SC395:

Image Generative Models in Computer Vision

Viraj Shah

Lecture 2

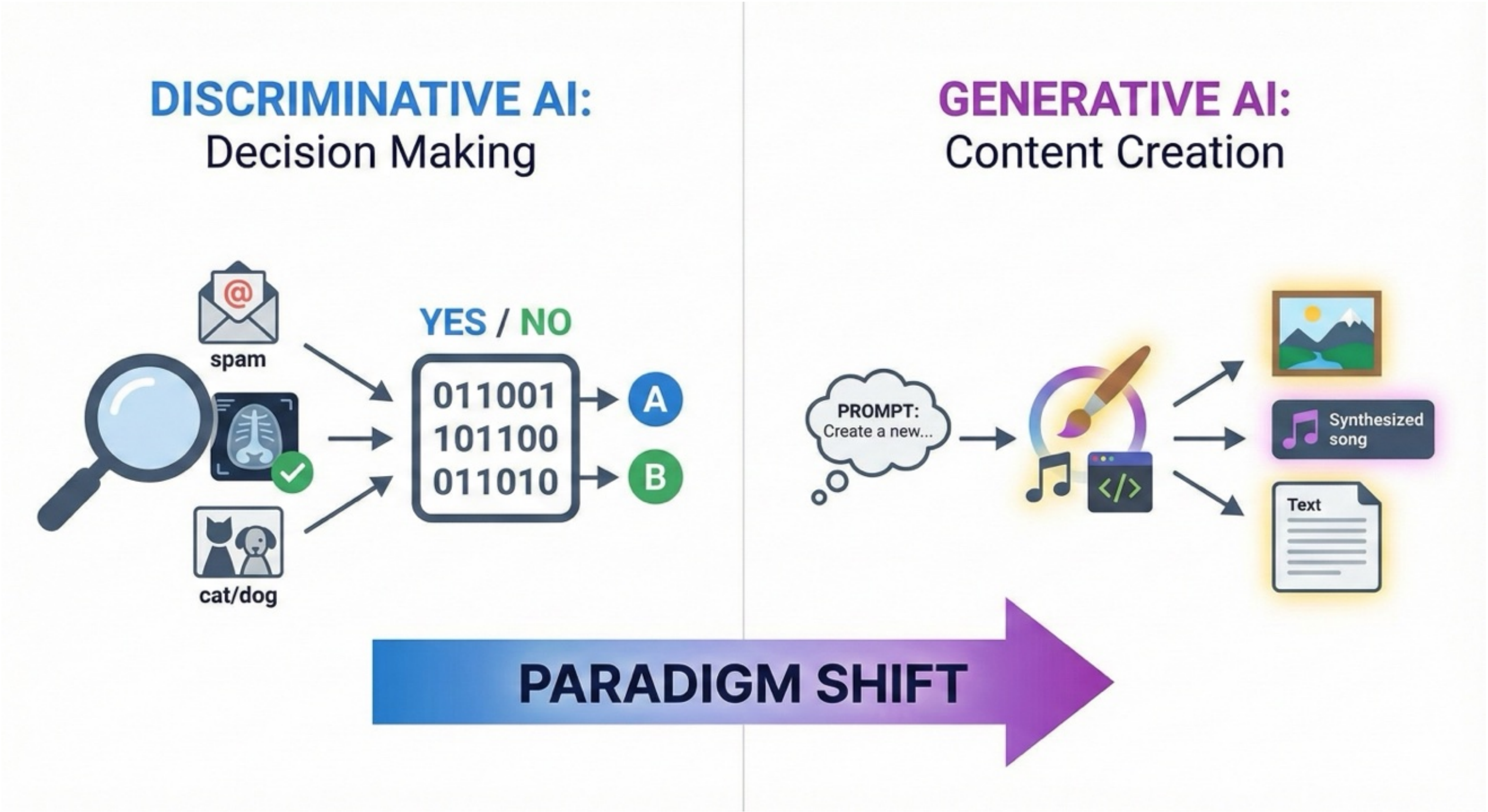
Jan 7^{th}, 2026

sc395.virajshah.com

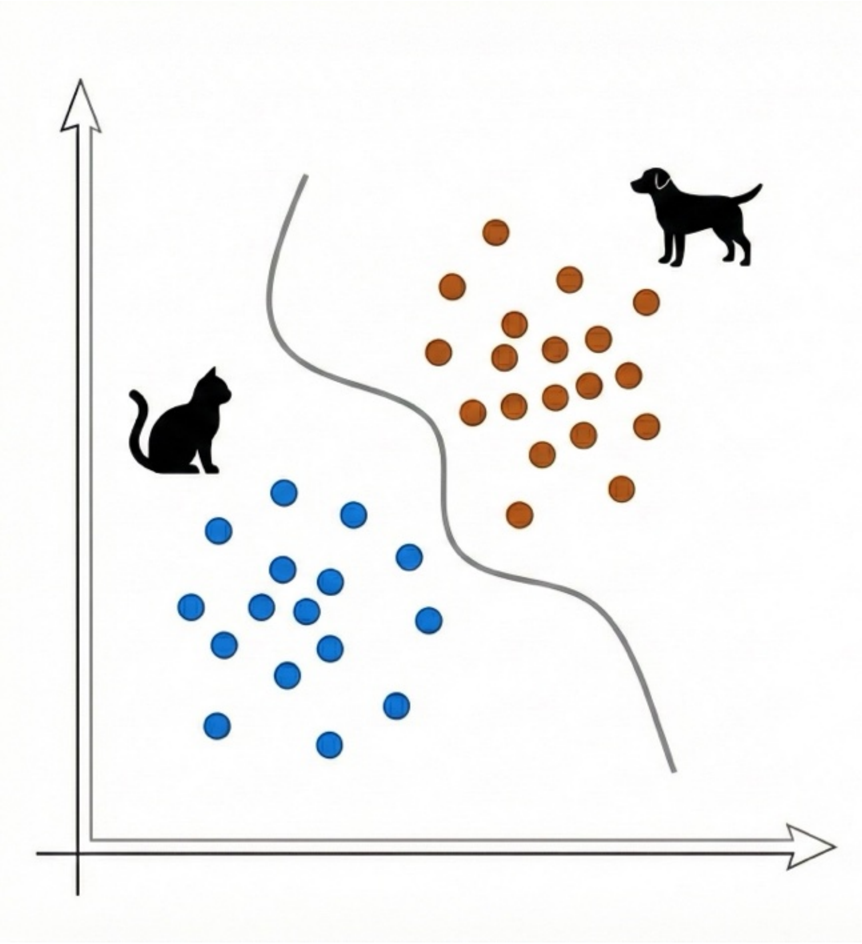
ILLINOIS

Lecture 2: GANs and Applications

Paradigm Shift: Discriminative Models → Generative Models

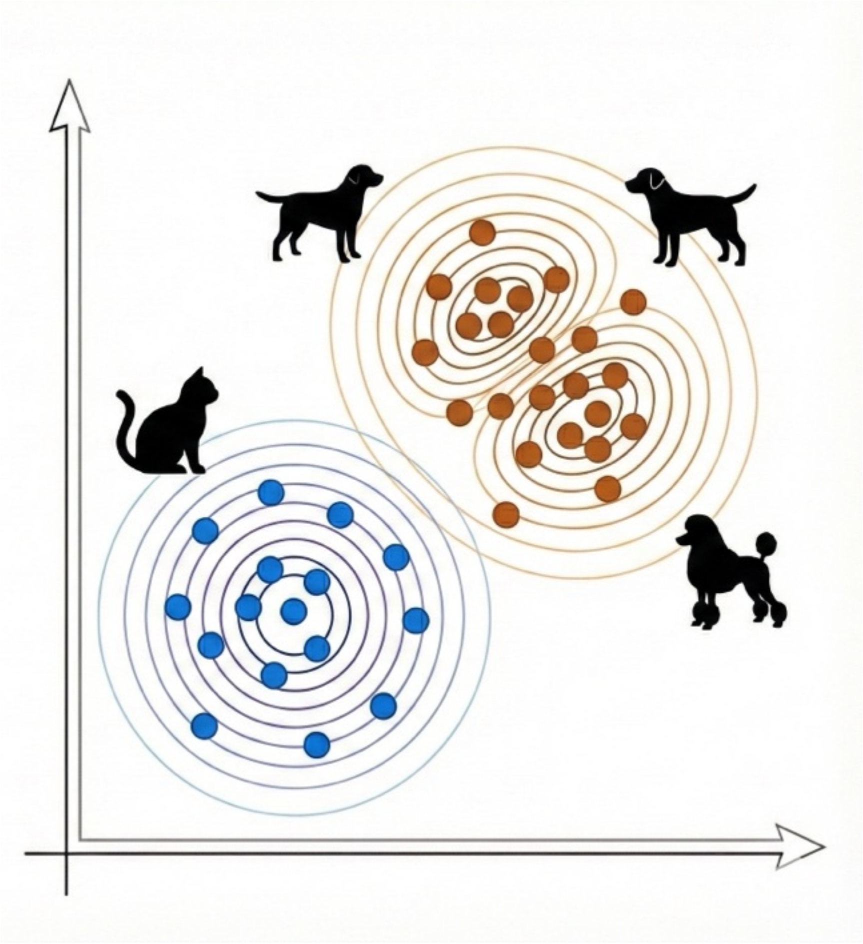


Discriminative Models vs. Generative Models



Predict the label given the Input

Learns $P(Y/X)$



Generate new content by understanding **the abstract patterns of the data**

Learns $P(X,Y)$

Naïve Attempt: Maximum Likelihood Estimation

- Our data distribution is $p_{data}(\boldsymbol{x})$: we have m samples: $(x^{(1)}, x^{(2)}, \ldots, x^{(m)})$

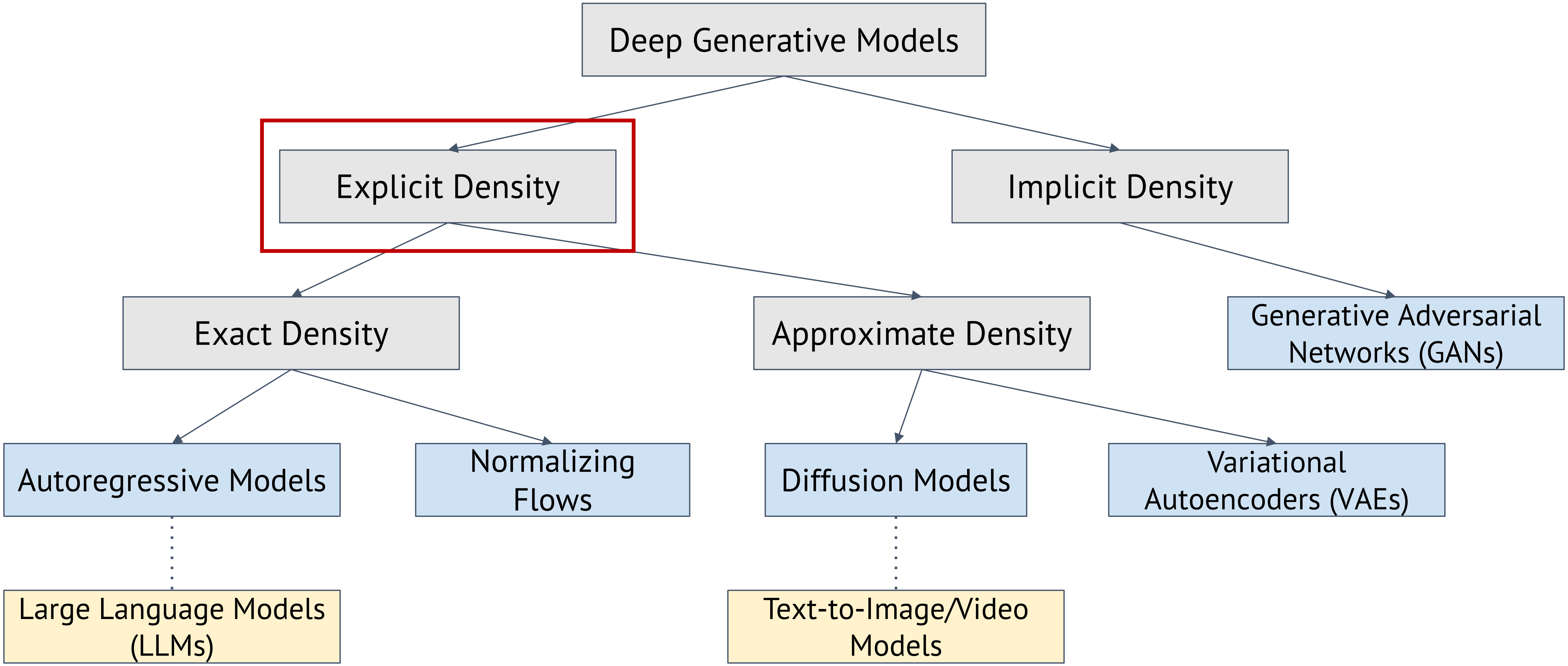
- Our generative model is parameterized by θ,
  - Likelihood of each sample is given by: $p_{model}\left(x^{(i)}; \theta\right)$

- MLE: choose the parameters for the model that maximize the likelihood of the training data:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{m} p_{model}\left(x^{(i)}; \theta\right)$$

Minimize KL Divergence

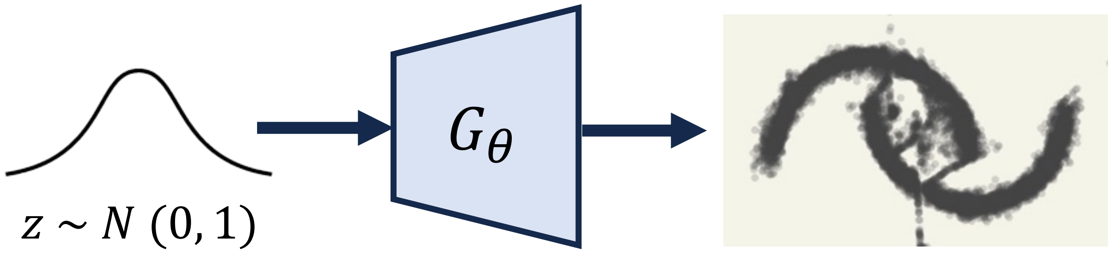
Deep Generative Models



Slide from Deep Generative Modeling, CMU, 18-789

Implicit Generative Models

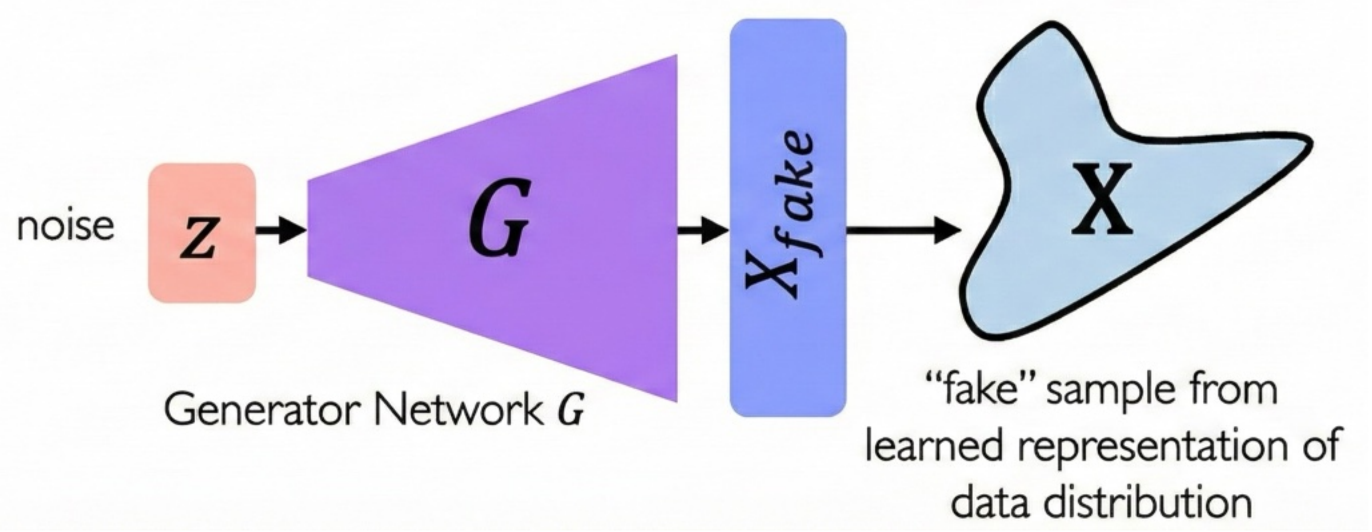
- No explicit definition of the density function p_{model}
- Interact only indirectly with p_{model}:
  - via sampling from it
  - No need to parameterize p_{model}
  - Instead, use latent variable z and generator G_θ such that $G_\theta(z) \sim p_{model}$



Samples from p_{model}

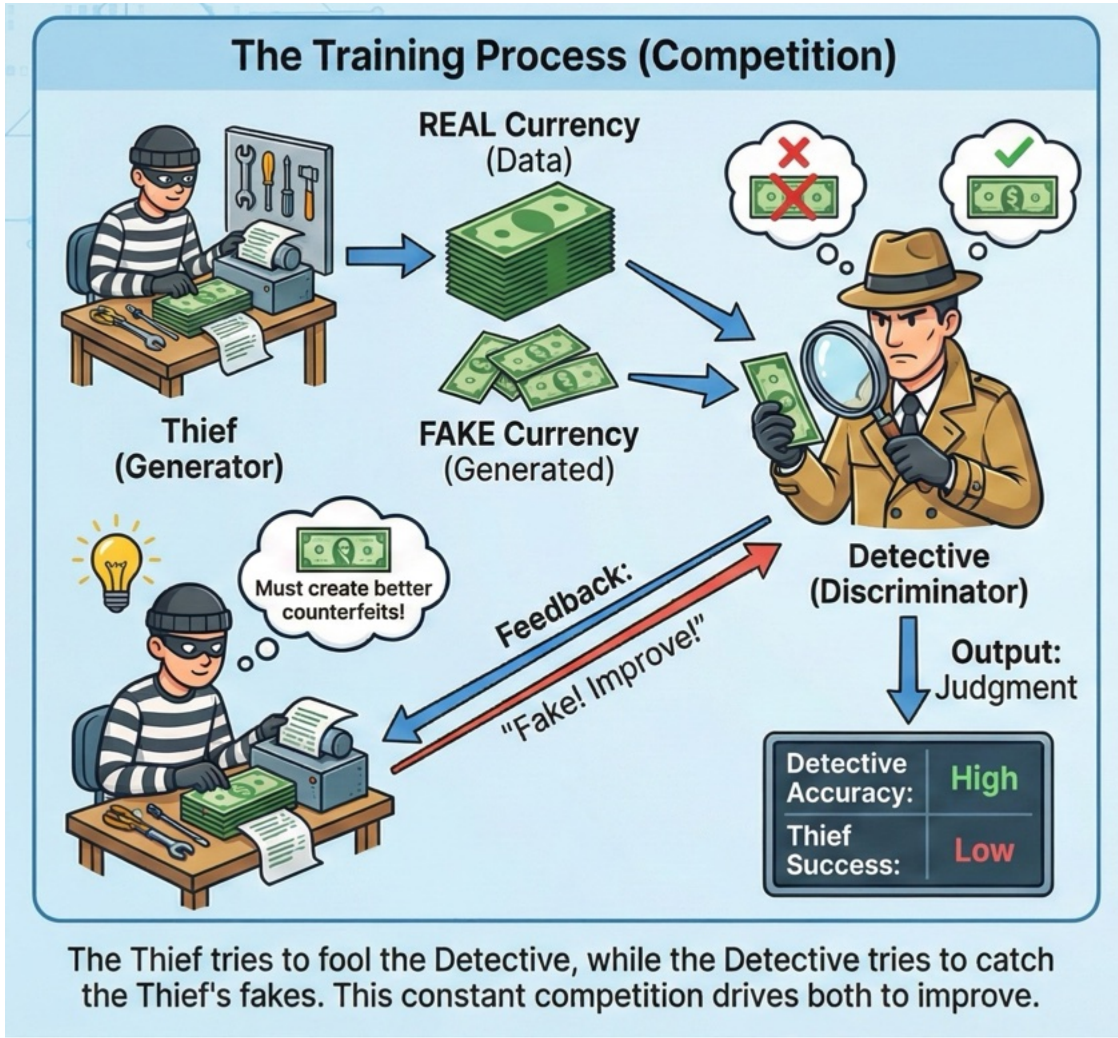
Implicit Models: What if we just want to sample?

