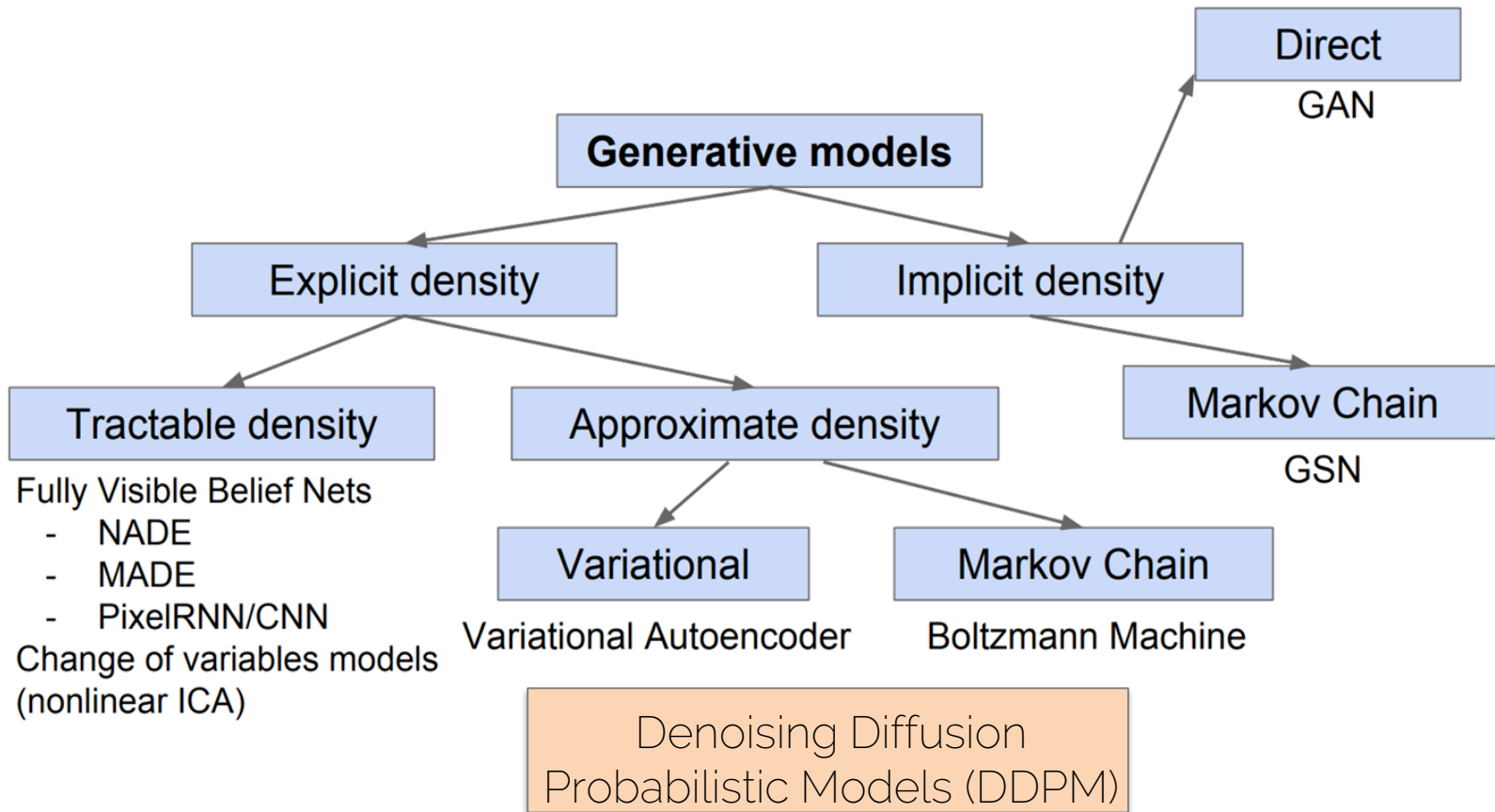
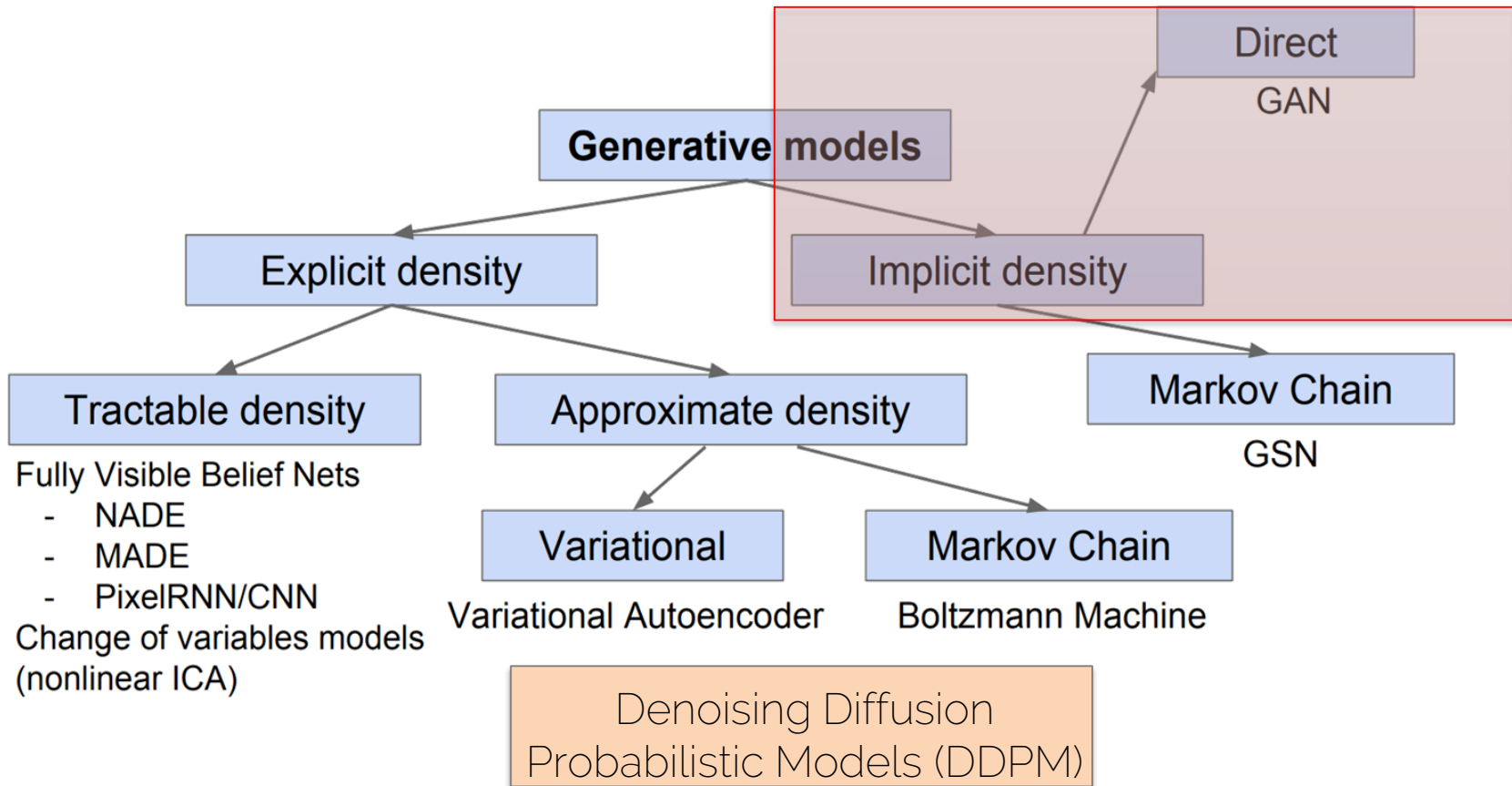


