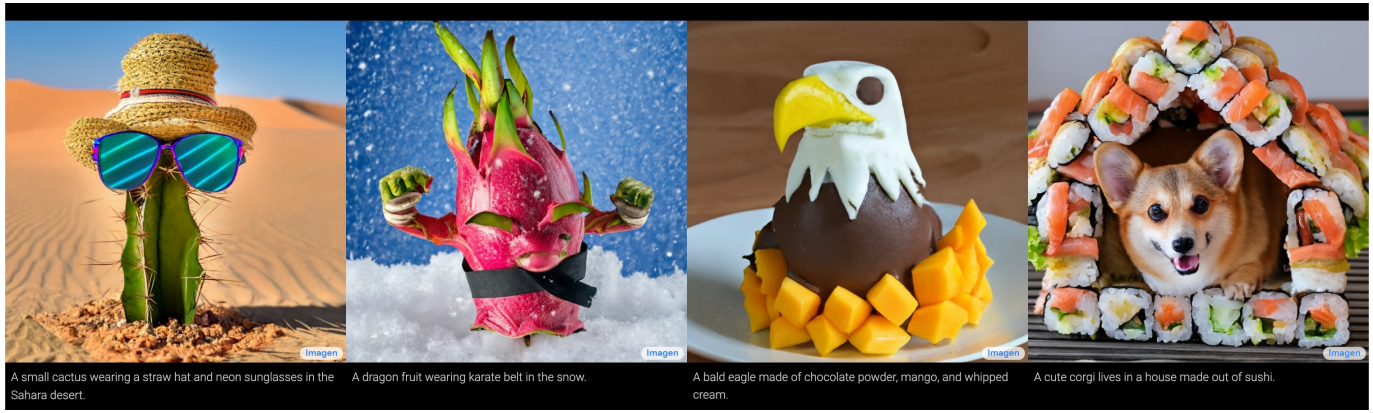
DALLE



https://openai.com/blog/dall-e/

23 May 2022

Imagen by Google AI

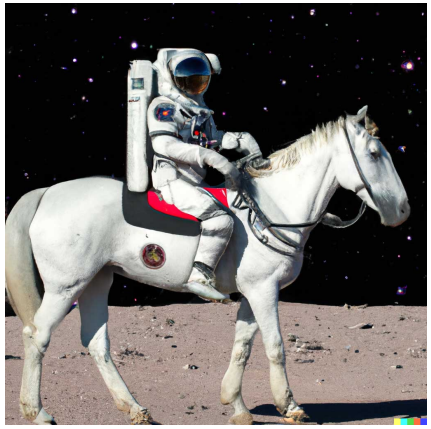


https://imagen.research.google/

Dalle-2 (Text-to-Image)

13 April 2022

A bowl of soup as a planet in the universe



An astronaut riding a horse in a photorealistic style

Teddy bears mixing sparkling chemicals as mad scientists

Diffusion Models

OpenAI: DALL-E3

Midjourney

music, audio, animation, video, physical etc....

Stable Diffusion 3

February 2024

Make-A-Video (Text-to-Video)

A confused grizzly bear in a calculus class

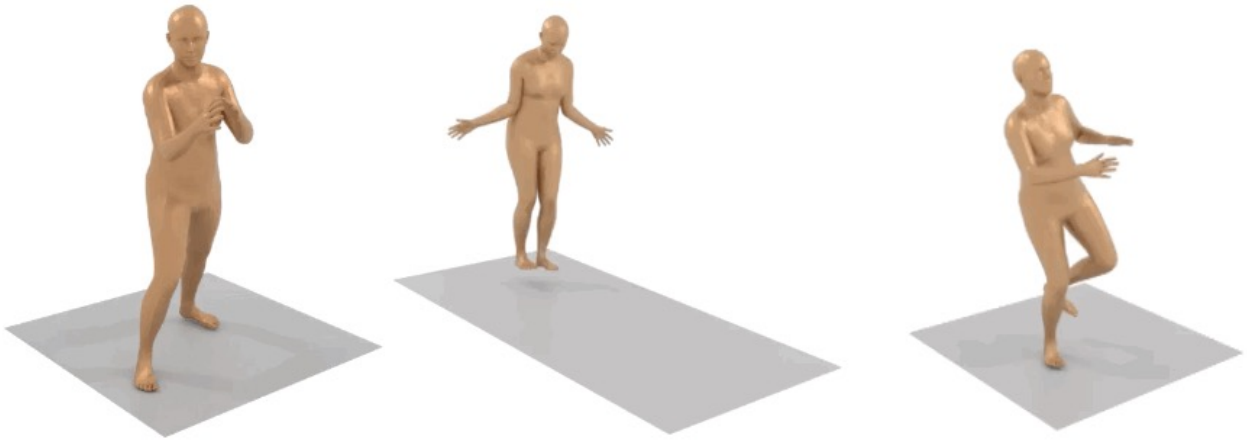
A golden retriever eating ice cream on a beautiful tropical beach at sunset, high resolution

A panda playing on a swing set

SORA (Text-to-Video)

February 2024

Human Motion Diffusion (Text-to-Motion)

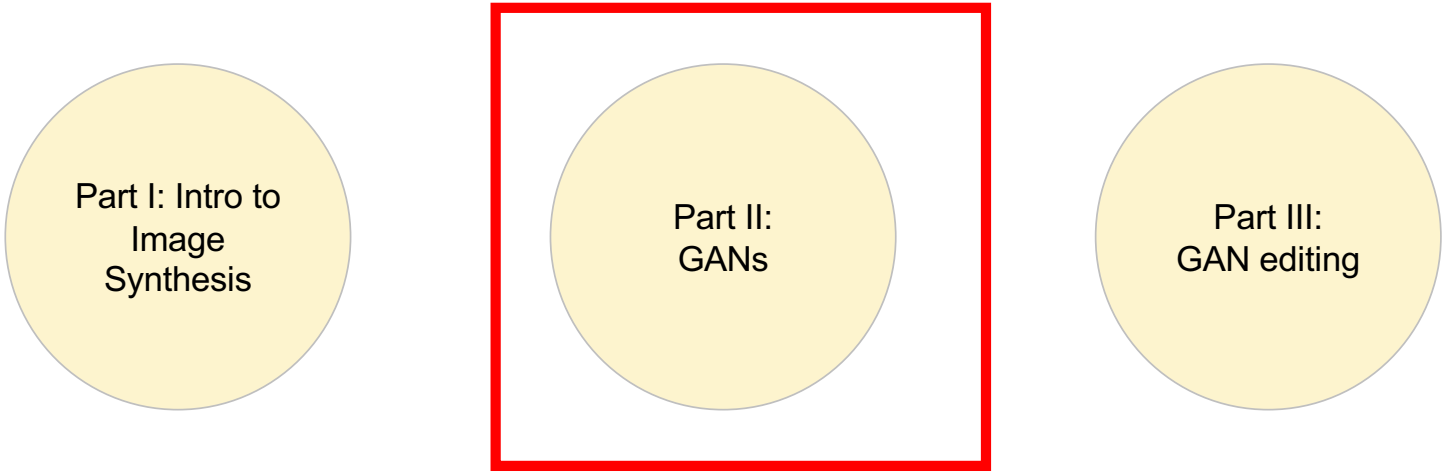


"A person punches in a manner consistent with martial arts."

"A person is skipping rope."

"a man kicks with something or someone with his left leg."

Today's lecture



Slides adapted from many resources:
[Fei-Fei Li & Andrej Karpathy & Justin Johnson & Ross Girshick & VGG]

Part II: GANs

- Relation among models
- Training objective
- Training GANs
- DCGAN
- Tricks for improving performance
- StyleGAN
- Metrics
- Conditional GANs
- Image Translation

Relation among models

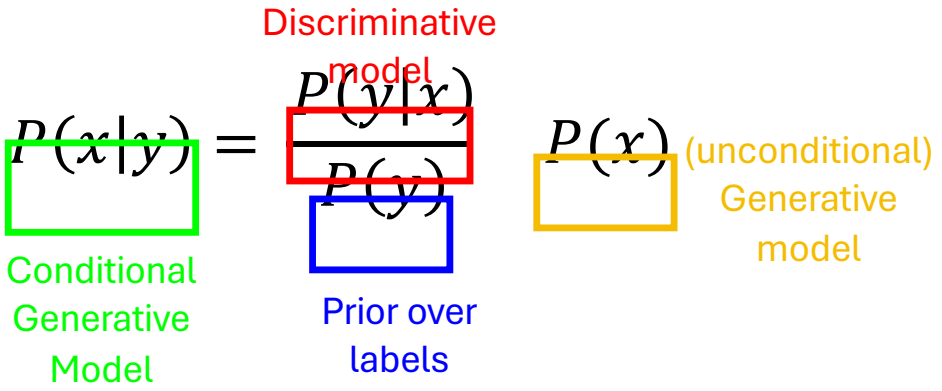
Relation among all models

Bayes' Rule:

$$P(x|y) = \frac{P(y|x)}{P(y)} P(x)$$

Relation among all models

Bayes' Rule:



What can we do with them?

- **Discriminative Model**: Learn a probability distribution p(y/x)
  - Assign labels to data
  - Feature Learning using labels
- **Generative Model**: Learn a probability distribution p(x)

- **Conditional Generative Model**: Learn p(x/y)

What can we do with them?

- **Discriminative Model**: Learn a probability distribution p(y/x)
  - Assign labels to data
  - Feature Learning using labels
- **Generative Model**: Learn a probability distribution p(x)
  - Detect Outliers
  - Unsupervised Feature Learning

- **Conditional Generative Model**: Learn p(x/y)

What can we do with them?

- **Discriminative Model**: Learn a probability distribution p(y/x)
  - Assign labels to data
  - Feature Learning using labels
- **Generative Model**: Learn a probability distribution p(x)
  - Detect Outliers
  - Unsupervised Feature Learning
  - Sample to **generate** new data
- **Conditional Generative Model**: Learn p(x/y)

What can we do with them?

- **Discriminative Model**: Learn a probability distribution p(y/x)
  - Assign labels to data
  - Feature Learning using labels
- **Generative Model**: Learn a probability distribution p(x)
  - Detect Outliers
  - Unsupervised Feature Learning
  - Sample to **generate** new data
- **Conditional Generative Model**: Learn p(x/y)
  - Generate new data conditioned on input labels

Generative Adversarial Networks

Generative Adversarial Networks

- Assume data x_i drawn from distribution $p_{data}(x)$.
  - Want to sample from p_{data}

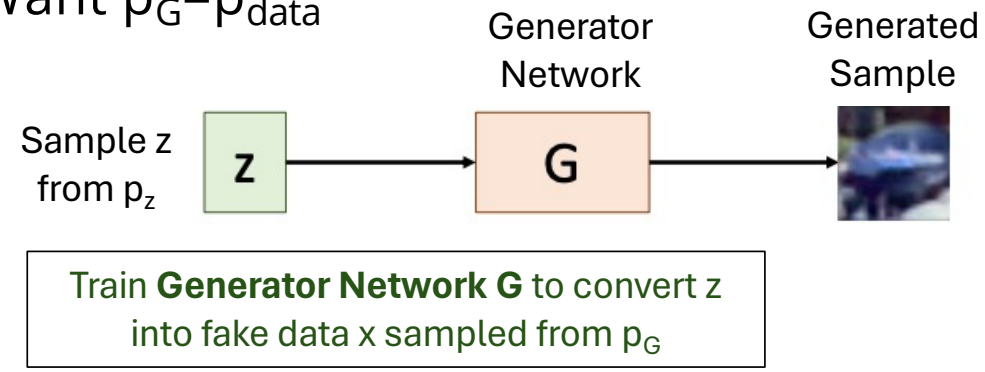
Introduced in 2014 by Ian Goodfellow

Generative Adversarial Networks

- Assume data x_i drawn from distribution $p_{data}(x)$.
  - Want to sample from p_{data}
- **Idea**: Introduce a latent variable z with simple prior p(z)
  - Sample z ~p(z) and pass to a Generator Network x=G(z)
  - X is a sample from the Generator distribution p_G.
  - Want $p_G=p_{data}$

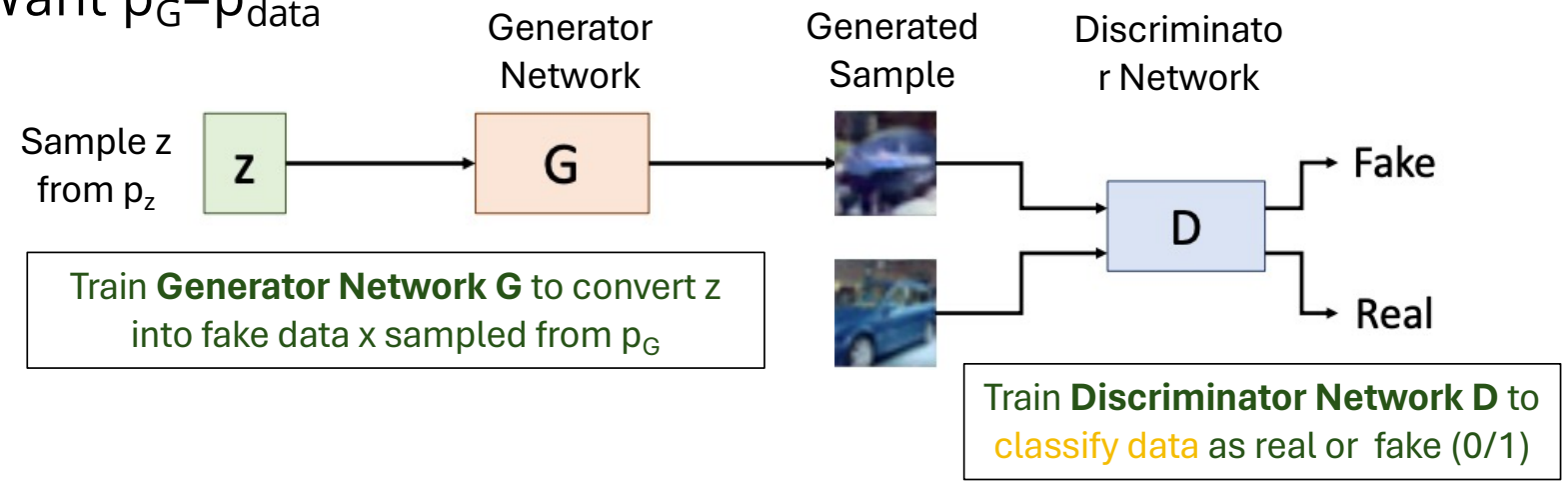
Generative Adversarial Networks

- Assume data x_i drawn from distribution $p_{data}(x)$.
  - Want to sample from p_{data}
- **Idea**: Introduce a latent variable z with simple prior p(z)
  - Sample z ~p(z) and pass to a Generator Network x=G(z)
  - X is a sample from the Generator distribution p_G.
  - Want $p_G=p_{data}$



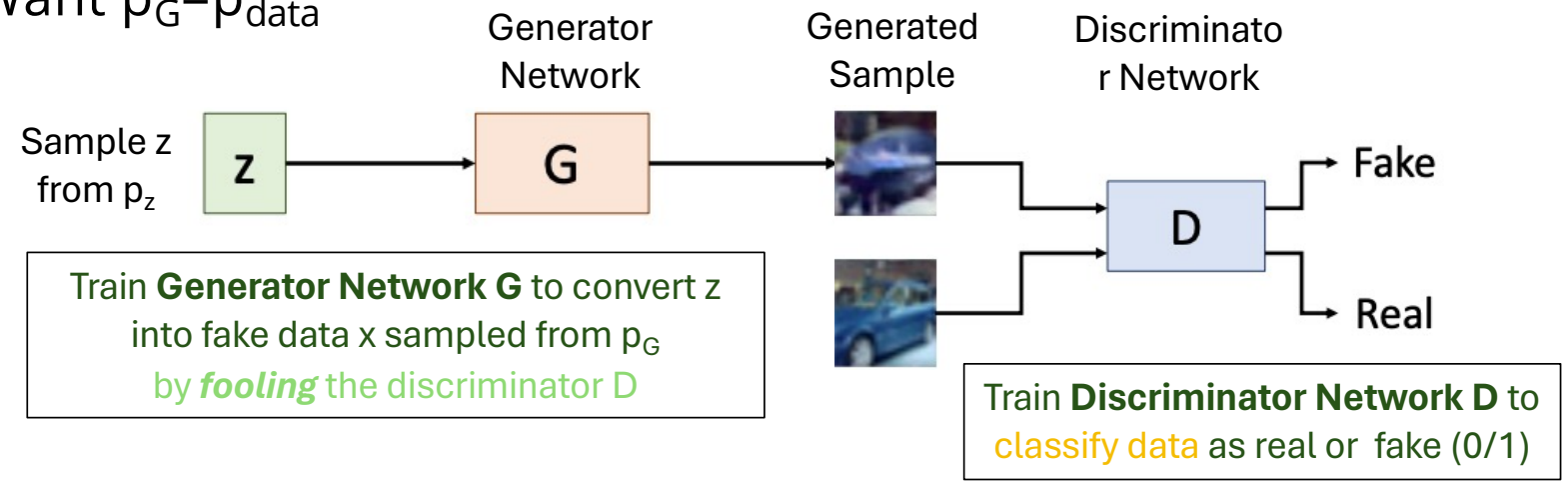
Generative Adversarial Networks

- Assume data x_i drawn from distribution $p_{data}(x)$.
  - Want to sample from p_{data}
- **Idea**: Introduce a latent variable z with simple prior p(z)
  - Sample z ~p(z) and pass to a Generator Network x=G(z)
  - X is a sample from the Generator distribution p_G.
  - Want $p_G=p_{data}$



Generative Adversarial Networks

- Assume data x_i drawn from distribution $p_{data}(x)$.
  - Want to sample from p_{data}
- **Idea**: Introduce a latent variable z with simple prior p(z)
  - Sample z ~p(z) and pass to a Generator Network x=G(z)
  - X is a sample from the Generator distribution p_G.
  - Want $p_G=p_{data}$



GANs: Training Objective

GANs: Training Objective

- Jointly train generator G and discriminator D with a minimax game

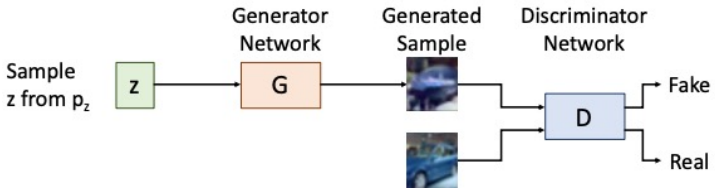
$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

GANs: Training Objective

- Jointly train generator G and discriminator D with a minimax game

Discriminator wants D(x) = 1 for real data

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$



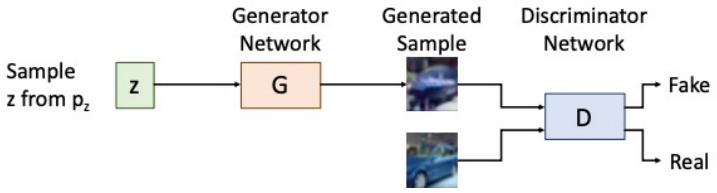
GANs: Training Objective

- Jointly train generator G and discriminator D with a minimax game

Discriminator wants D(x) = 1 for real data

Discriminator wants D(x) = 0 for fake data

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$



GANs: Training Objective

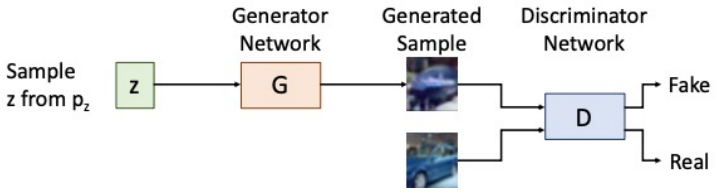
- Jointly train generator G and discriminator D with a minimax game

Discriminator wants D(x) = 1 for real data

Discriminator wants D(x) = 0 for fake data

$$\min_G \max_D \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

Generator wants D(x) = 1 for fake data



GANs: Training Objective

- Jointly train generator G and discriminator D with a minimax game

Train G and D using alternating gradient updates

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$

GANs: Training Objective

- Jointly train generator G and discriminator D with a minimax game

Train G and D using alternating gradient updates

$$\min_G \max_D \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

$$= \min_G \max_D V(G, D)$$

GANs: Training Objective

- Jointly train generator G and discriminator D with a minimax game

Train G and D using alternating gradient updates

$$\min_G \max_D \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

$$= \min_G \max_D V(G, D)$$

For t in 1, ... T:

1. (Update D) $D = D + \alpha_D \frac{\partial V}{\partial D}$
2. (Update G) $G = G - \alpha_G \frac{\partial V}{\partial G}$

GANs: Training Objective

- Jointly train generator G and discriminator D with a minimax game

Train G and D using alternating gradient updates

$$\min_G \max_D \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

$$= \min_G \max_D V(G, D)$$

We are not minimizing any overall loss! No training curves to look at!

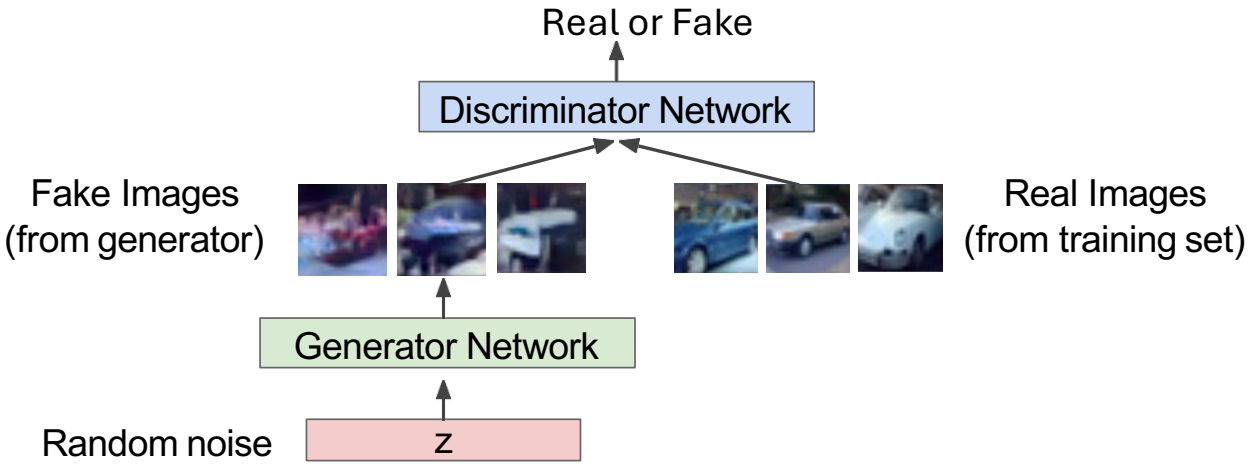
For t in 1, ... T:

1. (Update D) $D = D + \alpha_D \frac{\partial V}{\partial D}$
2. (Update G) $G = G - \alpha_G \frac{\partial V}{\partial G}$

Training GANs

Training GANs: Two-player game

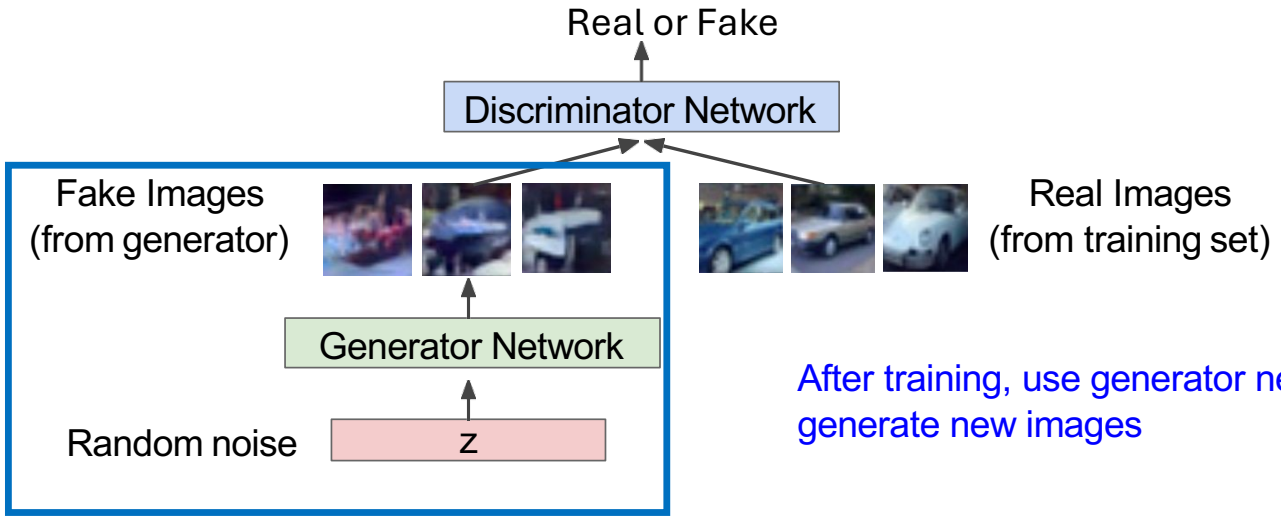
Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Training GANs: Two-player game