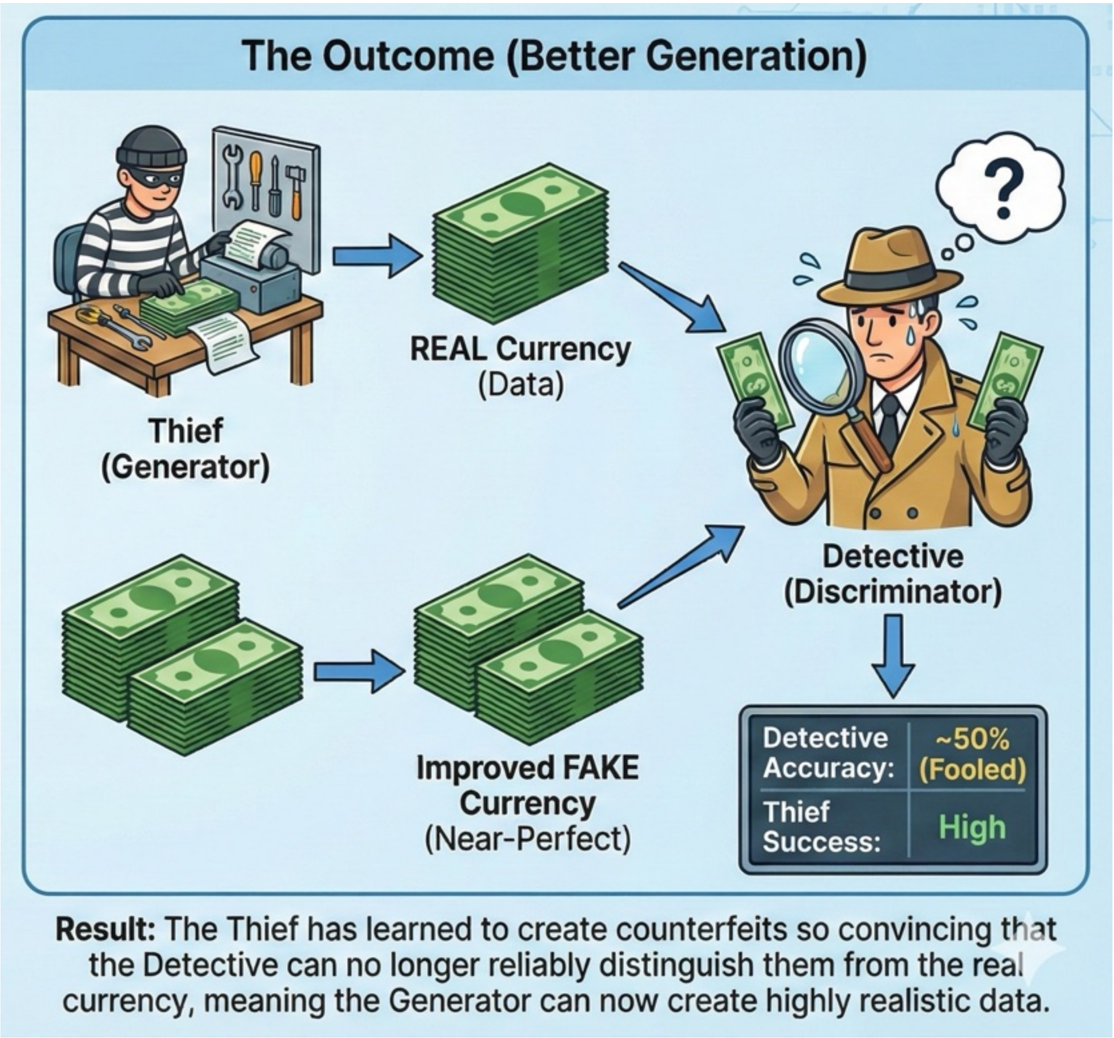
- **Idea**: don't explicitly model density, just sample to generate new instances
- **Problem**: want to sample from complex distribution -- can't do this directly!
- **Solution**: sample from something simple (e.g. noise), learn a transformation to the data distribution



Generative Adversarial Networks (GANs)

Obtain a generative model by having two neural networks competing with each other

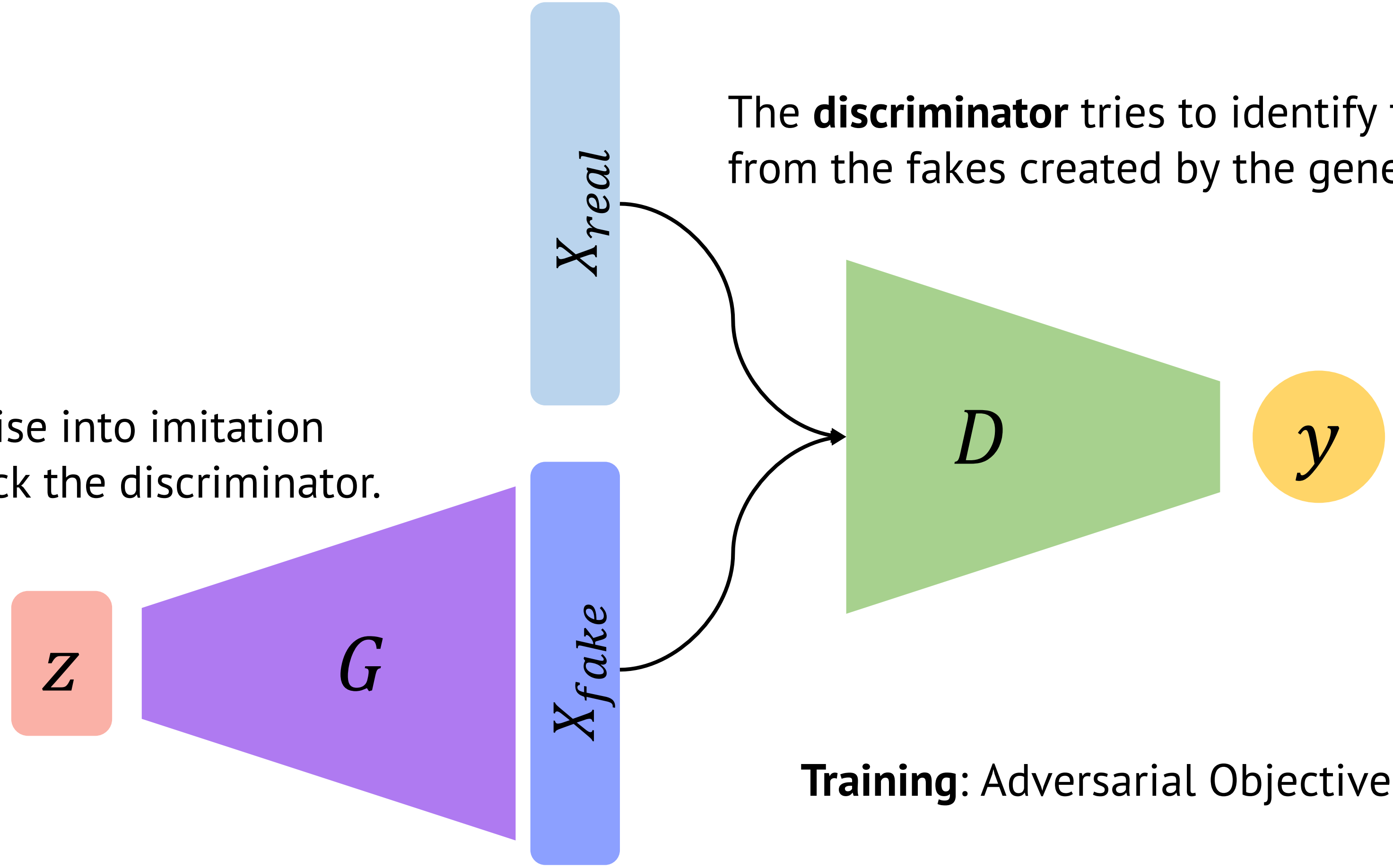




Generative Adversarial Networks (GANs)

The **discriminator** tries to identify the real data from the fakes created by the generator

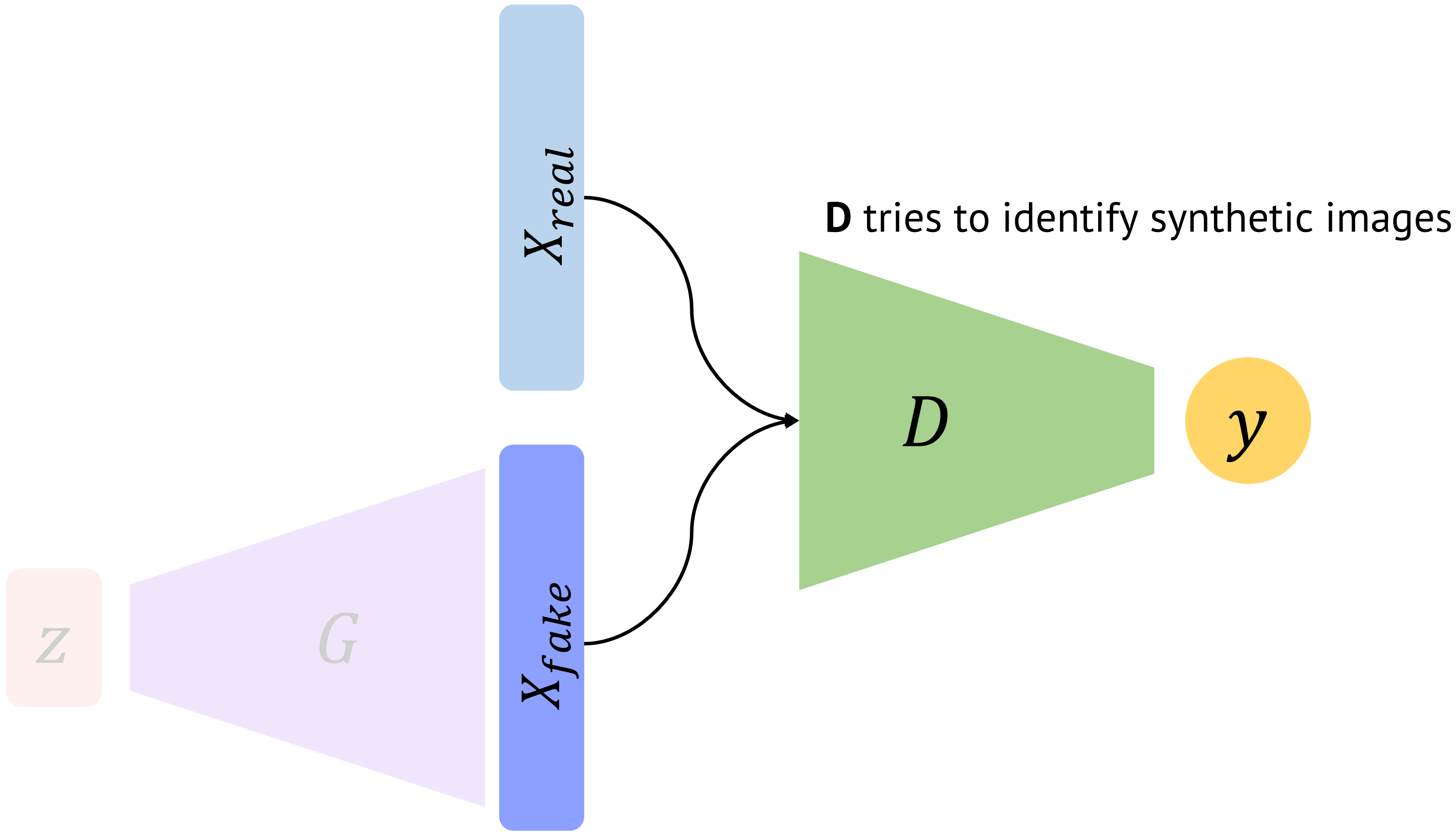
The **generator** turns noise into imitation of the data trying to trick the discriminator.



Training: Adversarial Objectives for G and D

Optimum: G reproduces true data distribution

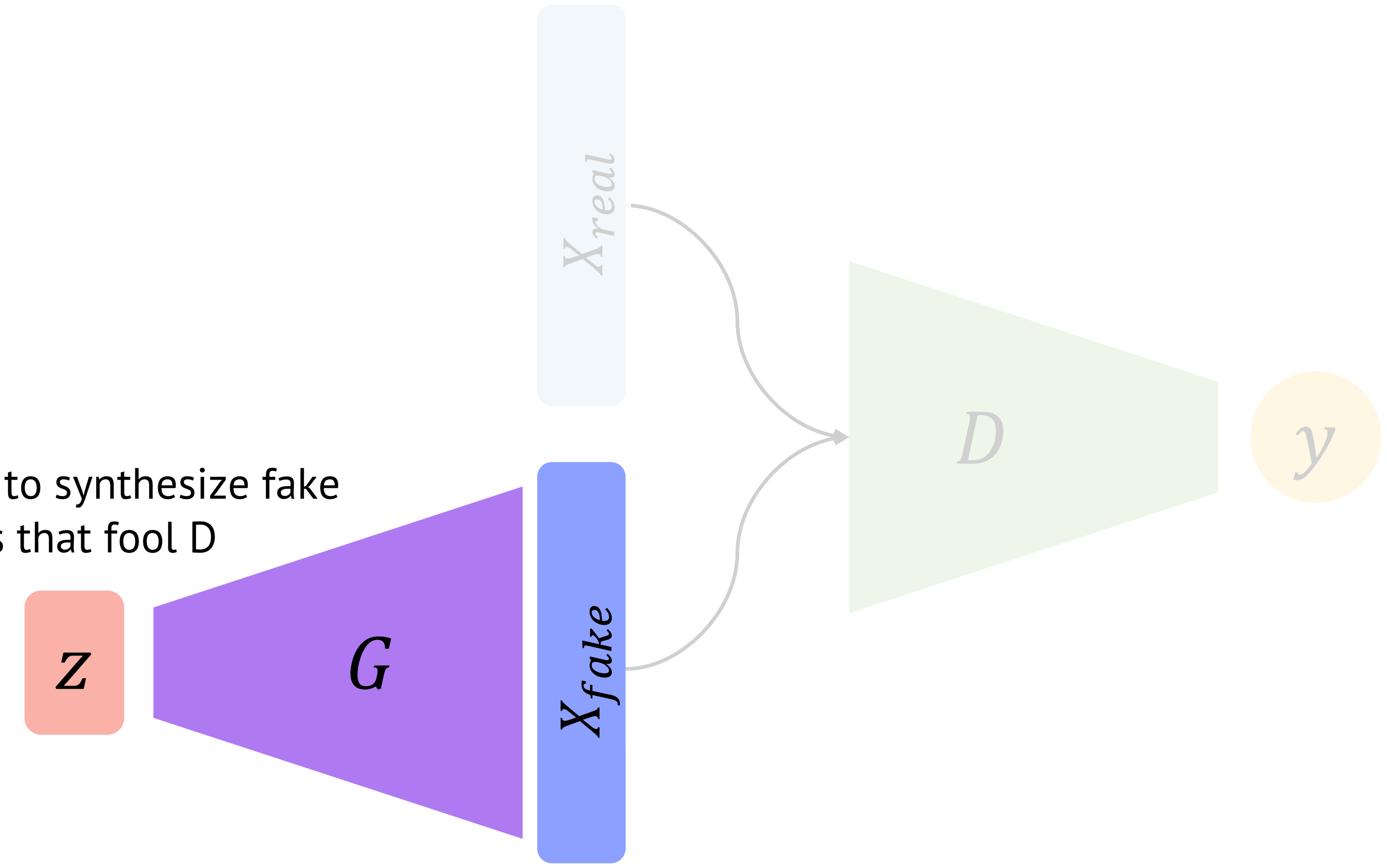
Generative Adversarial Networks (GANs)



$$\underbrace{\arg\max_{D}}\,\mathbb{E}_{\mathbf{z},\mathbf{x}}[\underbrace{\log D(G(\mathbf{z}))}_{\text{Fake}} + \underbrace{\log(1 - D(\mathbf{x}))}_{\text{Real}}]$$

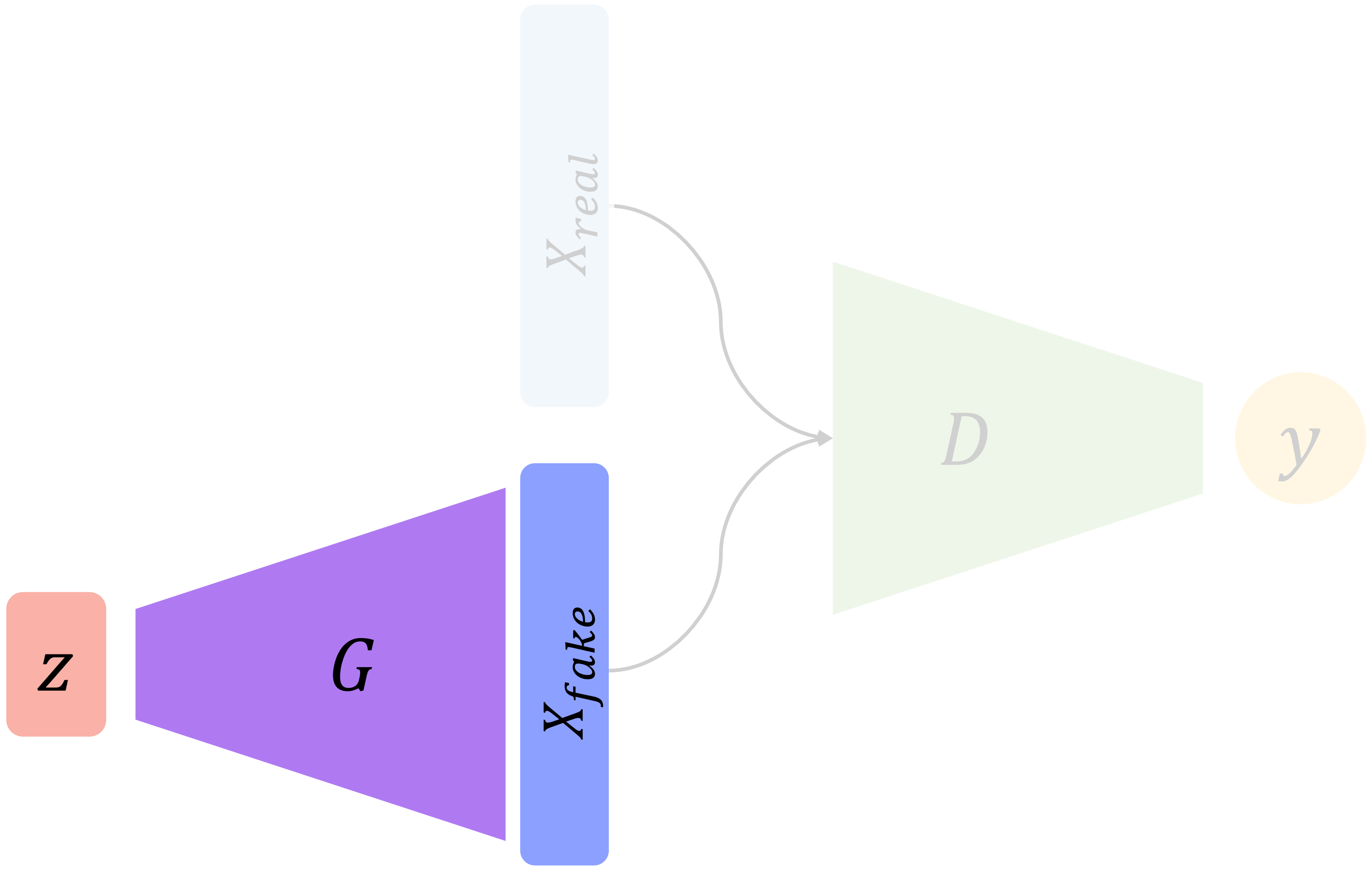
Generative Adversarial Networks (GANs)

G tries to synthesize fake images that fool D



$$\arg\min_{G} \mathbb{E}_{\mathbf{z},\mathbf{x}}[\underbrace{\log D(G(\mathbf{z}))}_{\textbf{Fake}} + \log(1 - D(\mathbf{x}))]$$

Generative Adversarial Networks (GANs)



After training, use the generator to synthesize new data

Optimal solution of the minimax game

Optimal discriminator cost:

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

$$\nabla_y [a \log y + b \log(1-y)] = 0 \implies y^* = \frac{a}{a+b} \quad \forall \quad [a, b] \in \mathbb{R}^2 \backslash [0, 0]$$

$$\implies D^*(x) = \frac{p_{\text{data}}(x)}{(p_{\text{data}}(x) + p_g(x))}$$

Generator Objective under Optimal Discriminator

$$V(G, D^*) = \mathbb{E}_{x\sim p_{\text{data}}}[\log D^*(x)] + \mathbb{E}_{x\sim p_g}[\log(1 - D^*(x))]$$

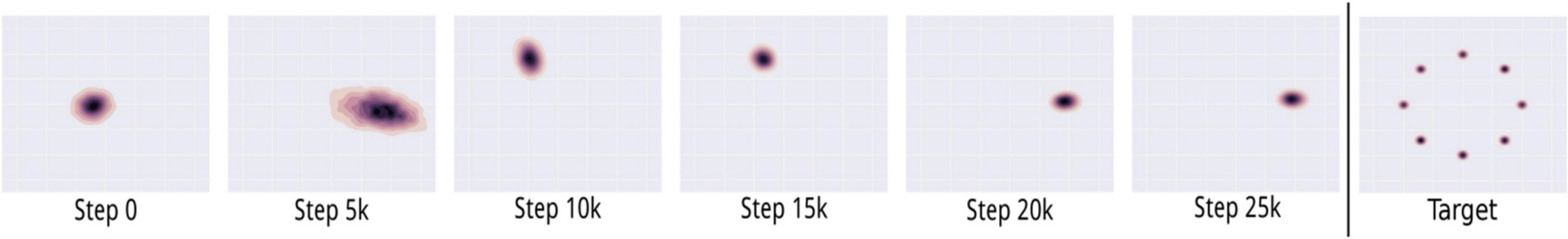
Generator Objective under Optimal Discriminator

$$
\begin{aligned}
V(G, D^*) &= \mathbb{E}_{x \sim p_{\text{data}}} [\log D^*(x)] + \mathbb{E}_{x \sim p_g} [\log(1 - D^*(x))] \\
&= \mathbb{E}_{x \sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(x)}\right] + \mathbb{E}_{x \sim p_g} \left[\log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)}\right] \\
&= -\log(4) + \underbrace{KL\left(p_{\text{data}} \middle\| \left(\frac{p_{\text{data}} + p_g}{2}\right)\right) + KL\left(p_{\text{g}} \middle\| \left(\frac{p_{\text{data}} + p_g}{2}\right)\right)}_{\text{(Jensen-Shannon Divergence (JSD) of } p_{\text{data}} \text{ and } p_g) \geq 0}
\end{aligned}
$$

$$
V(G^*, D^*) = -\log(4) \ \text{ when } p_g = p_{\text{data}}
$$

Mode Collapse

Try:
https://poloclub.github.io/ganlab/



Standard GAN training collapses when the true distribution is a mixture of gaussians (Figure from Metz et al 2016)

Back to our GAN objective

- Is it feasible to run the inner optimization to completion?
- For this specific objective, would it create problems if we were able to do so?