# Generative Neural Networks

# Taxonomy of Generative Models



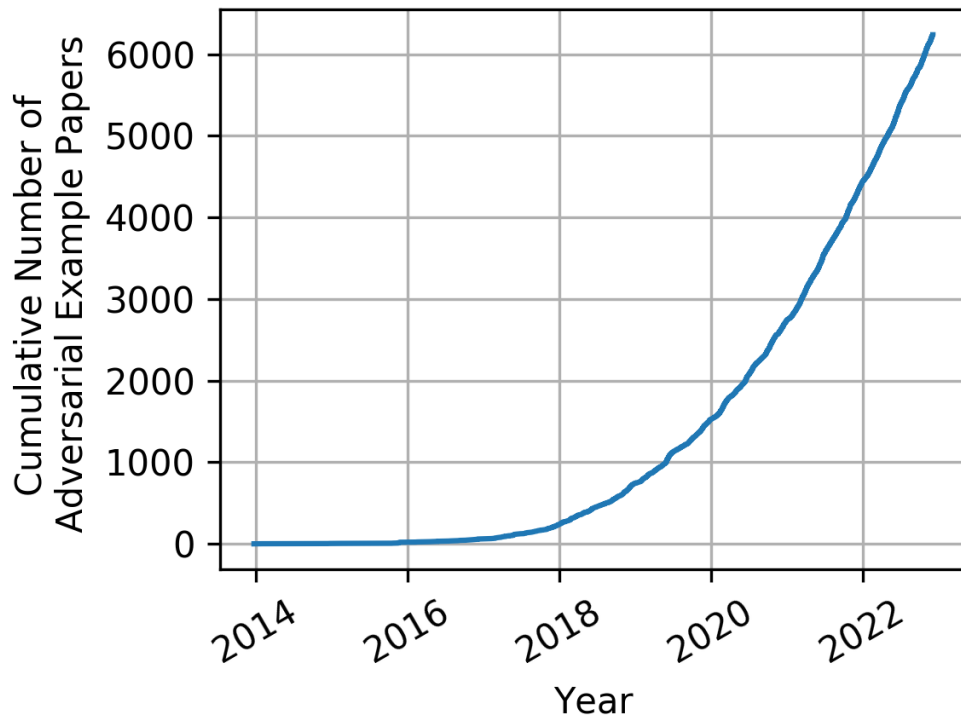
# Taxonomy of Generative Models



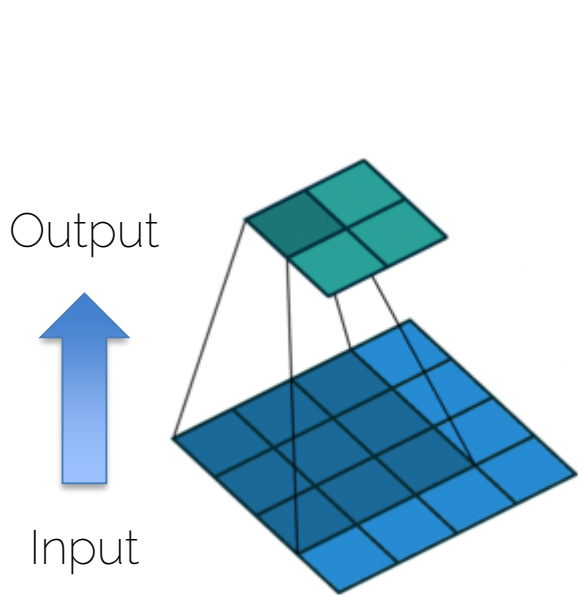
# Generative Adversarial Networks (GANs)

# Generative Adversarial Networks (GANs)

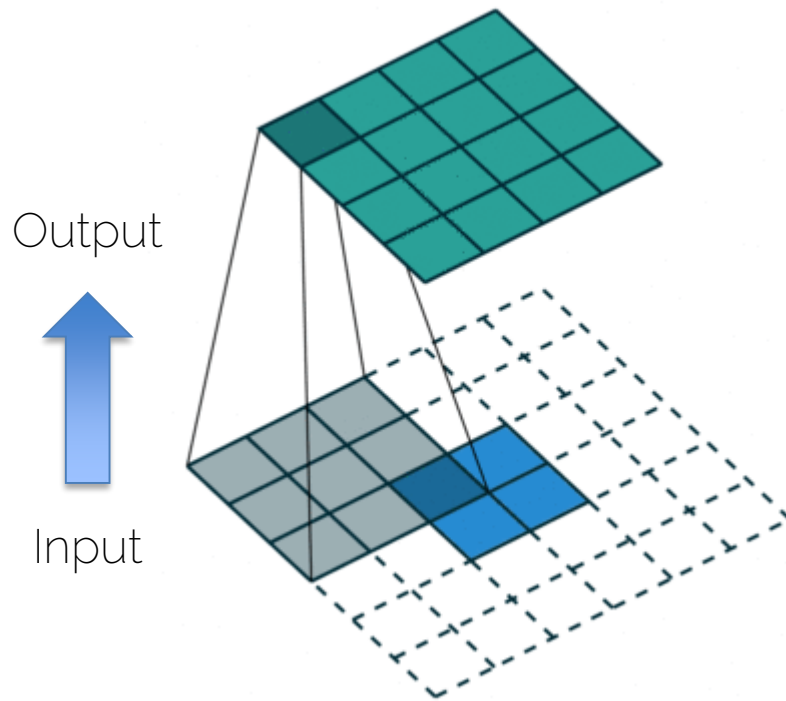
List of All (arXiv) Adversarial Example Papers