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

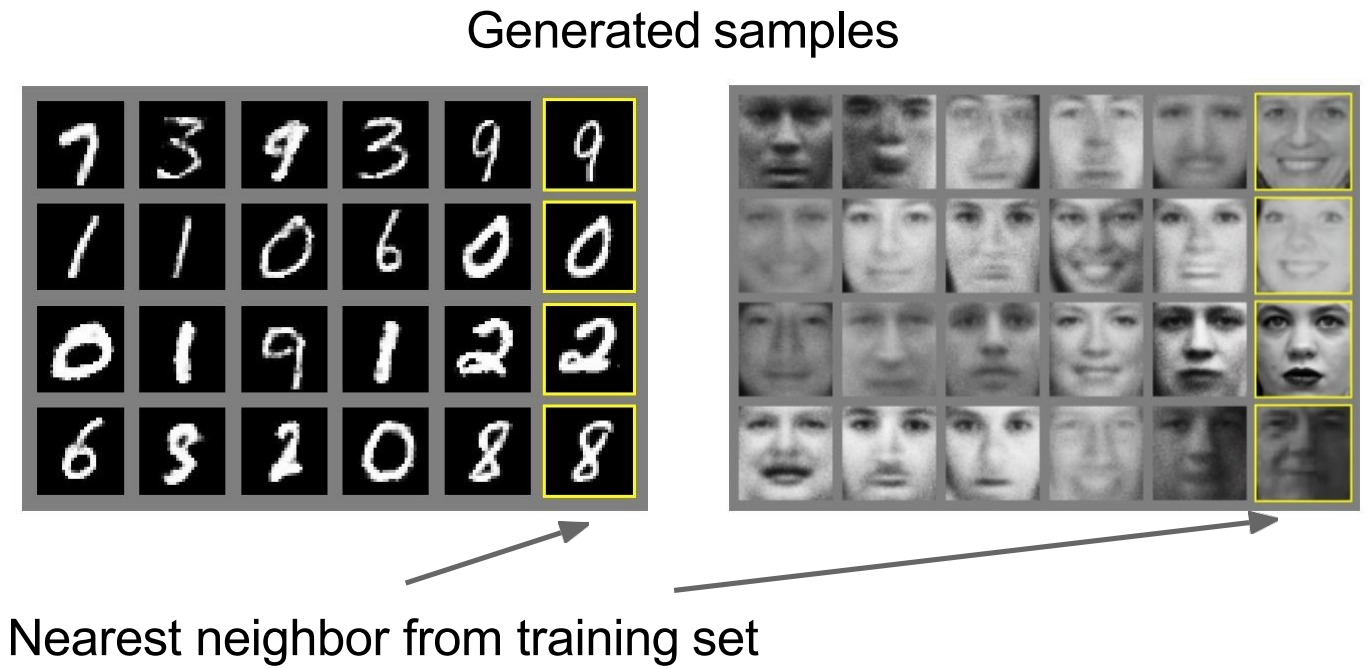


After training, use generator network to generate new images

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

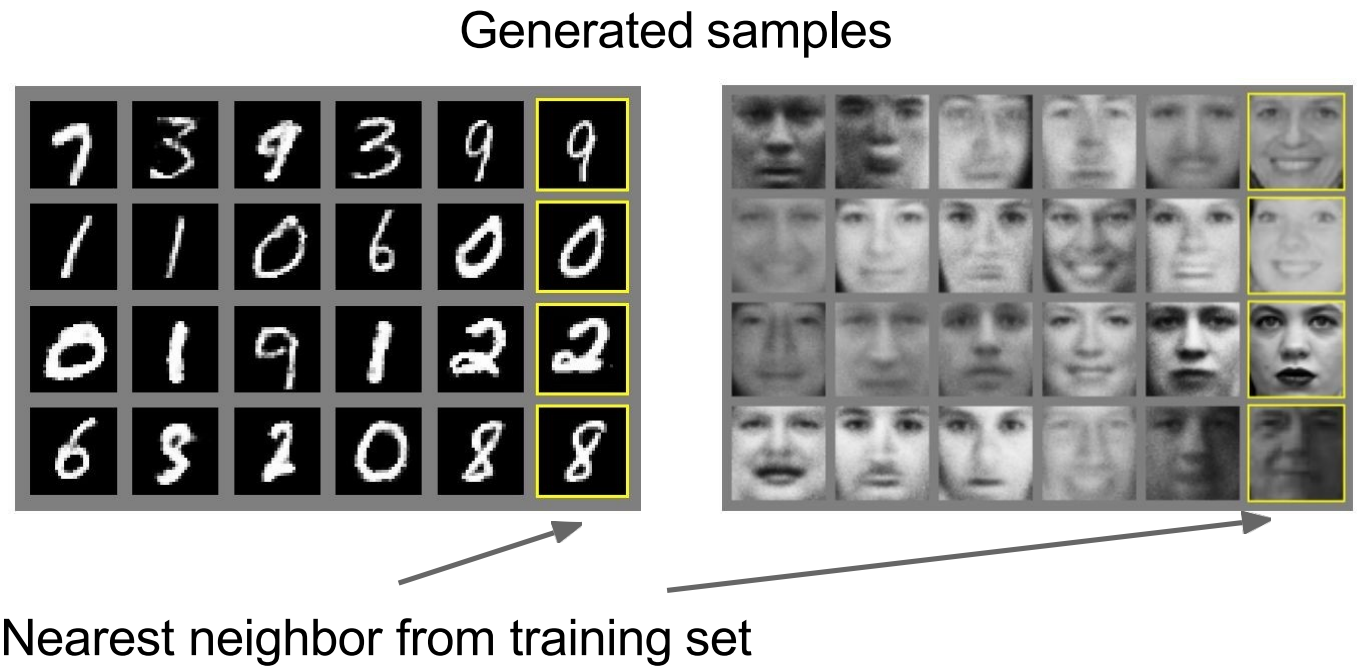
Examples of GANs!

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014



Training GANS is hard! Mode-collapse

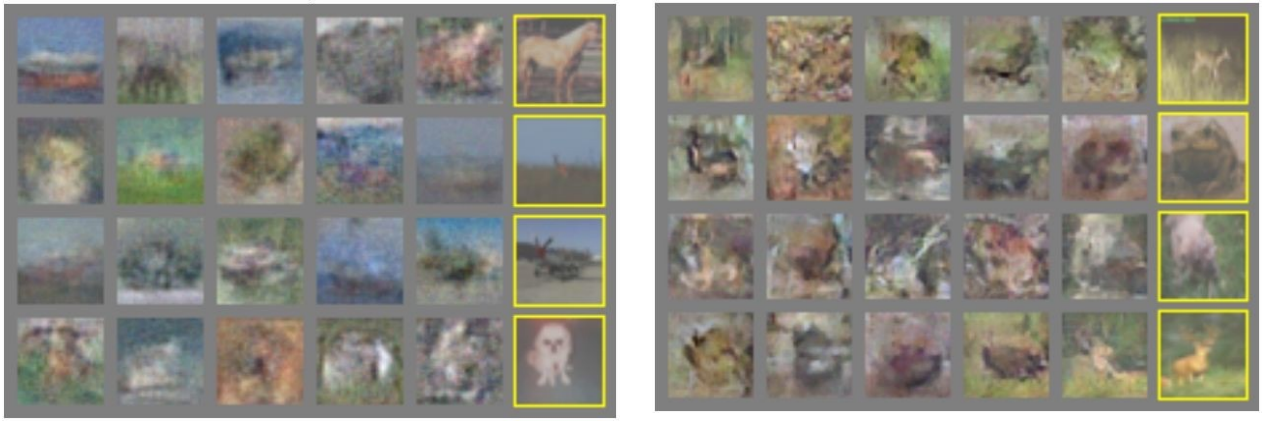
Generator can “memorize” real images



Training GANS is hard!

Generator can produce “fuzzy” images

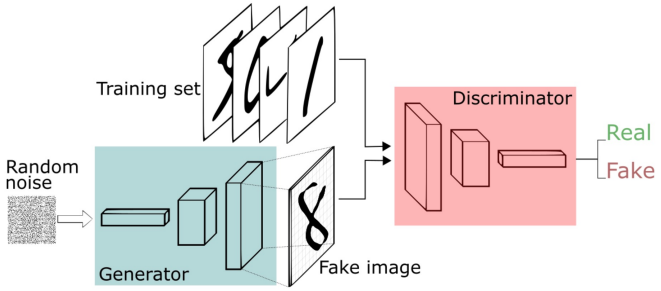
Generated samples (CIFAR-10)



Nearest neighbor from training set

Training GANS is hard!

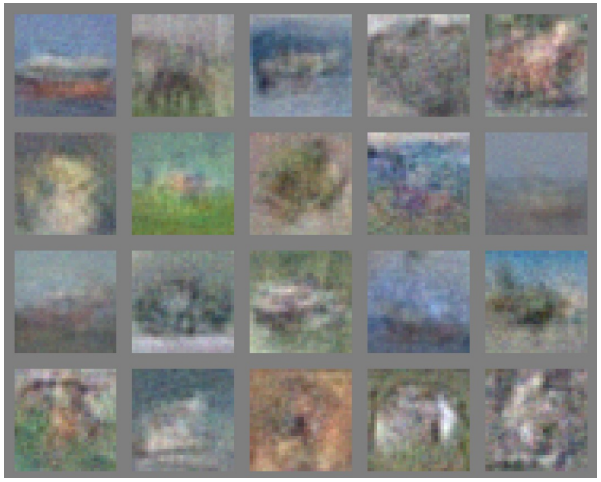
- Training GANs is still very hard
  - Many problems exist
  - Non-convergence
    - The models never converge and worse they become unstable
  - Mode collapse
    - The generator produces a single or limited modes
      - i.e. the images are not as diverse as the true data
- Many tricks exist



Deep Convolutional (DC) GAN

Original GAN – Goodfellow 2014

- Original GAN paper
- Uses only fully connected layers
  - Limited to generating small images
- Discriminator
  - Binary cross entropy loss



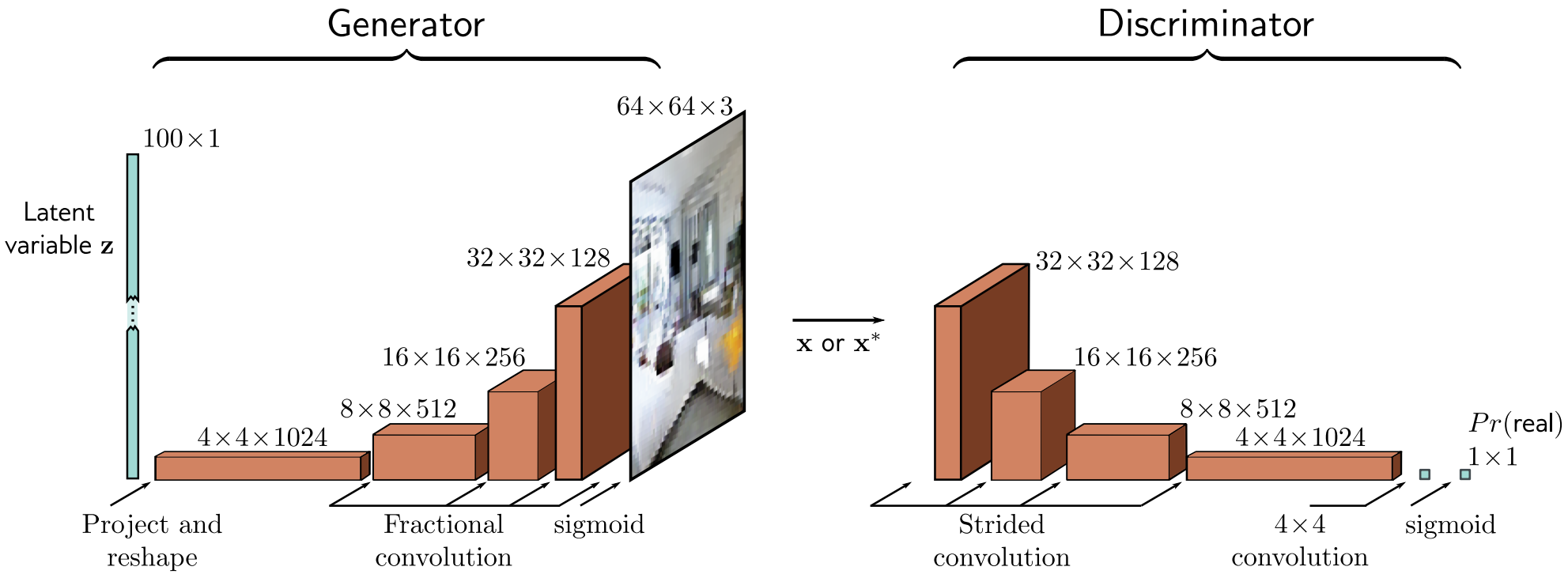
DCGAN 2015/2016

How to do GANs with convolutional layers?

- Replace any pooling layers with strided convolutions (discriminator) and transposed-strided convolutions (generator)
- Use batchnorm in both the generator and the discriminator
- Use LeakyReLU activation in the discriminator for all layers
- Use ReLU activation in generator for all layers except for the output, which uses Tanh
  - Later people recommend using LeakyReLU in both G and D

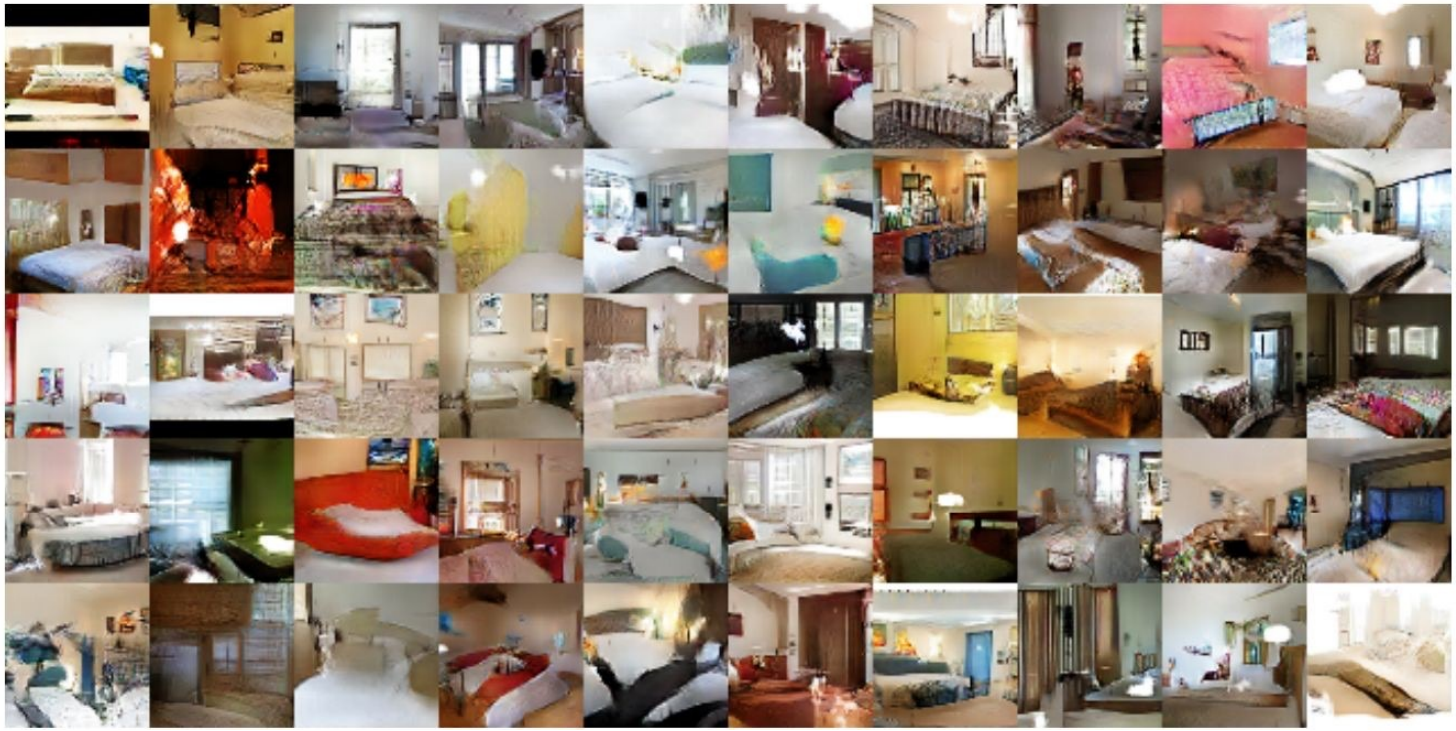
[Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." , ICLR 2016]

Deep Convolutional (DC) GAN



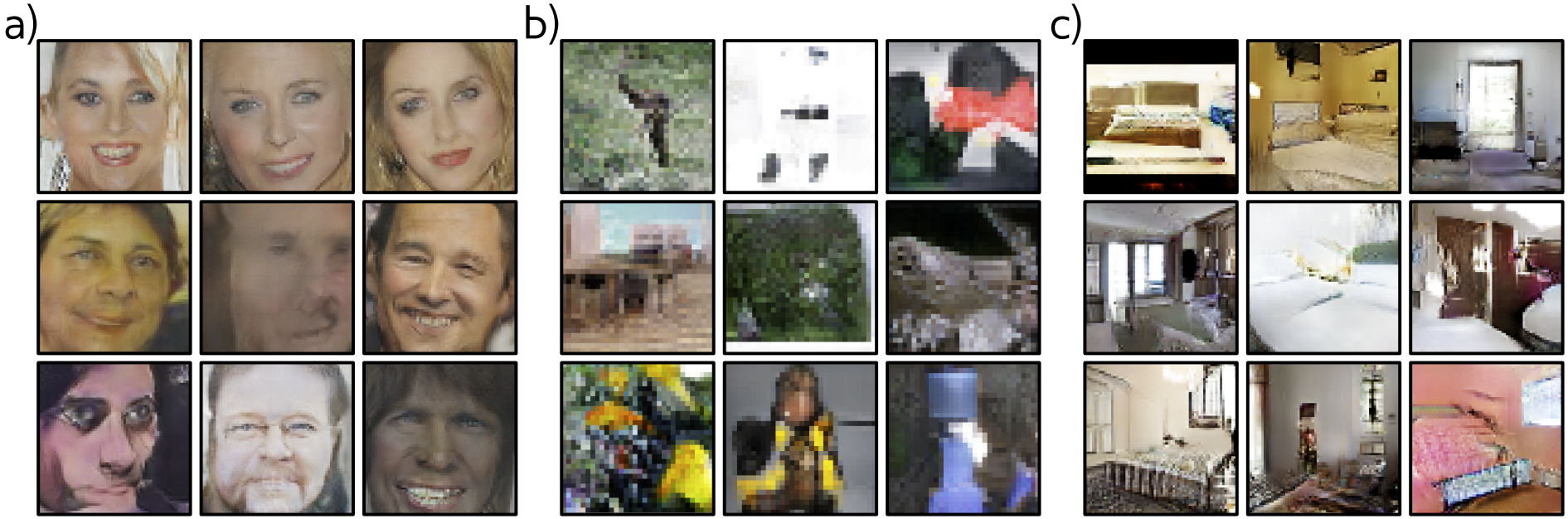
DC GAN Results

Samples from the model look much better!



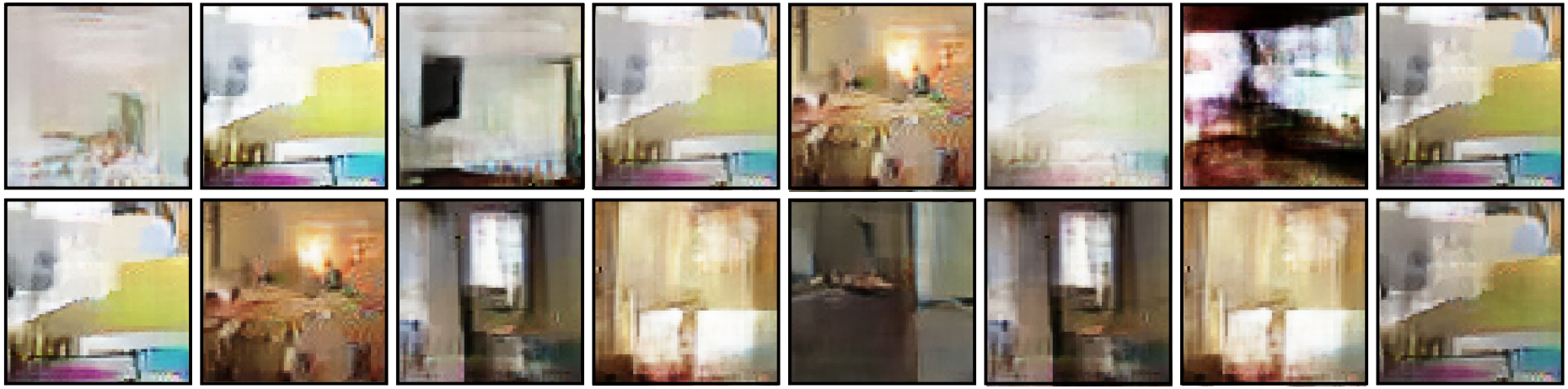
Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

DC GAN Results

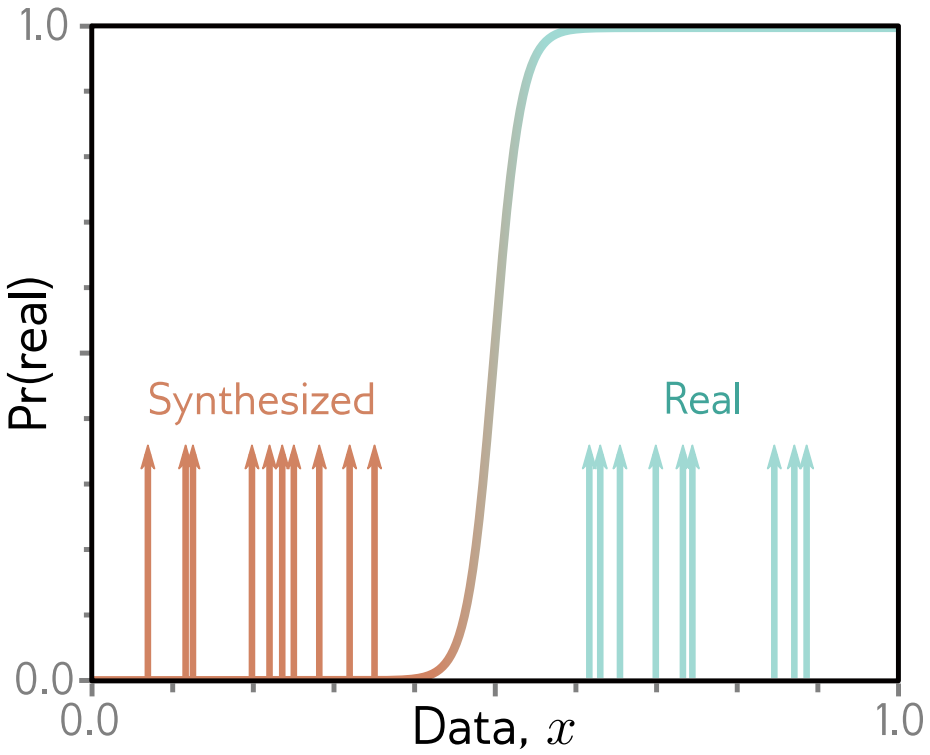


Mode collapse

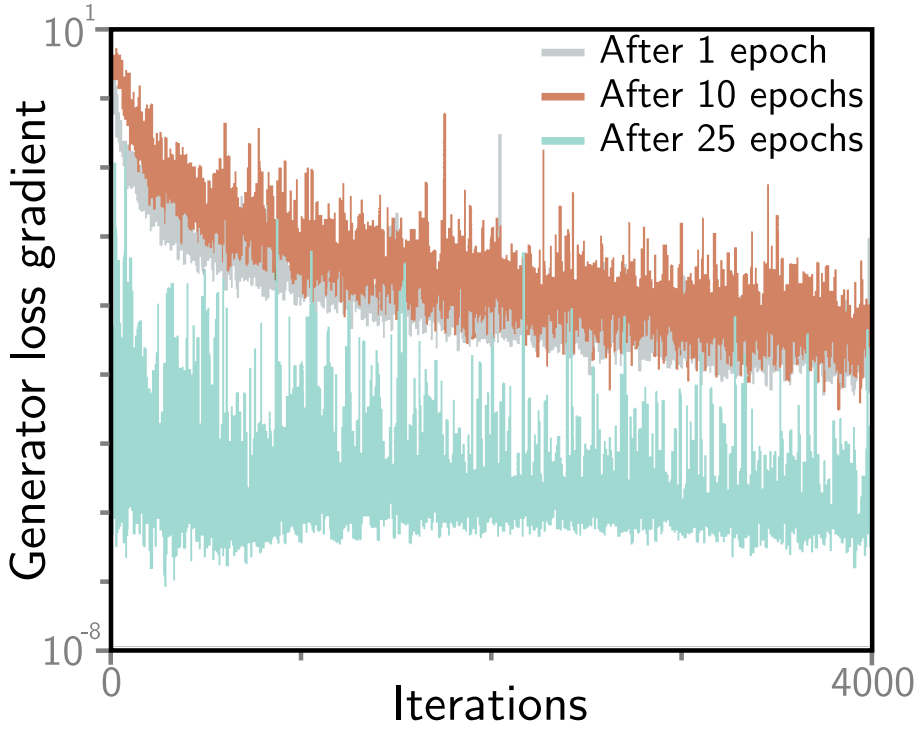
Training GANs is difficult. Often fails.