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

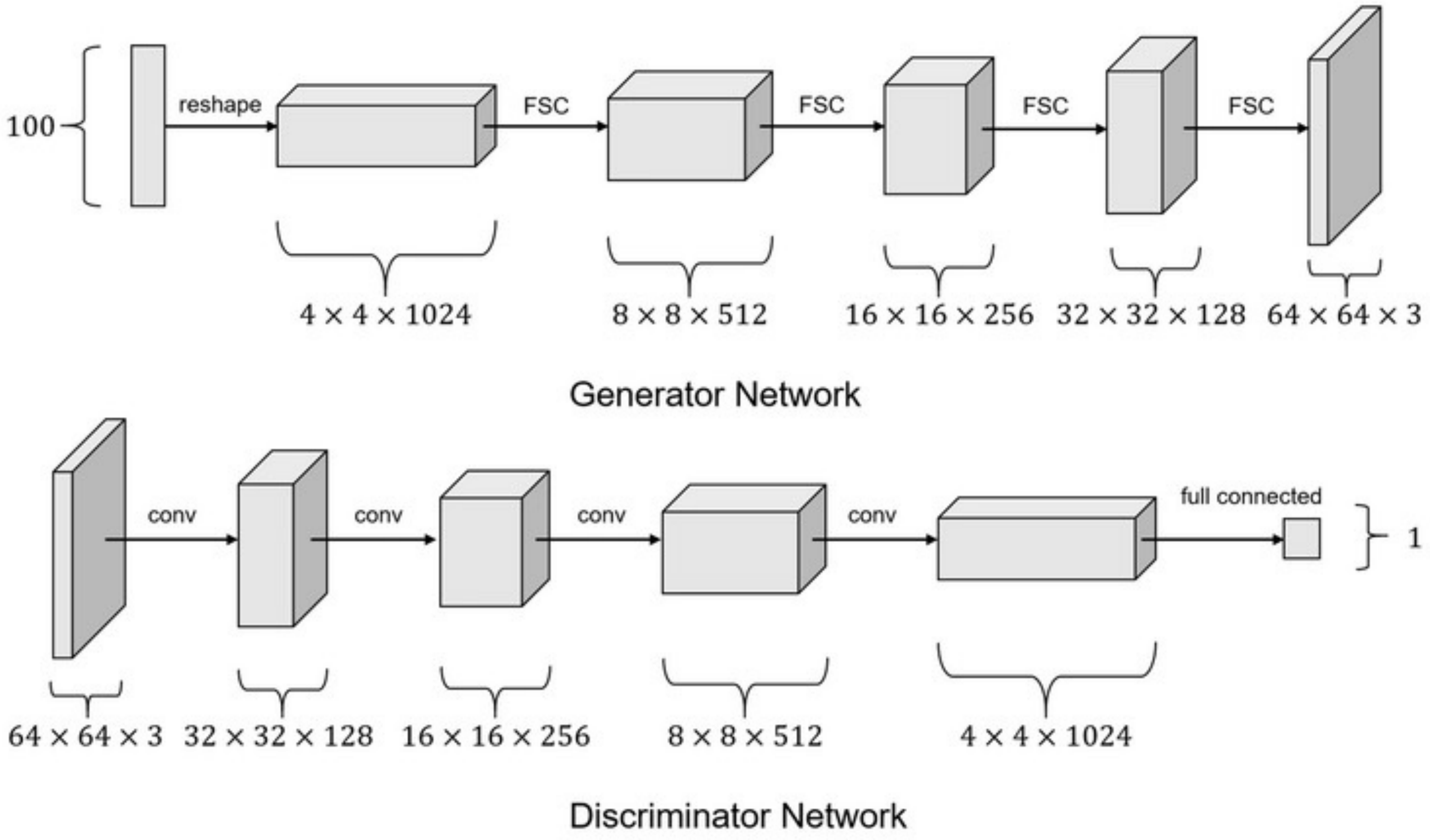
$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \boxed{\mathbb{E}_{z\sim q(z)}[\log(1 - D(G(z)))]}$$

$$\nabla_{\theta_G} V(D,G) = \nabla_{\theta_G} \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

- $\nabla_a \log(1-\sigma(a)) = \frac{-\nabla_a \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)\,(1-\sigma(a))}{1-\sigma(a)} = -\sigma(a) = -D\left(G(z)\right)$
- Gradient goes to 0 if D is confident, i.e. $D\left(G(z)\right) \to 0$
- Minimize $\boxed{-\mathbb{E}_{z\sim q(z)}\left[\log D\left(G(z)\right)\right]}$ for **Generator** instead (keep Discriminator as it is)

Slide from Svetlana Lazebnik

Deep Convolutional GAN (DCGAN)



- Remove max-pooling and mean-pooling
- Upsample using transposed convolutions in the generator
- Downsample with strided convolutions and average pooling

DCGAN, Radford et al., 2016

DCGAN - Key Results

- Good samples on datasets with 3M images (Faces, Bedrooms) for the first time

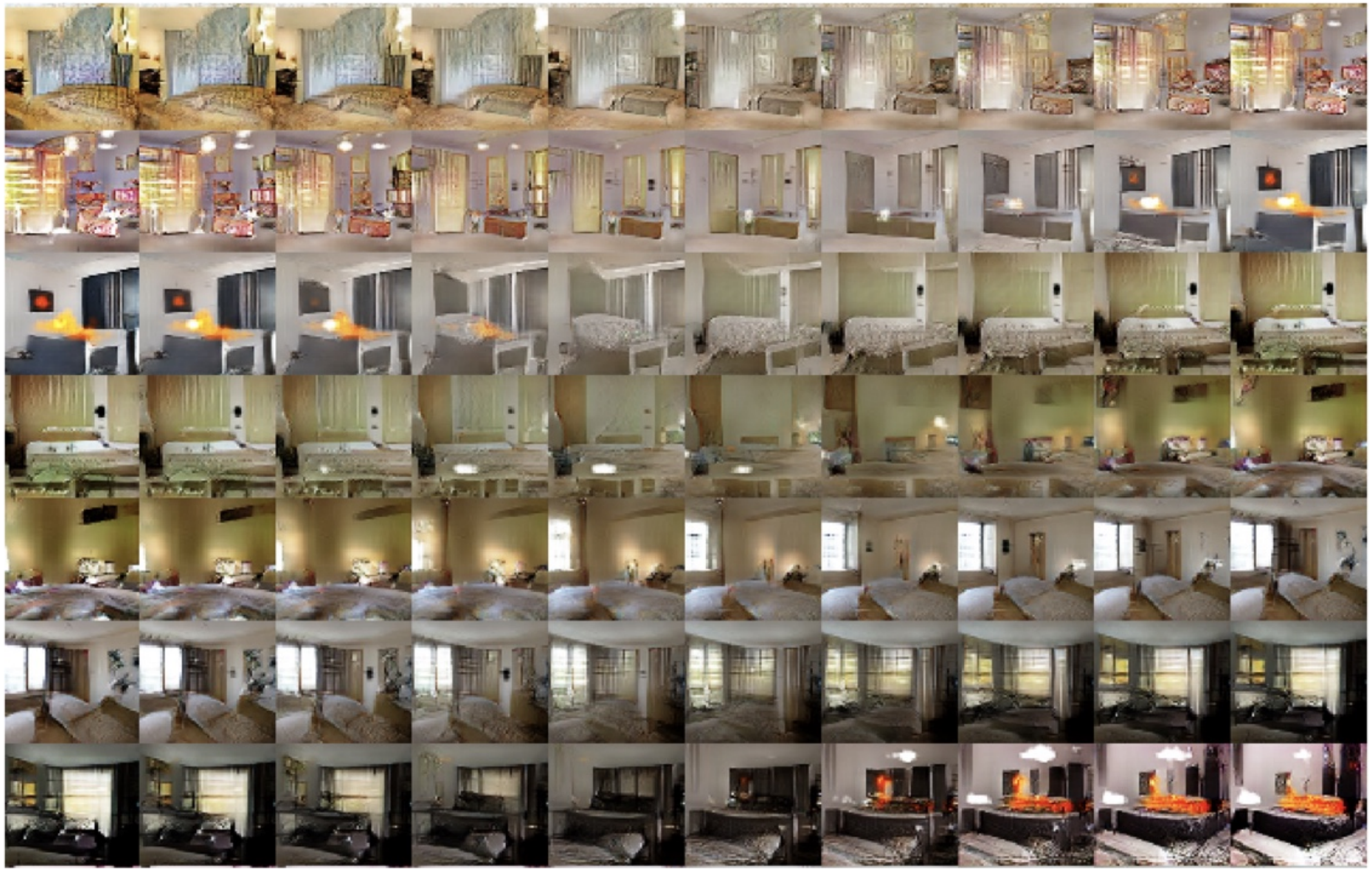
DCGAN, Radford et al., 2016

DCGAN - Key Results

DCGAN, Radford et al., 2016

DCGAN - Key Results

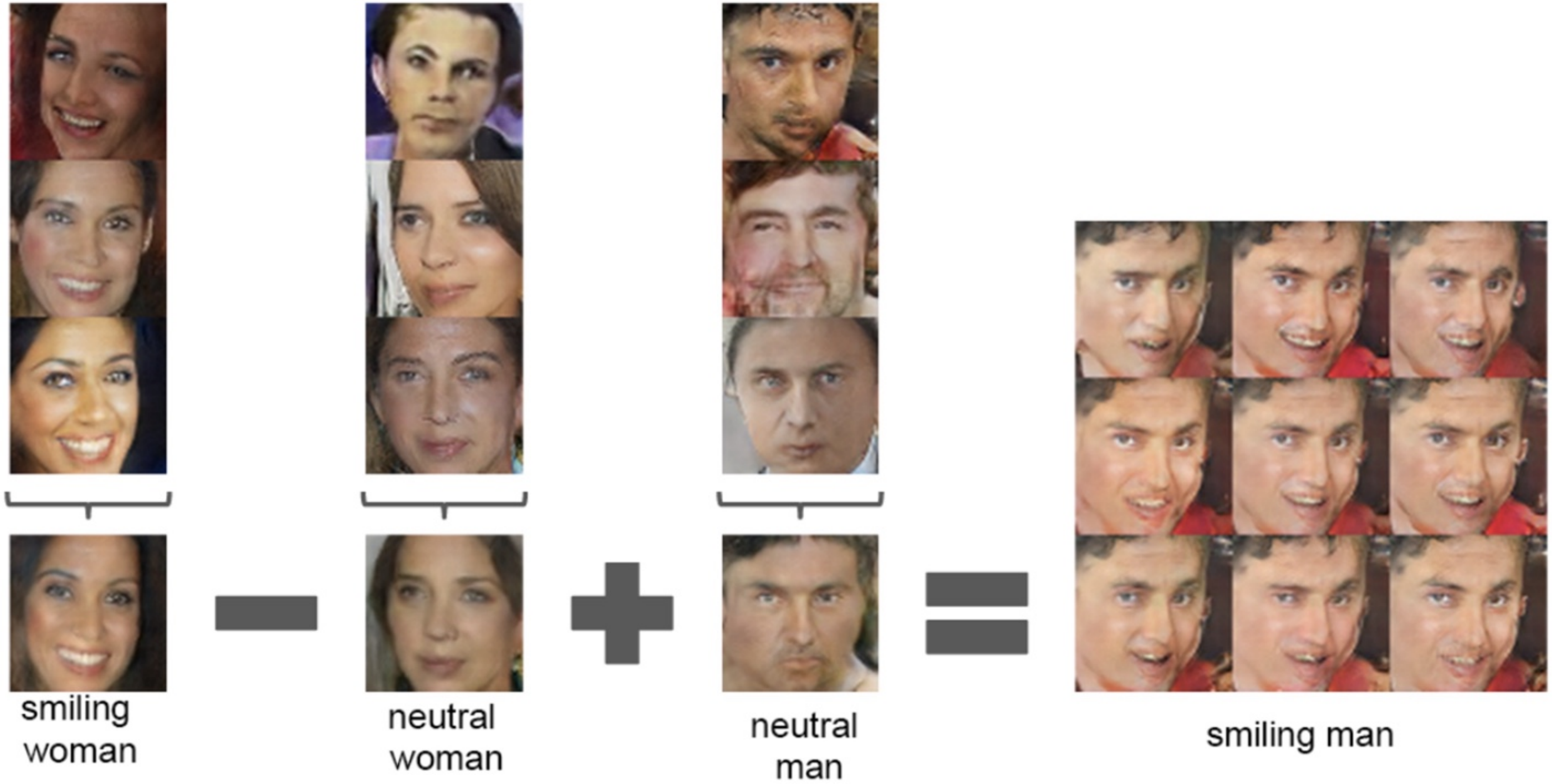
- Smooth interpolations in high dimensions



DCGAN, Radford et al., 2016

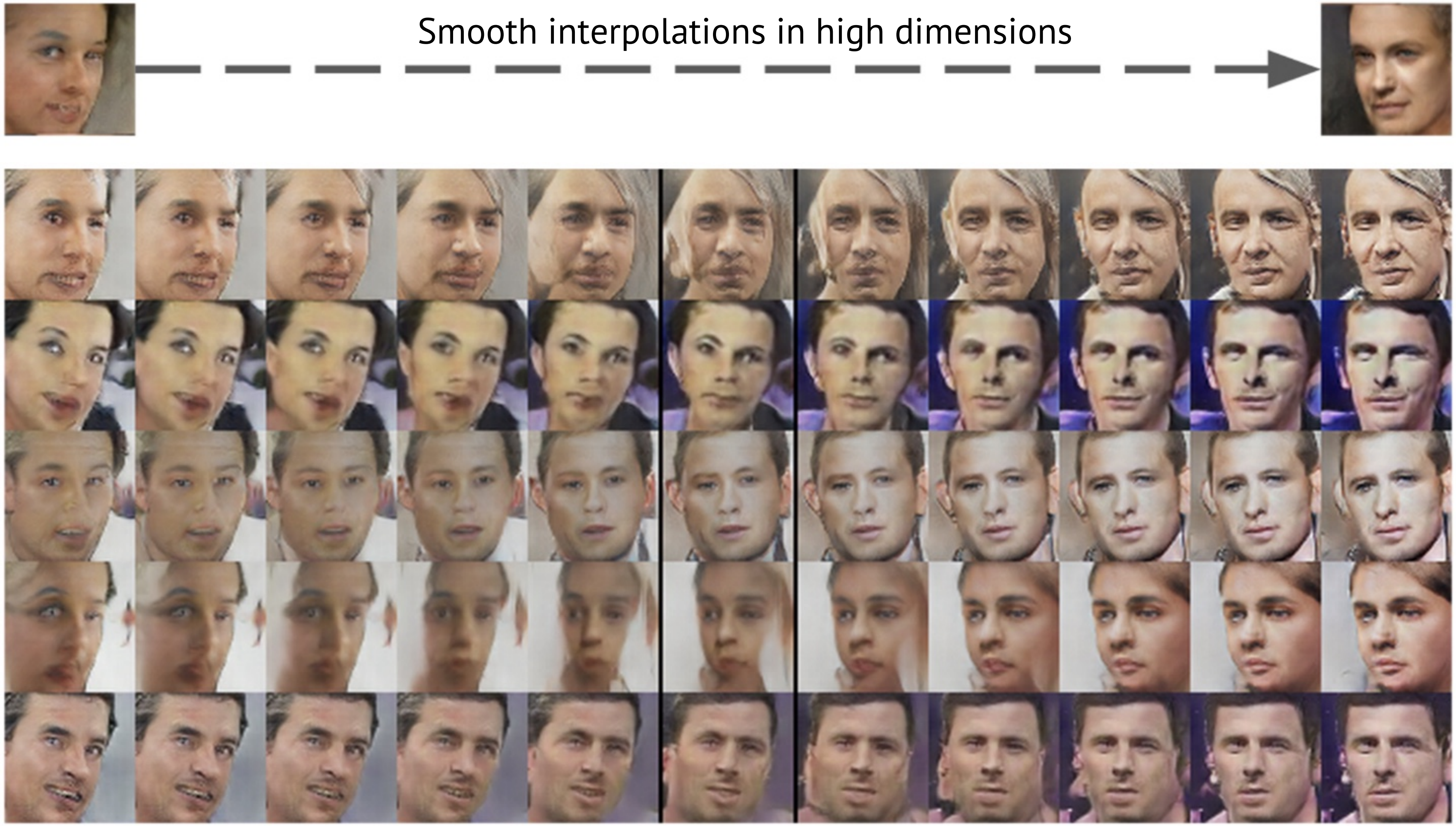
DCGAN - Key Results

- Vector Arithmetic in Latent Space (z)



DCGAN, Radford et al., 2016

DCGAN - Key Results

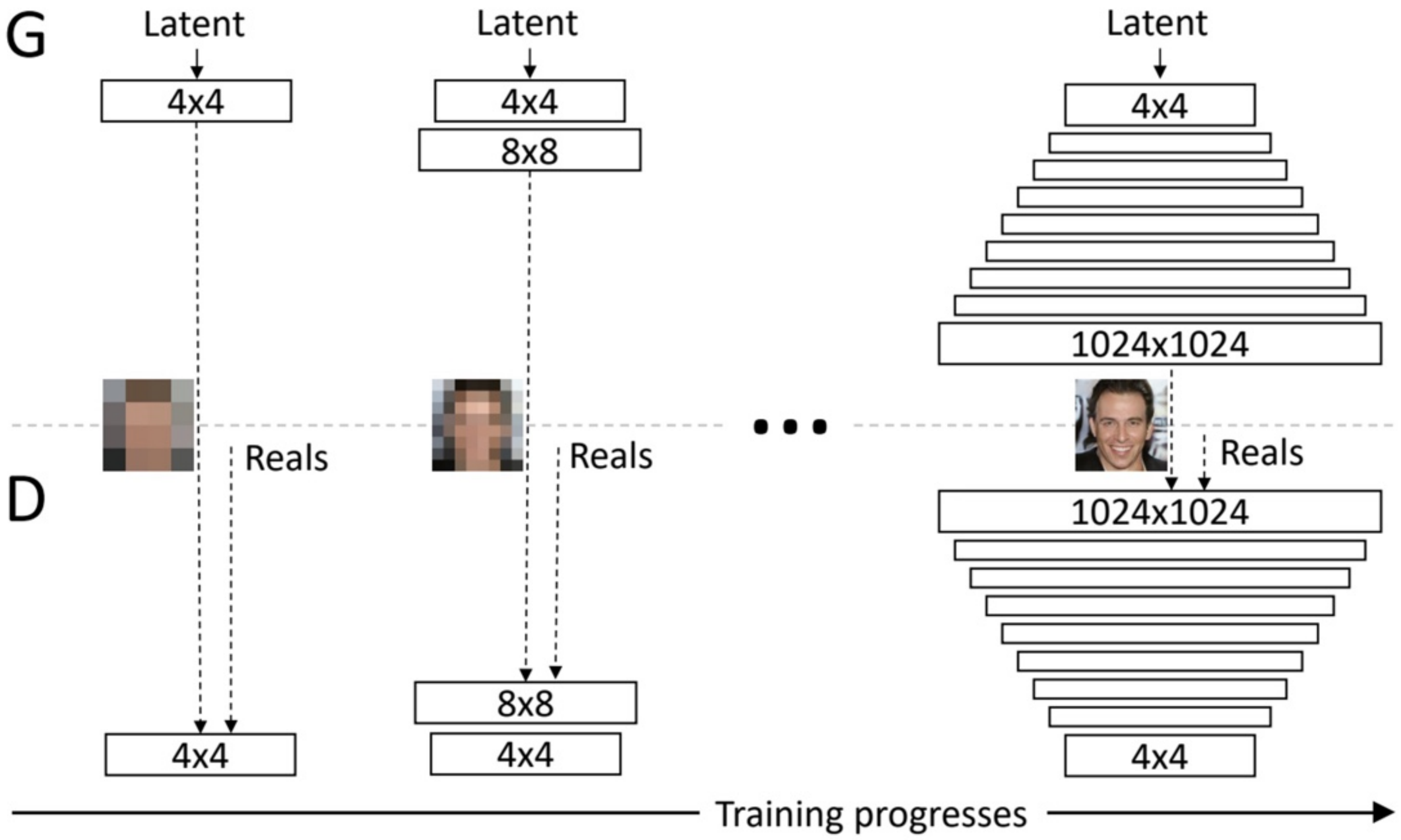


DCGAN, Radford et al., 2016

GANs: Progress

Progressive growing of GANs

First high quality samples from any generative models



Progressive GANs, Karras et al 2017

Progressive growing of GANs

Progressive GANs, Karras et al 2017

Progressive growing of GANs

Mao et al. (2016b) (128 × 128) Gulrajani et al. (2017) (128 × 128) Our (256 × 256)