# Convolution and Deconvolution

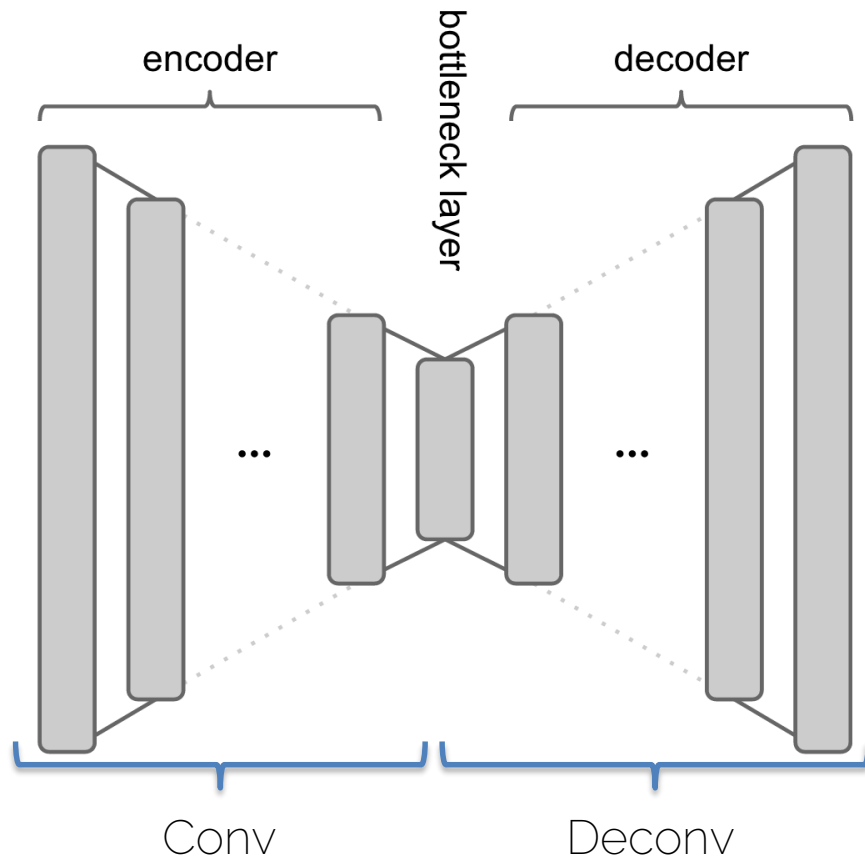


Convolution  
no padding, no stride

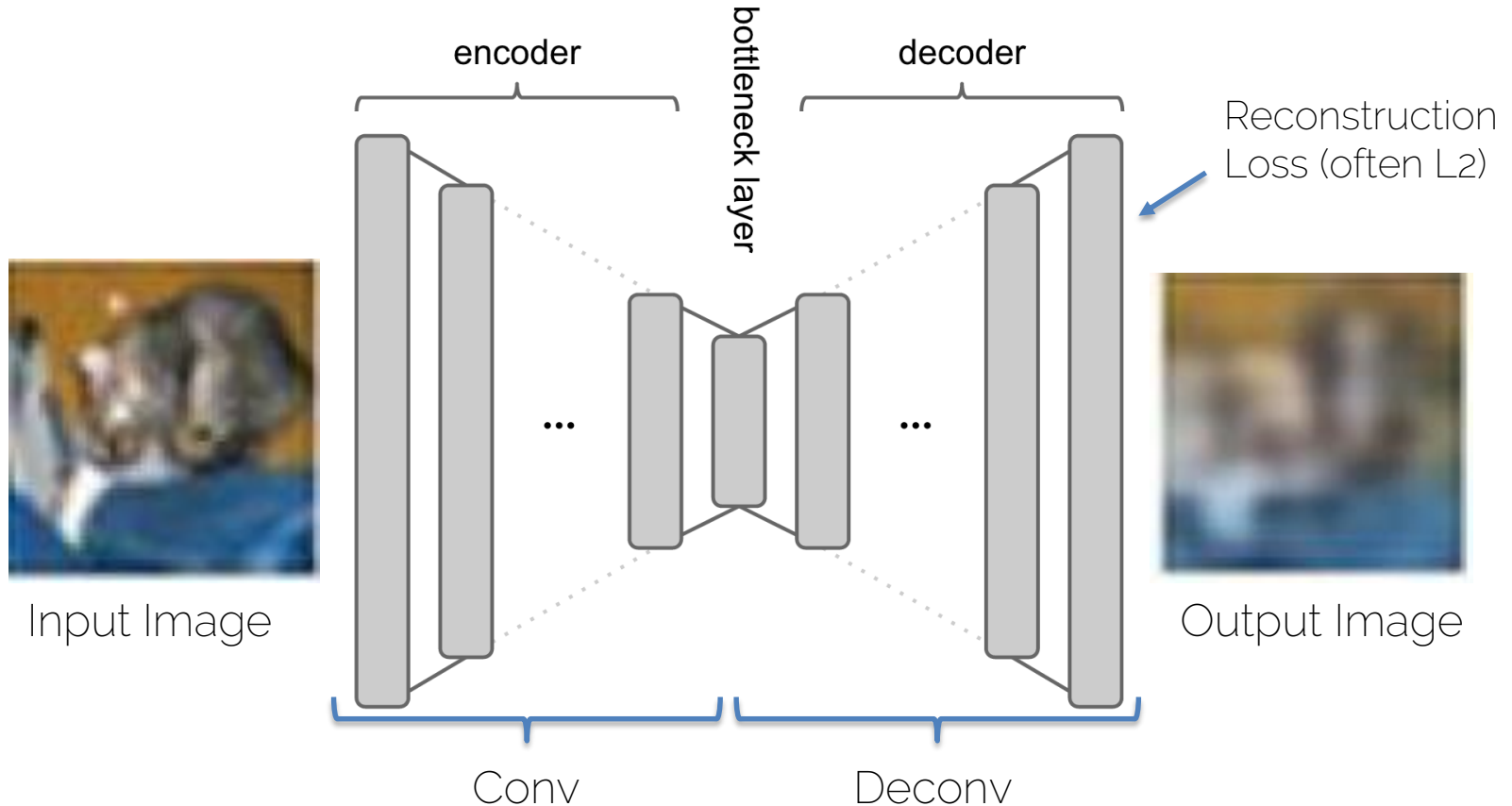


Transposed convolution  
no padding, no stride