GAN training instability



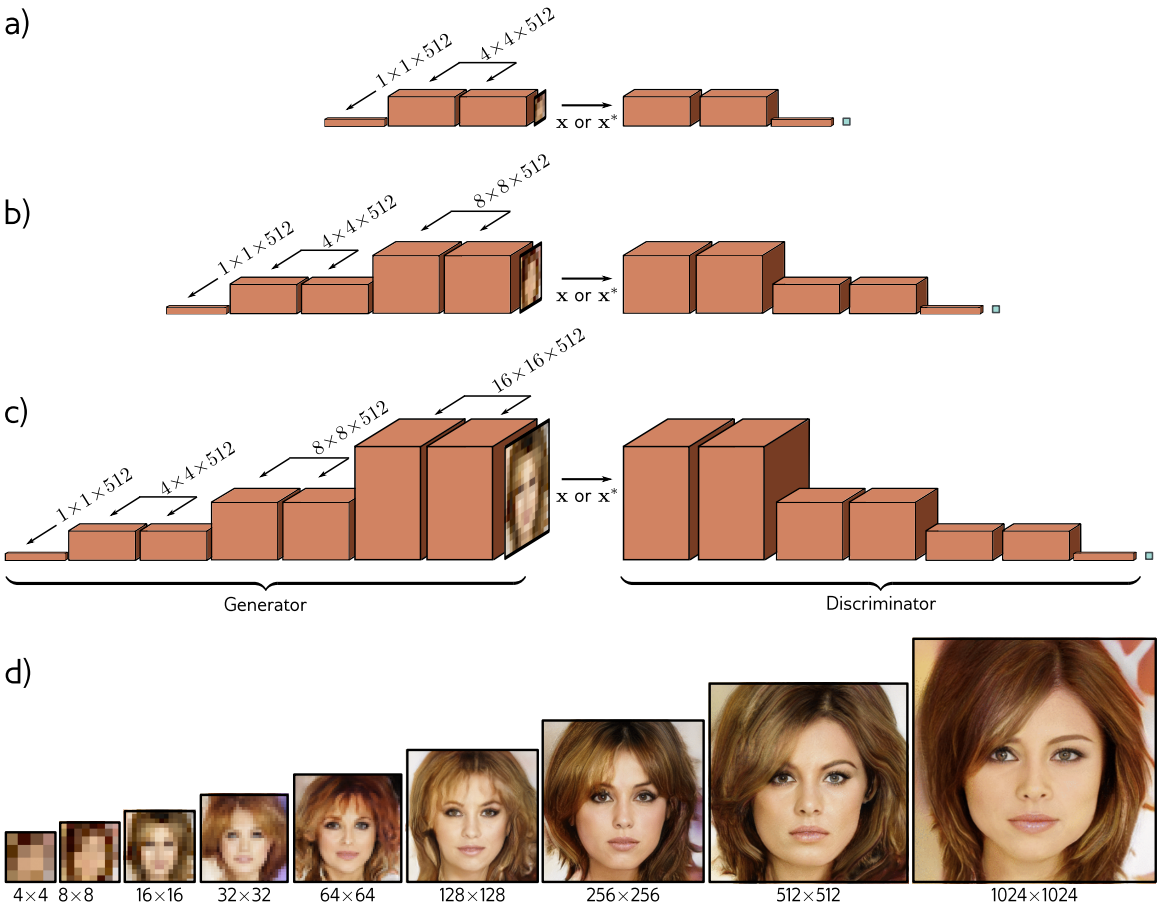
GAN training instability



Fix generator and continue training discriminator, then generator gradients disappear.

Tricks for improving performance

Trick 1: Progressive growing



Trick 2: Minibatch discrimination

- Add in statistics across minibatch as input to discriminator
- Forces generated batch to have similar variation to real batch

Trick 3: Truncation

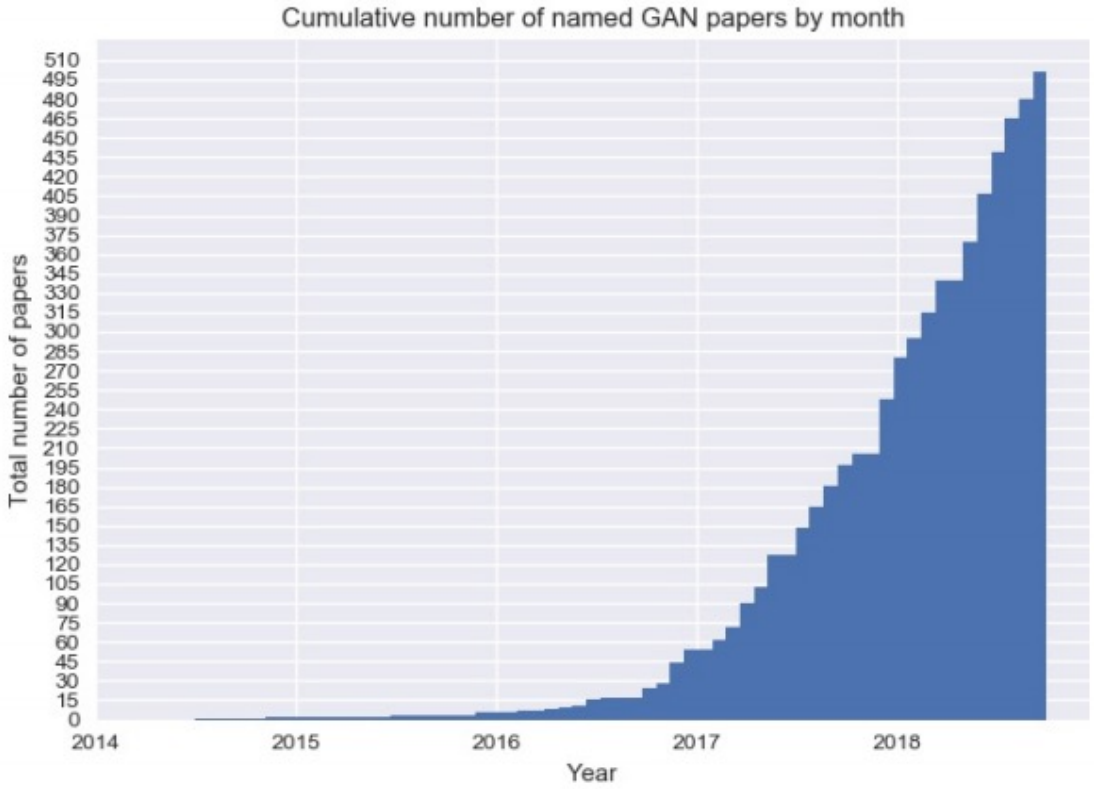
Only choose random values of latent variables that are less than a threshold τ.

Example results

Interpolation

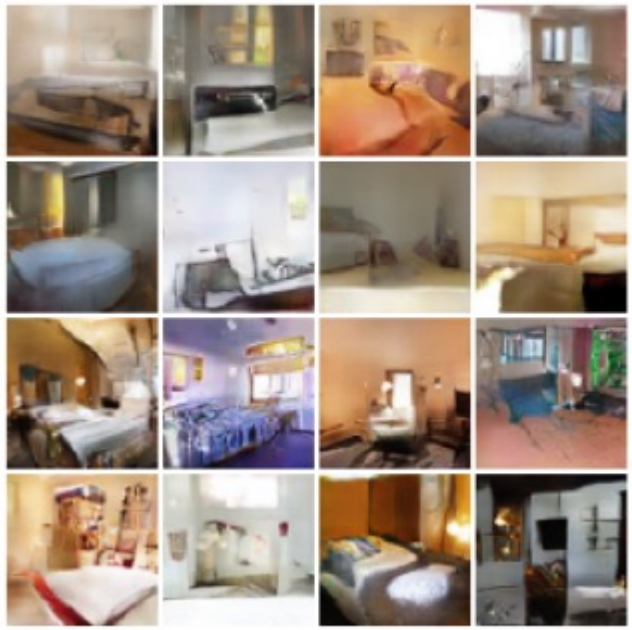
StyleGAN

Generative Adversarial Networks



GAN improvements – Wassestein Loss

Wasserstein GAN (WGAN)

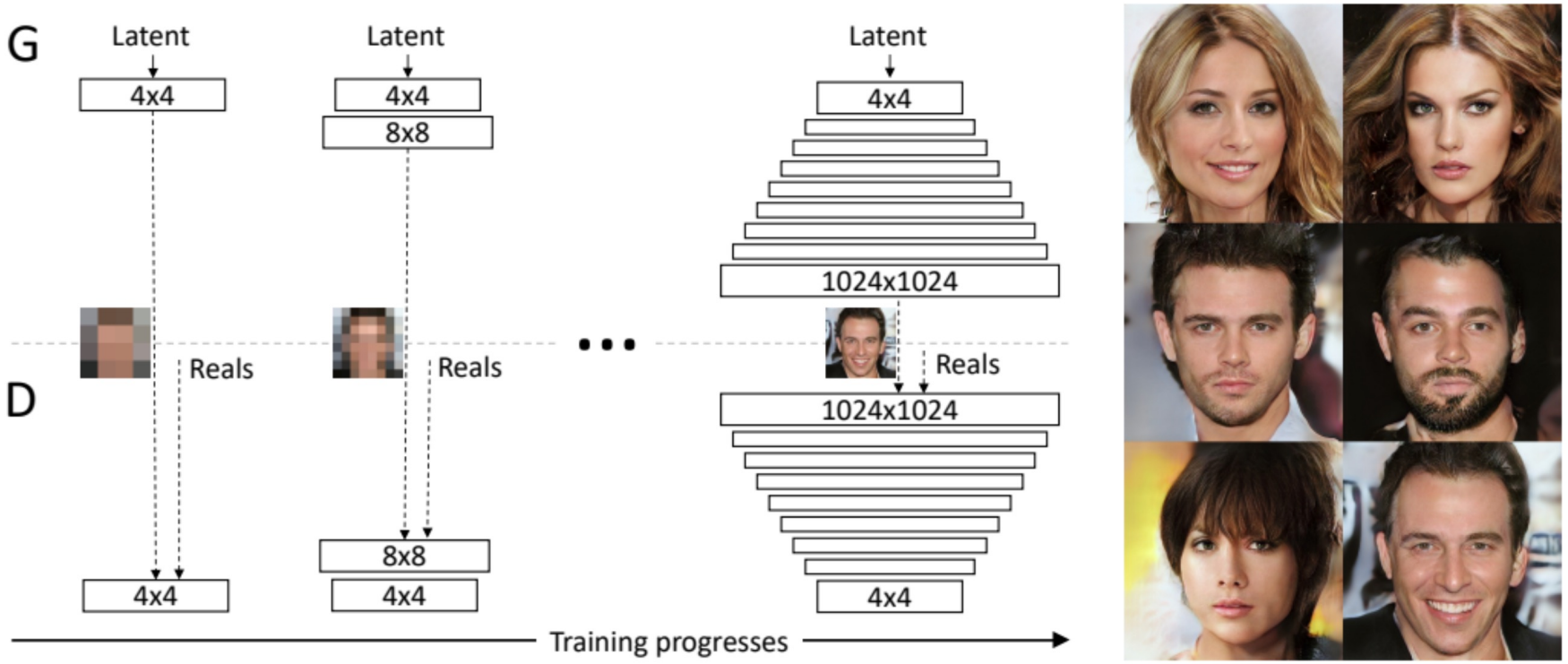


Arjovsky, Chintala, and Bouttou, "Wasserstein GAN", 2017

WGAN with Gradient Penalty (WGAN-GP)

Gulrajani et al, "Improved Training of Wasserstein GANs", NeurIPS 2017

Progressive GAN, Karras et al ICLR 2018



Karras et al, “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

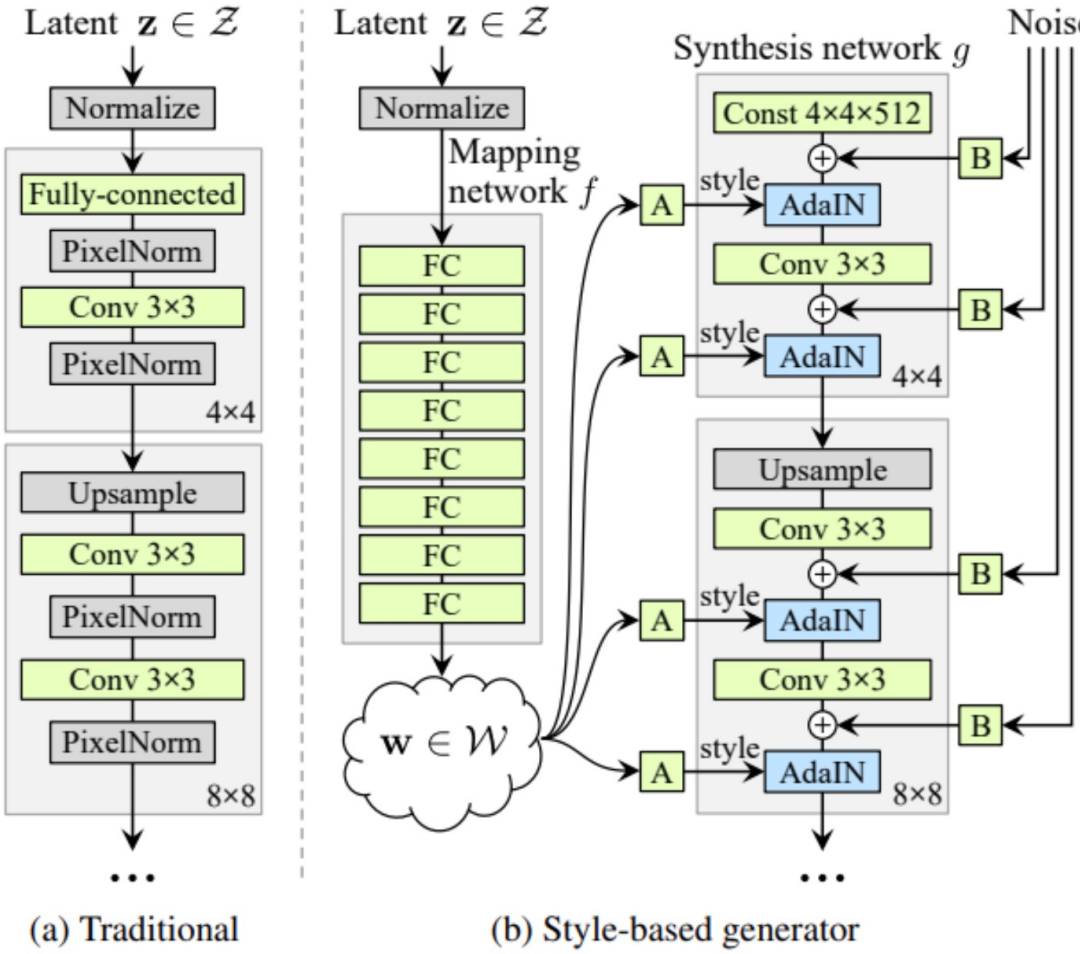
GAN improvements – High Resolution Progressive GAN, Karras et al ICLR 2018

256 x 256 bedrooms

1024 x 1024 faces

Karras et al, “Progressive Growing of GANs for Improved Quality, Stability, and Variation”, ICLR 2018

StyleGAN, Karras et al CVPR 2019



(a) Traditional (b) Style-based generator

$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

AdaIN: [Huang and Belongie, ICCV 2017]
normalization followed by a modulation

Figure 1. While a traditional generator [30] feeds the latent code though the input layer only, we first map the input to an intermediate latent space $\mathcal{W}$, which then controls the generator through adaptive instance normalization (AdaIN) at each convolution layer. Gaussian noise is added after each convolution, before evaluating the nonlinearity. Here "A" stands for a learned affine transform, and "B" applies learned per-channel scaling factors to the noise input. The mapping network f consists of 8 layers and the synthesis network g consists of 18 layers — two for each resolution ($4^2 - 1024^2$). The output of the last layer is converted to RGB using a separate 1×1 convolution, similar to Karras et al. [30]. Our generator has a total of 26.2M trainable parameters, compared to 23.1M in the traditional generator.

GAN improvements – High Resolution StyleGAN, Karras et al CVPR 2019

512 x 384 cars

1024 x 1024 faces

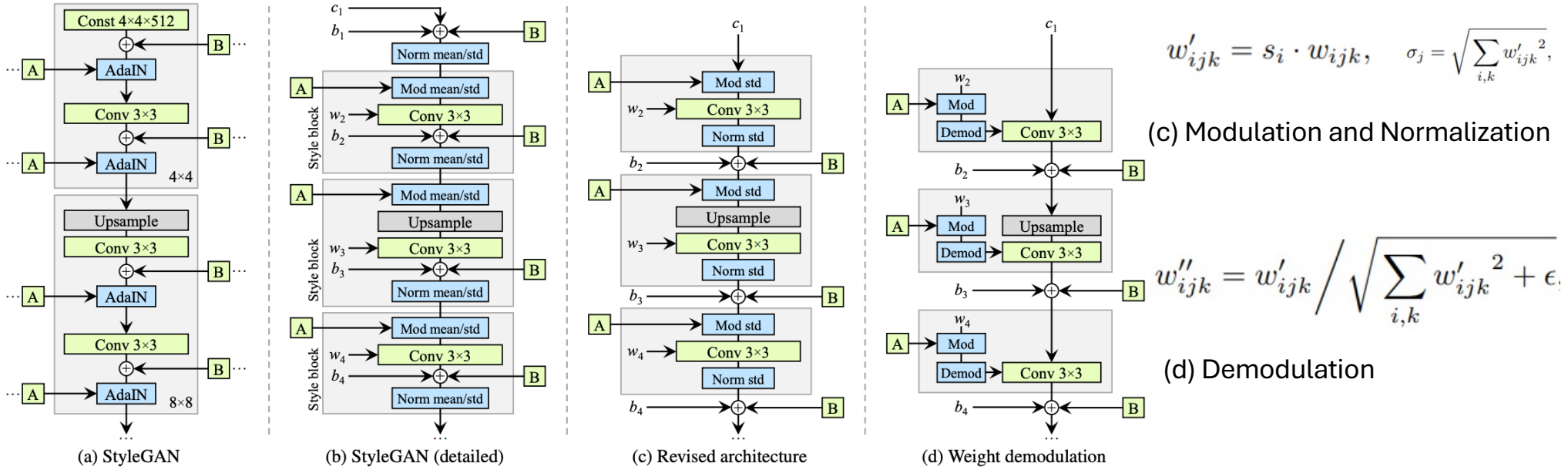
Karras et al, "A Style-Based Generator Architecture for Generative Adversarial Networks", CVPR 2019

StyleGAN Interpolations
Karras et al CVPR 2019

StyleGAN artifacts

Figure 1. Instance normalization causes water droplet -like artifacts in StyleGAN images. These are not always obvious in the generated images, but if we look at the activations inside the generator network, the problem is always there, in all feature maps starting from the 64x64 resolution. It is a systemic problem that plagues all StyleGAN images.

StyleGAN2, Karras et al CVPR 2020



$$w'_{ijk} = s_i \cdot w_{ijk}, \qquad \sigma_j = \sqrt{\sum_{i,k} {w'_{ijk}}^2},$$

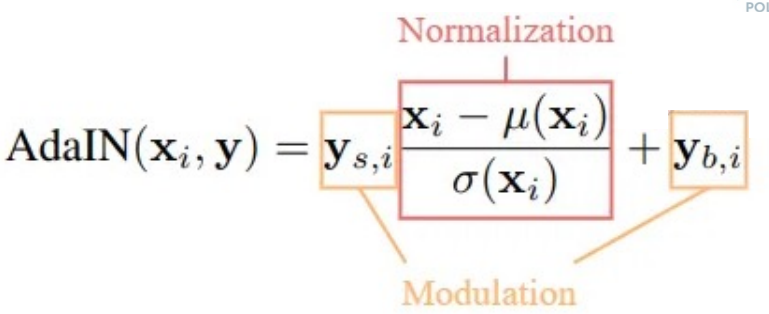
(c) Modulation and Normalization

$$w''_{ijk} = w'_{ijk} \Big/ \sqrt{\sum_{i,k} {w'_{ijk}}^2 + \epsilon}$$

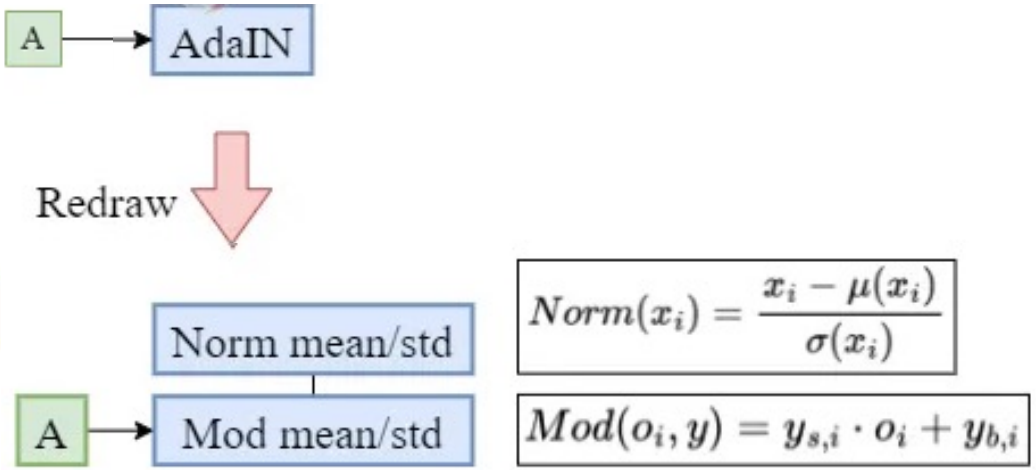
(d) Demodulation

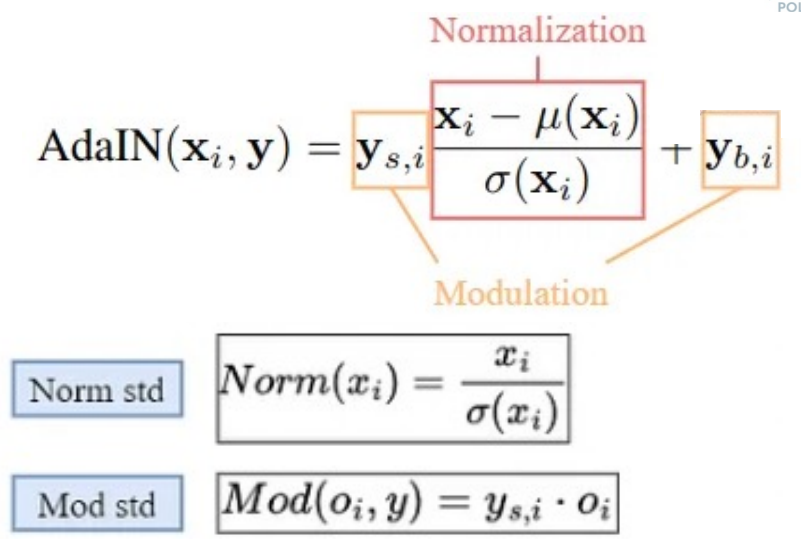
Figure 2. We redesign the architecture of the StyleGAN synthesis network. (a) The original StyleGAN, where A denotes a learned affine transform from $\mathcal{W}$ that produces a style and B is a noise broadcast operation. (b) The same diagram with full detail. Here we have broken the AdaIN to explicit normalization followed by modulation, both operating on the mean and standard deviation per feature map. We have also annotated the learned weights (w), biases (b), and constant input (c), and redrawn the gray boxes so that one style is active per box. The activation function (leaky ReLU) is always applied right after adding the bias. (c) We make several changes to the original architecture that are justified in the main text. We remove some redundant operations at the beginning, move the addition of b and B to be outside active area of a style, and adjust only the standard deviation per feature map. (d) The revised architecture enables us to replace instance normalization with a "demodulation" operation, which we apply to the weights associated with each convolution layer.