Karras et al 2017

Progressive growing of GANs

Progressive growing of GANs

Progressive GANs, Karras et al 2017

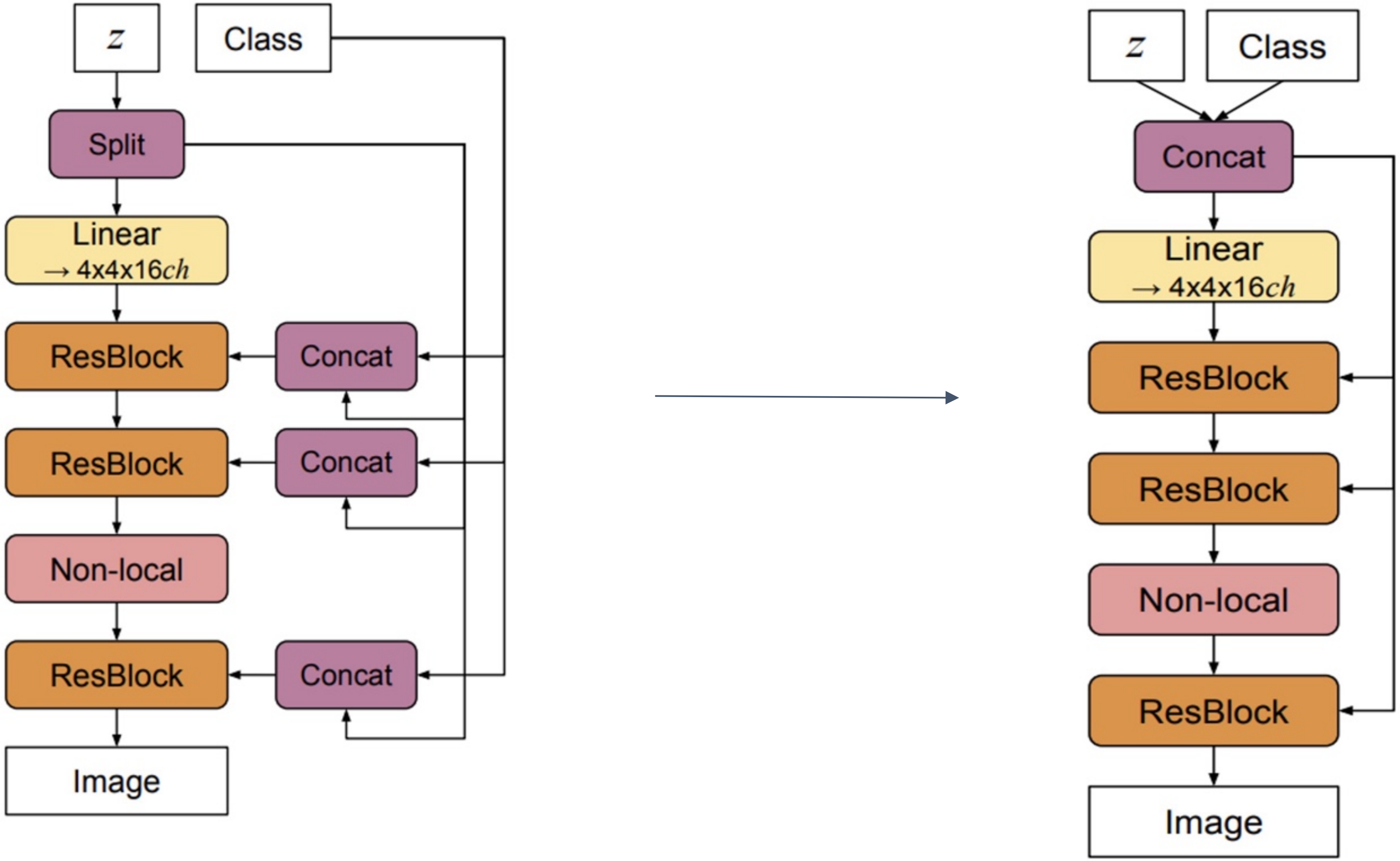
BigGAN

BigGANs, Brock et al 2018

BigGAN

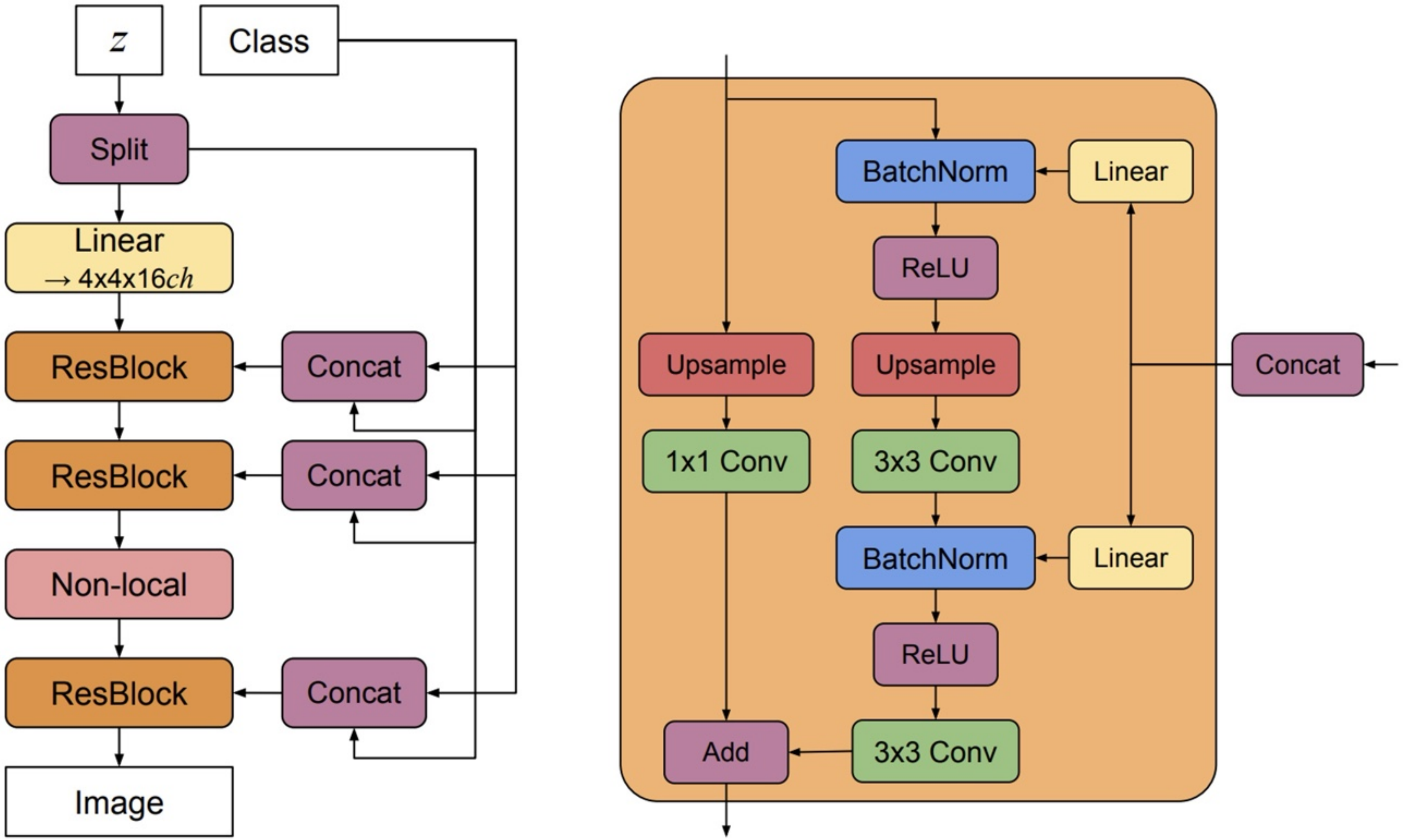
BigGANs, Brock et al 2018

BigGAN and BigGAN-deep



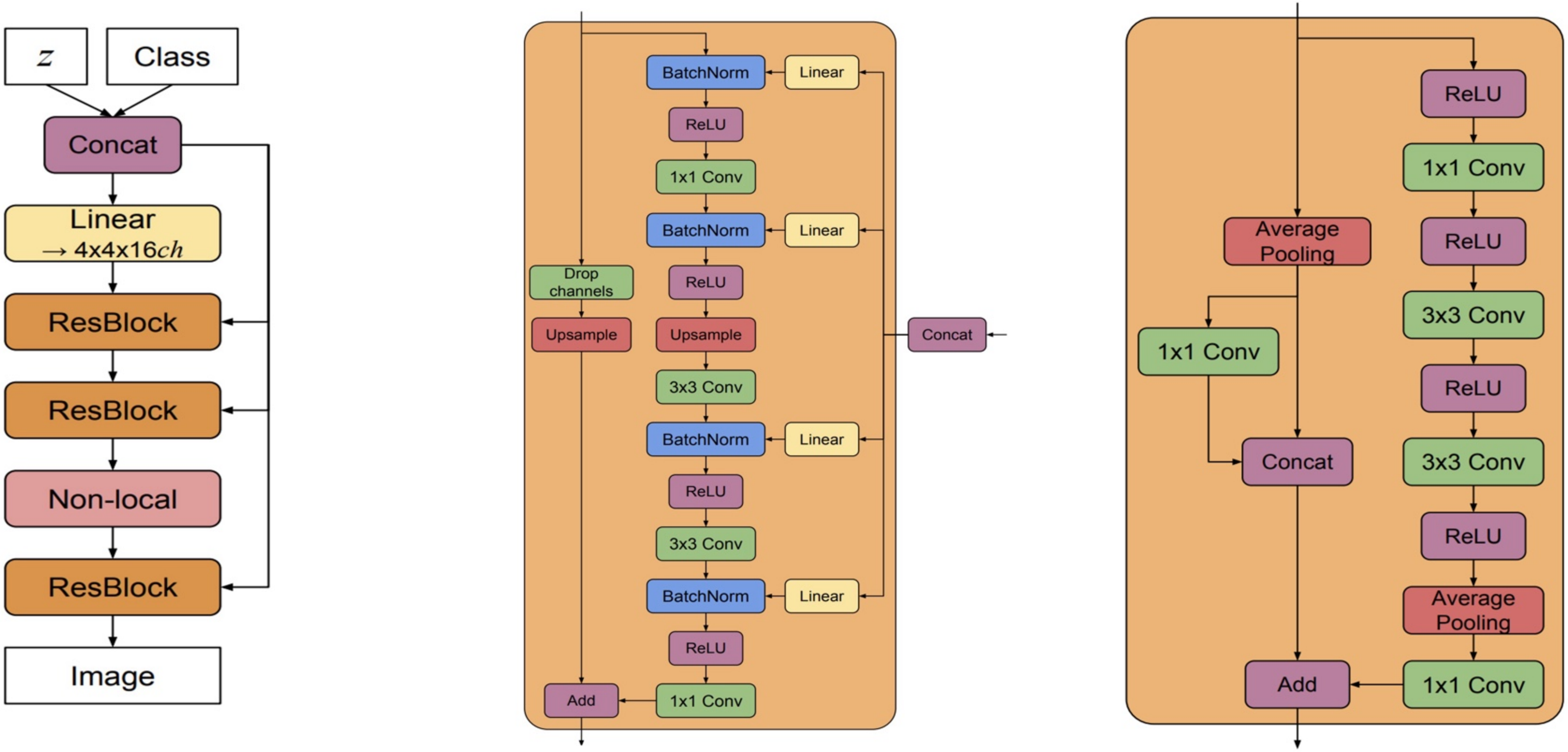
BigGANs, Brock et al 2018

BigGAN



BigGANs, Brock et al 2018

BigGAN-deep



BigGANs, Brock et al 2018

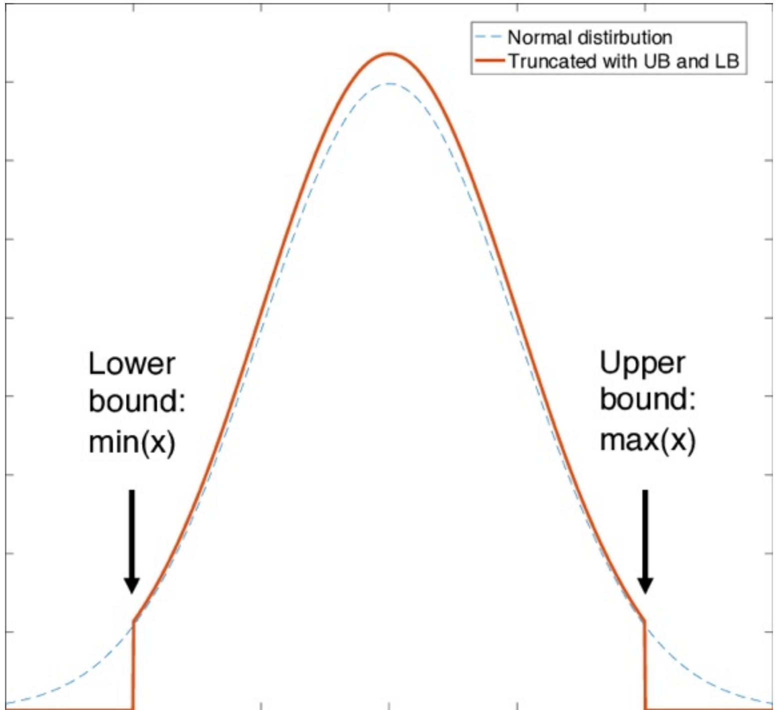
BigGAN

- Salient bits
  - Increase your batch size (as much as you can)
  - Use Cross-Replica (Sync) Batch Norm
  - Increase your model size
  - Wider helps as much as deeper
  - Fuse class information at all levels
  - Hinge Loss
  - Orthonormal regularization & Truncation Trick (on z)

BigGANs, Brock et al 2018

BigGAN - Truncation Trick

(a) (b)



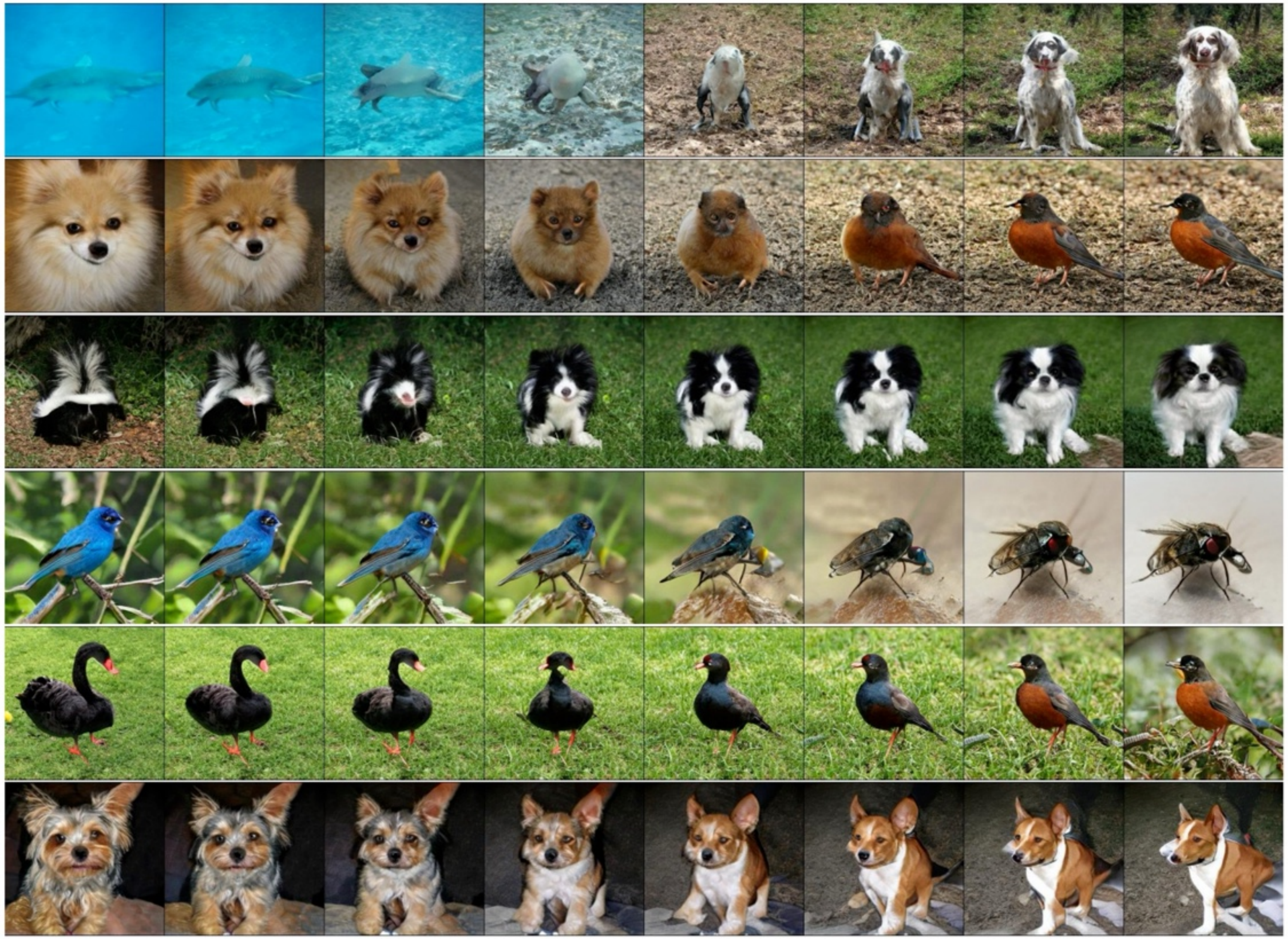
BigGANs, Brock et al 2018

BigGAN

(a) 128×128 (b) 256×256 (c) 512×512 (d)