# Autoencoder

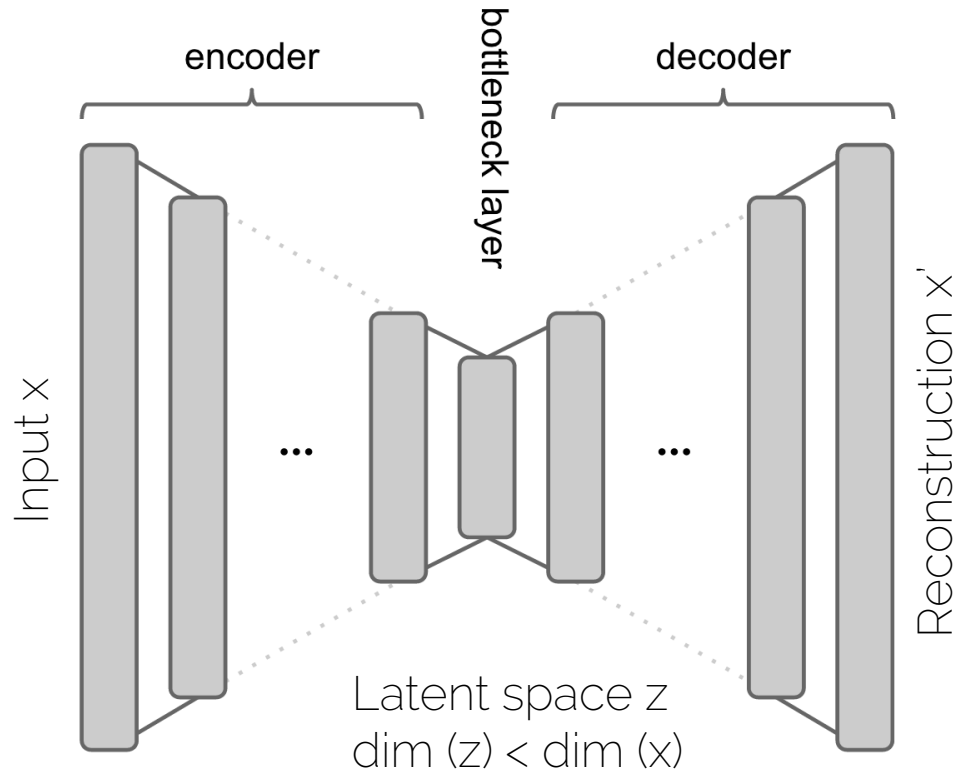


# Reconstruction: Autoencoder

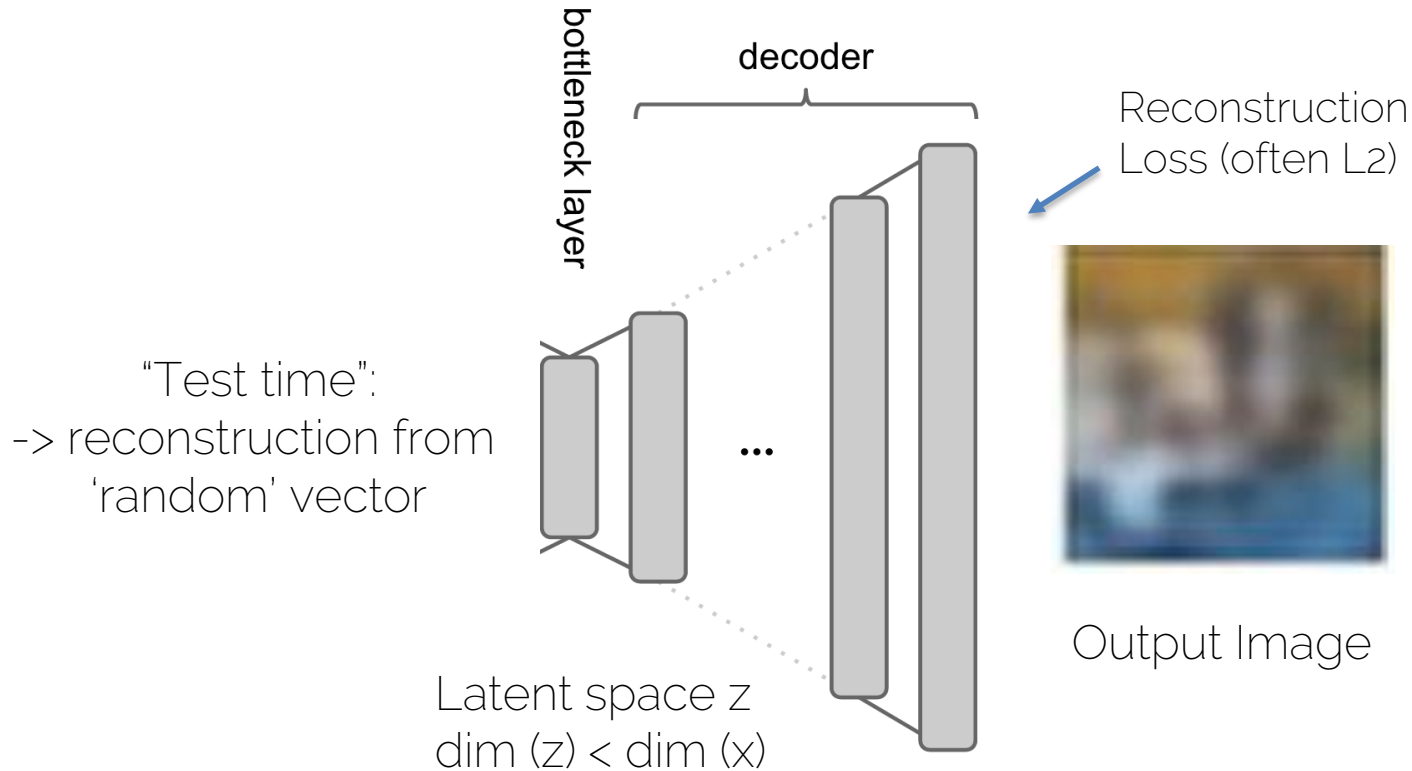




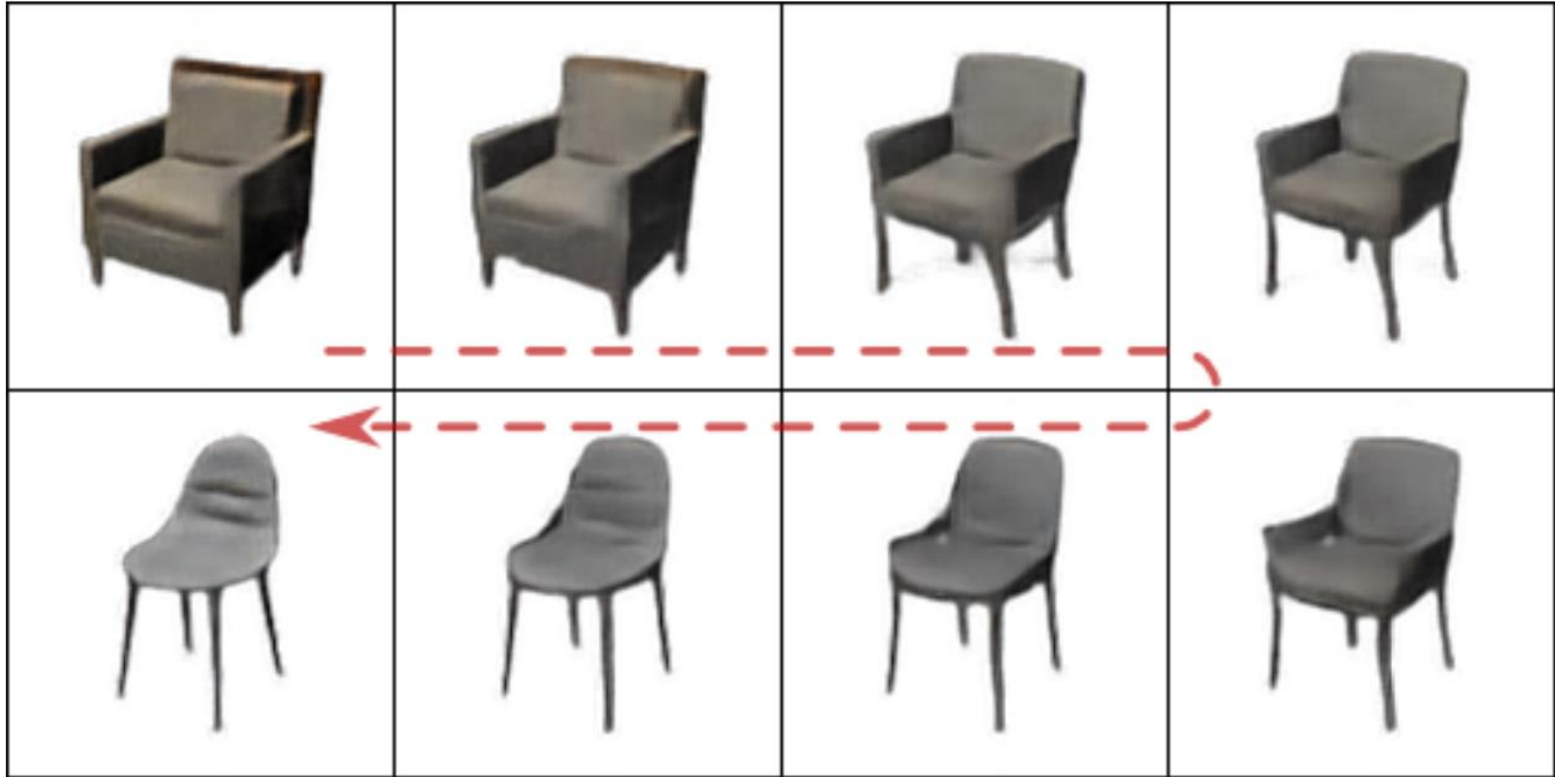
# Training Autoencoders



# Decoder as Generative Model



# Decoder as Generative