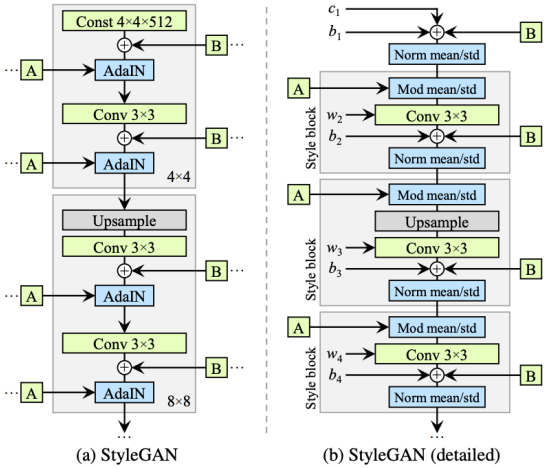
StyleGAN2, Karras et al CVPR 2020



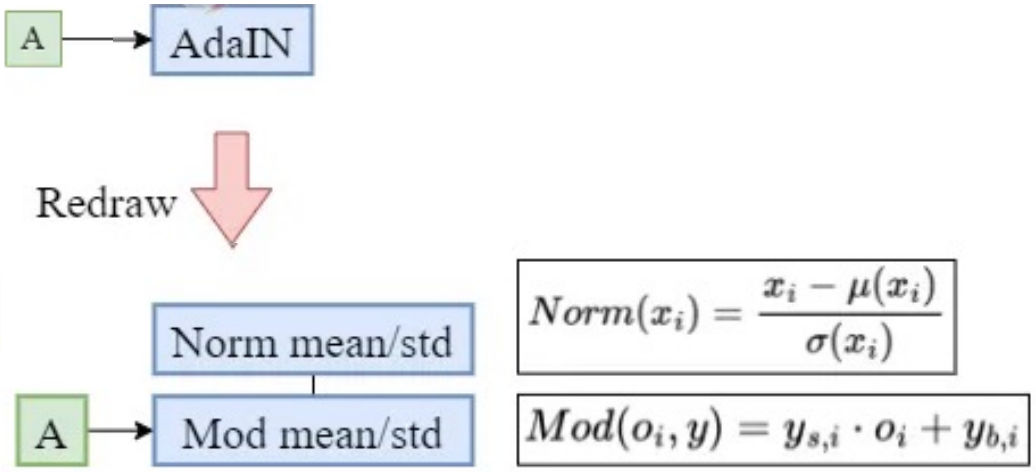
StyleGAN2, Karras et al CVPR 2020

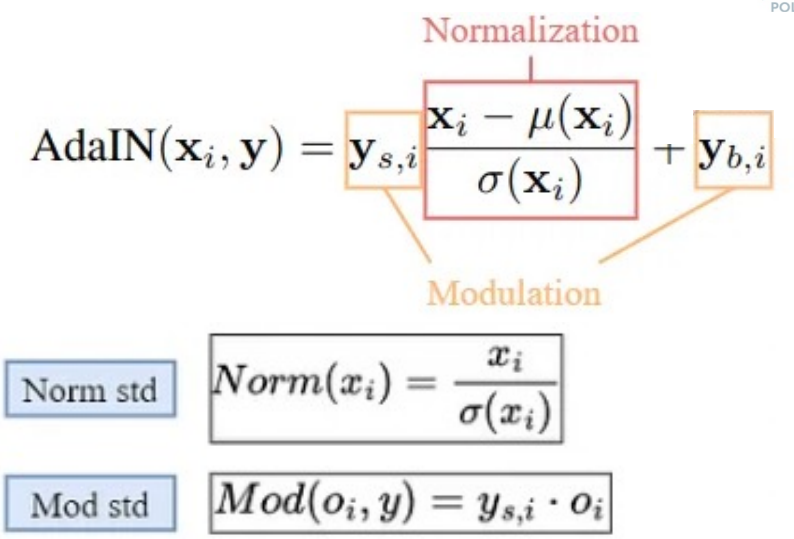


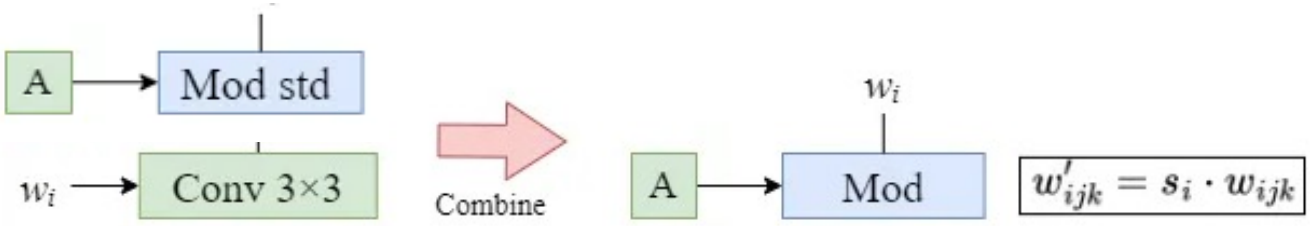


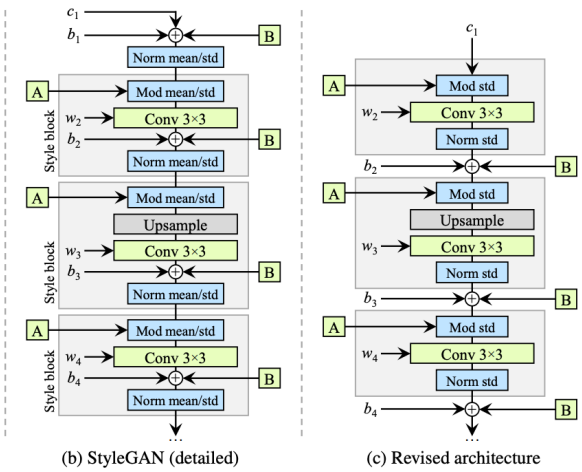


StyleGAN2, Karras et al CVPR 2020

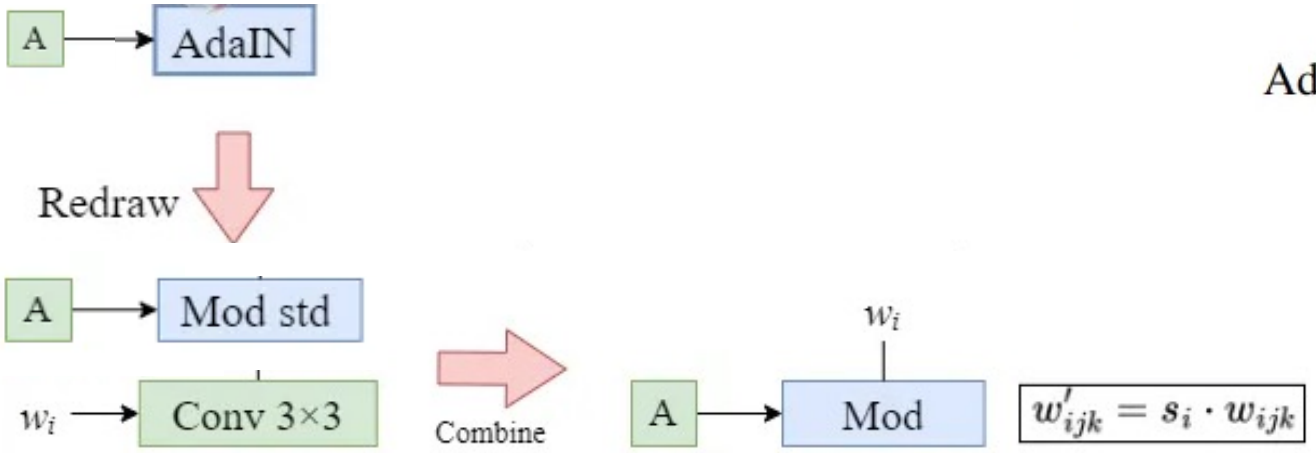


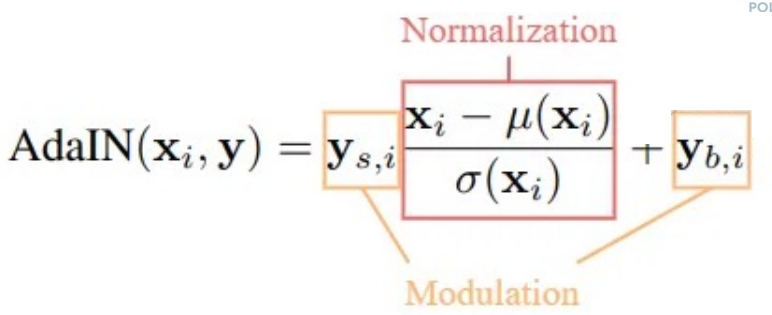


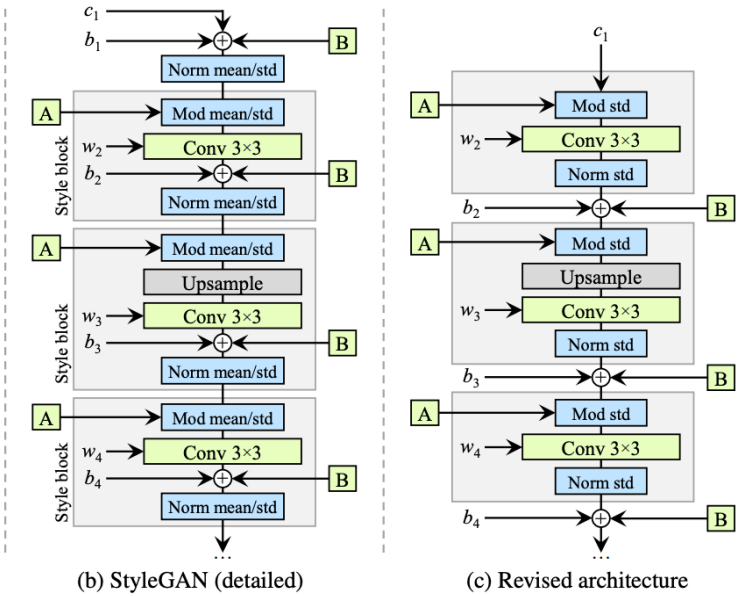




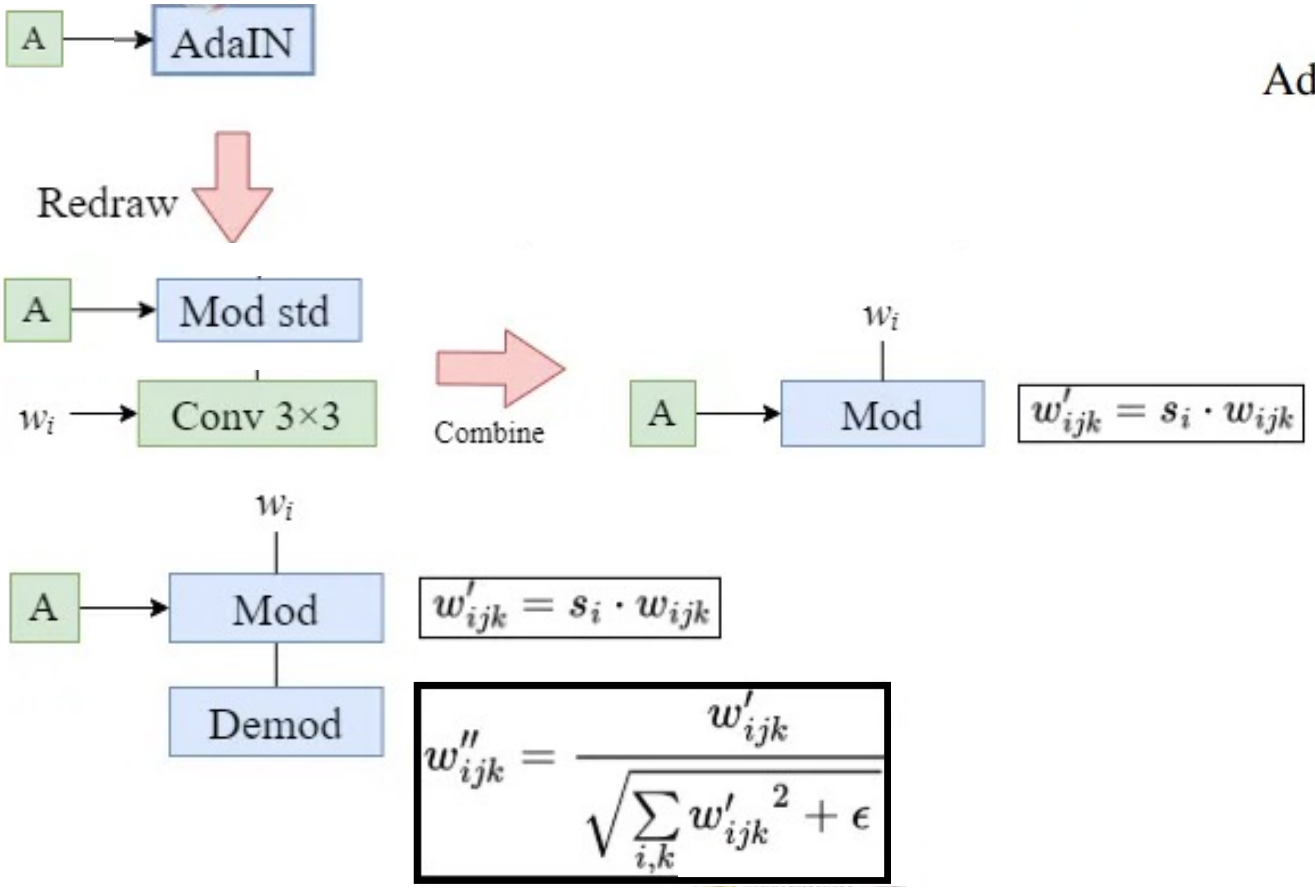
StyleGAN2, Karras et al CVPR 2020

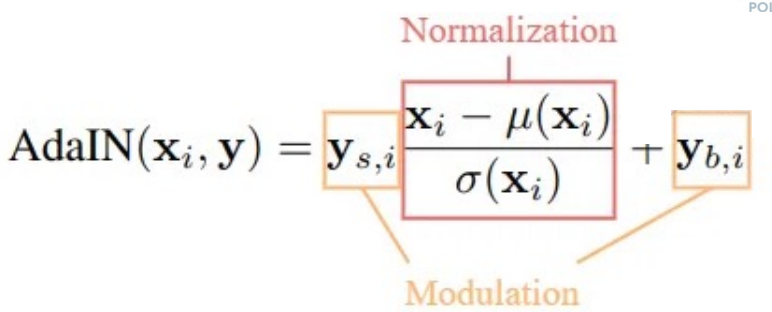


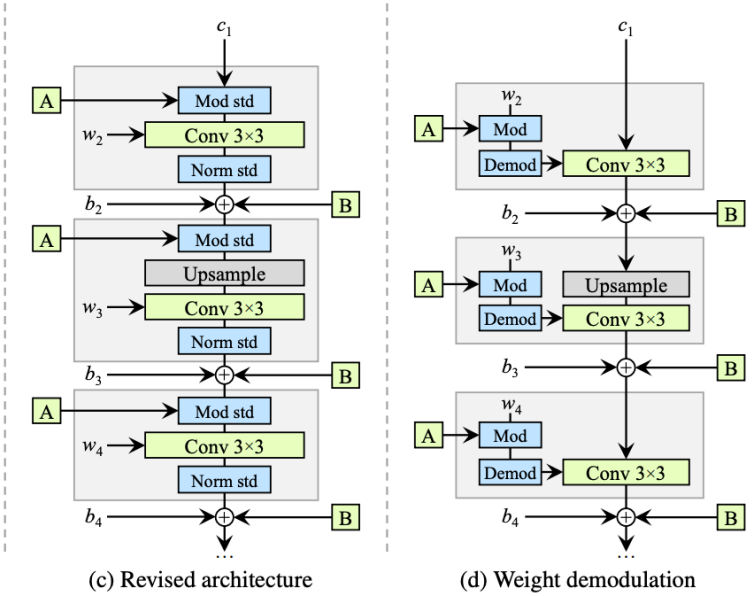




StyleGAN2, Karras et al CVPR 2020

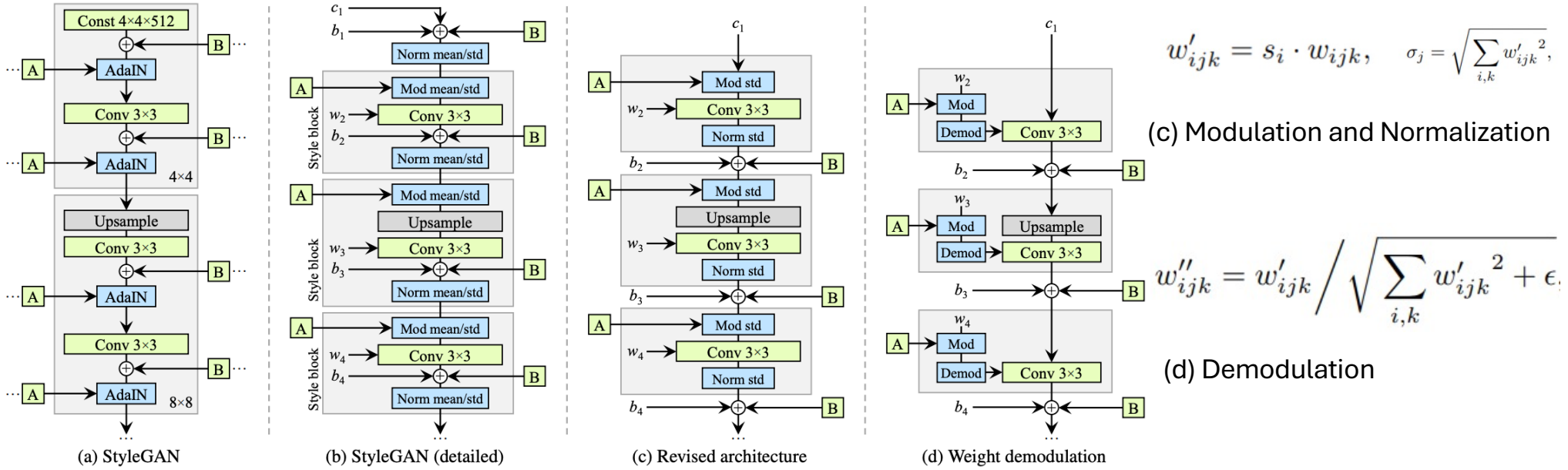






(c) Revised architecture (d) Weight demodulation

StyleGAN2, Karras et al CVPR 2020



$$w'_{ijk} = s_i \cdot w_{ijk}, \qquad \sigma_j = \sqrt{\sum_{i,k} {w'_{ijk}}^2},$$

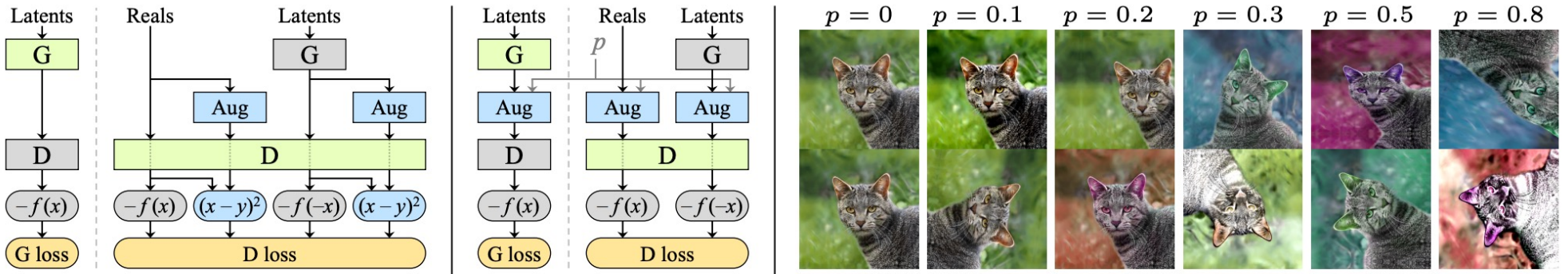
(c) Modulation and Normalization

$$w''_{ijk} = w'_{ijk} \Big/ \sqrt{\sum_{i,k} {w'_{ijk}}^2 + \epsilon}$$

(d) Demodulation

Figure 2. We redesign the architecture of the StyleGAN synthesis network. (a) The original StyleGAN, where A denotes a learned affine transform from $\mathcal{W}$ that produces a style and B is a noise broadcast operation. (b) The same diagram with full detail. Here we have broken the AdaIN to explicit normalization followed by modulation, both operating on the mean and standard deviation per feature map. We have also annotated the learned weights (w), biases (b), and constant input (c), and redrawn the gray boxes so that one style is active per box. The activation function (leaky ReLU) is always applied right after adding the bias. (c) We make several changes to the original architecture that are justified in the main text. We remove some redundant operations at the beginning, move the addition of b and B to be outside active area of a style, and adjust only the standard deviation per feature map. (d) The revised architecture enables us to replace instance normalization with a "demodulation" operation, which we apply to the weights associated with each convolution layer.

StyleGAN2 - ADA, Karras et al NeurIPS 2020

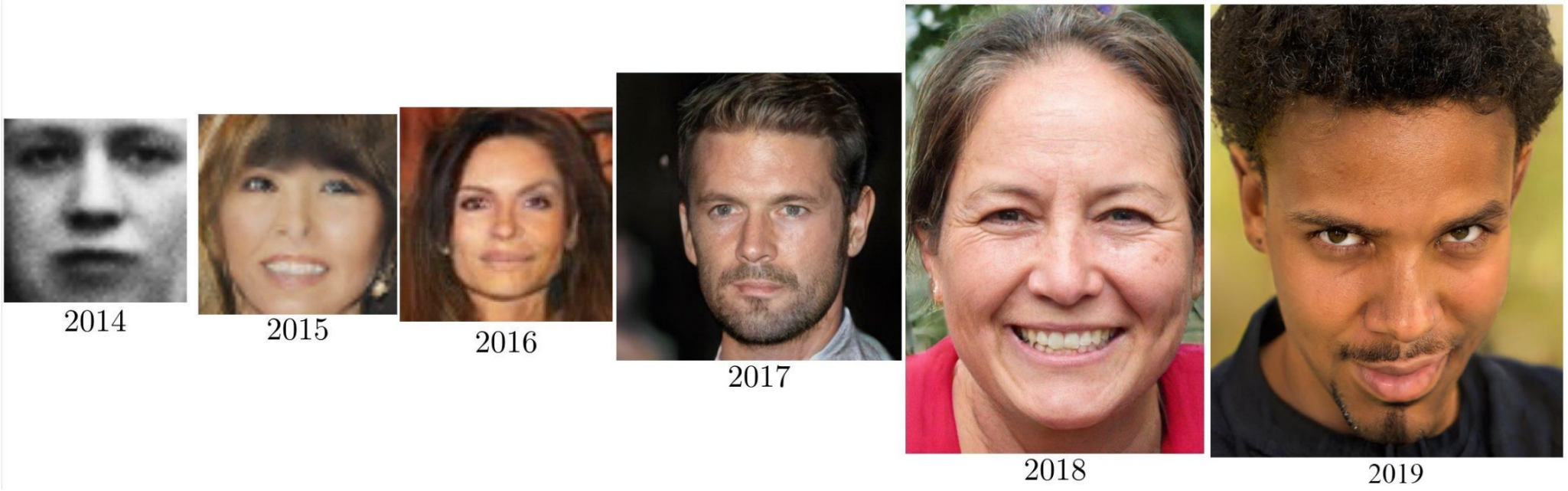


(a) bCR (previous work) (b) Ours (c) Effect of augmentation probability p

Figure 2: (a,b) Flowcharts for balanced consistency regularization (bCR) [43] and our stochastic discriminator augmentations. The blue elements highlight operations related to augmentations, while the rest implement standard GAN training with generator G and discriminator D [12]. The orange elements indicate the loss function and the green boxes mark the network being trained. We use the non-saturating logistic loss [12] $f(x) = \log(\text{sigmoid}(x))$. (c) We apply a diverse set of augmentations to every image that the discriminator sees, controlled by an augmentation probability p.

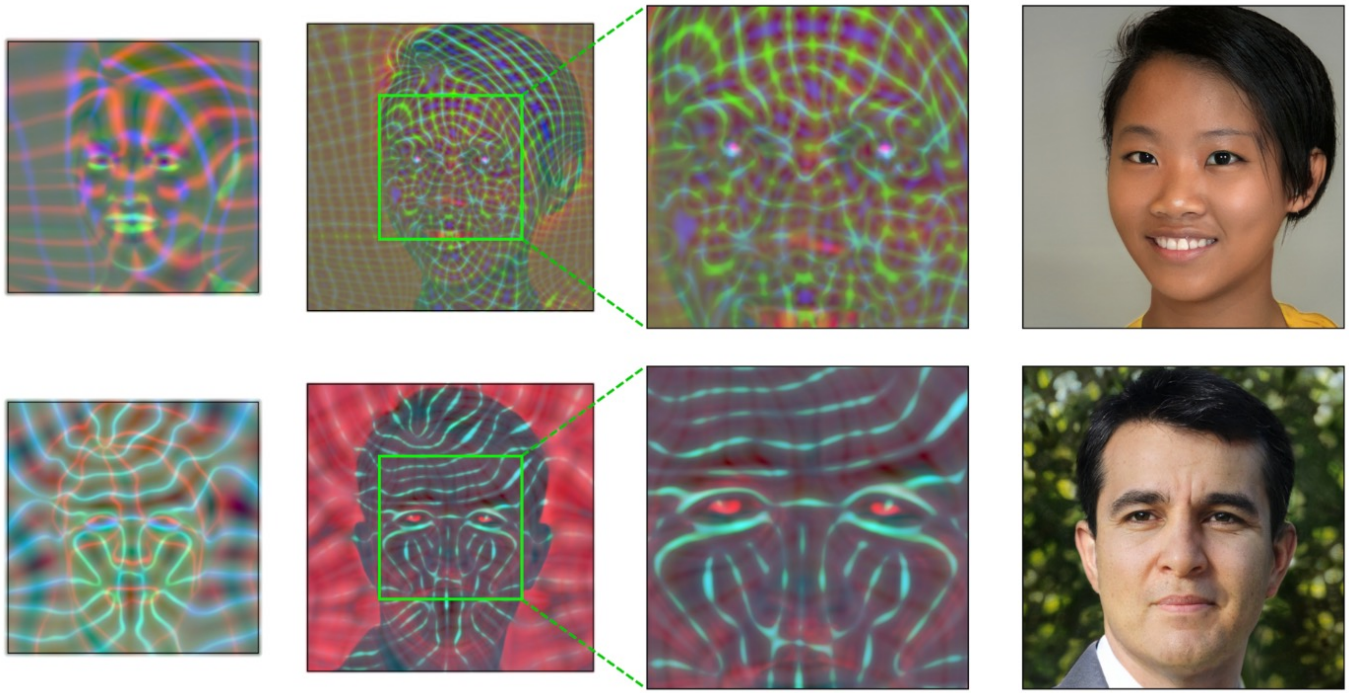
https://github.com/NVlabs/stylegan2-ada

GAN progress on face generation



Source: https://twitter.com/goodfellow_ian/status/1084973596236144640/
+StyleGANv2

StyleGAN3, Karras NeurIPS 2021



https://nvlabs.github.io/stylegan3/

Examples

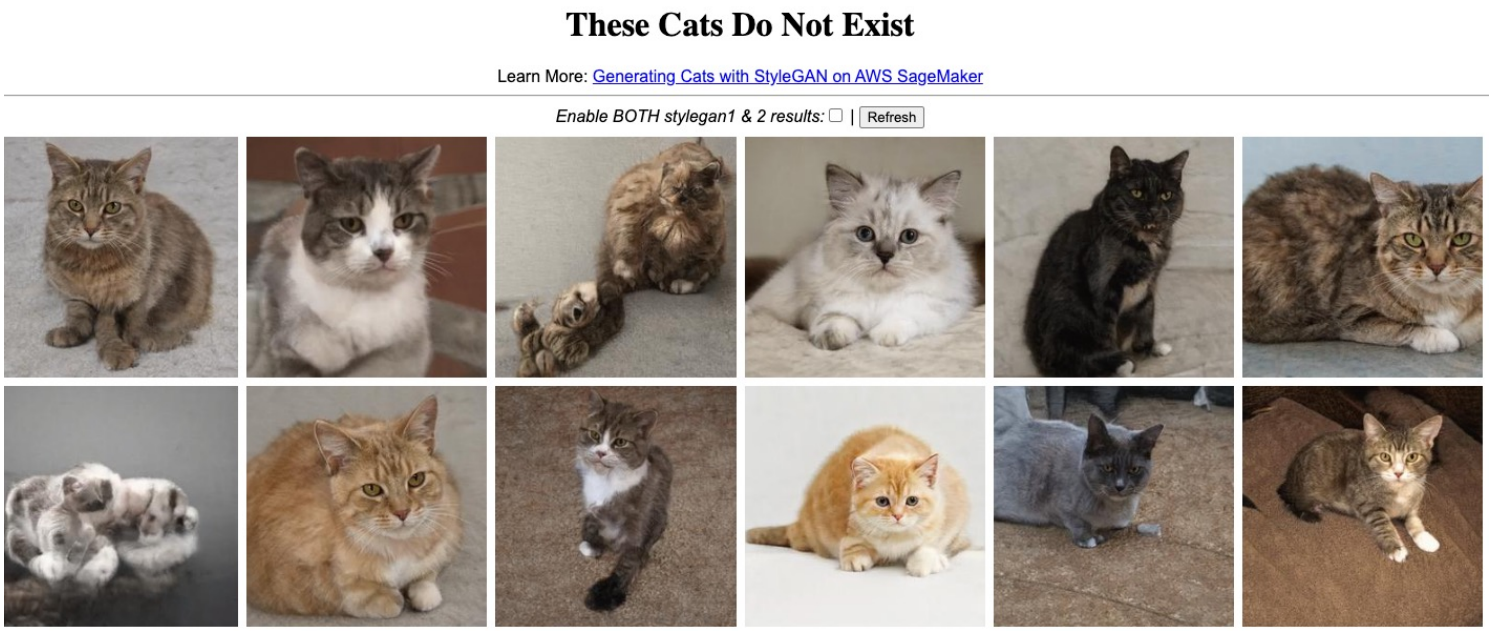
- http://www.whichfaceisreal.com

You are **correct**. The image on the left is real.