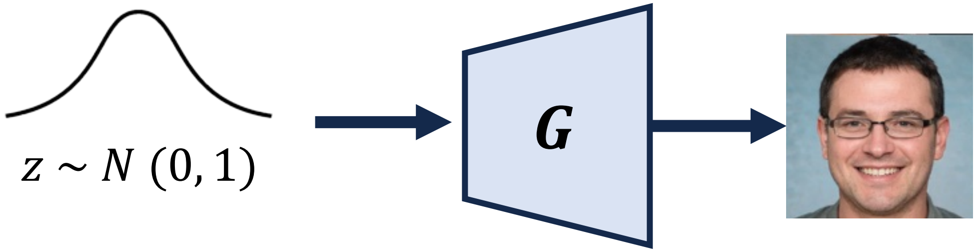
BigGANs, Brock et al 2018

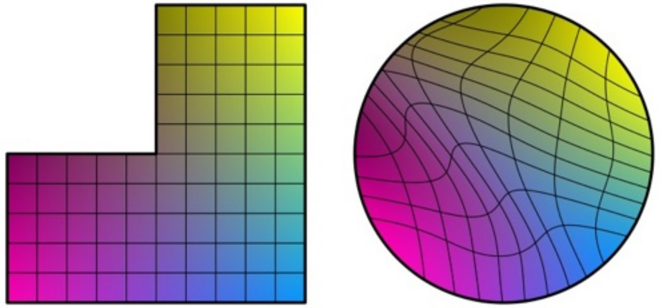
BigGAN



BigGANs, Brock et al 2018

StyleGAN



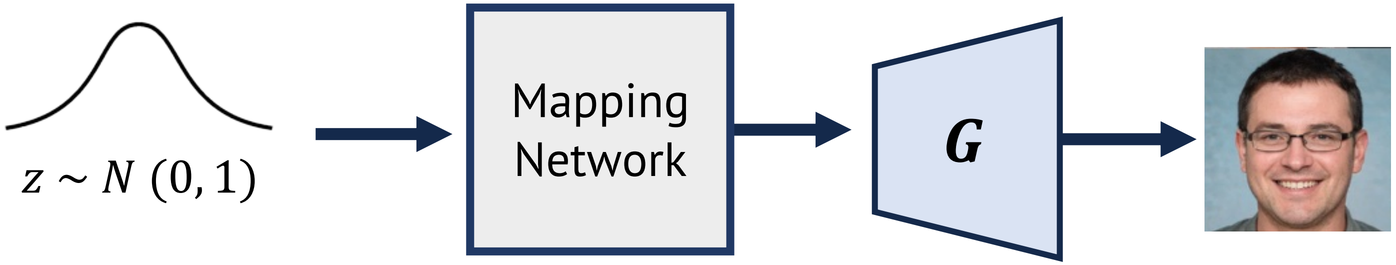


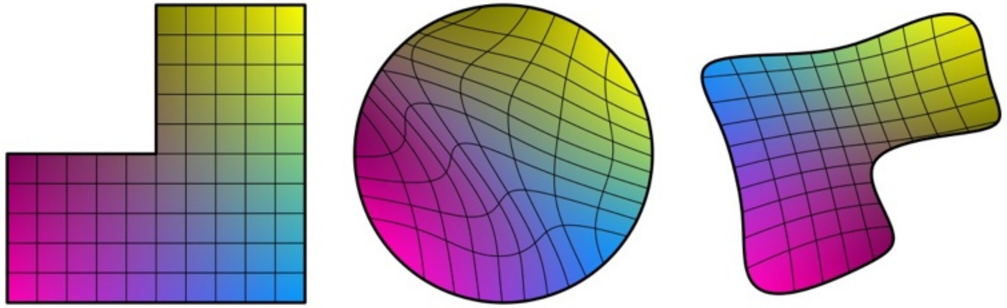
(a) Distribution of features in training set

(b) Mapping from $\mathcal{Z}$ to features

Ref: StyleGAN, Karras et al. CVPR '19; StyleGAN2, Karras et al. CVPR '20

StyleGAN

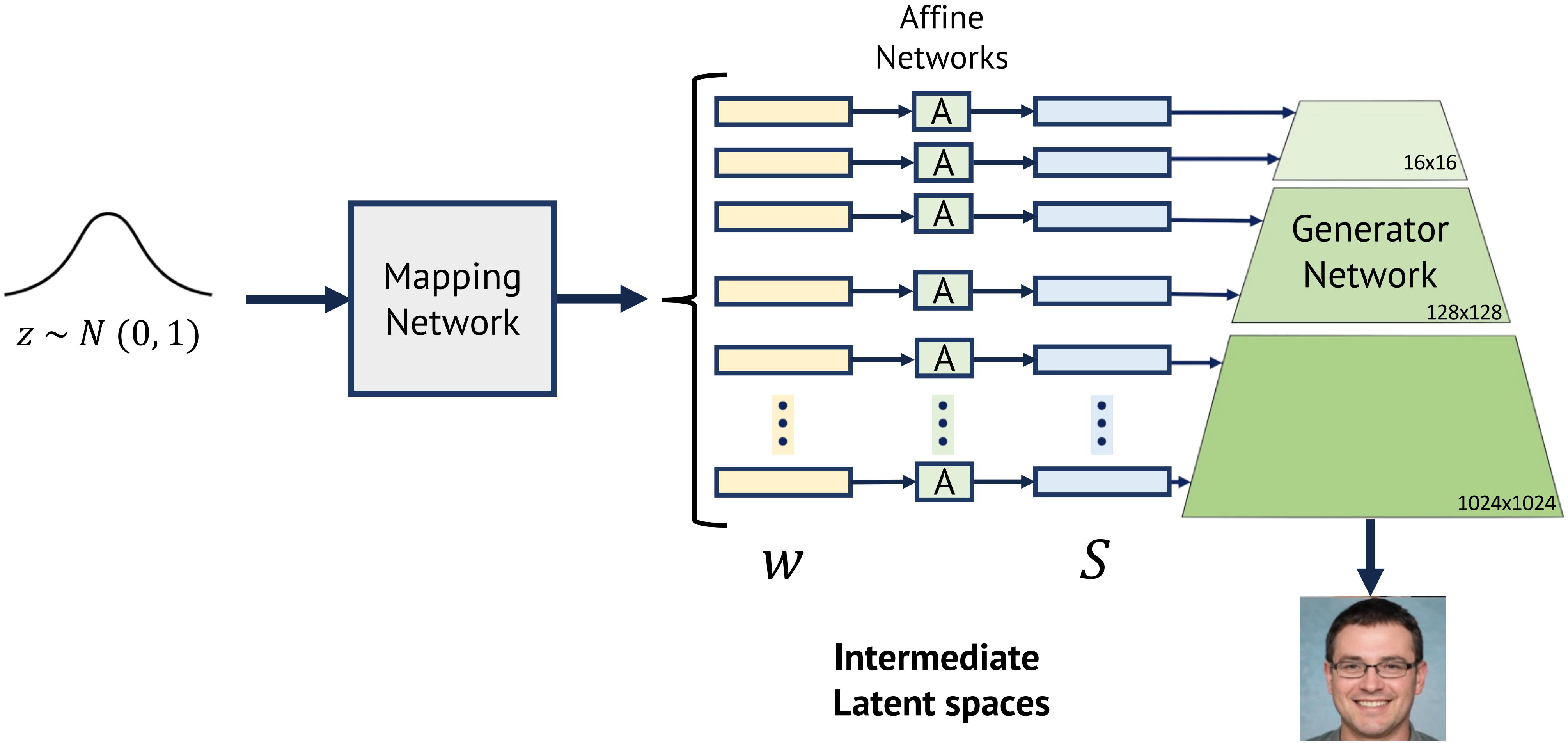




(a) Distribution of features in training set (b) Mapping from $\mathcal{Z}$ to features (c) Mapping from $\mathcal{W}$ to features

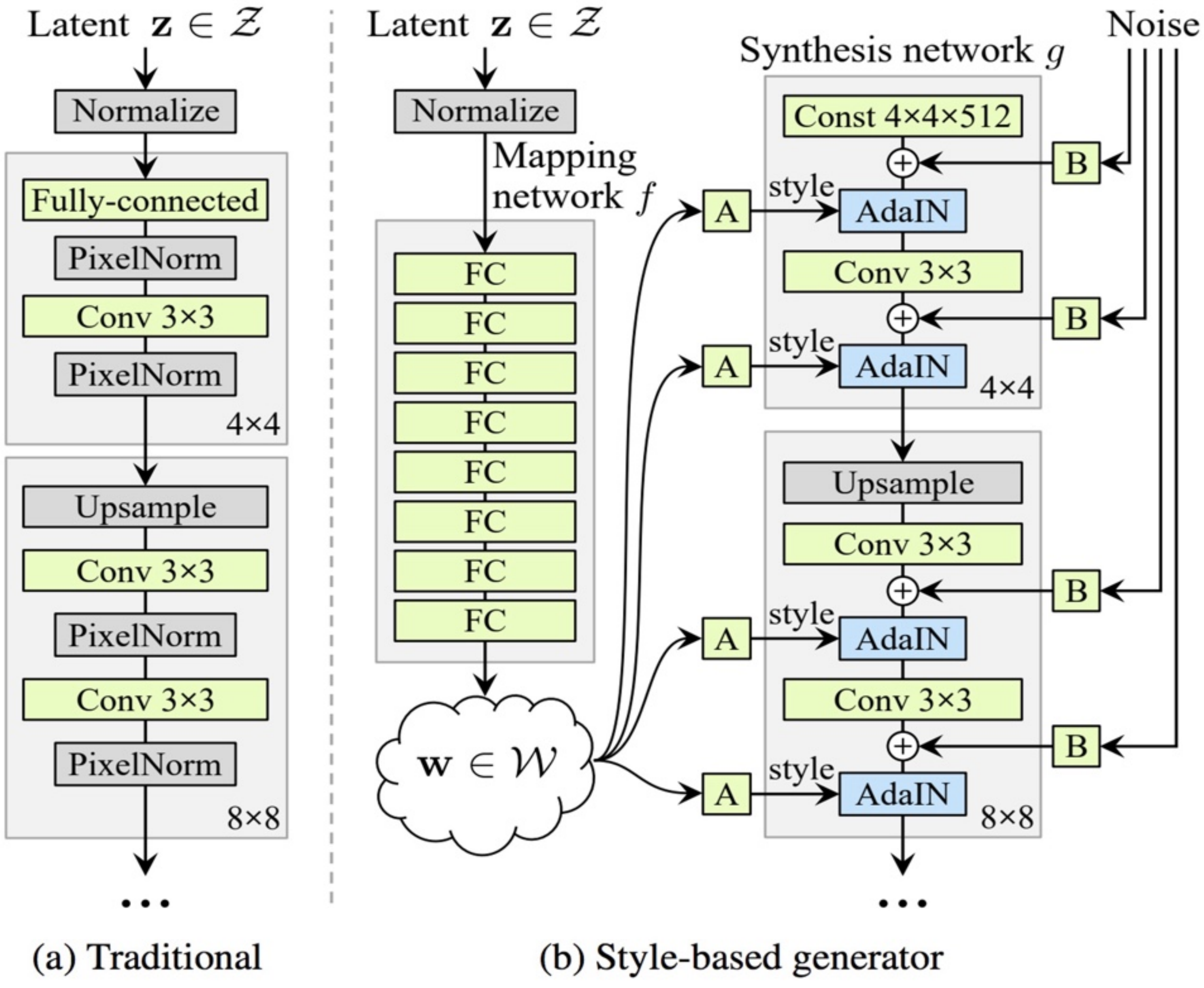
Ref: StyleGAN, Karras et al. CVPR ’19; StyleGAN2, Karras et al. CVPR ‘20

StyleGAN



Ref: StyleGAN, Karras et al. CVPR '19; StyleGAN2, Karras et al. CVPR '20

StyleGAN



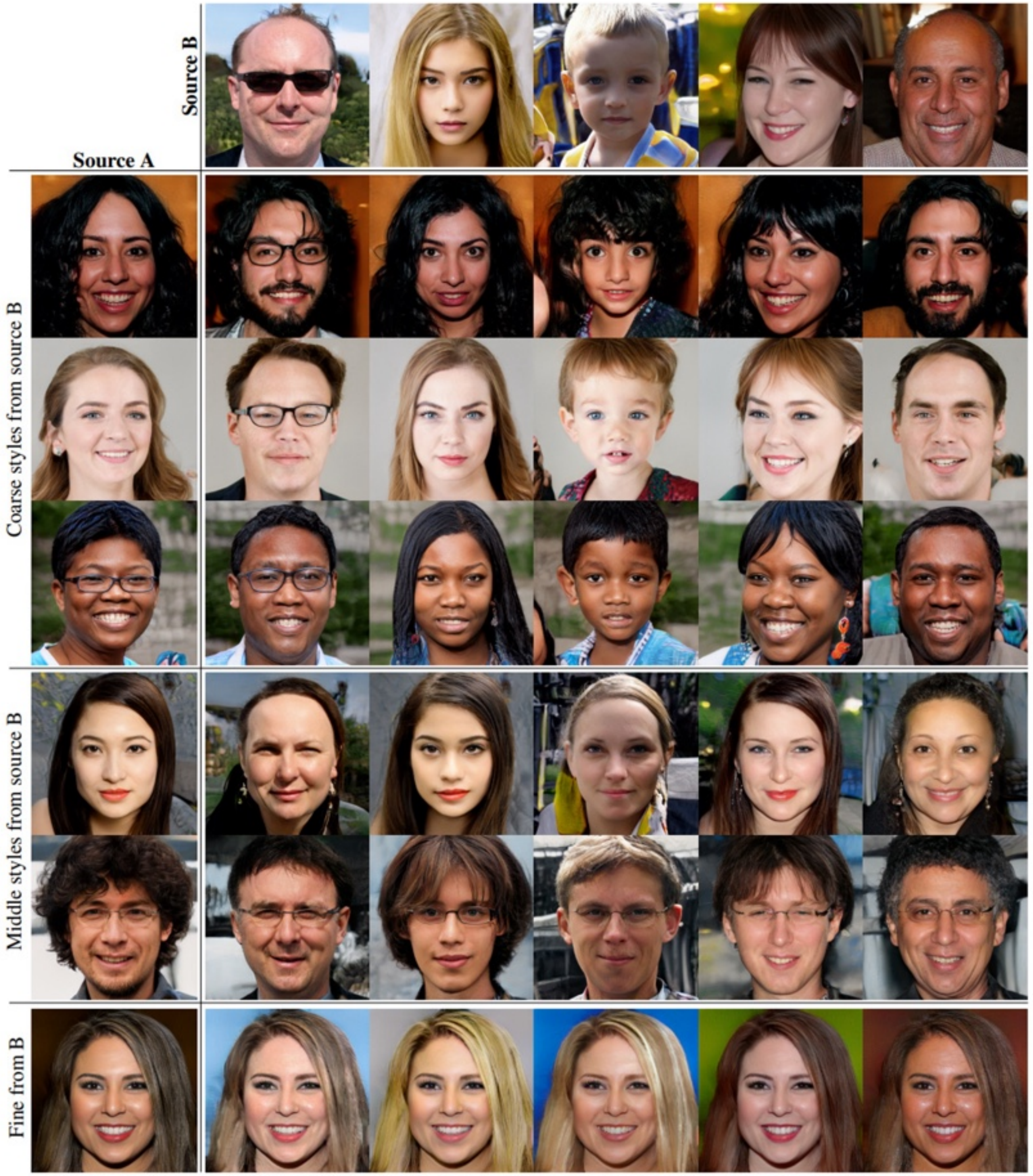
$$\mathrm{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

Instance normalization =

normalize per channel per sample

Ref: StyleGAN, Karras et al. CVPR '19; StyleGAN2, Karras et al. CVPR '20

StyleGAN – "Style" Transfer

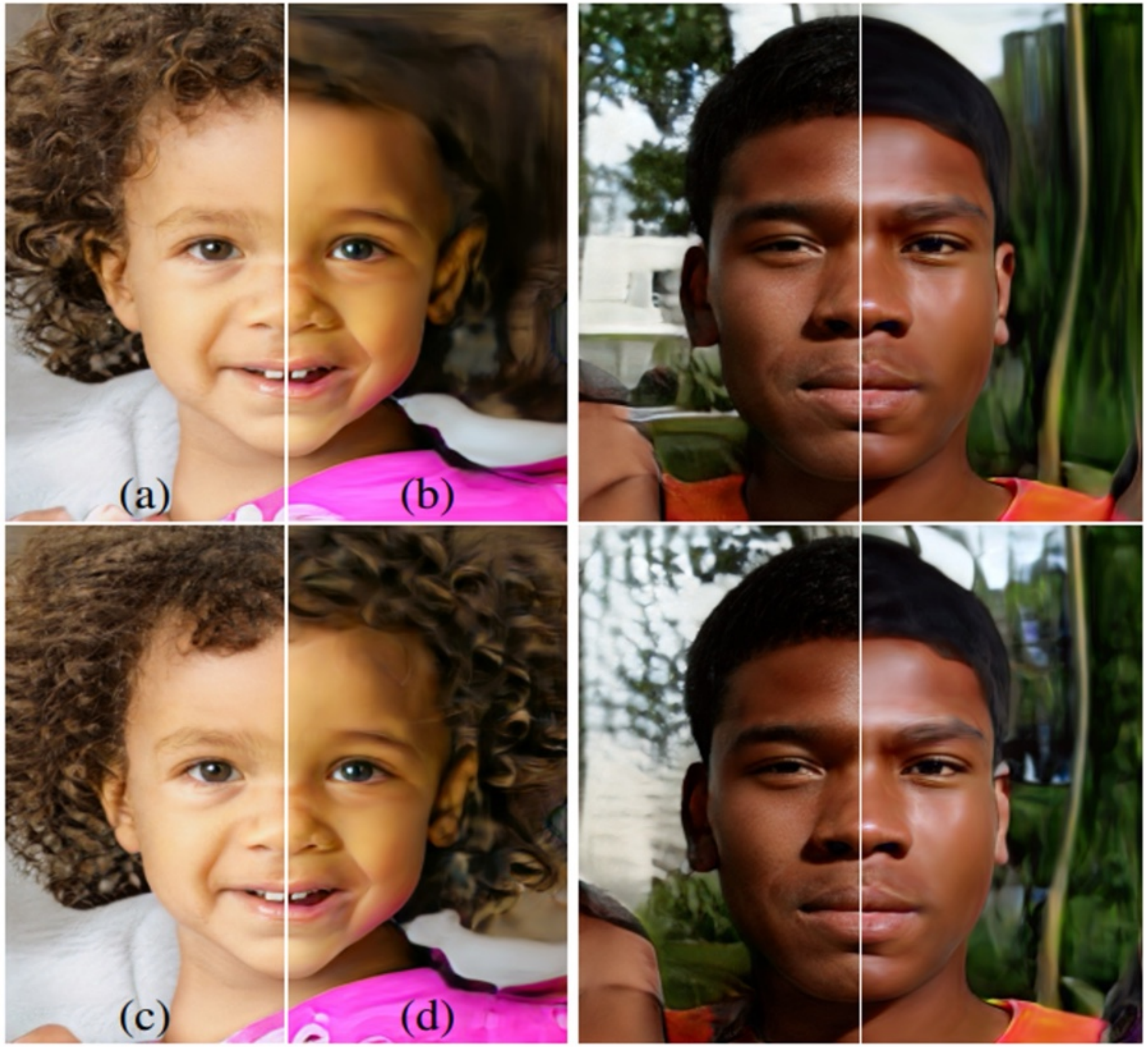


Ref: StyleGAN, Karras et al. CVPR '19;
StyleGAN2, Karras et al. CVPR '20

StyleGAN

Ref: StyleGAN, Karras et al. CVPR '19; StyleGAN2, Karras et al. CVPR '20

StyleGAN - Effect of adding noise

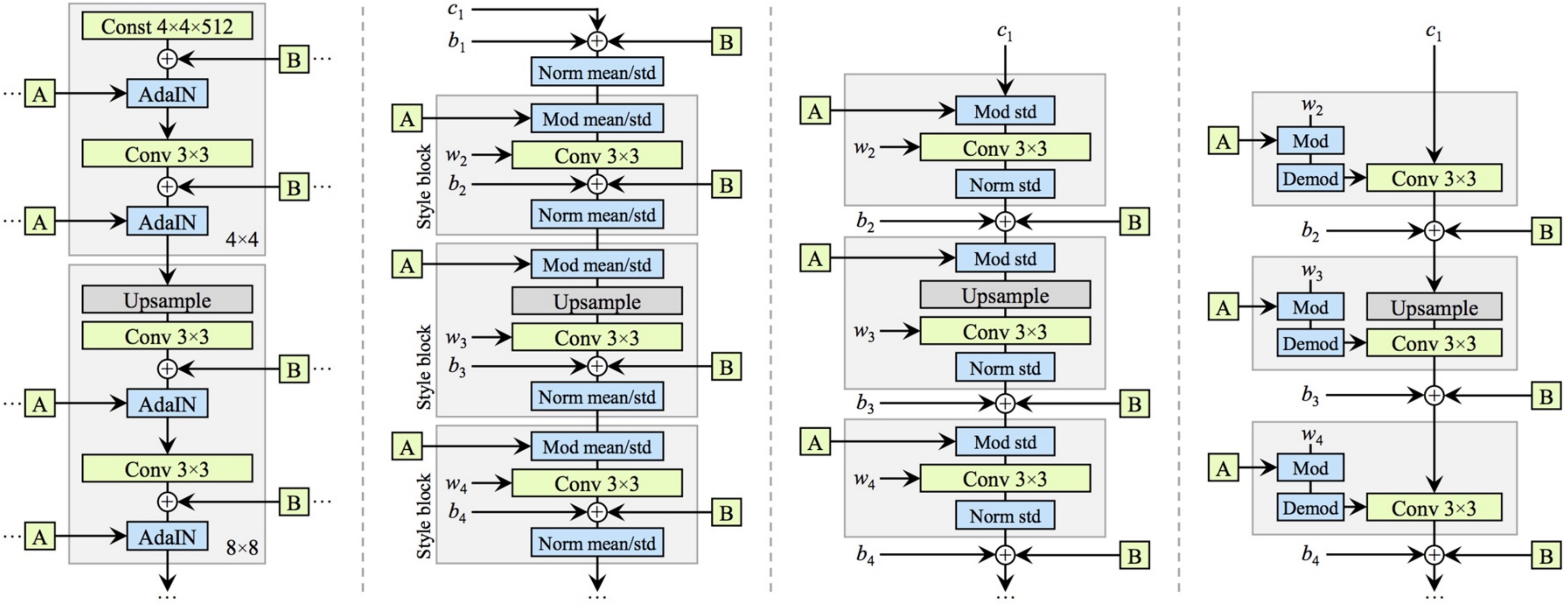


StyleGAN-v2

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.