Model



Interpolation between two chair models

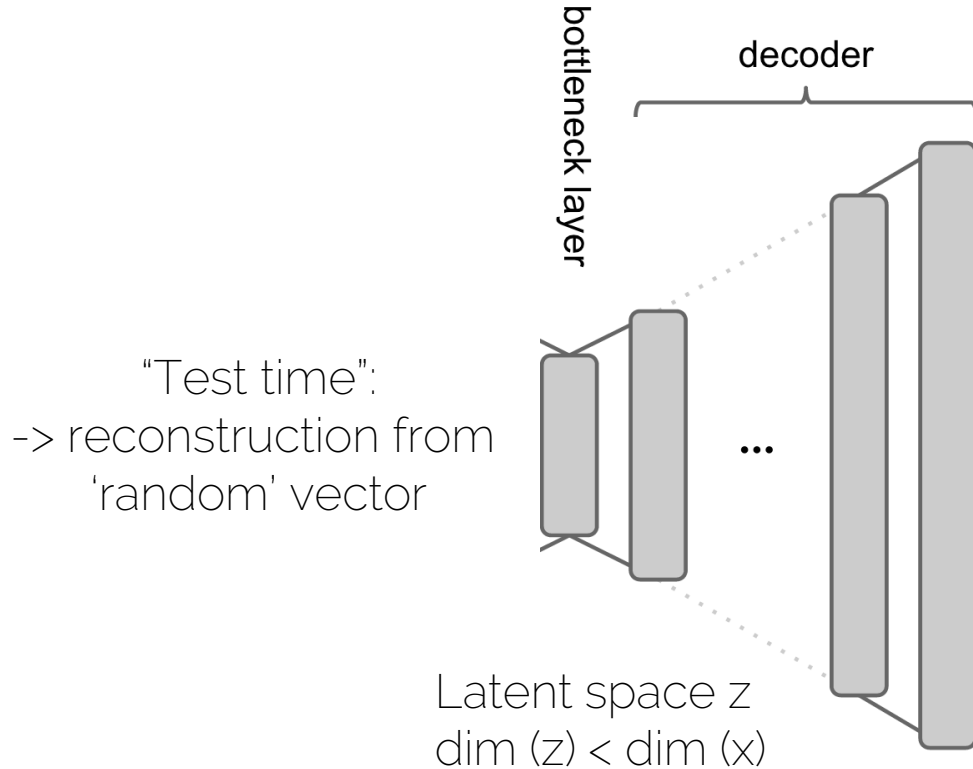
# Decoder as Generative Model

1



Morphing between  
chair models

# Decoder as Generative Model

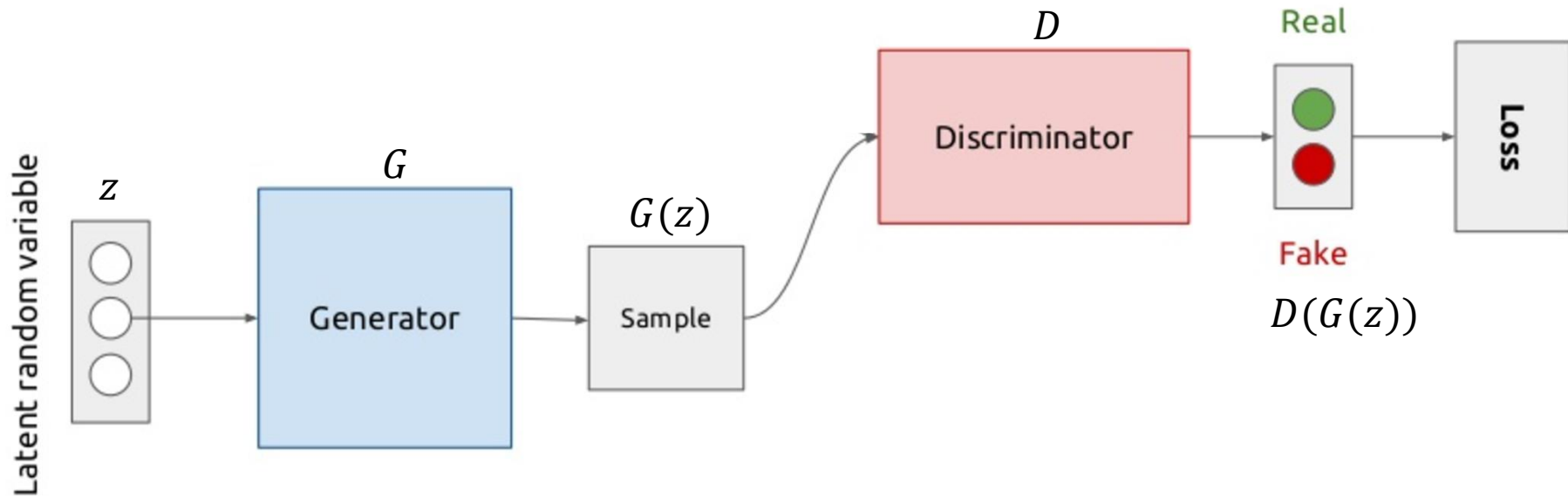