Play again.

Examples

- https://thiscatdoesnotexist.com/



Which face is real?

- Cool website
  - http://www.whichfaceisreal.com

- Others
  - https://thisrentaldoesnotexist.com/
  - https://thiscatdoesnotexist.com/
  - https://thishorsedoesnotexist.com/
  - https://www.thiswaifudoesnotexist.net/

Metrics

Data augmentation in GANs

- Data augmentation is helpful during GAN training, if done to **real** AND **fake** images
  - Need differentiable augmentations
  - https://github.com/mit-han-lab/data-efficient-gans

[Zhao Zhengli et al. "Image Augmentations for GAN Training." arXiv 2006.02595, 2020]

FID - Frechet Inception Distance

- A metric attempting to quantitatively evaluate the quality of generated images
- Main Idea
  - Use features learned for classification, and see how similar they are for real and generated images
- How to:
  - Take the 2048 dimensional output of the global pooling layer in Inception v3
  - Compute the mean and covariance matrix of the features for real and fake images
  - Calculate the Frechet distance between the real and generated images

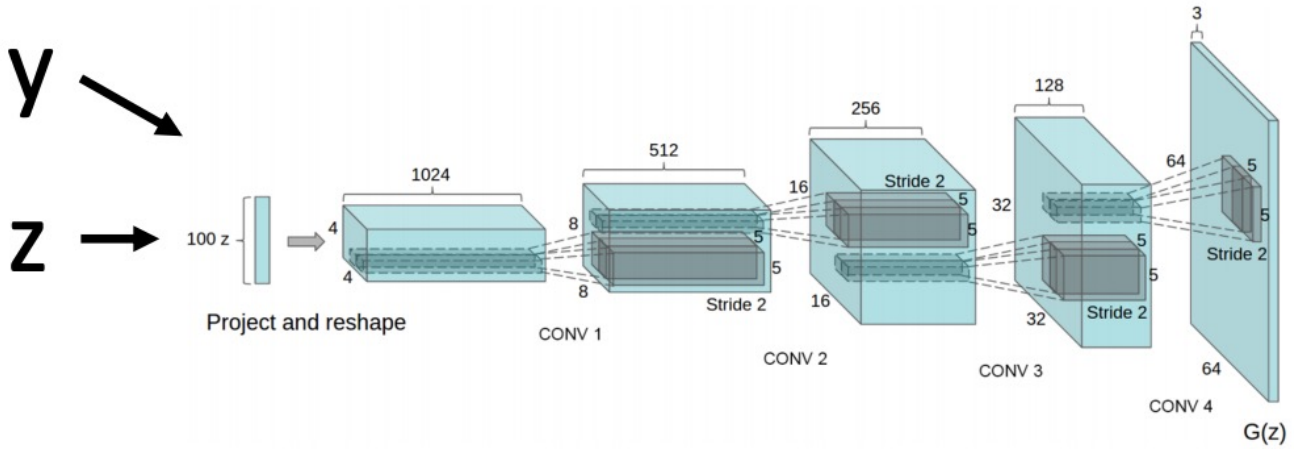
$$\|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \operatorname{Tr}\left(\boldsymbol{C} + \boldsymbol{C}_w - 2(\boldsymbol{C}\boldsymbol{C}_w)^{1/2}\right)$$

[Mercel. Martin, de al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In NeurIPS 2017]

Conditional GANs

Conditional GANs

- Conditional GAN: learn p(x|y) instead of p(x)
  - Make generator and discriminator both take label y as an additional input



Recall: Batch Normalization

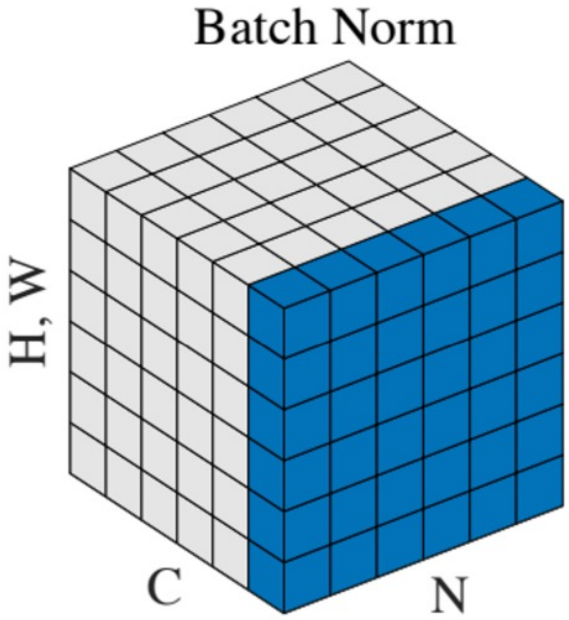
- A way to keep the weights in the gaussian range
- Idea: "Normalize" the outputs of a layer so they have zero mean and unit variance
  - Helps reduce "internal covariate shift"
  - improves optimization
- Before non-linearity – run batch normalization empirically on the specific batch

$$\hat{x} = \frac{x - E[x]}{\sqrt{Var[x]}}$$

This is a differentiable function, so we can use it as an operator in our networks and backprop through it!

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [Ioffe and Szegedy 2015]

Recall: Batch Normalization



[Wu and He, “Group Normalization”, ECCV 2018]

Recall: Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

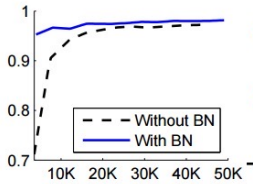
$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

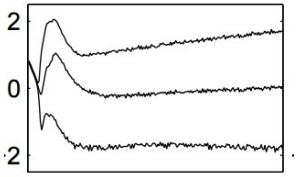
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift [Ioffe and Szegedy 2015]

Test accuracy

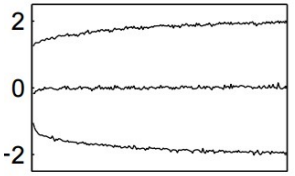


(a)

Responses of different layers over iterations



(b) Without BN



(c) With BN

Conditional GANs

Batch Normalization

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

$$y_{i,j} = \gamma_j \hat{x}_{i,j} + \beta_j$$

Learn a separate scale and shift for each different label y

Conditional Batch Normalization

$$\mu_j = \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$

$$\sigma_j^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$

$$\hat{x}_{i,j} = \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \epsilon}}$$

$$y_{i,j} = \boldsymbol{\gamma_j^y} \hat{x}_{i,j} + \boldsymbol{\beta_j^y}$$

Dumoulin et al, “A learned representation for artistic style”, ICLR 2017

Conditional GANs

Miyato et al, “Spectral Normalization for Generative Adversarial Networks”, ICLR 2018

Conditional GANs
BigGAN

Brock et al, "Large Scale GAN Training for High Fidelity Natural Image Synthesis", ICLR 2019

512x512 images on ImageNet

Conditional GANs
More than labels: Text to Image

Zhang et al, "StackGAN++: Realistic Image Synthesis with Stacked Generative Adversarial Networks.", TPAMI 2018
Zhang et al, "StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks.", ICCV 2017
Reed et al, "Generative Adversarial Text-to-Image Synthesis", ICML 2016

Image Translation

Image translation: Pix2Pix

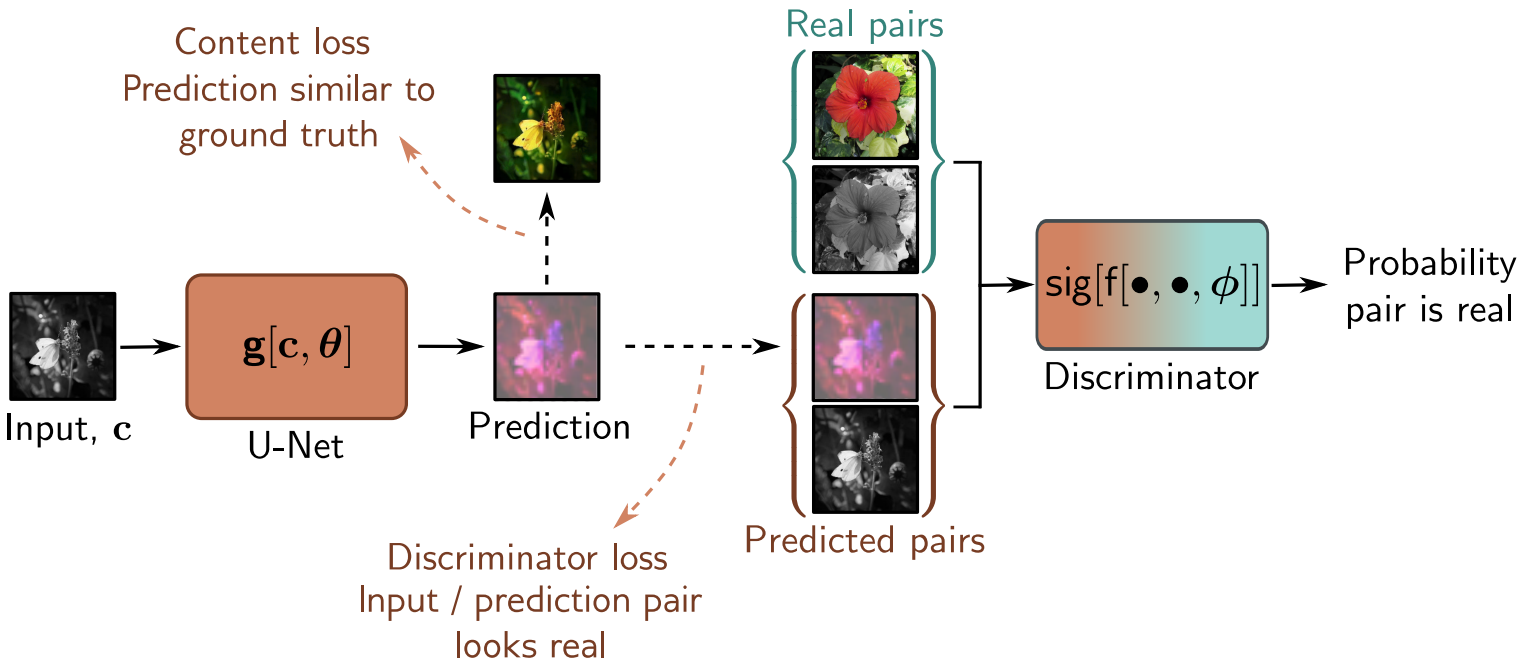


Image translation: Pix2Pix

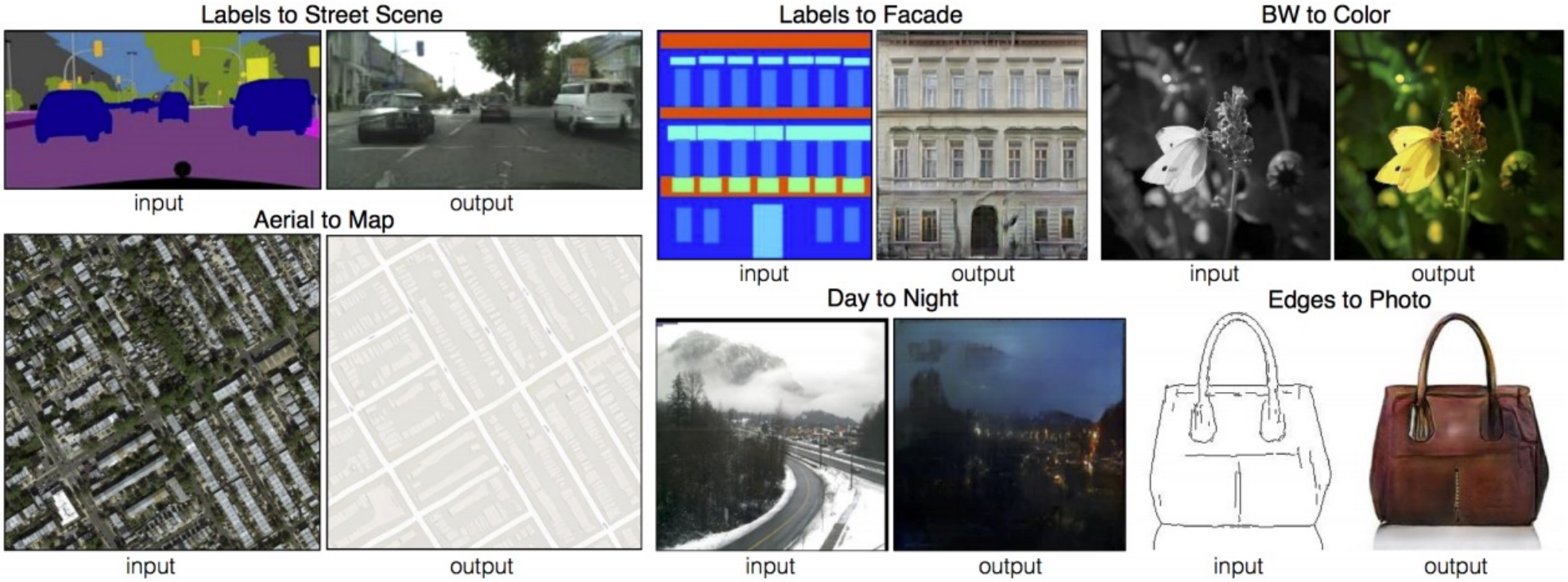


Image translation: SRGAN

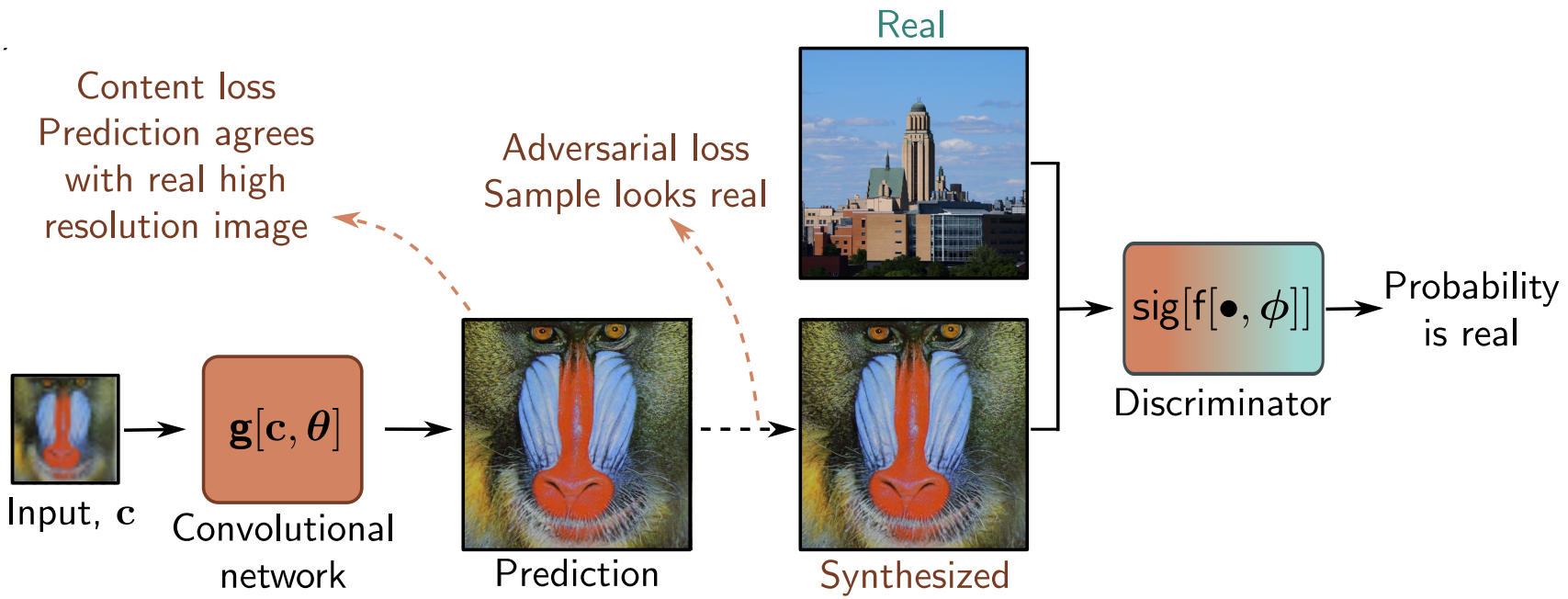


Image translation: SRGAN

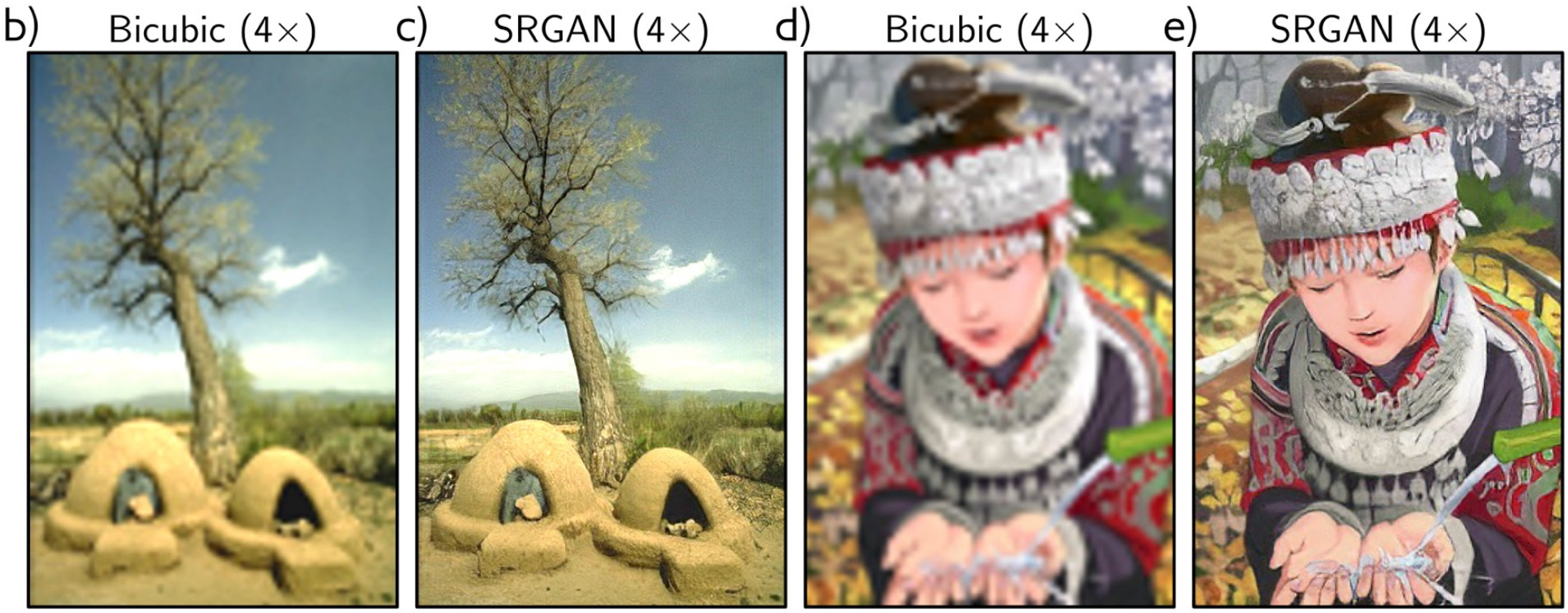


Image translation: CycleGAN

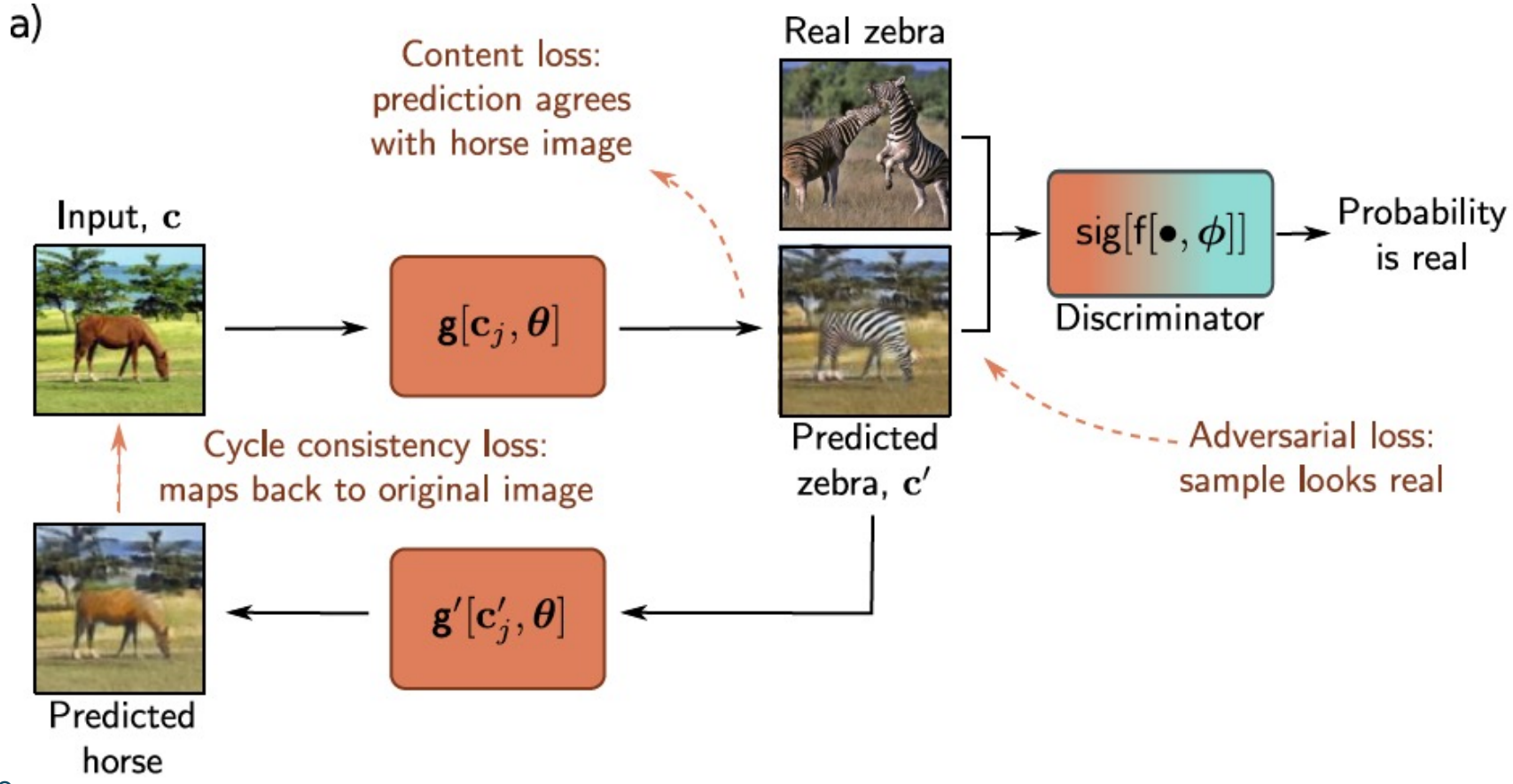


Image translation: CycleGAN

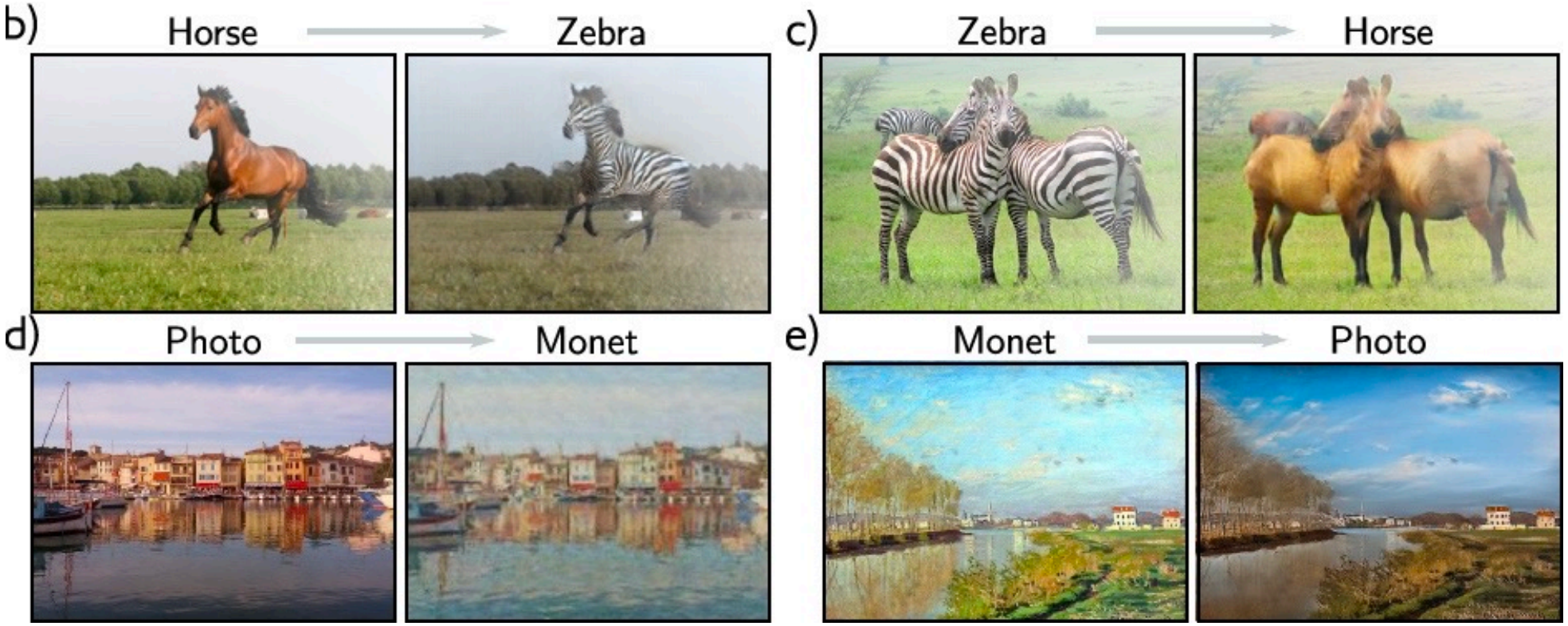
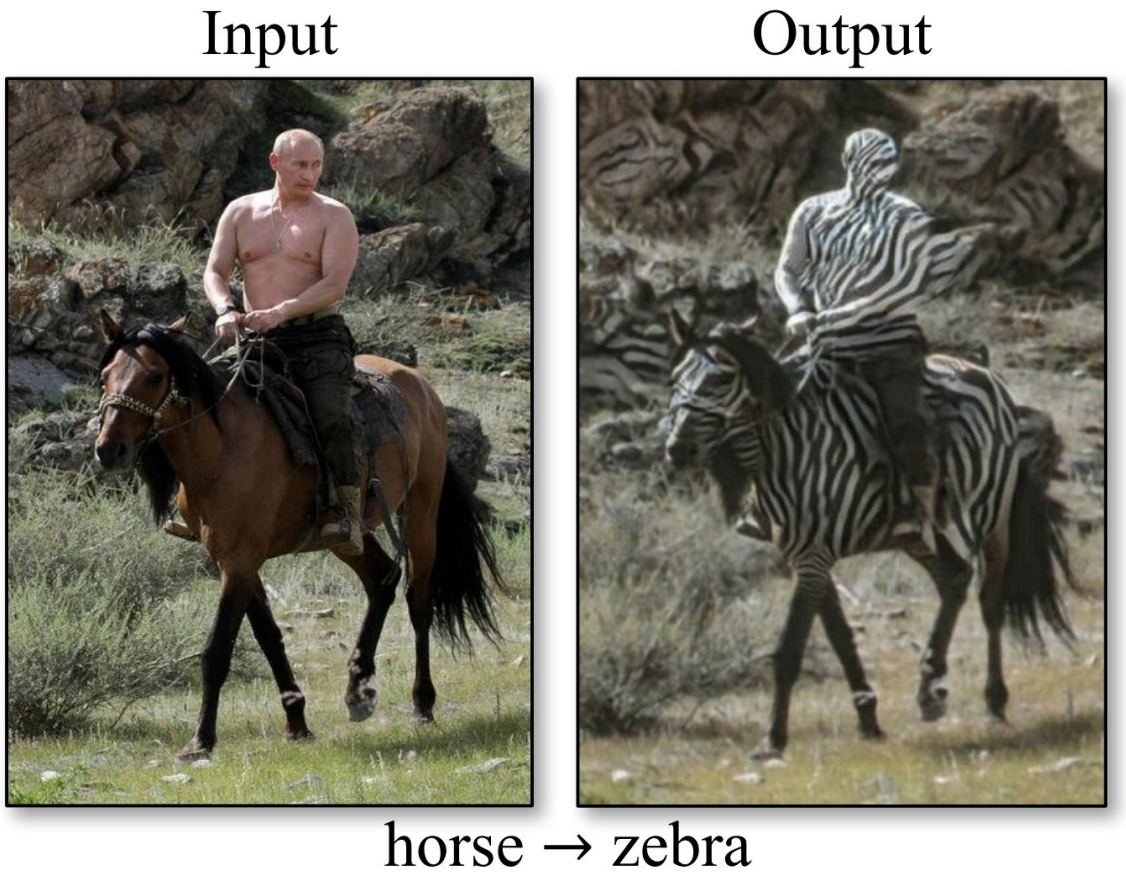