Ref: StyleGAN, Karras et al. CVPR '19; StyleGAN2, Karras et al. CVPR '20

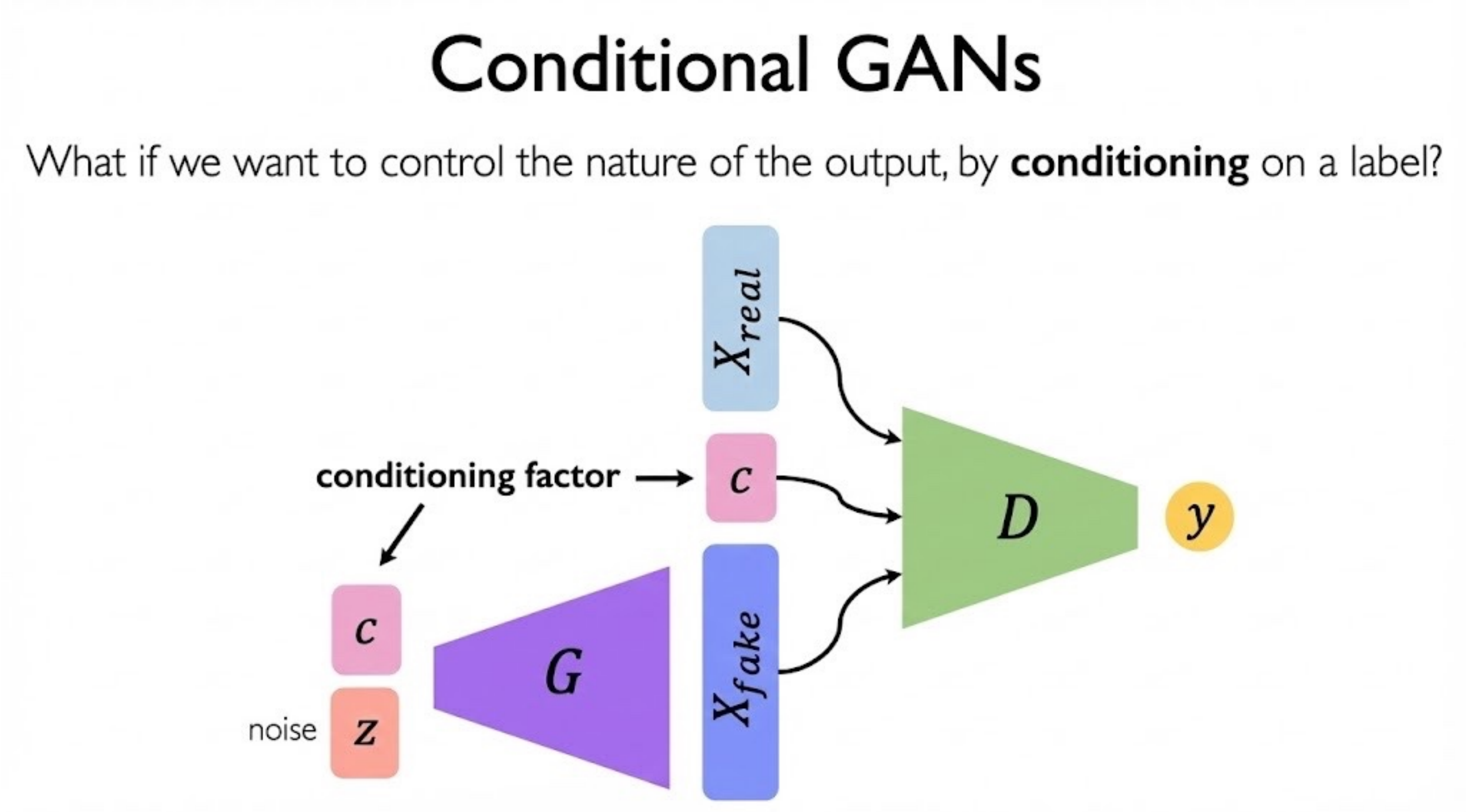
StyleGAN-v2



(a) StyleGAN (b) StyleGAN (detailed) (c) Revised architecture (d) Weight demodulation

Ref: StyleGAN, Karras et al. CVPR '19; StyleGAN2, Karras et al. CVPR '20

Conditional Generation using GANs



Slide from © Alexander Amini and Ava Amini
MIT 6.S191: Introduction to Deep Learning

Pix2Pix: Conditional Image-to-Image Generation

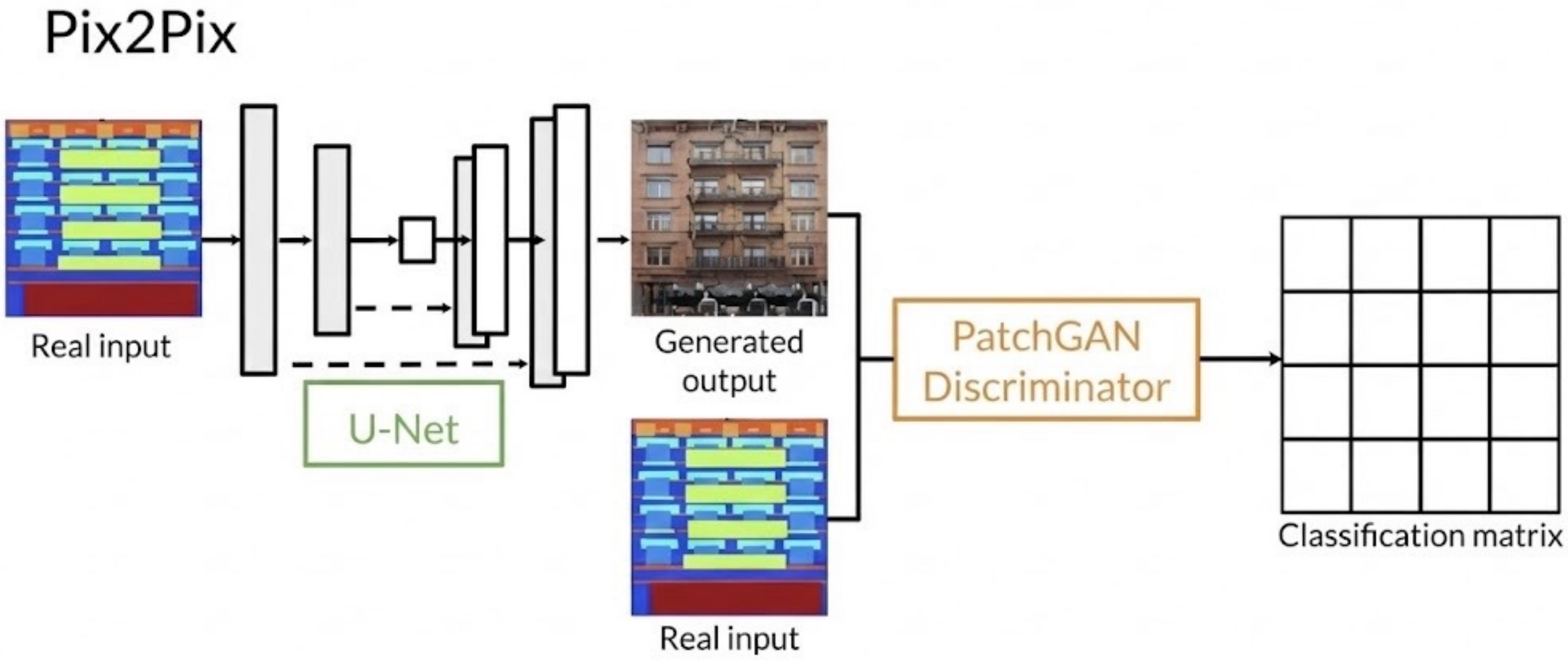
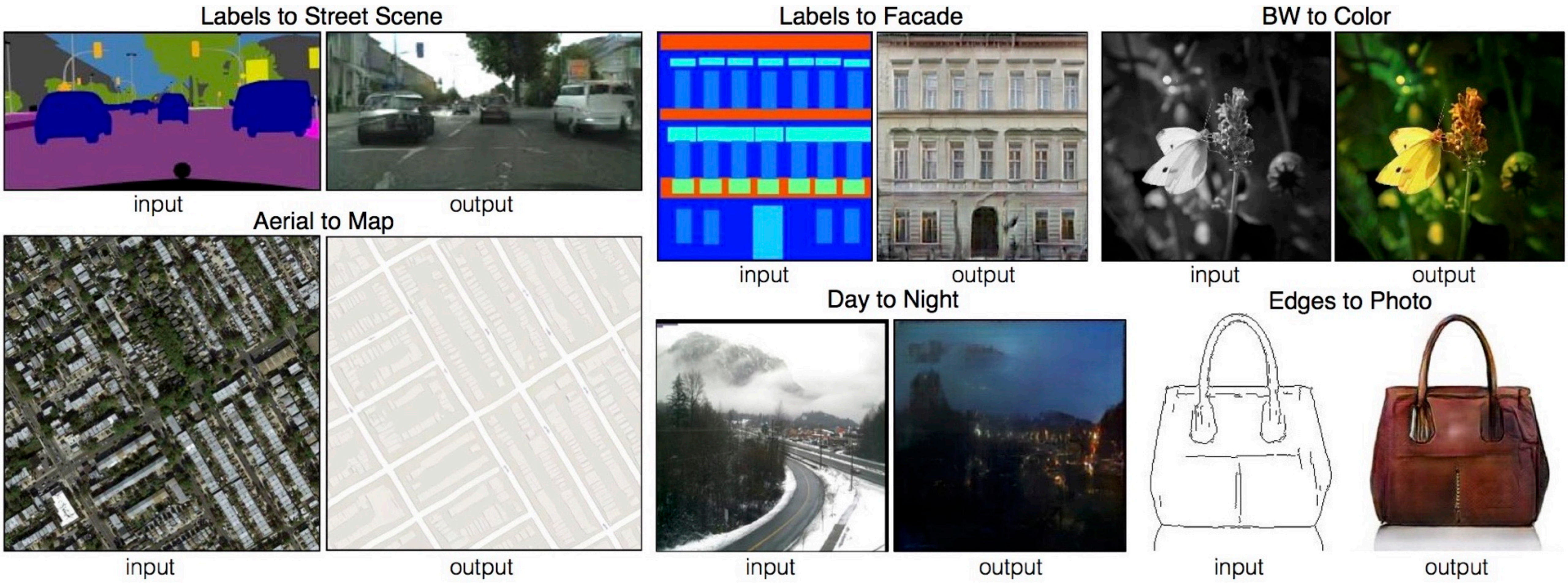
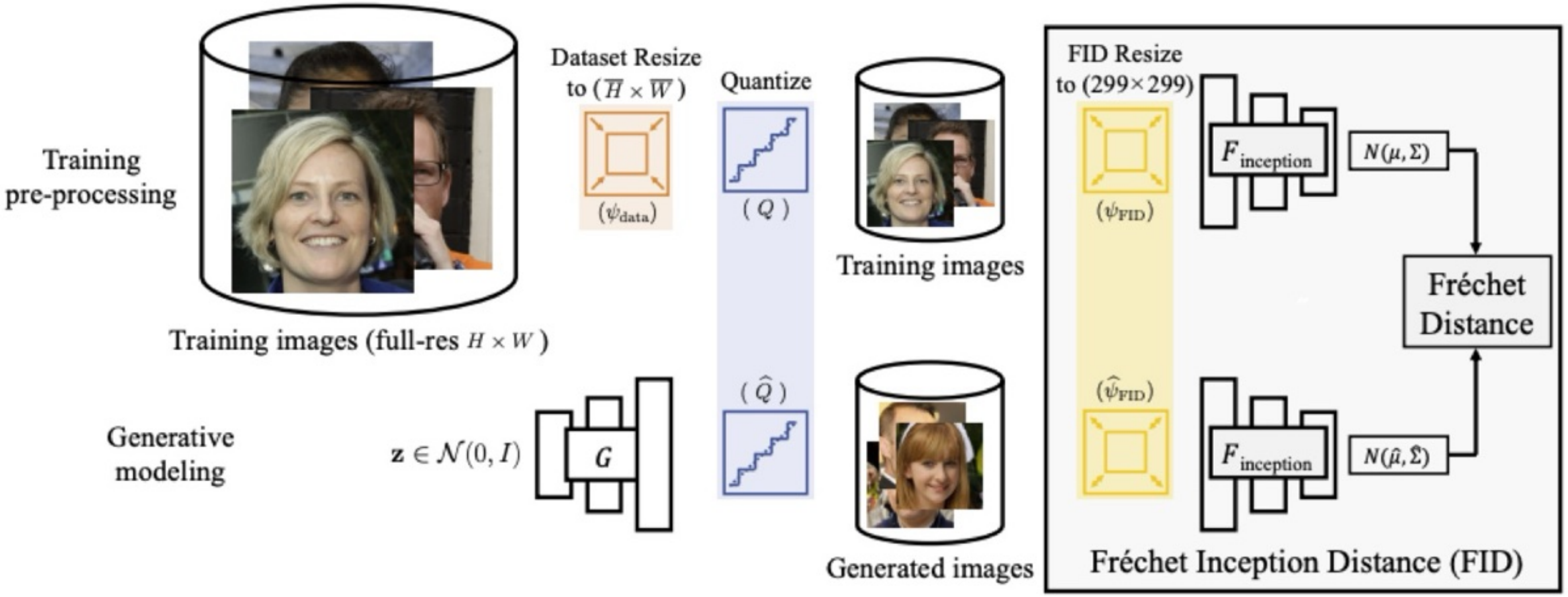


Image available from: https://arxiv.org/abs/1611.07004

Pix2Pix: Results



How to evaluate performance of GAN?



Fréchet Inception Distance (FID)

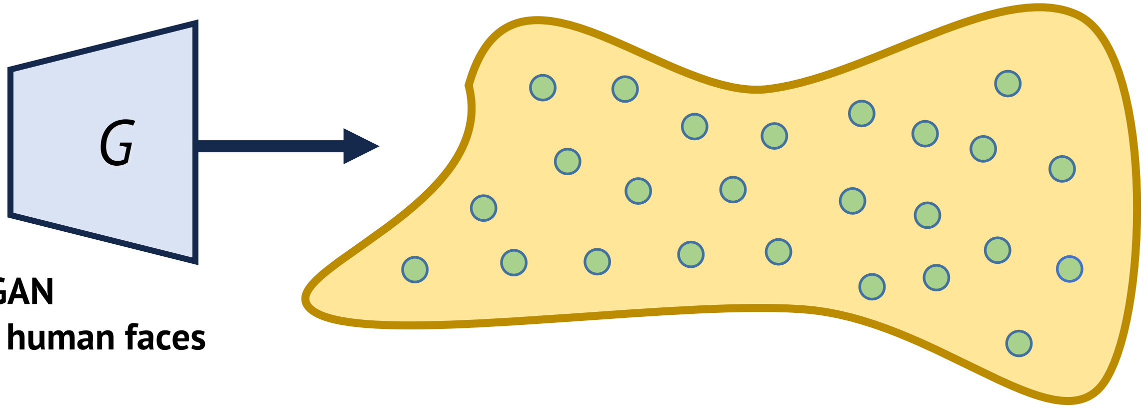
$$\mathbf{FID} = ||\mu - \hat{\mu}||_2^2 + \mathbf{Tr}(\Sigma + \hat{\Sigma} - 2(\Sigma\hat{\Sigma})^{1/2})$$

Slide from Jun-Yun Zhu, CMU

GANs: Applications

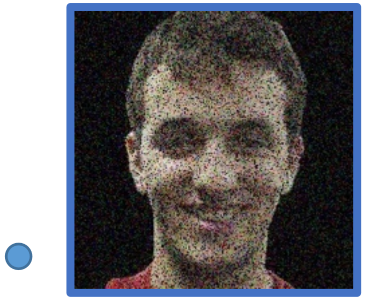
- Rich representation of the image data
- Smooth latent space with interpolation capabilities
- Fast, single step inference

Image Restoration using GANs



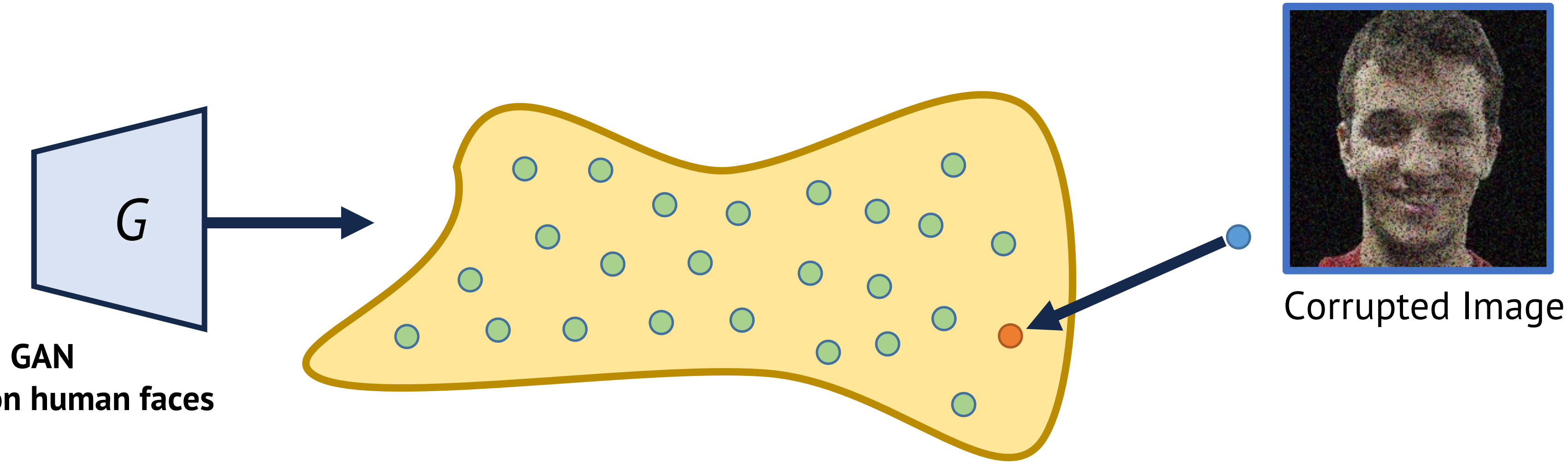
GAN
trained on human faces

GAN Image Space ≈ Set of all human faces



Corrupted Image

Image Restoration using GANs



GAN
trained on human faces

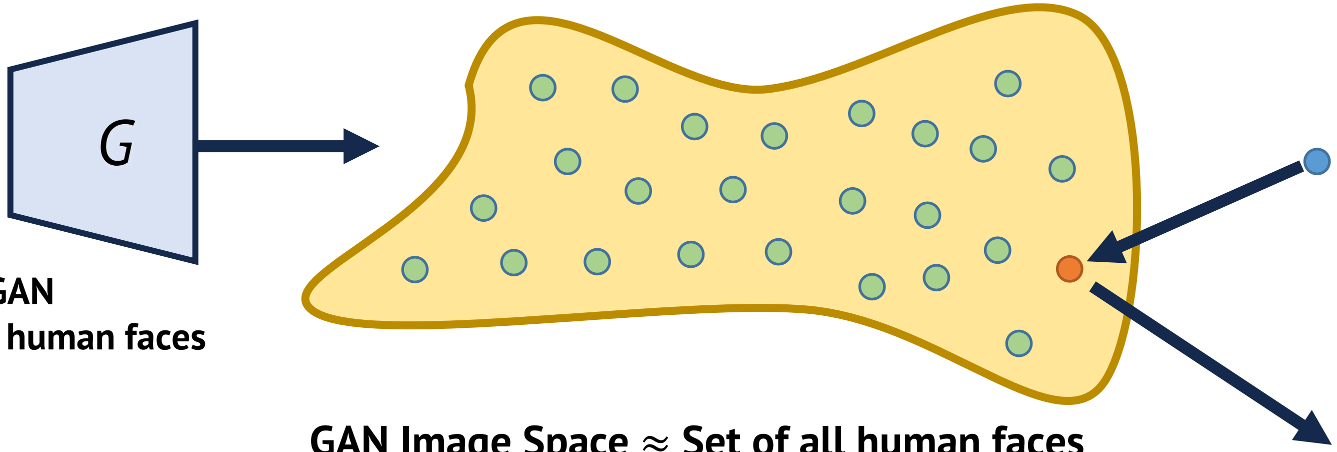
Corrupted Image

Find the closest image in the GAN Image Space

GAN Image Space ≈ Set of all human faces

Solving Inverse Problems using GAN priors, V. Shah et al., ICASSP '18

Image Restoration using Generative Models



GAN
trained on human faces

GAN Image Space ≈ Set of all human faces

Corrupted Image

Find the closest image in the GAN Image Space

$$w^* = \underset{z}{\operatorname{argmin}} \|y - G(w)\|_2$$

$\sim G(w^*)$

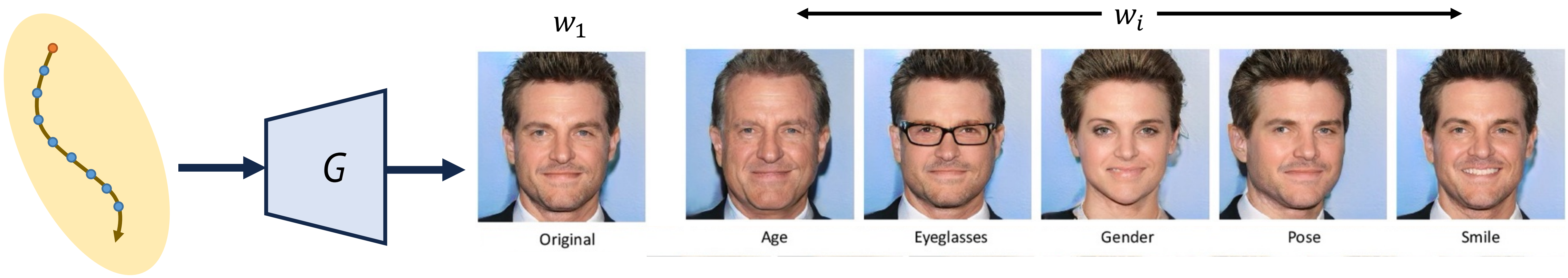
Restored Image

Provable algorithms for solving:

- Image Restoration
- Image Denoising, Inpainting etc.