Reconstruction Loss  
Often L2, i.e., sum of squared dist.  
-> L2 distributes error equally  
-> mean is opt.  
-> res. Is blurry

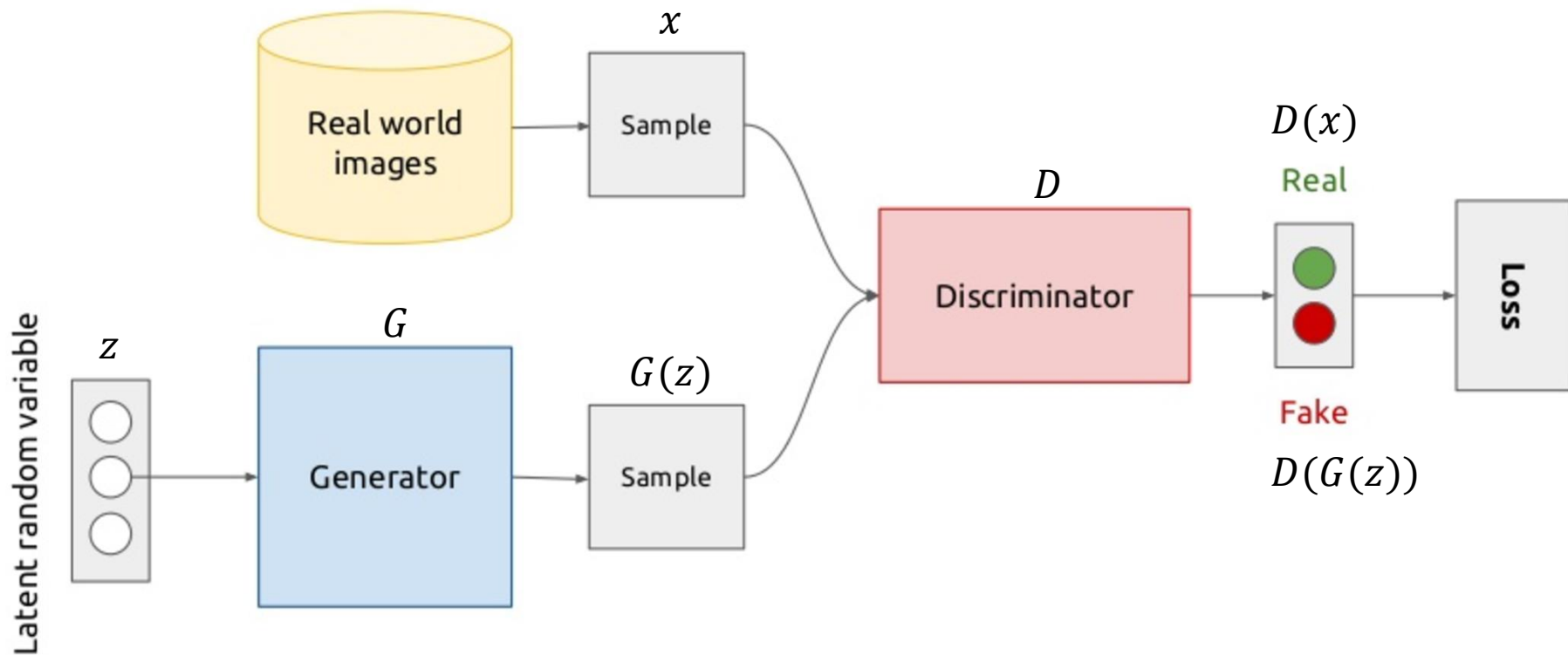


Instead of L2, can we  
"learn" a loss function?