Image translation: CycleGAN



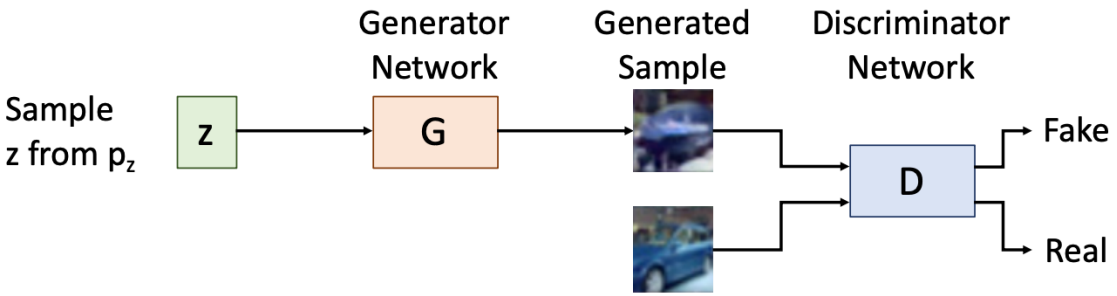
Conditional GANs
Unpaired Image-to-Image Translation: CycleGAN

CycleGAN

Jun-Yan Zhu, Taesung Park et. al. ICCV 2017
https://www.youtube.com/watch?v=9reHvktowLY

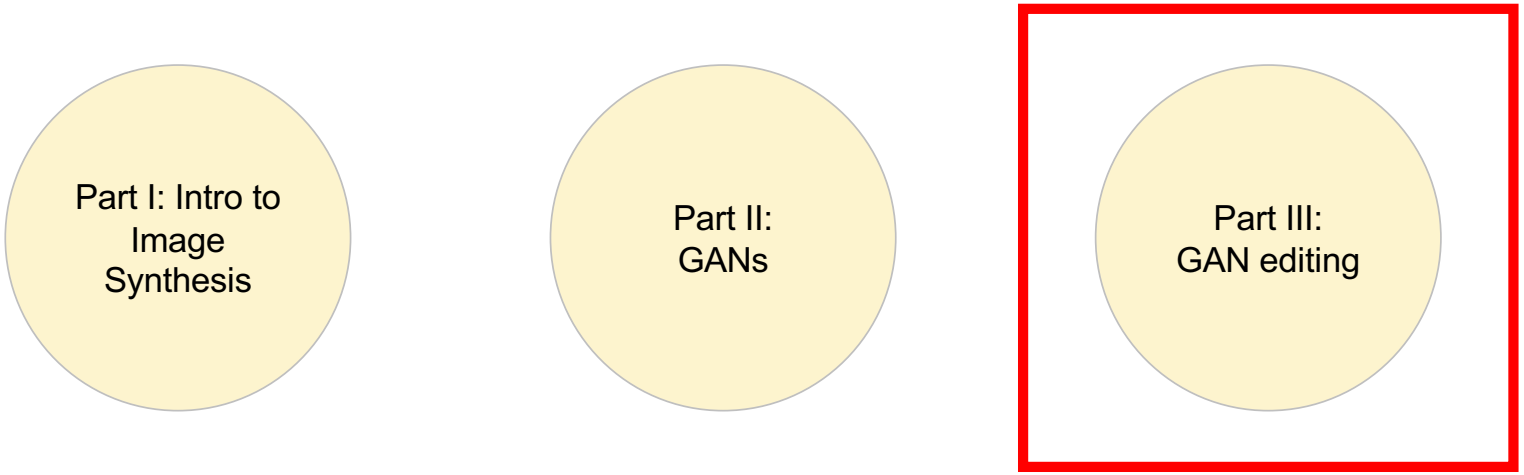
GAN summary

Jointly train two networks:
Discriminator: Classify data as real or fake
Generator: Generate data that fools the discriminator



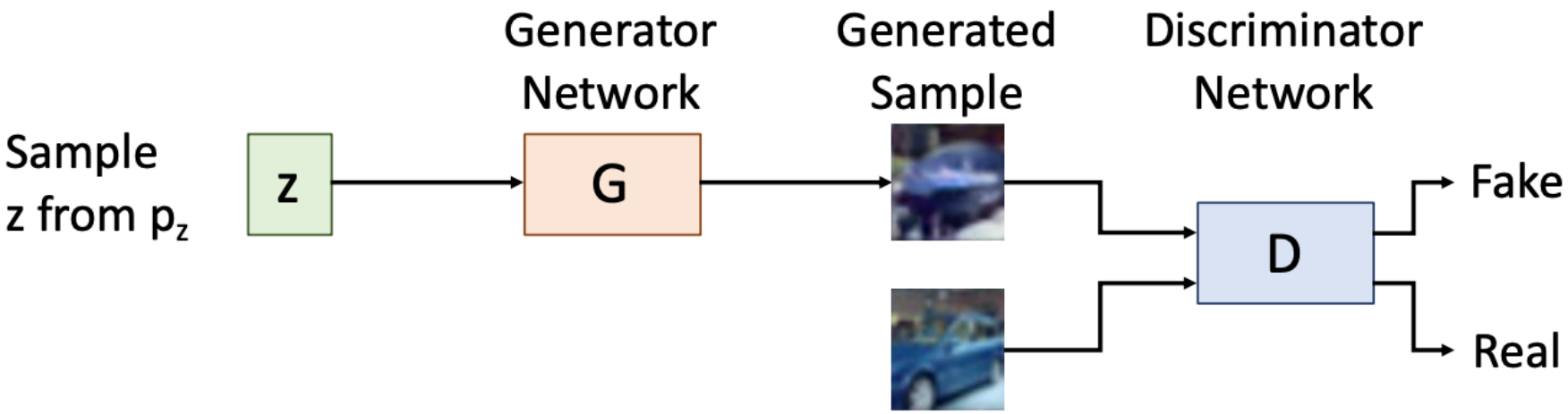
Under some assumptions, generator converges to true data distribution
Many applications! Very active area of research!

Today's lecture



Slides adapted from many resources:
[Fei-Fei Li & Andrej Karpathy & Justin Johnson & Ross Girshick & VGG]

Latent space of a GAN



Part III: GAN editing

- Interpolation in latent space
- GAN Math
- GAN inversion
- Learn and apply latent directions
- CLIP + StyleGAN

Part III: GAN editing

- **Interpolation in latent space**
- GAN Math
- GAN inversion
- Learn and apply latent directions
- CLIP + StyleGAN

Generate random images

https://github.com/NVlabs/stylegan2-ada

```
python generate.py --outdir=out --trunc=1 --seeds=666-700 --
network=https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada-pytorch/pretrained/ffhq.pkl
```

GAN Interpolation

Interpolating between points in latent z space

GAN Interpolation

Interpolating between points in latent z space

[Radford, Alec, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." , ICLR 2016]

Latent space of a GAN

z_1 $a*z_1+(1-a)*z_2$ z_2

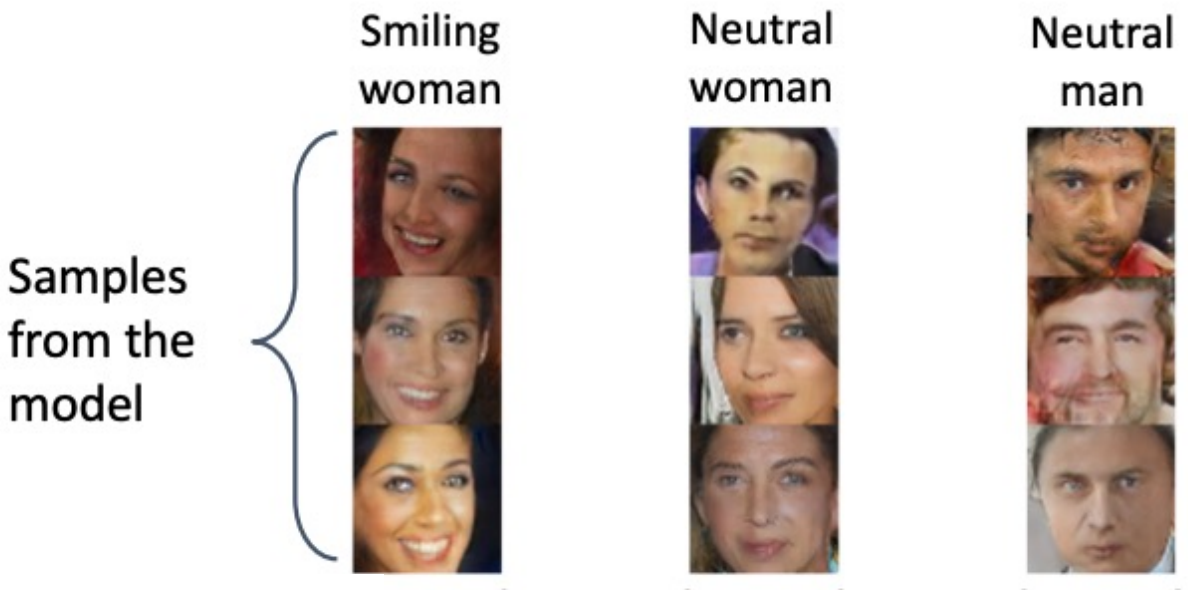
https://github.com/NVlabs/stylegan2-ada

```
python generate.py --outdir=out --dlatents=out/dlatents.npz --network=https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada/pretrained/ffhq.pkl
```

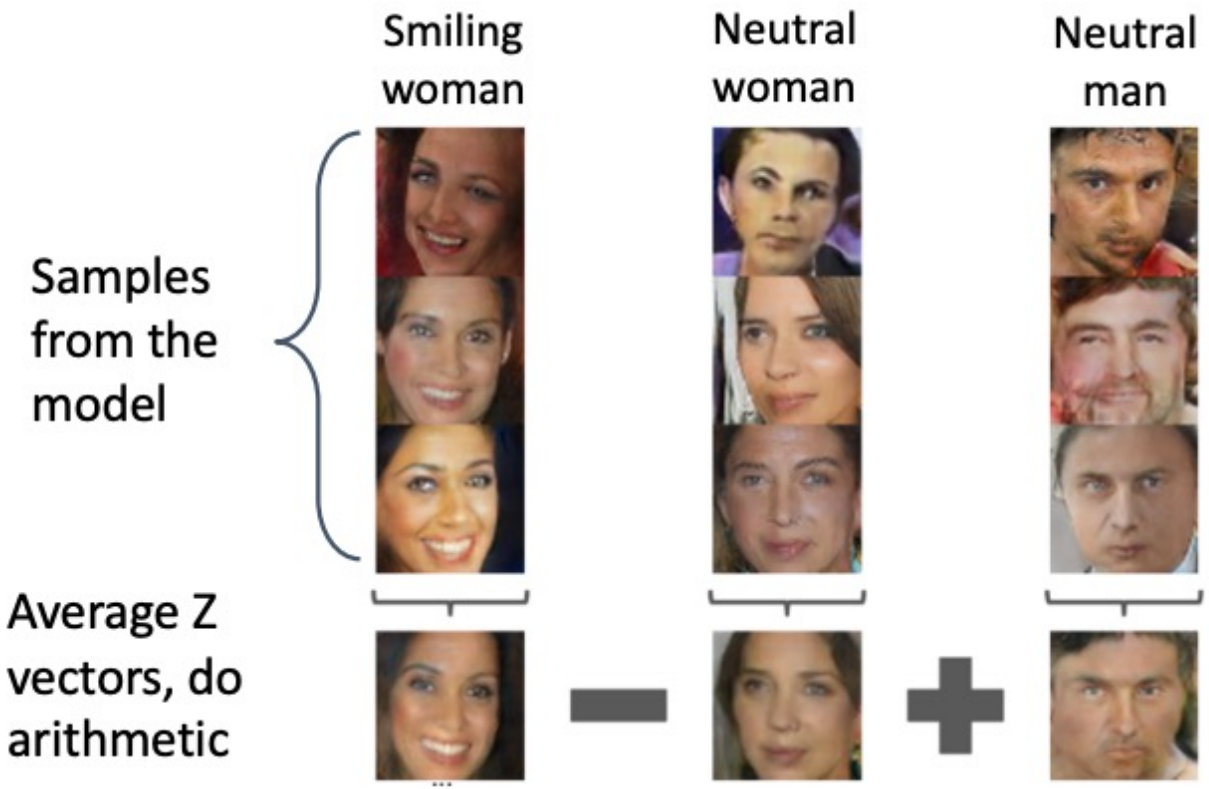
Part III: GAN editing

- Interpolation in latent space
- **GAN Math**
- GAN inversion
- Learn and apply latent directions
- CLIP + StyleGAN

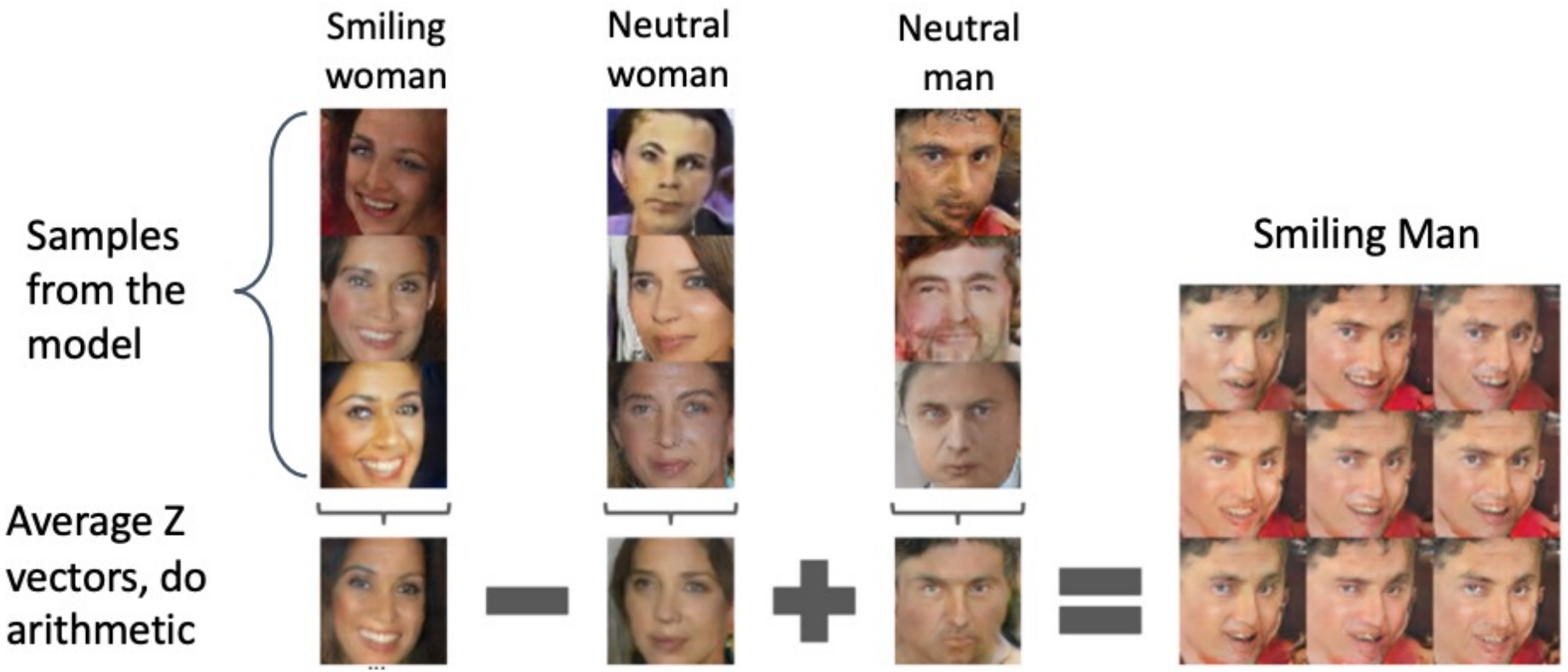
GAN Vector Math



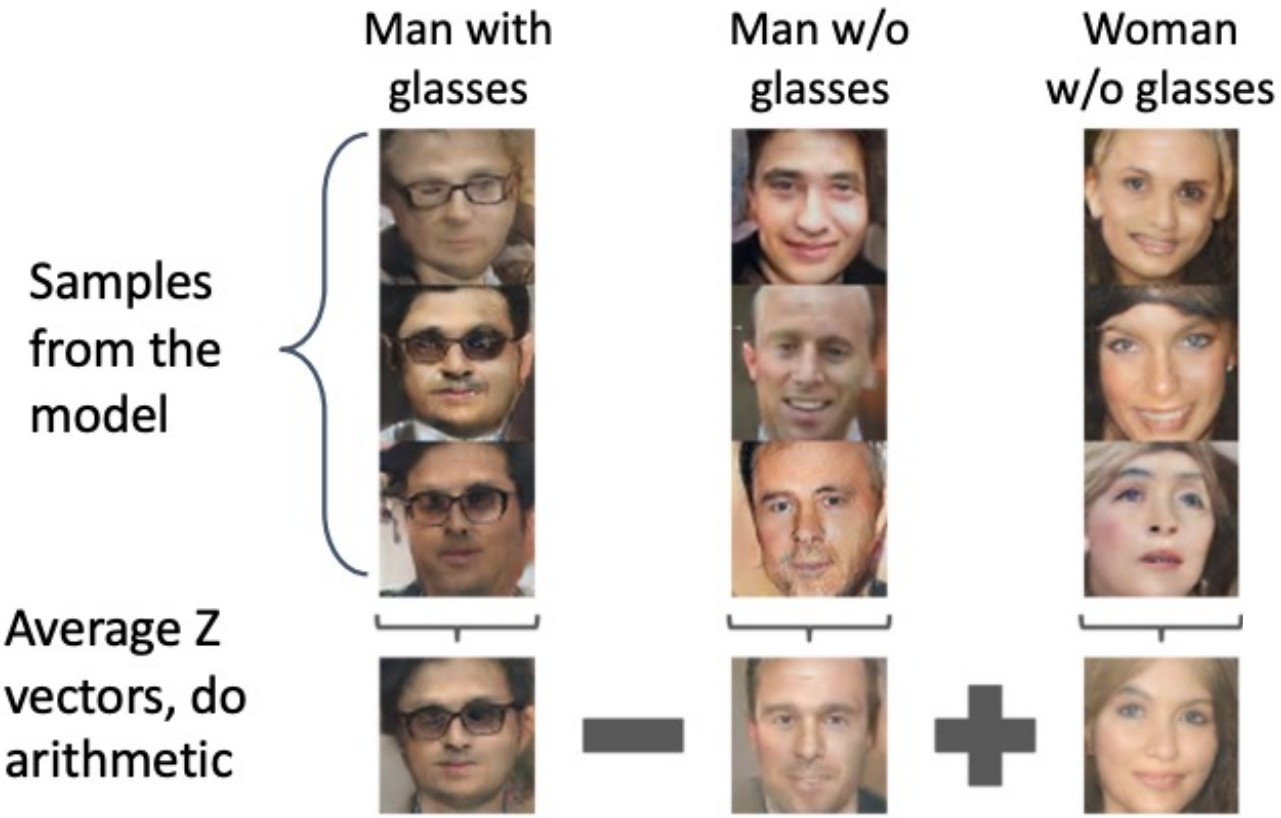
GAN Vector Math



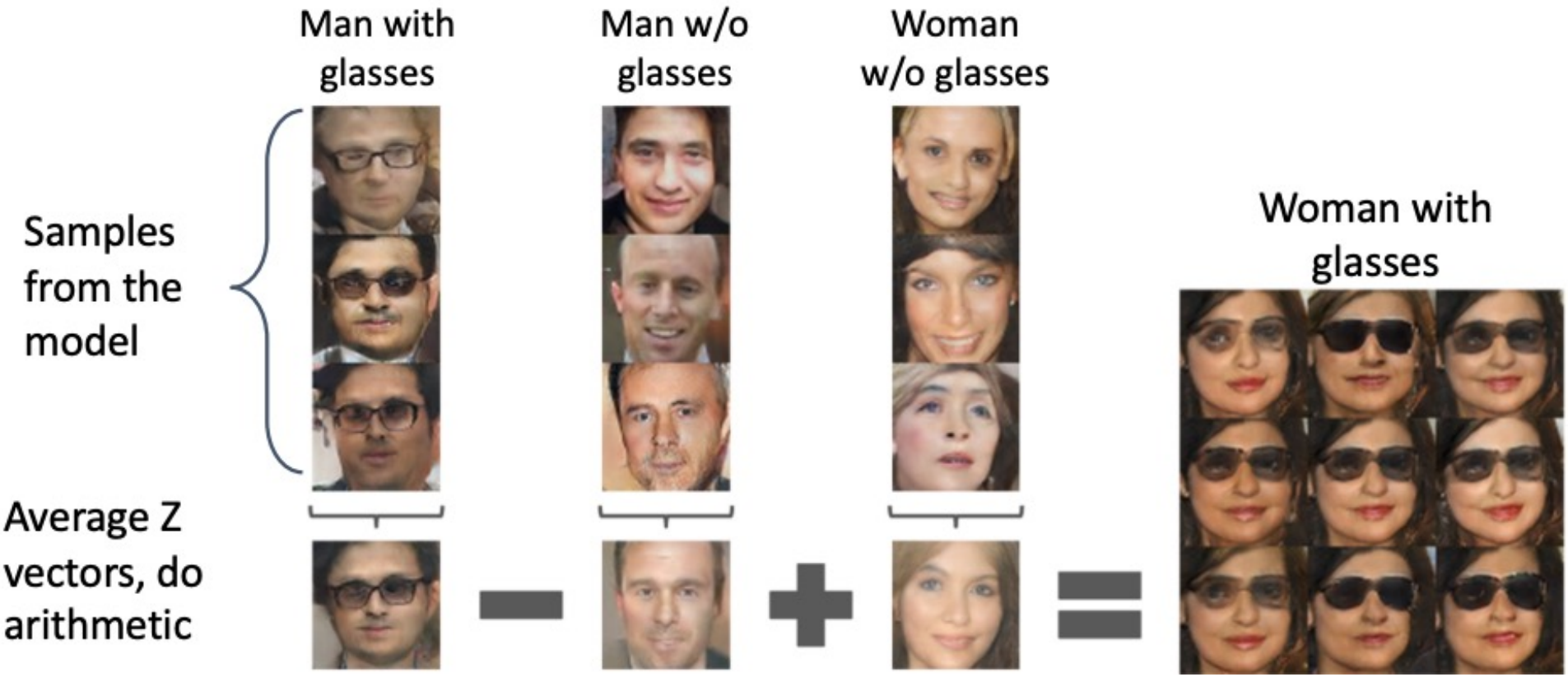
GAN Vector Math



GAN Vector Math



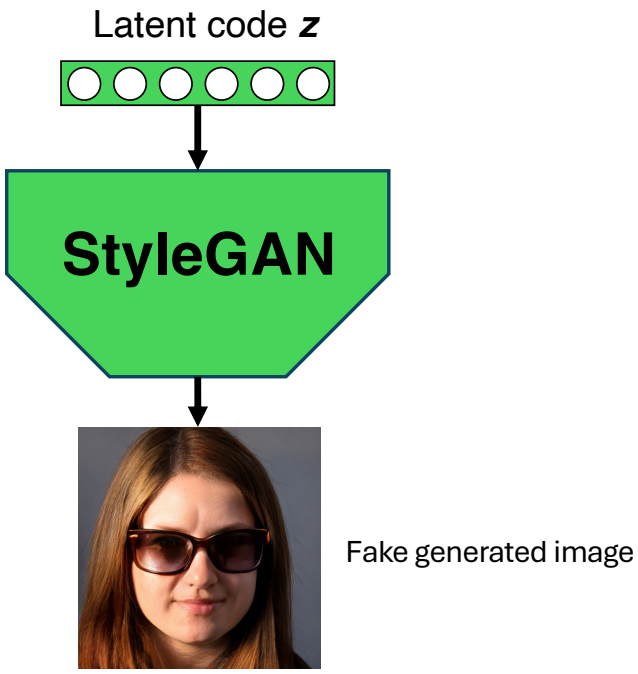
GAN Vector Math



Part III: GAN editing

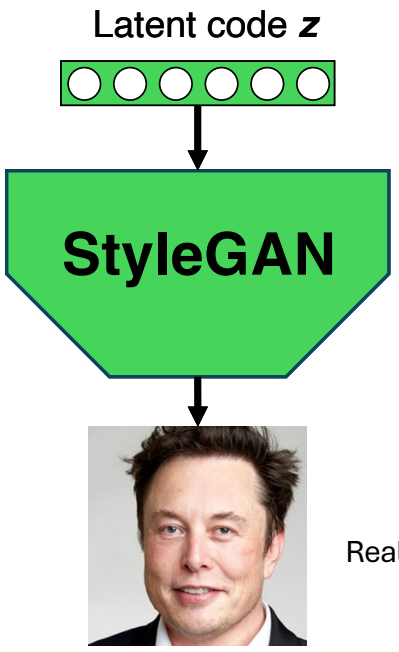
- Interpolation in latent space
- GAN Math
- **GAN inversion**
- Learn and apply latent directions
- CLIP + StyleGAN