Solving Inverse Problems using GAN priors, V. Shah et al., ICASSP '18

GAN Inversion is necessary for Image Editing



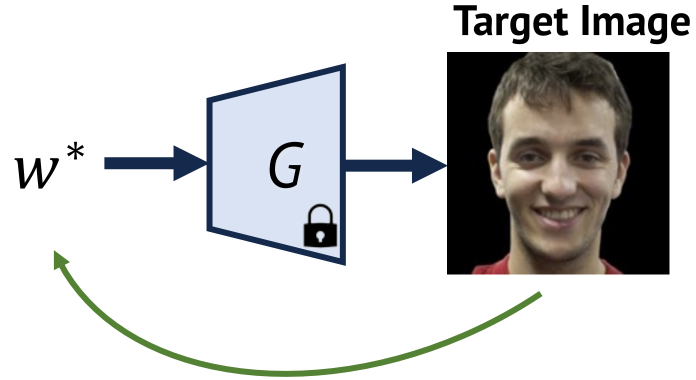
$w_1 \rightarrow w_2 \rightarrow w_3$ ...

Edit a real image?

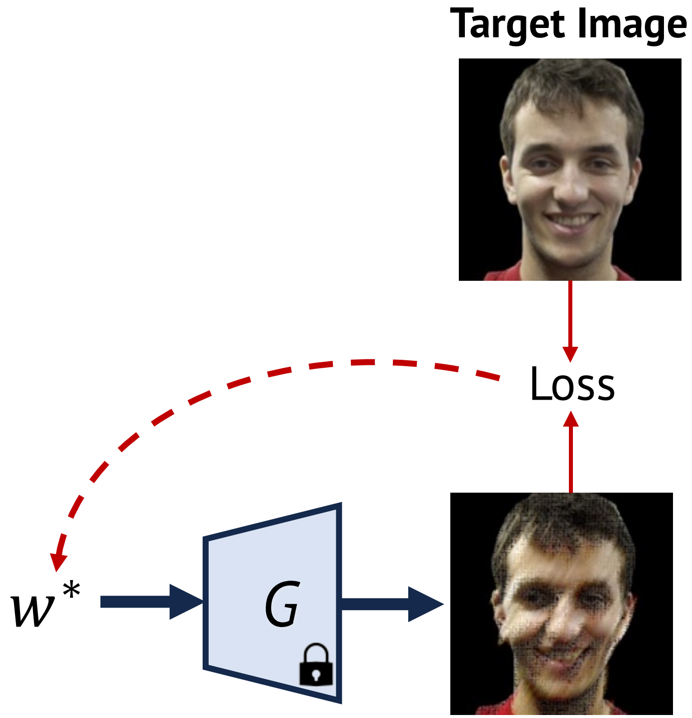
Corresponding Latent code must be known!

Ref: InterfaceGAN, Shen et al.. CVPR '20

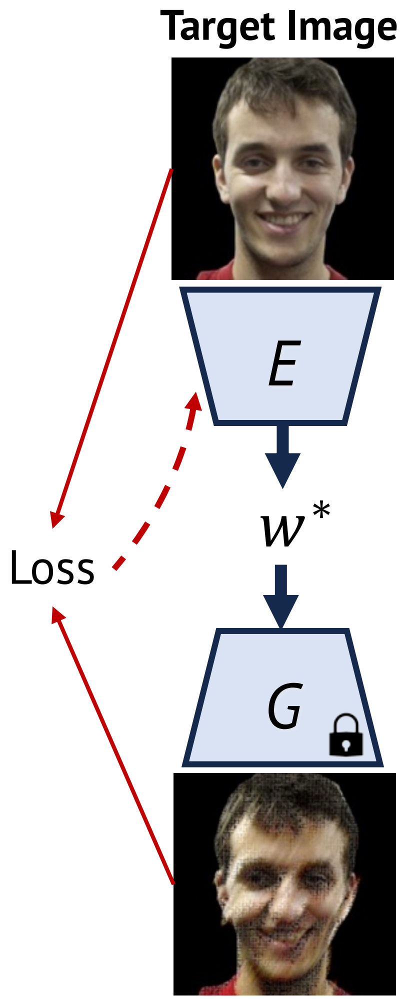
GAN Inversion



GAN Inversion



Optimization-based

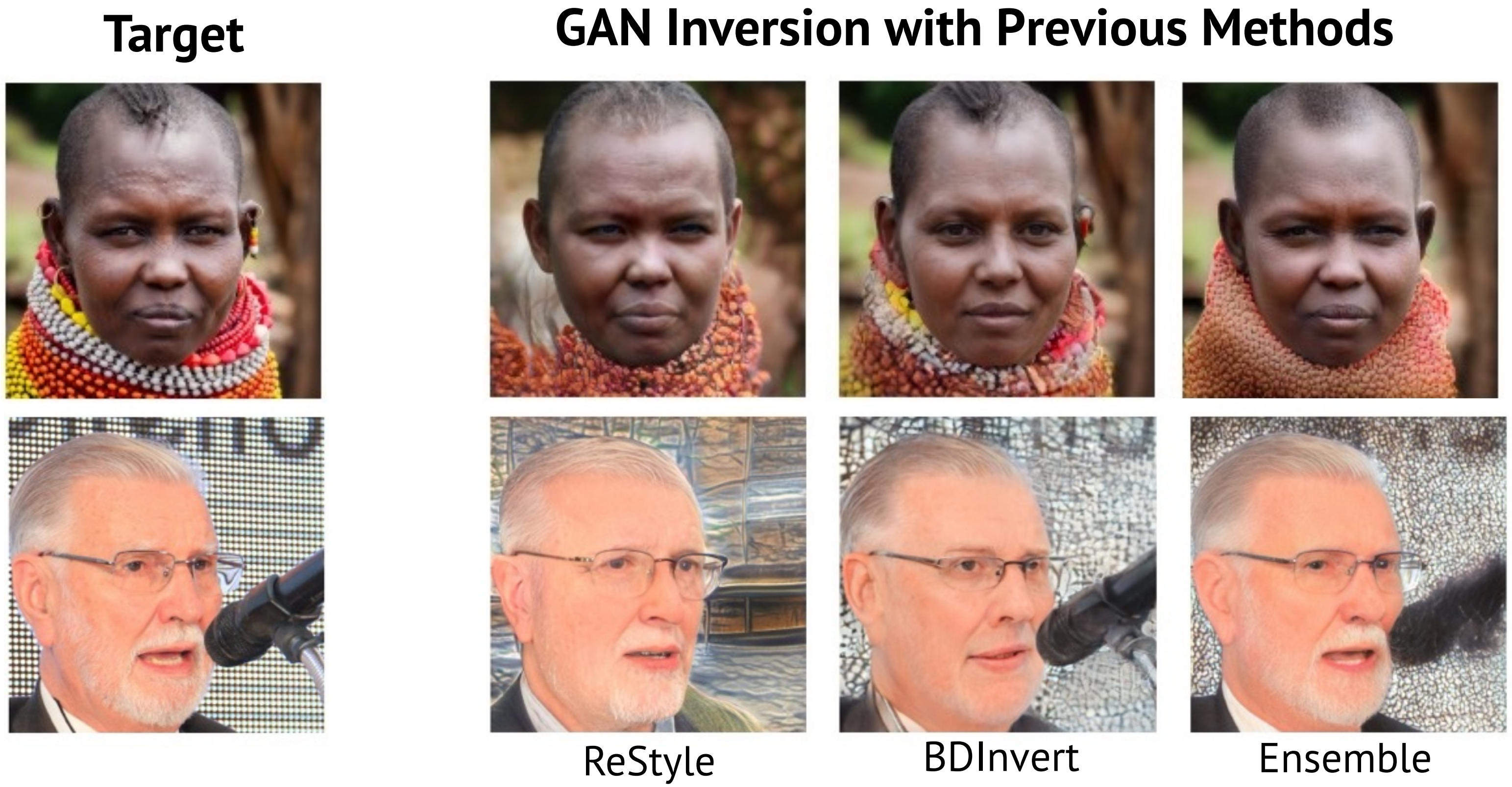


Encoder-based

backprop

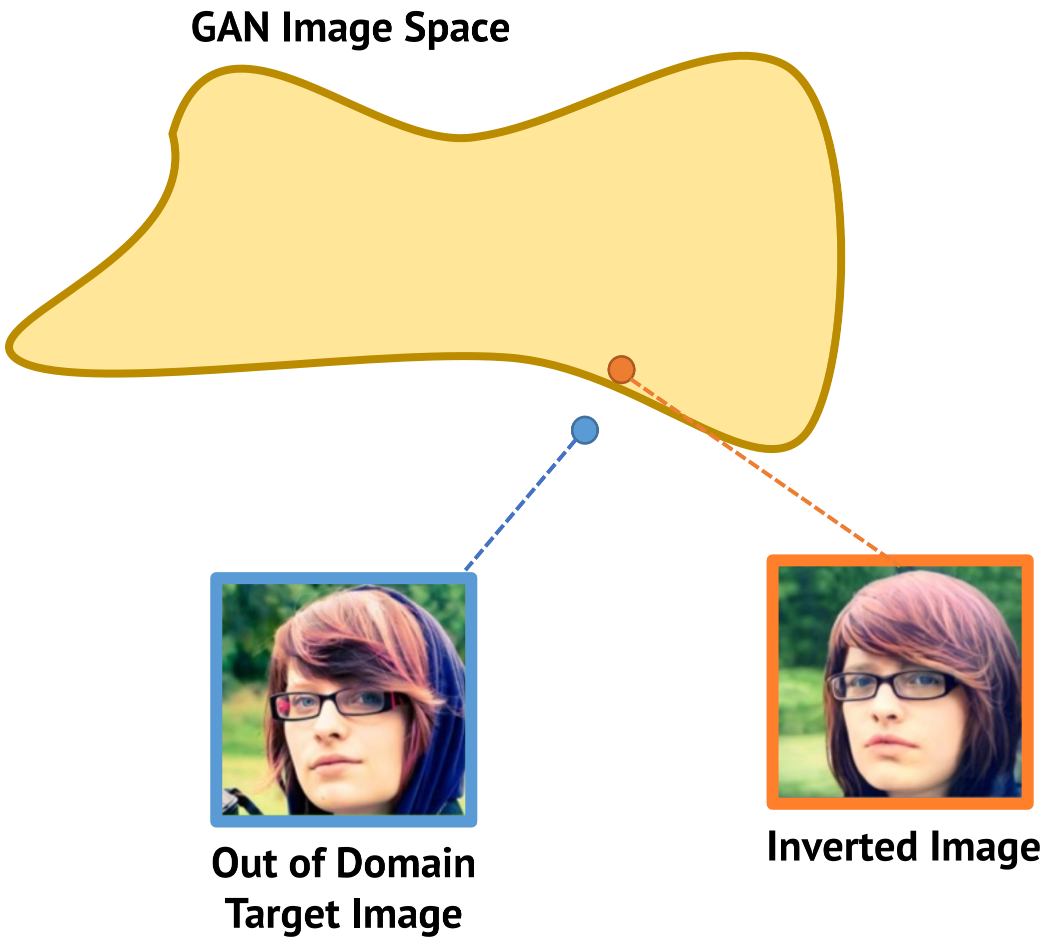
Ref: GAN Inversion Survey, Xia et al. '21; Image2StyleGAN, Abdal et al. ICCV '19; e4e, Tov et al. SIGGRAPH 2020

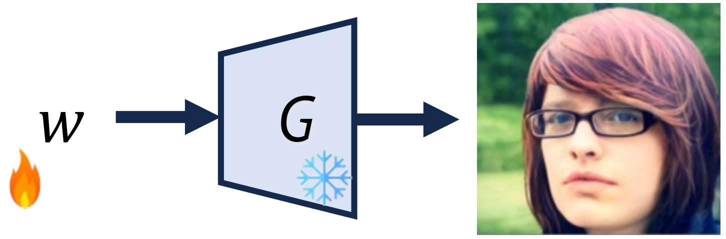
GAN Inversion: Most Techniques Fail on out-of-domain Images



Ref: ReStyle, Alauf et al., ICCV '21
BDInvert, Kang et al., ICCV '21
Ensemble, Cai et al., CVPR '21

Key Challenge in Inverting a Frozen GAN

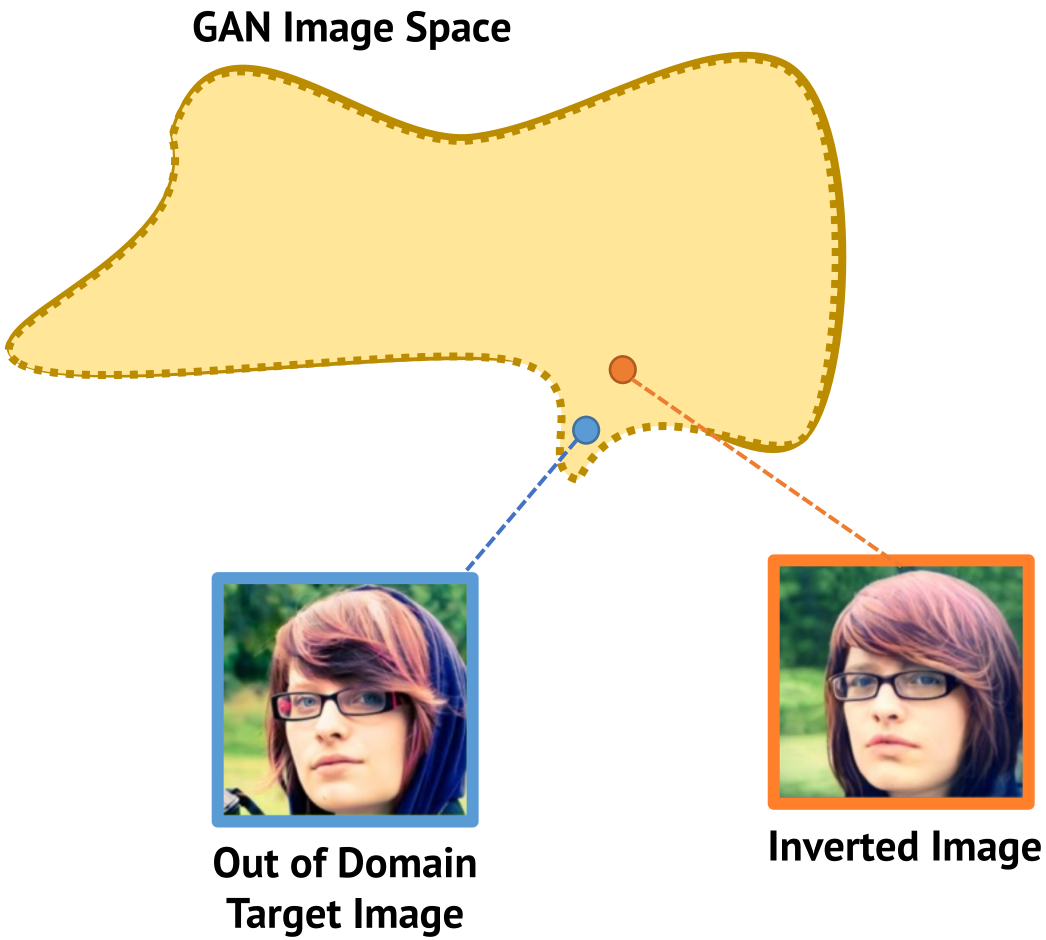


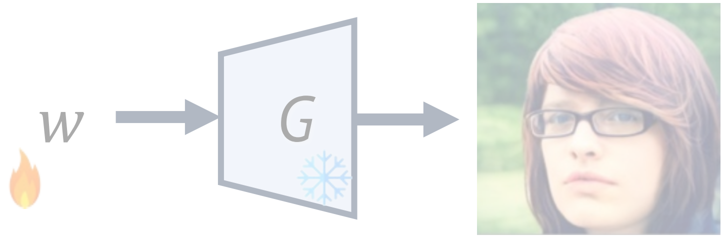


Generator is *Fixed*, Latent is *Flexible*

Near-perfect GAN Inversion, Q. Feng, V. Shah et al., '22; **Make It So**, A. Bhattad, V. Shah et al., '23

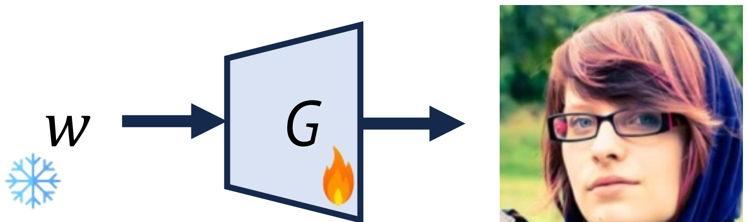
Key Challenge in Inverting a Frozen GAN





Generator is *Frozen,* Latent is *Trainable*

Proposed Idea:



Generator is *Flexible*, Latent is *Fixed*

Near-perfect GAN Inversion, Q. Feng, V. Shah et al., '22; **Make It So**, A. Bhattad, V. Shah et al., '23

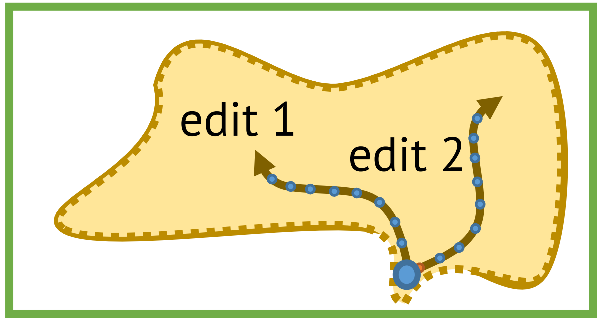
Our Method Achieves Near-perfect GAN Inversion

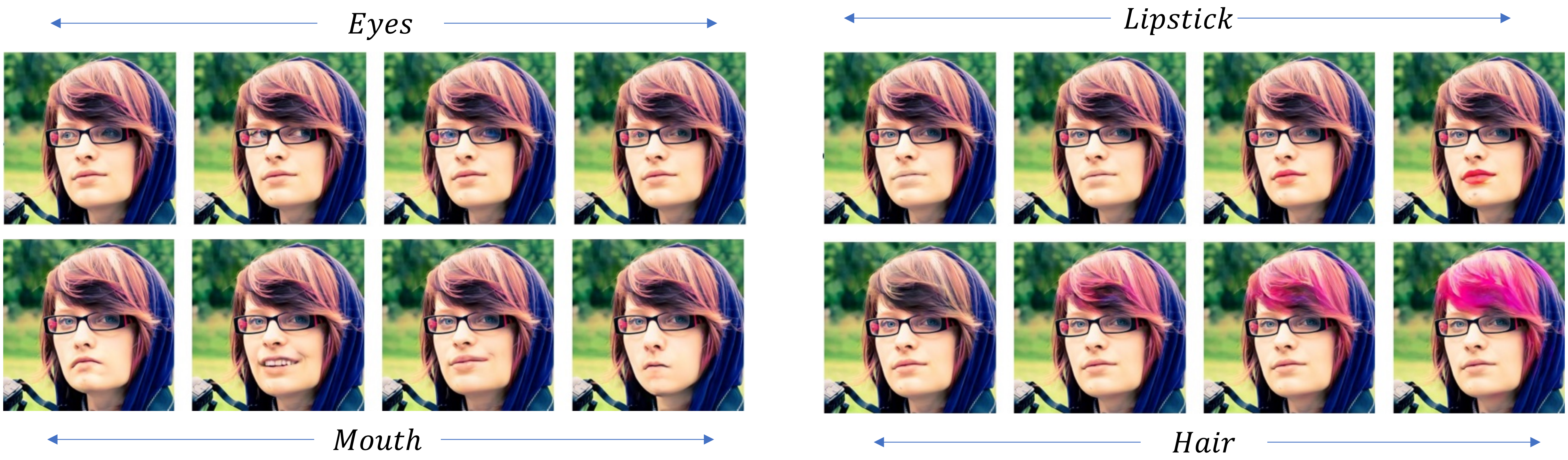
Target **GAN Inversion with Previous Methods** **Ours**