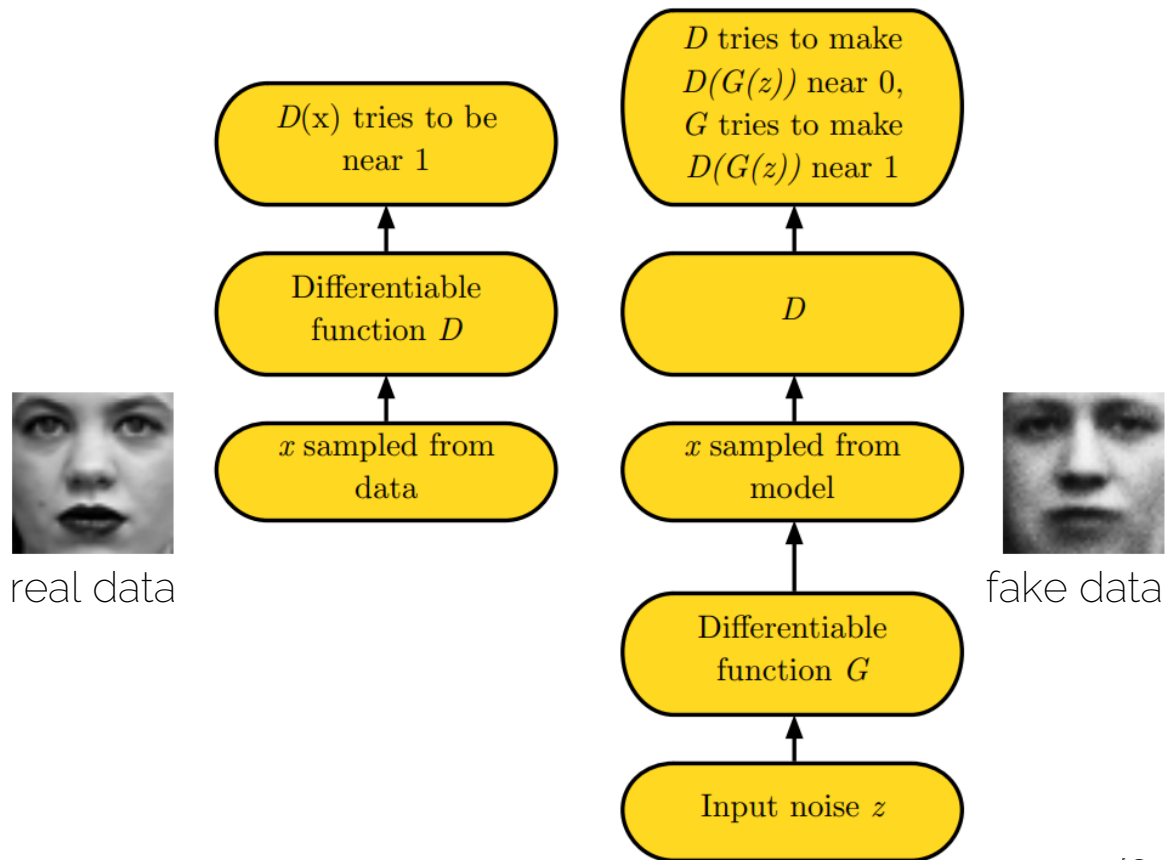
# Generative Adversarial Networks (GANs)



# Generative Adversarial Networks (GANs)



# Generative Adversarial Networks (GANs)





# GANs: Loss Functions

Discriminator loss

$$J^{(D)} = \underbrace{-\frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}) - \frac{1}{2} \mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))}_{\text{binary cross entropy}}$$

Generator loss

$$J^{(G)} = -J^{(D)}$$

- Minimax Game:
  - G minimizes probability that D is correct
  - Equilibrium is saddle point of discriminator loss

-> D provides supervision (i.e., gradients) for G

# GANs: Loss Functions

Discriminator loss

$$J^{(D)} = -\frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}) - \frac{1}{2} \mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))$$

Generator loss

$$J^{(G)} = -\frac{1}{2} \mathbb{E}_{\mathbf{z}} \log D(G(\mathbf{z}))$$

- Heuristic Method (often used in practice)
  - G maximizes the log-probability of D being mistaken
  - G can still learn even when D rejects all generator samples

# Alternating Gradient Updates

- Step 1: Fix  $G$ , and perform gradient step to

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\log D(\mathbf{x}) - \frac{1}{2}\mathbb{E}_{\mathbf{z}}\log(1 - D(G(\mathbf{z})))$$

- Step 2: Fix  $D$ , and perform gradient step to

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{\mathbf{z}}\log D(G(\mathbf{z}))$$