Real images: GAN inversion



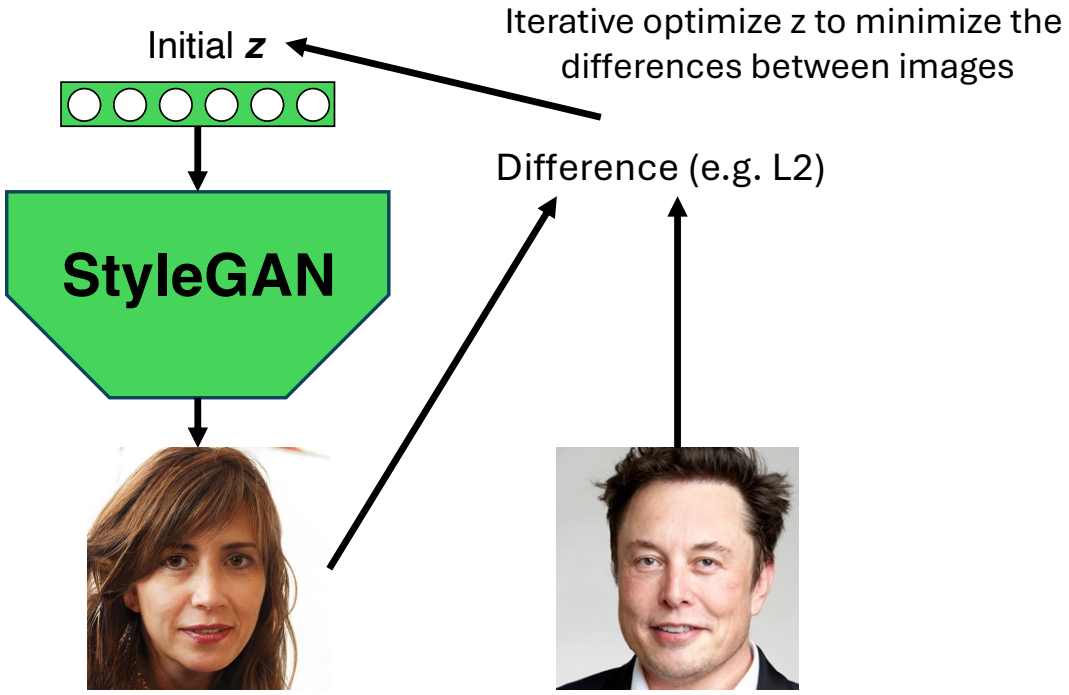
Fake generated image

Real images: GAN inversion

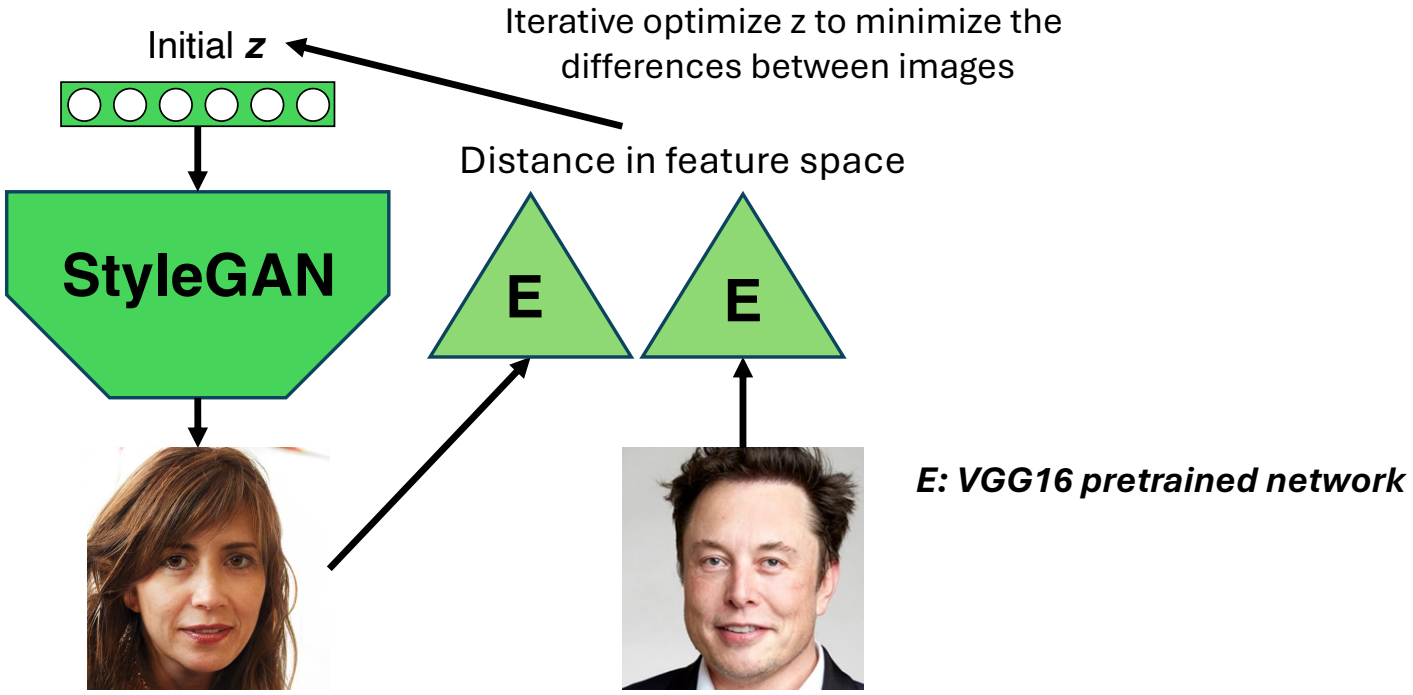


How to find z_o that generates Elon Musk?

Real images: GAN inversion



Real images: GAN inversion



Real images: GAN inversion

```
python projector.py --outdir=out --target=targetimg.png \ --
network=https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada/pretrained/ffhq.pkl
```

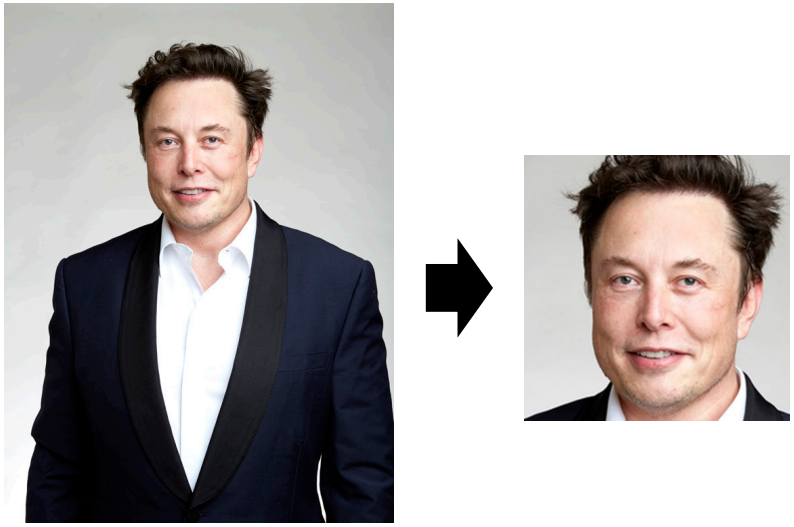
Real images: GAN inversion

```
python projector.py --outdir=out --target=targetimg.png \ --
network=https://nvlabs-fi-cdn.nvidia.com/stylegan2-ada/pretrained/ffhq.pkl
```

Align Images with FFHQ

https://github.com/happy-jihye/FFHQ-Alignment

Align Images with FFHQ

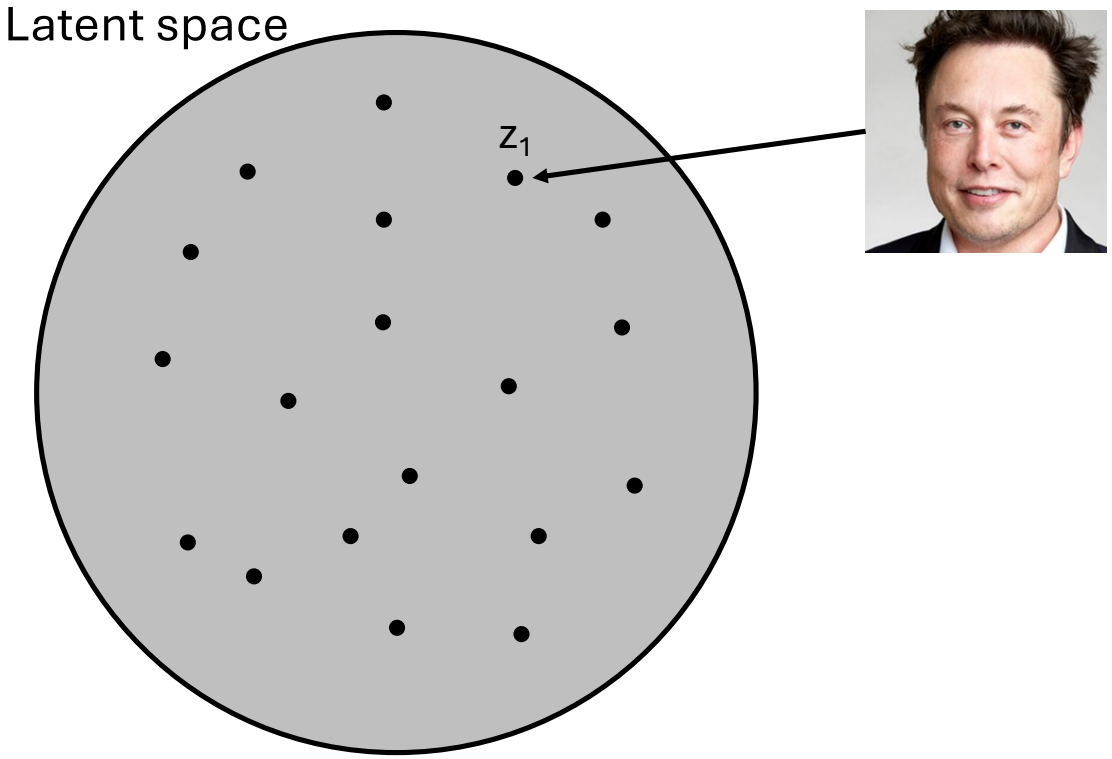


https://github.com/happy-jihye/FFHQ-Alignment

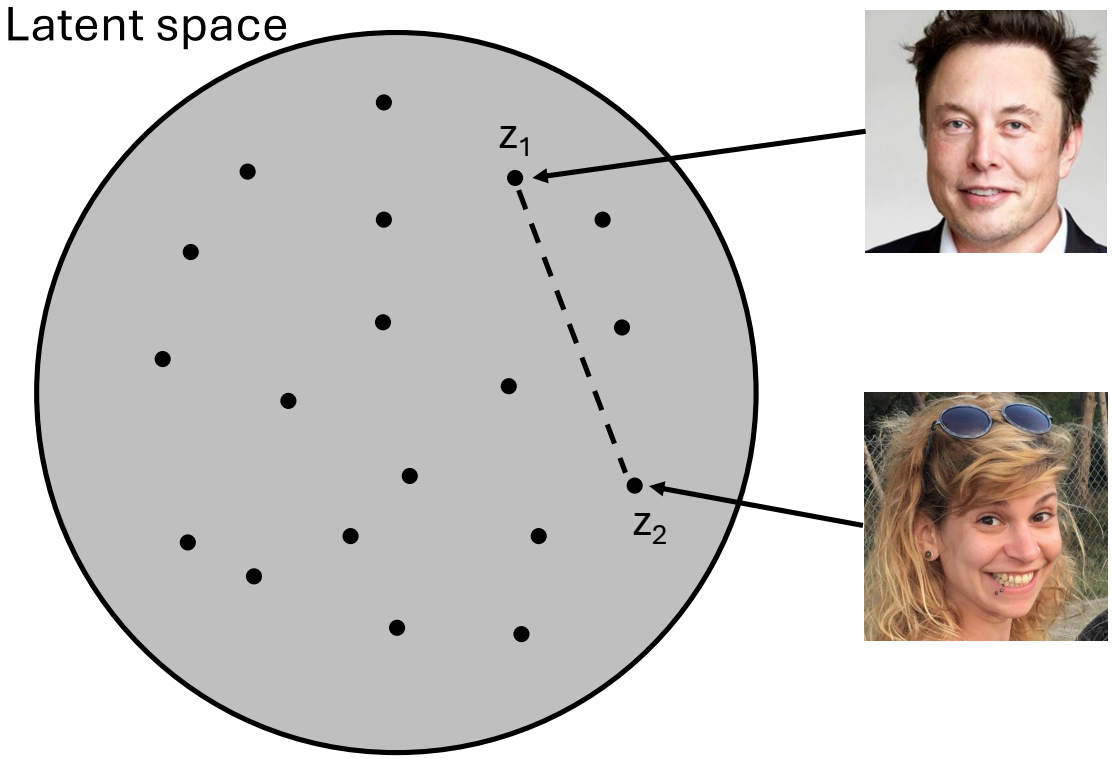
Part III: GAN editing

- Interpolation in latent space
- GAN Math
- GAN inversion
- **Learn and apply latent directions**
- CLIP + StyleGAN

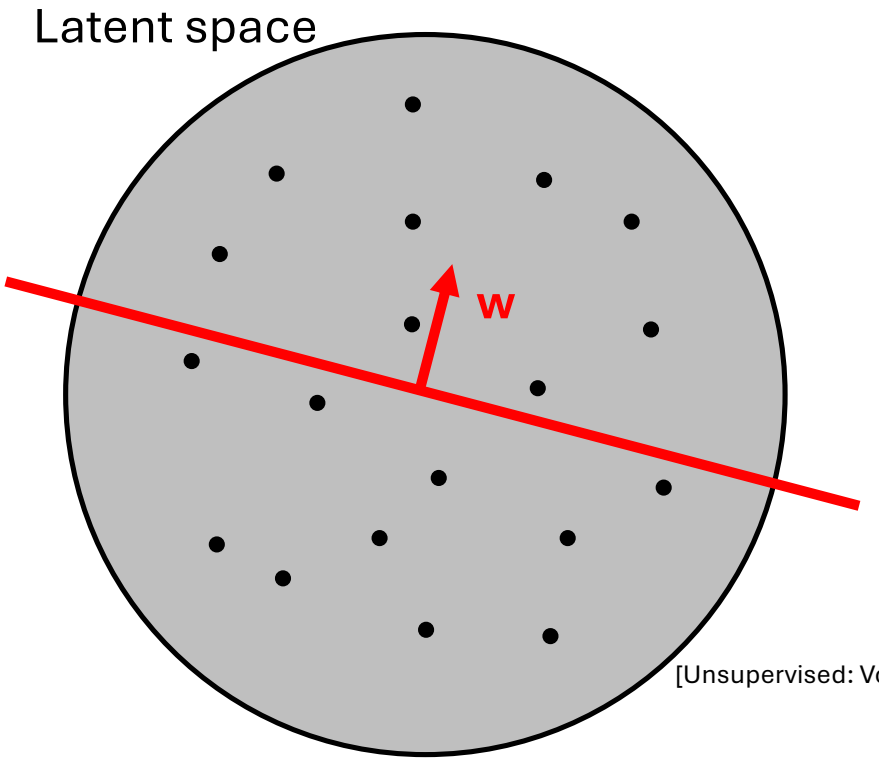
Latent directions



Latent directions



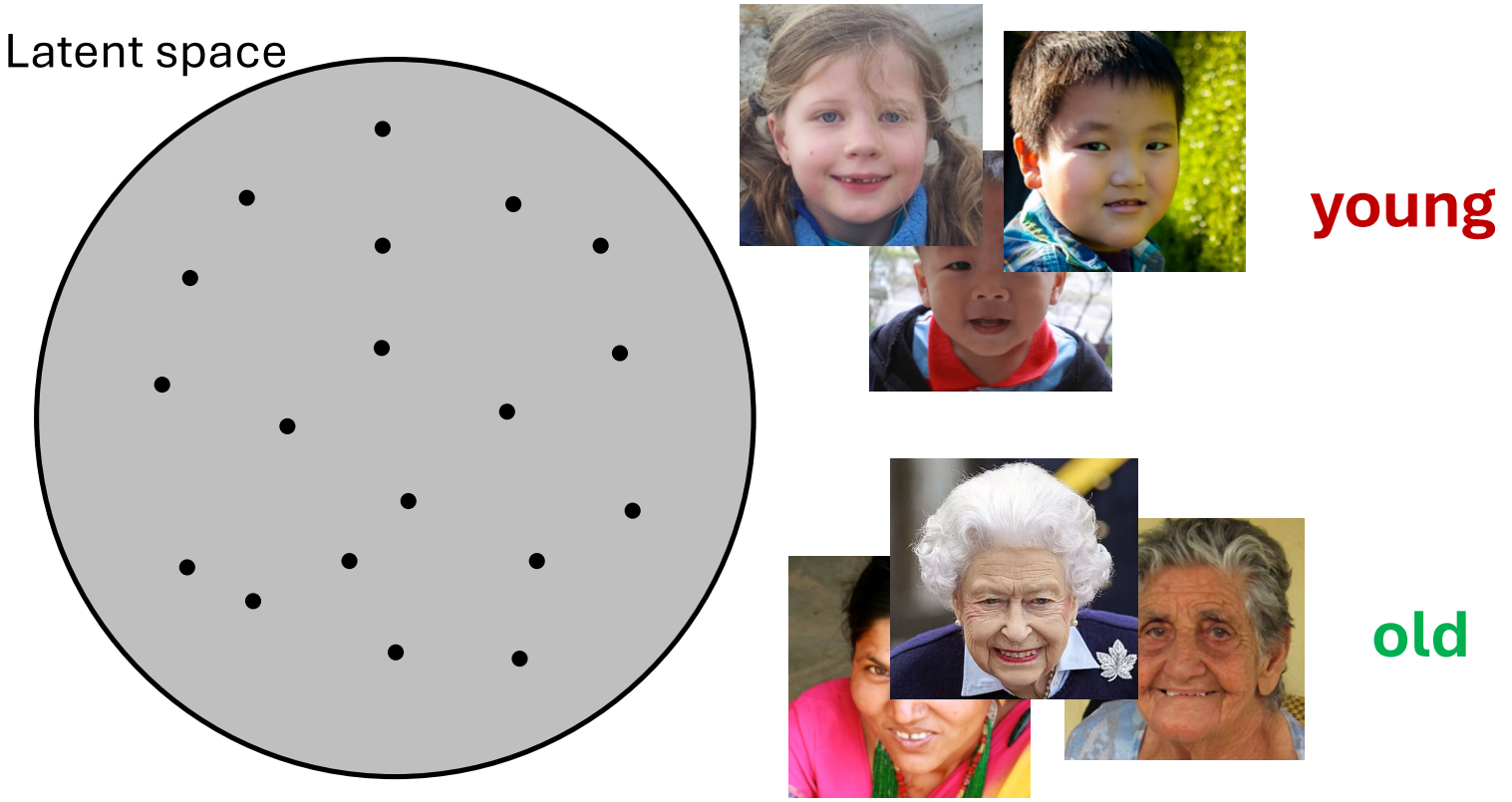
Latent directions



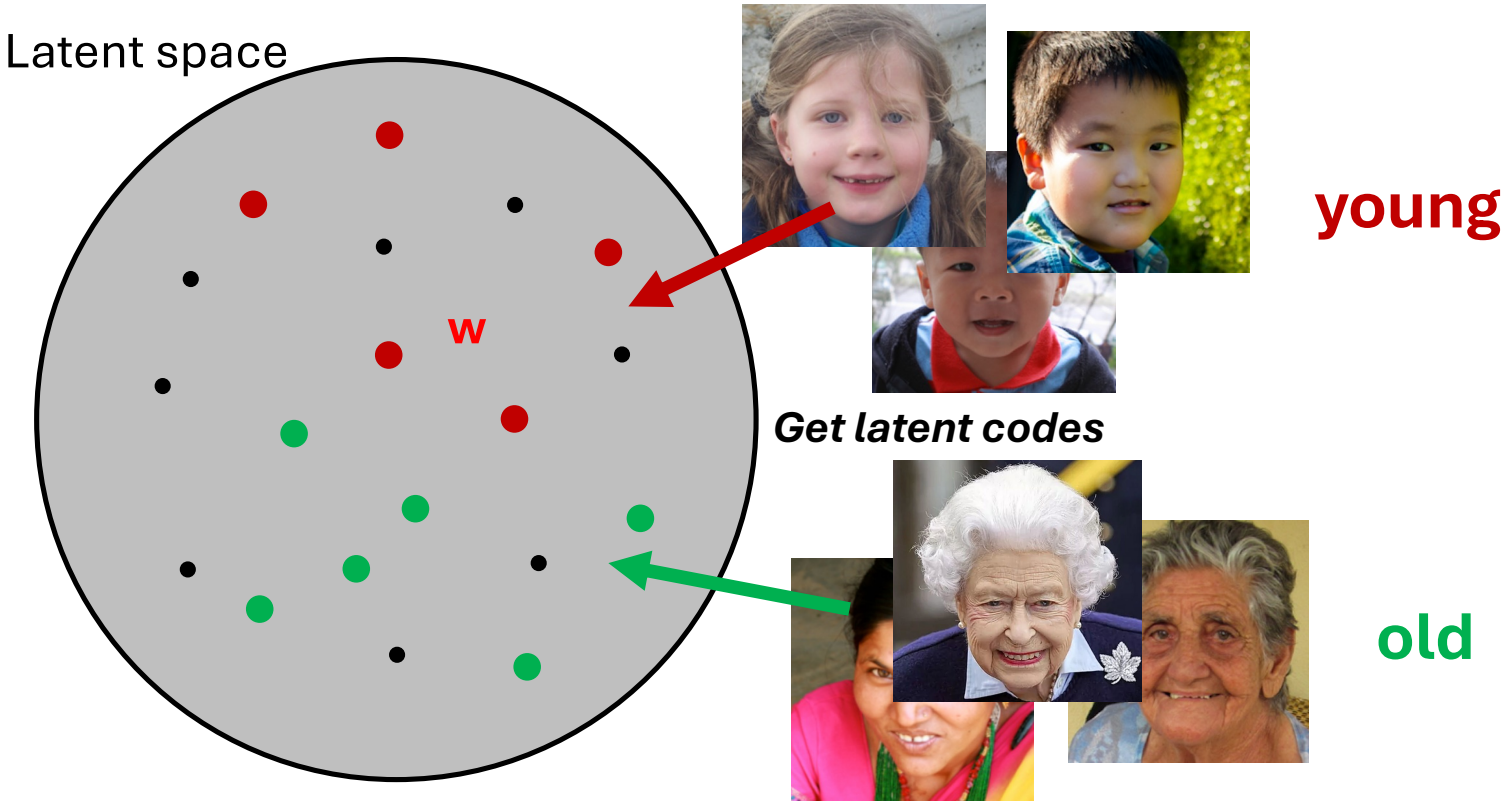
- Learn meaningful latent directions
- Either supervised or unsupervised
- Apply these directions in any image
- Example of directions in faces:
  - Aging
  - Smiling
  - Hair or skin color
  - Gender

[Unsupervised: Voynov and Babenko. "Unsupervised discovery of interpretable directions in the gan latent space." In ICML 2020]