ReStyle BDInvert Ensemble

Ref: ReStyle, Alauf et al., ICCV ‘21
BDInvert, Kang et al., ICCV ’21
Ensemble, Cai et al., CVPR ‘21

Most off-the-shelf Editing methods work!





using **StyleSpace Editing Directions**

Ref: StyleSpace, CVPR 2021

One-shot Image Stylization

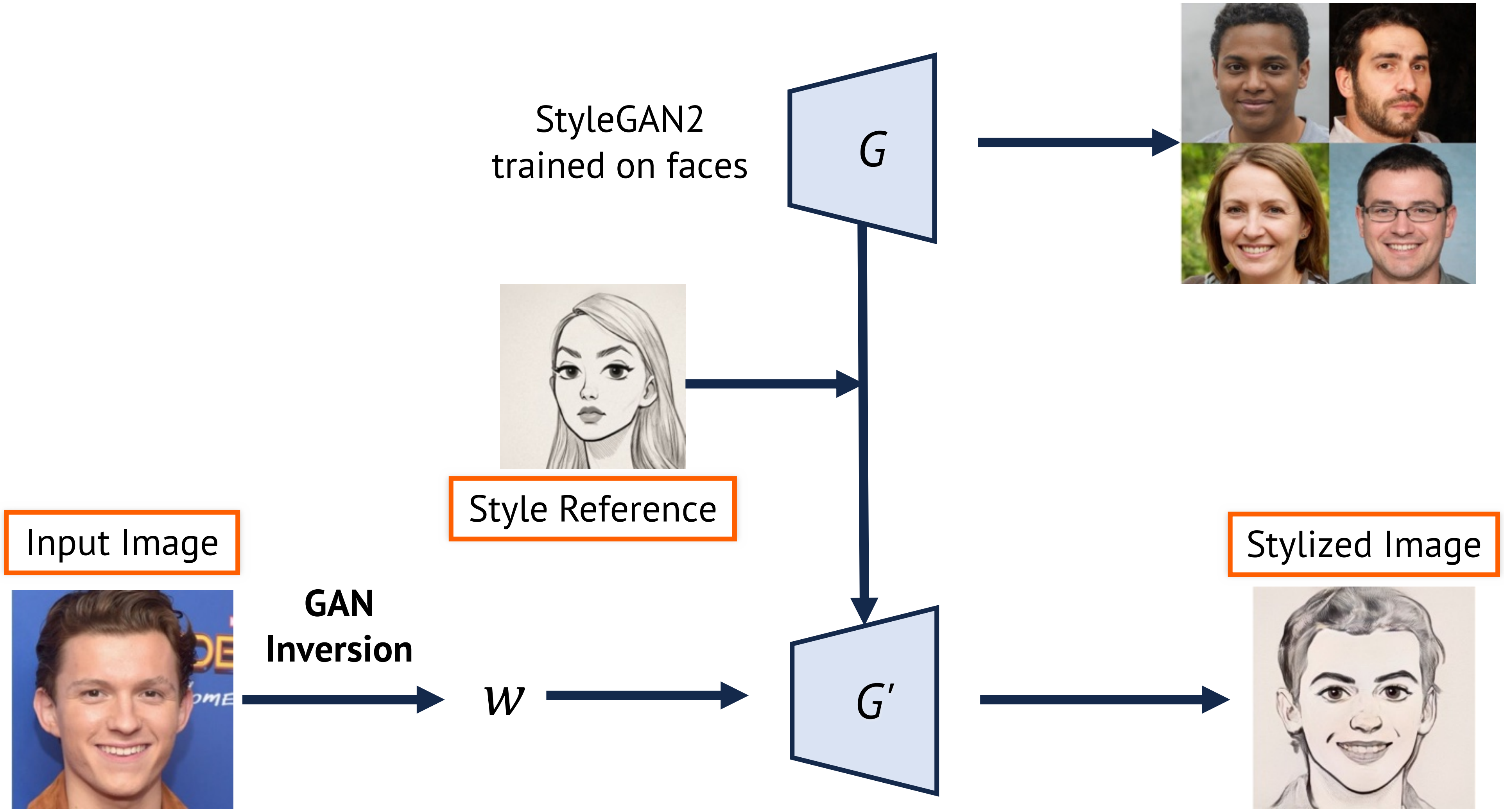
Style Reference

Input Image

Stylized Image

Stylizing an Input Image in the style of a Reference Style using only *one* example

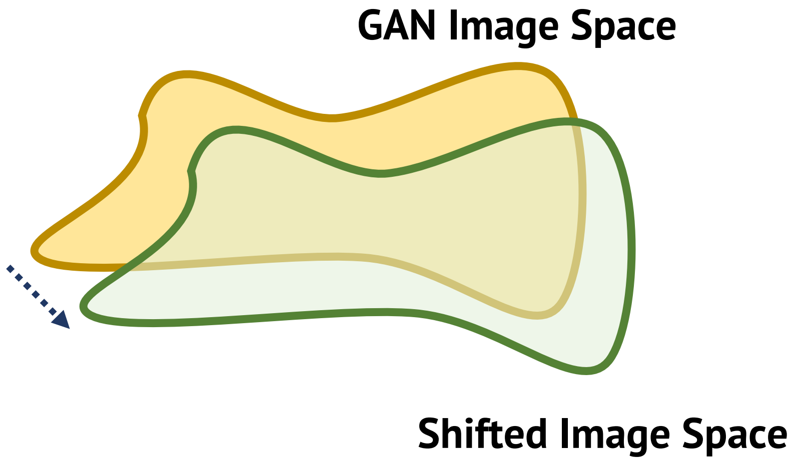
Pre-trained GAN for one-shot Stylization

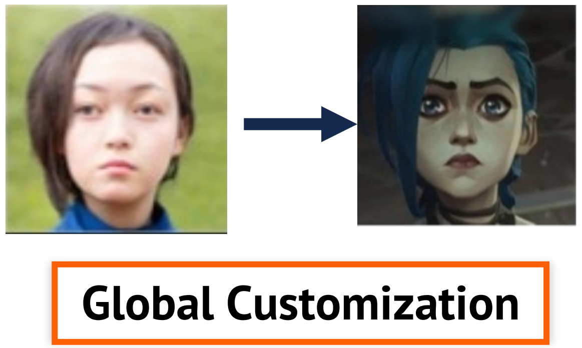


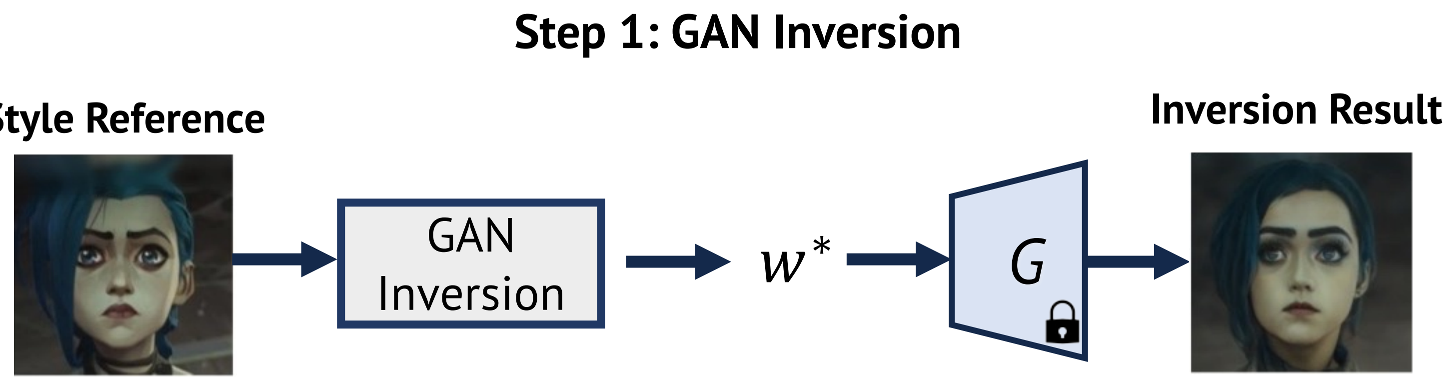
Fine-tuned StyleGAN2

JoJoGAN, Chong et al. ECCV '22

How to perform global customization of StyleGAN2?







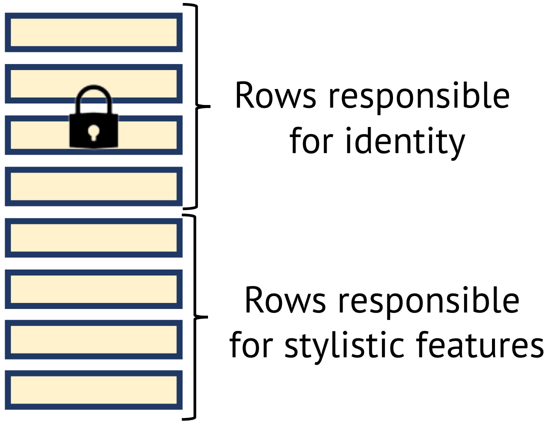
Ref: GAN Inversion Survey, Xia et al.'21; e4e, Tov et al. SIGGRAPH 2020

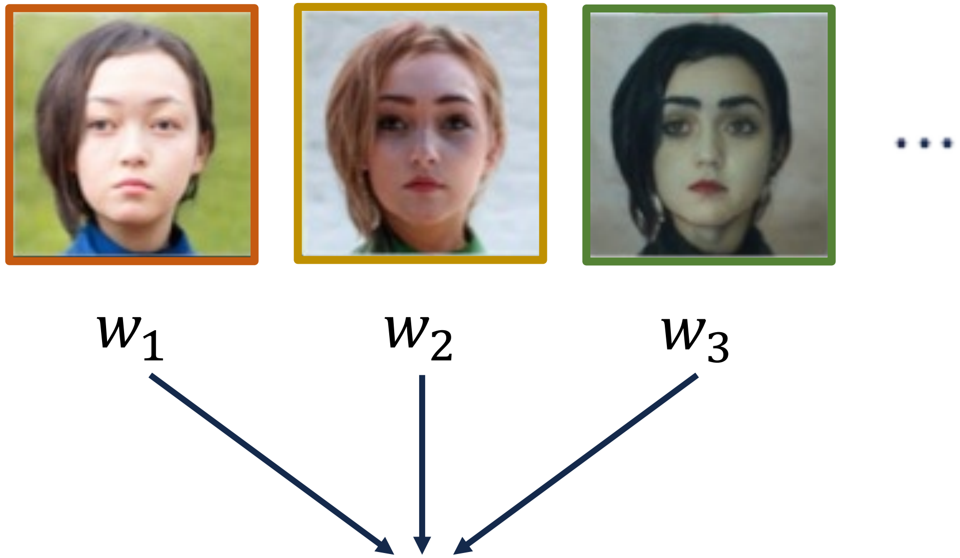
JoJoGAN: Style-mixing

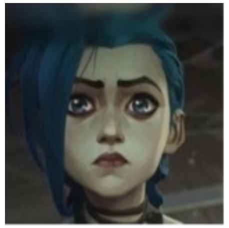
Step 2. Use Style-mixing property to Create a Training Set

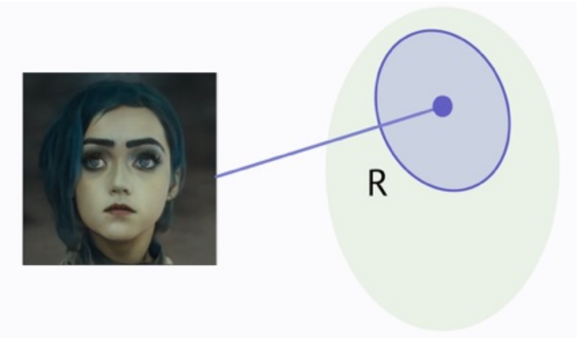
Inversion result

w^*







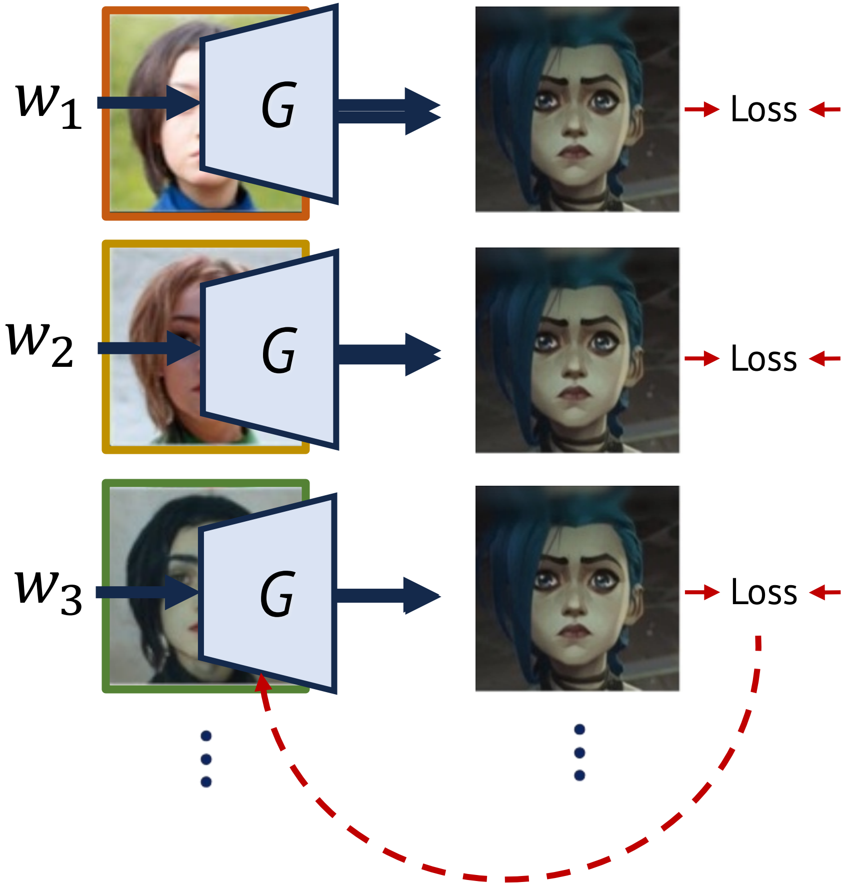


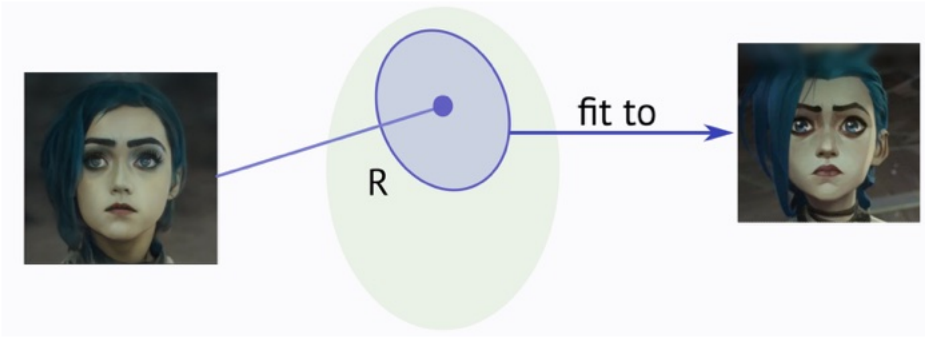
S or W space

Ref: StyleGAN, Karras et al. CVPR '19; StyleGAN2, Karras et al. CVPR '20

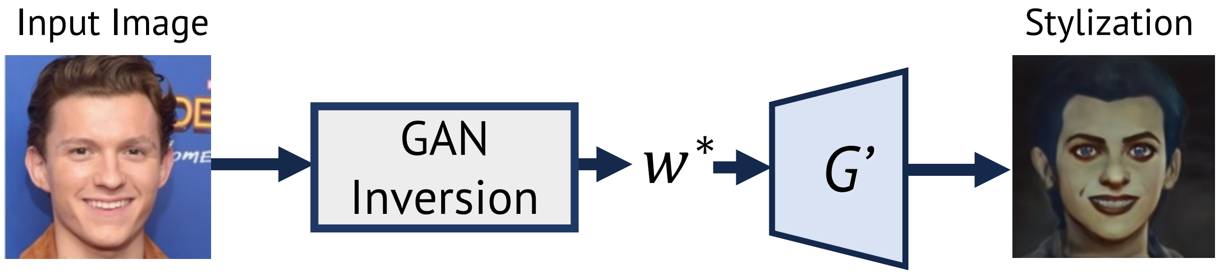
How to Fine-tune StyleGAN2 ? : JoJoGAN

Fine-tune *G* using pixel-level losses



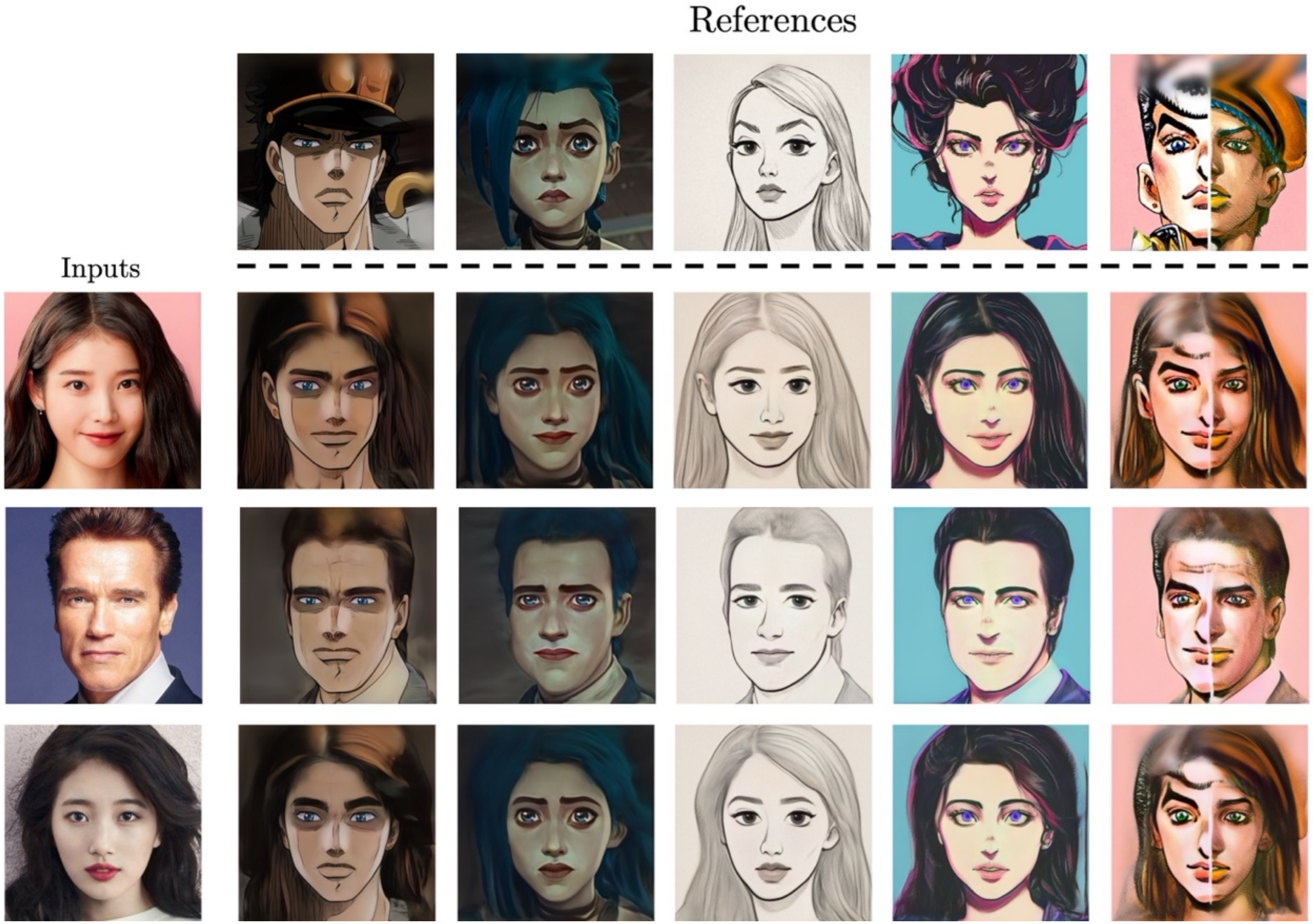


Inference on fine-tuned *G*



Ref: JoJoGAN, Chong et al. ECCV '22

JoJoGAN: Results



GANs: Summary

- Tremendous progress in training algorithm, architecture, and conditioning mechanism
- Rich understanding of image features with disentangled representation
- Smooth interpolations in latent space
- Vast variety of applications of pre-trained GANs in all engineering domains through,
  - GAN conditioning
  - GAN Inversion + latent space traversal
  - GAN customization (fine-tuning)
- Are GANs still relevant?! : current and future trends