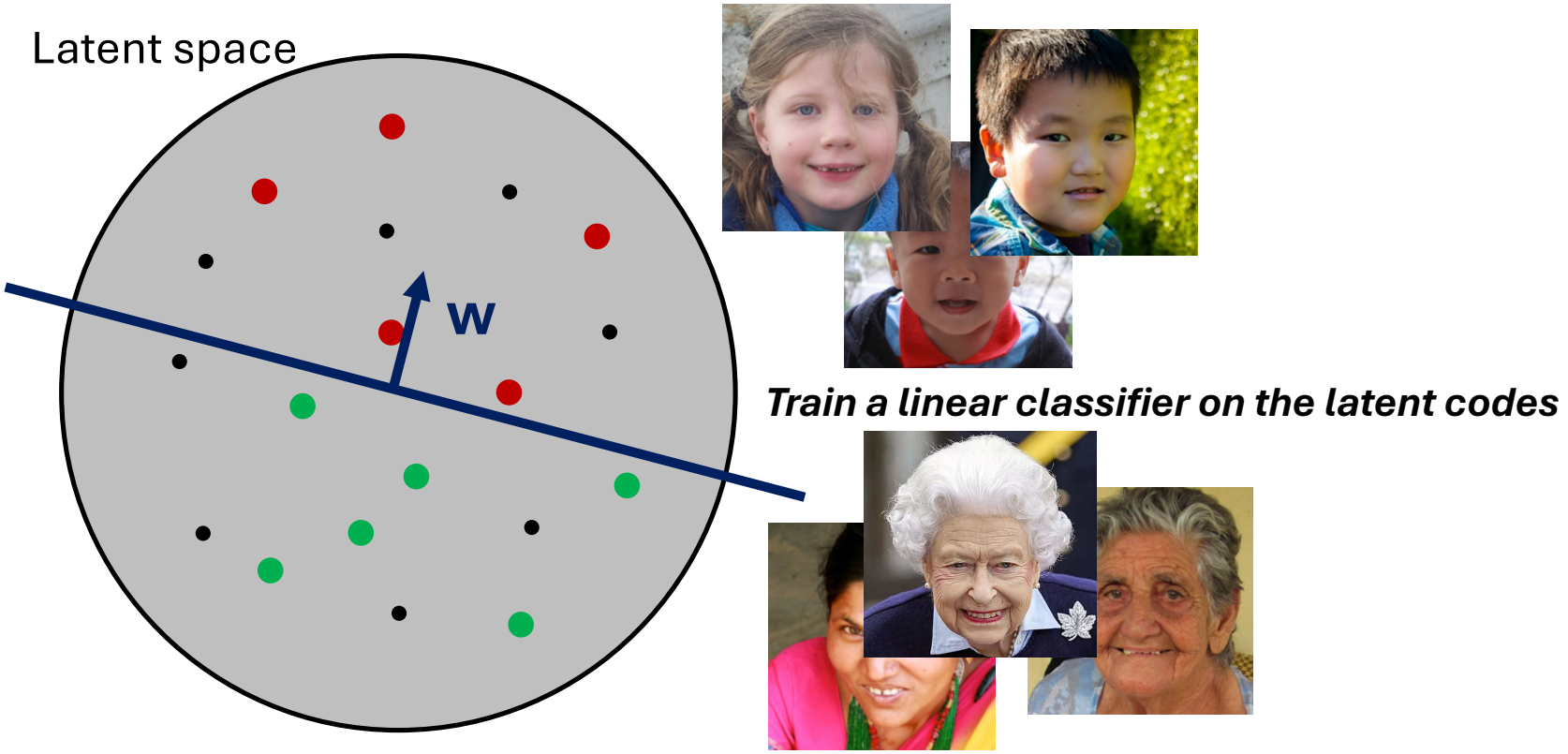
Supervised Learning of Latent directions



Supervised Learning of Latent directions

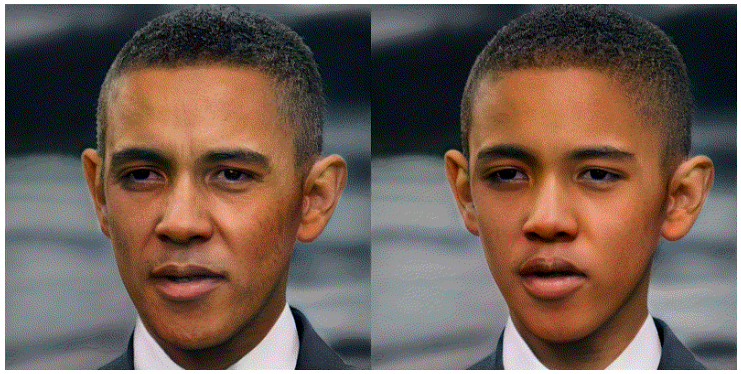


Supervised Learning of Latent directions

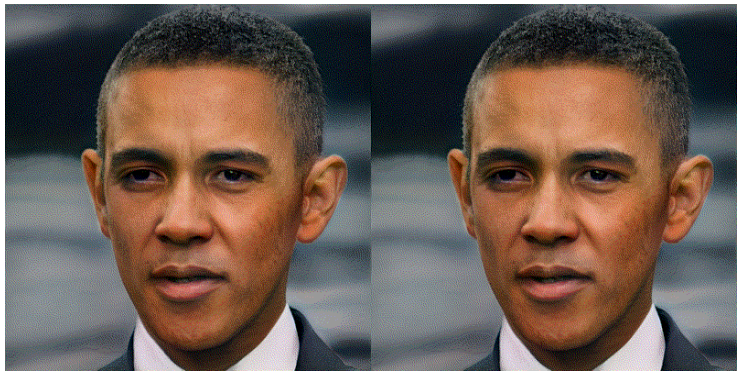


Supervised Learning of Latent directions

aging

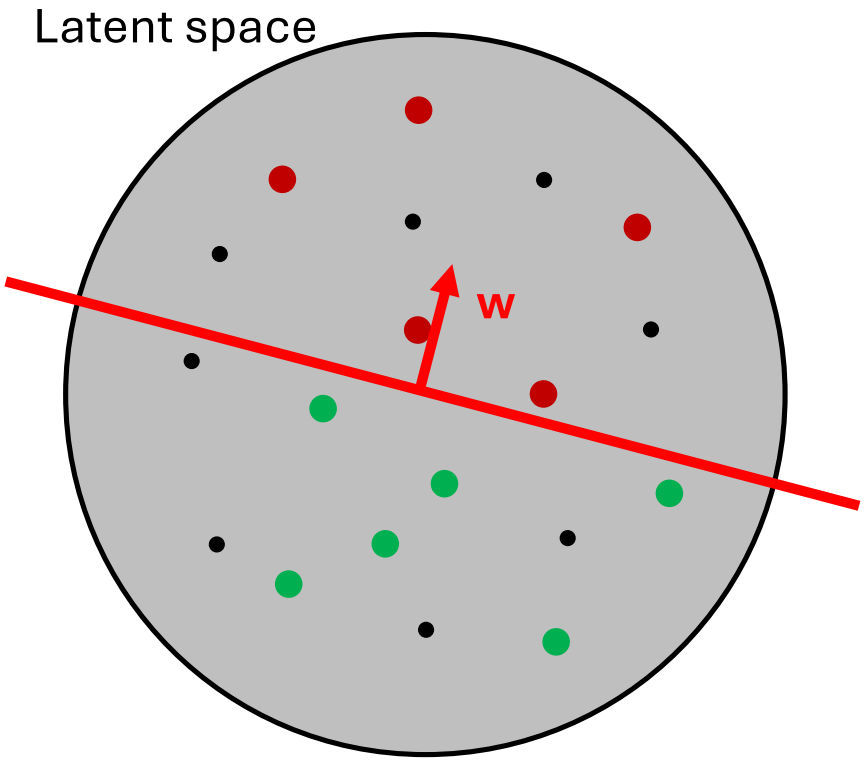


smiling



$z_{new} = z + w*m$

Supervised Learning of Latent directions



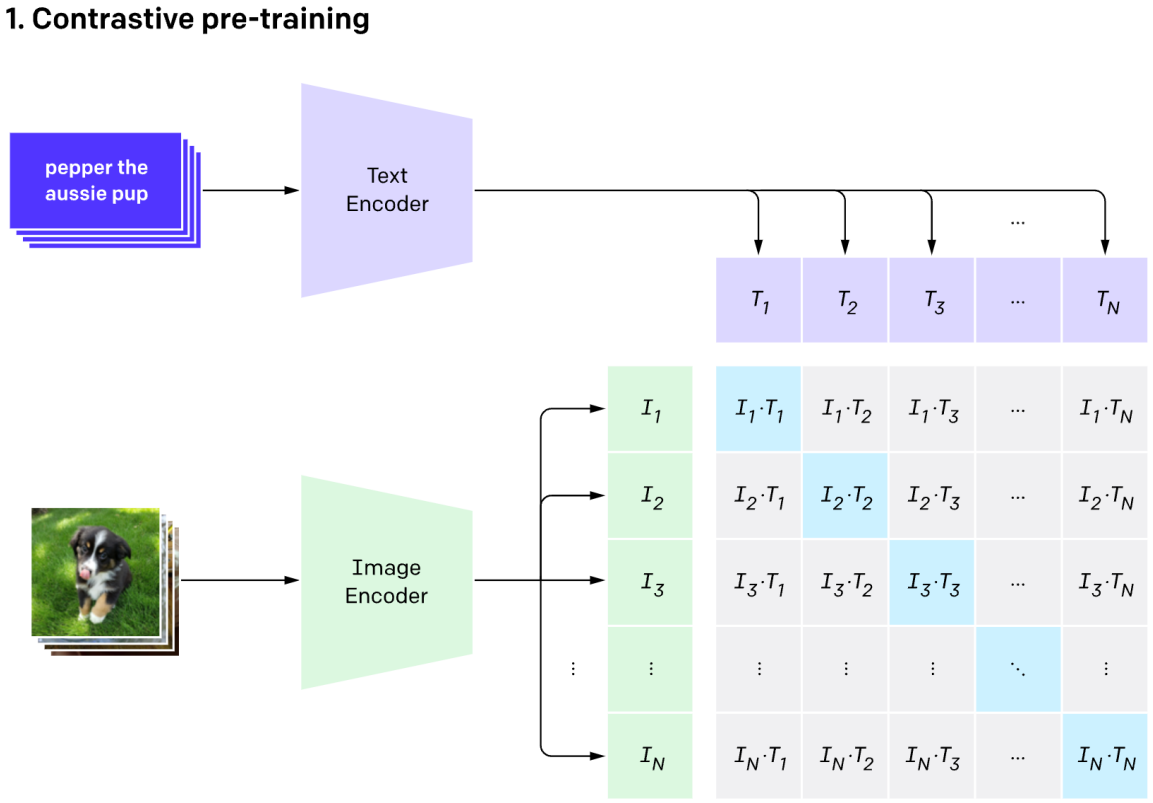
$z_{new} = z + w*m$

m:magnitude

Part III: GAN editing

- Interpolation in latent space
- GAN Math
- GAN inversion
- Learn and apply latent directions
- **CLIP + StyleGAN**

CLIP


1. Contrastive pre-training
pepper the aussie pup
Text Encoder
Image Encoder
T_1 T_2 T_3 ... T_N
I_1 I_2 I_3 ⋮ I_N
$I_1 \cdot T_1$ $I_1 \cdot T_2$ $I_1 \cdot T_3$ ... $I_1 \cdot T_N$
$I_2 \cdot T_1$ $I_2 \cdot T_2$ $I_2 \cdot T_3$ ... $I_2 \cdot T_N$
$I_3 \cdot T_1$ $I_3 \cdot T_2$ $I_3 \cdot T_3$ ... $I_3 \cdot T_N$
$I_N \cdot T_1$ $I_N \cdot T_2$ $I_N \cdot T_3$ ... $I_N \cdot T_N$

CLIP

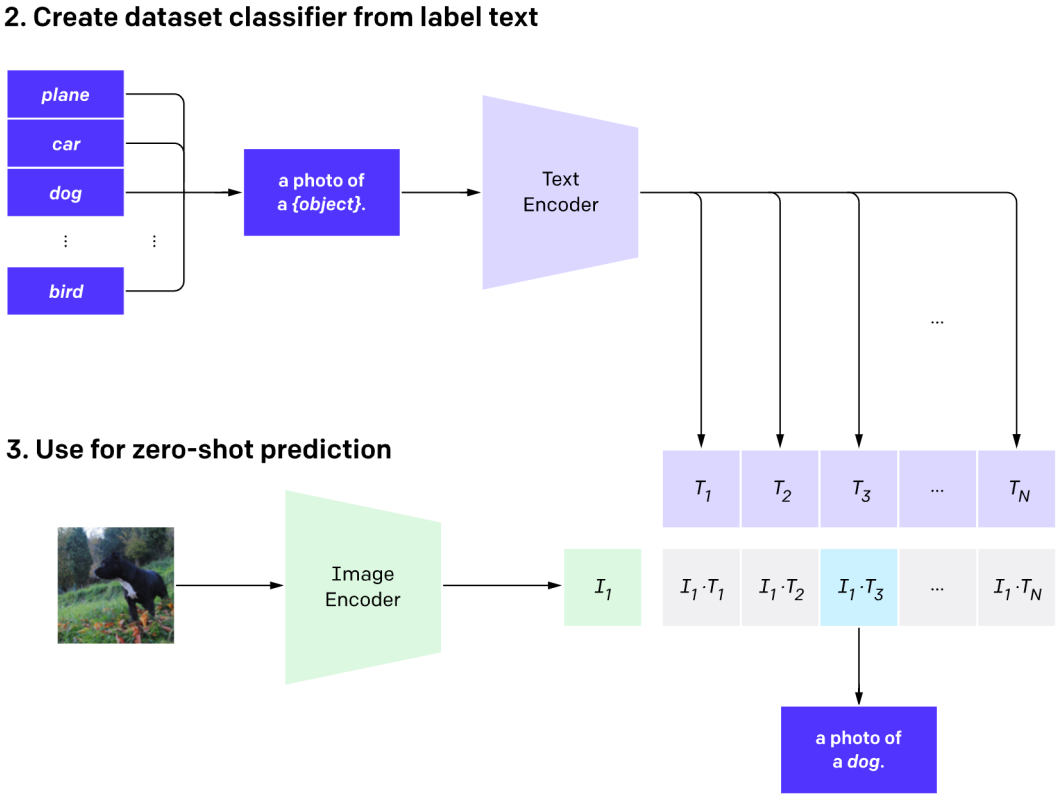

2. Create dataset classifier from label text
plane
car
dog
bird
a photo of a {object}.
Text Encoder
...
T_1
T_2
T_3
...
T_N
3. Use for zero-shot prediction
Image Encoder
I_1
$I_1 \cdot T_1$
$I_1 \cdot T_2$
$I_1 \cdot T_3$
...
$I_1 \cdot T_N$
a photo of a dog.

Image generation from text

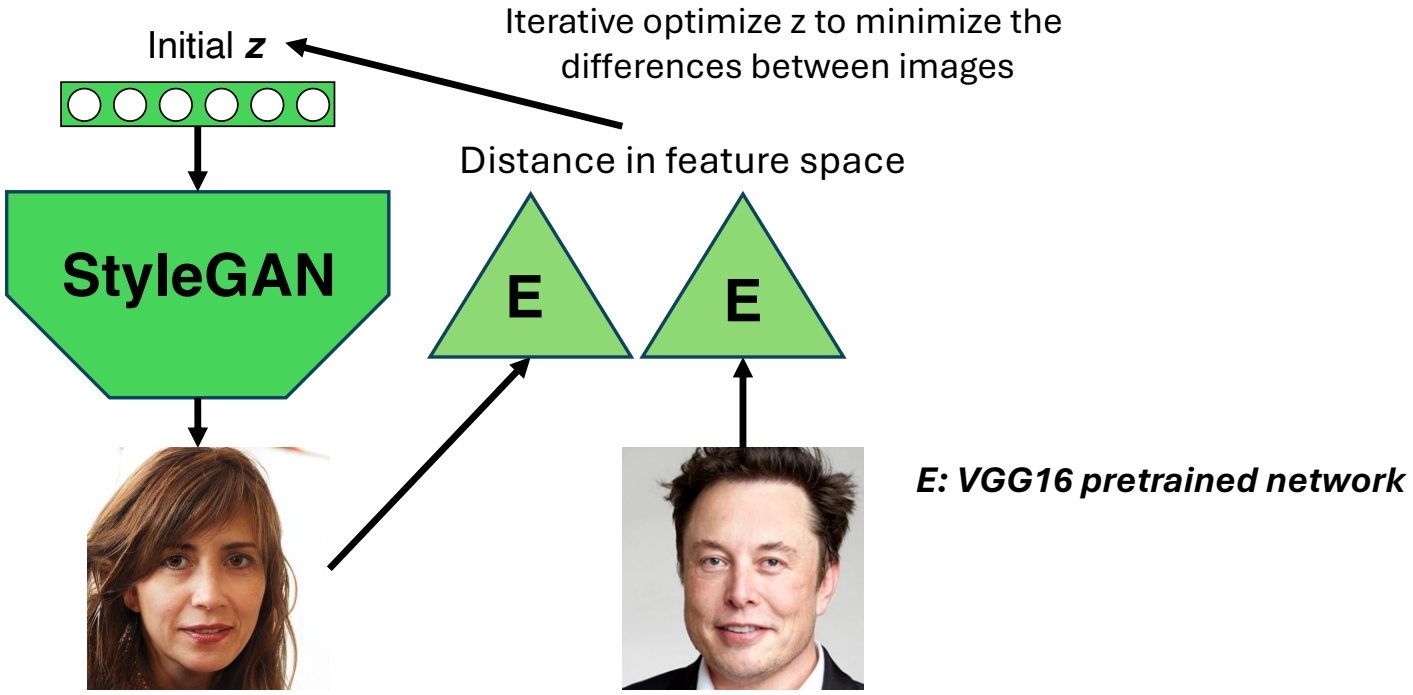


Image generation from text

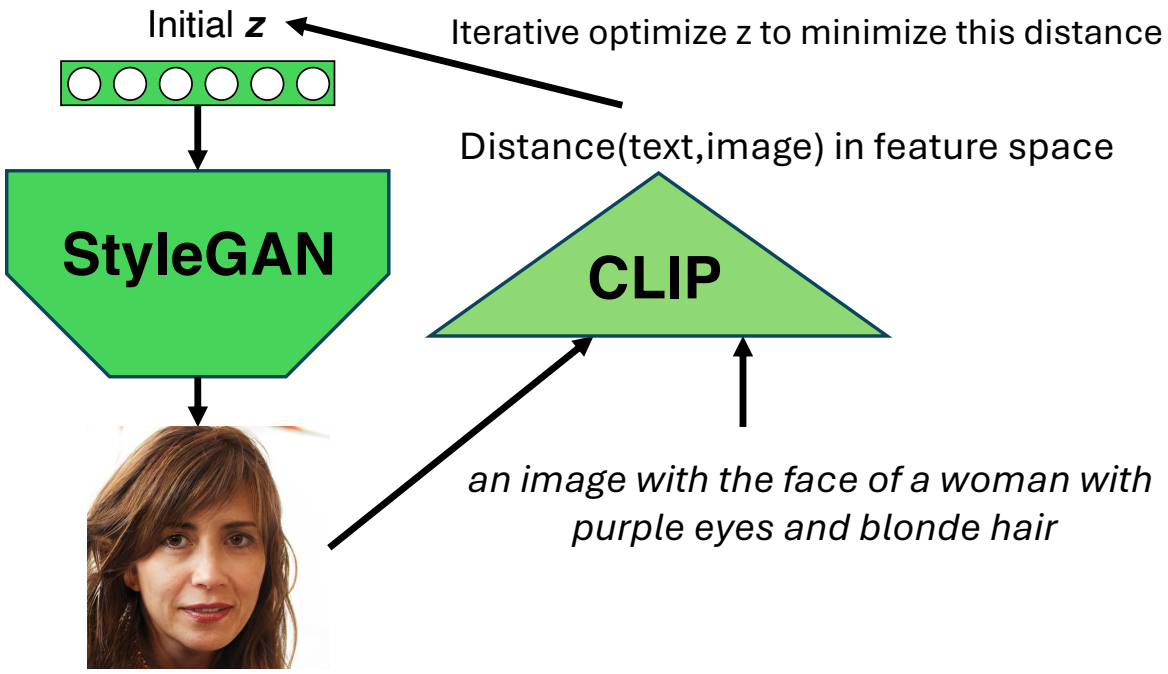


Image generation from text

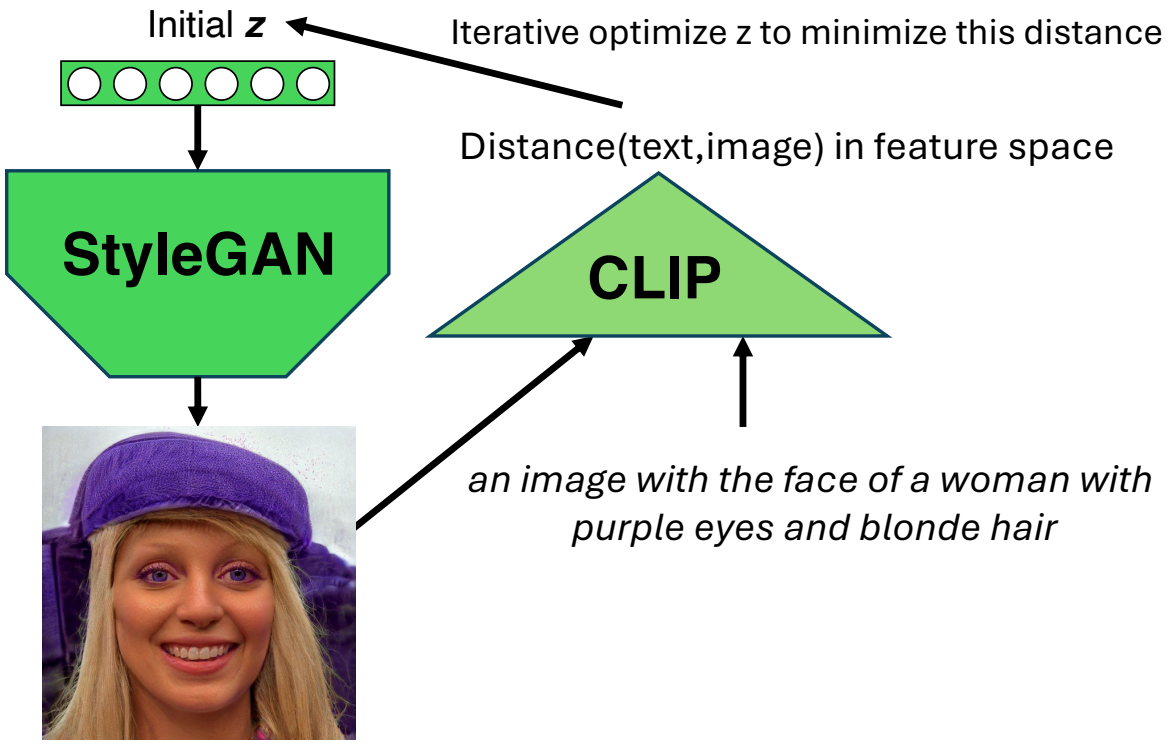
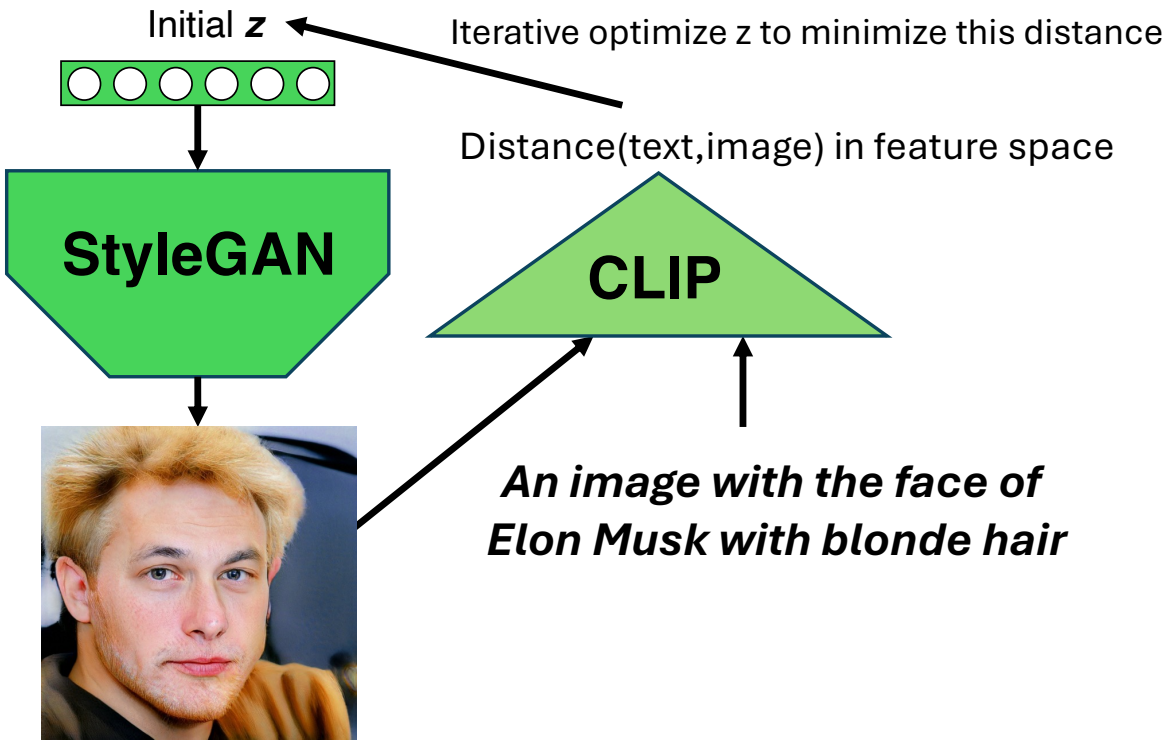


Image generation from text



Thank you