# Vanilla GAN

**for** number of training iterations **do**

**for**  $k$  steps **do**

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

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

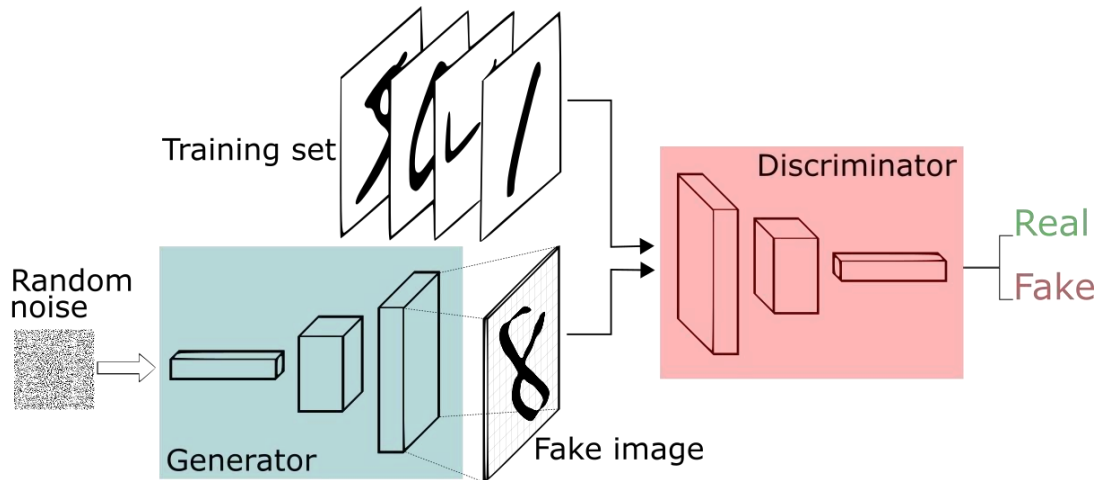
**end for**

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

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

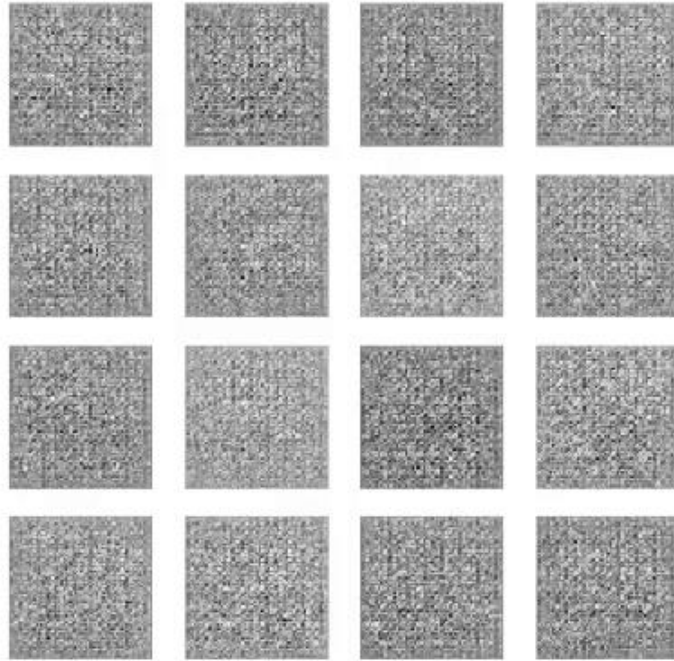
**end for**

# Putting it all Together



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

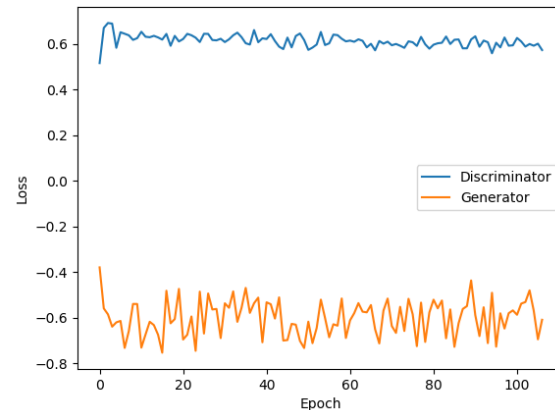
# Training a GAN



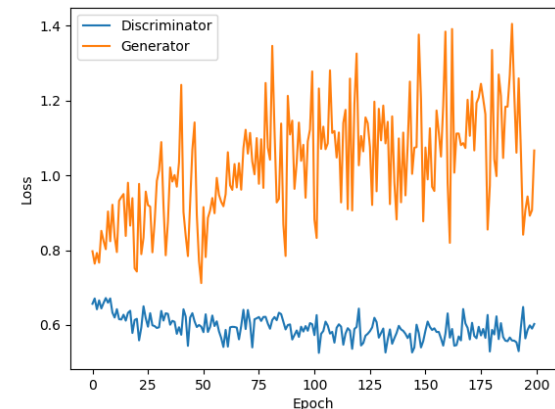
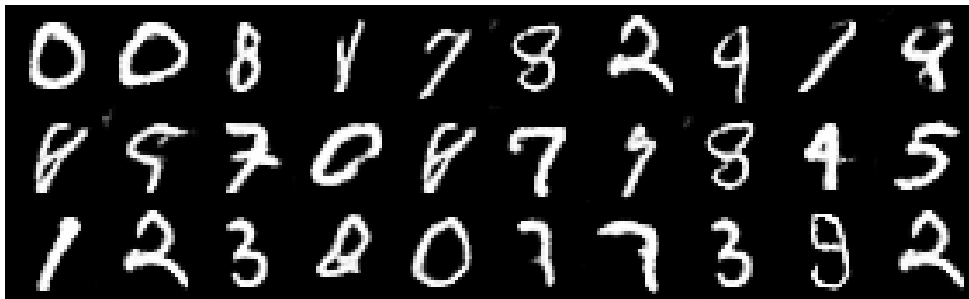
<https://medium.com/ai-society/gans-from-scratch-1-a-deep-introduction-with-code-in-pytorch-and-tensorflow-cb03cdcdba0f>

# GANs: Loss Functions

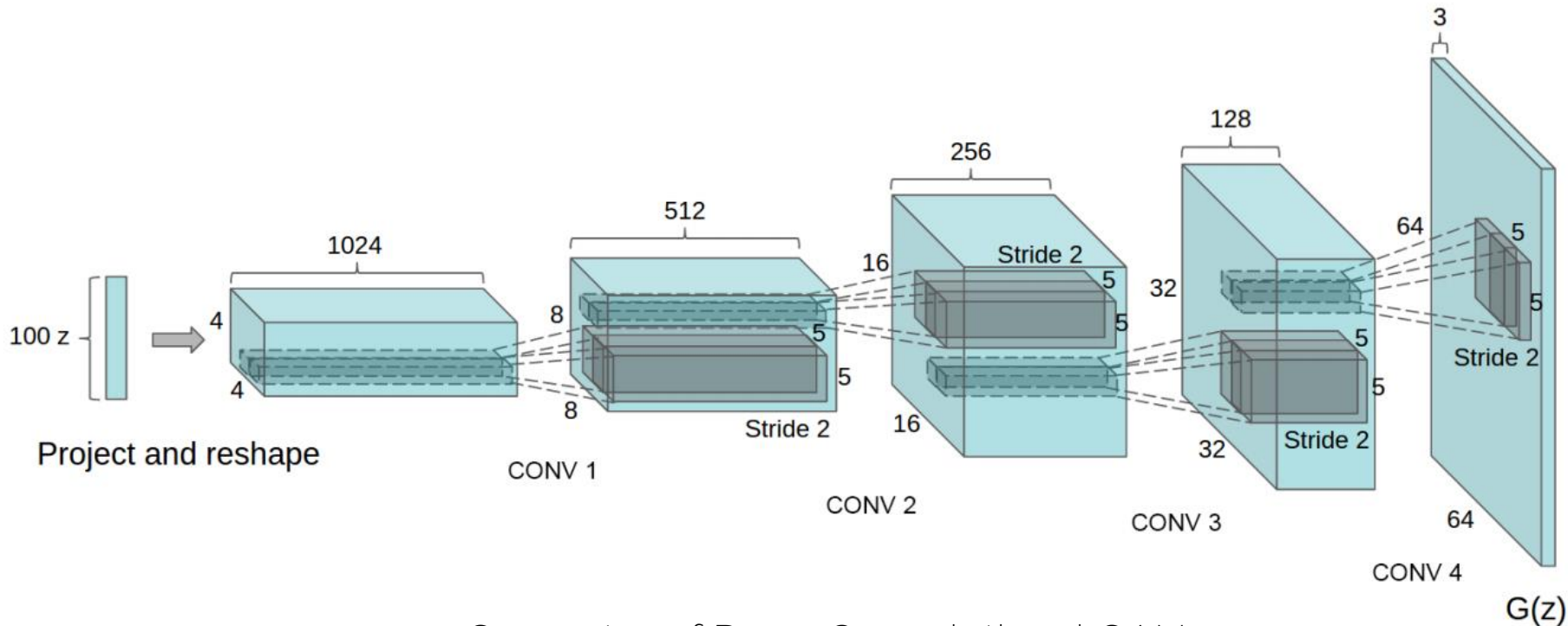
Minimax



Heuristic



# DCGAN: Generator



Generator of Deep Convolutional GANs



# DCGAN: Results



Results on MNIST

# DCGAN: Results



Results on CelebA (200k relatively well aligned portrait photos)

# DCGAN: Results

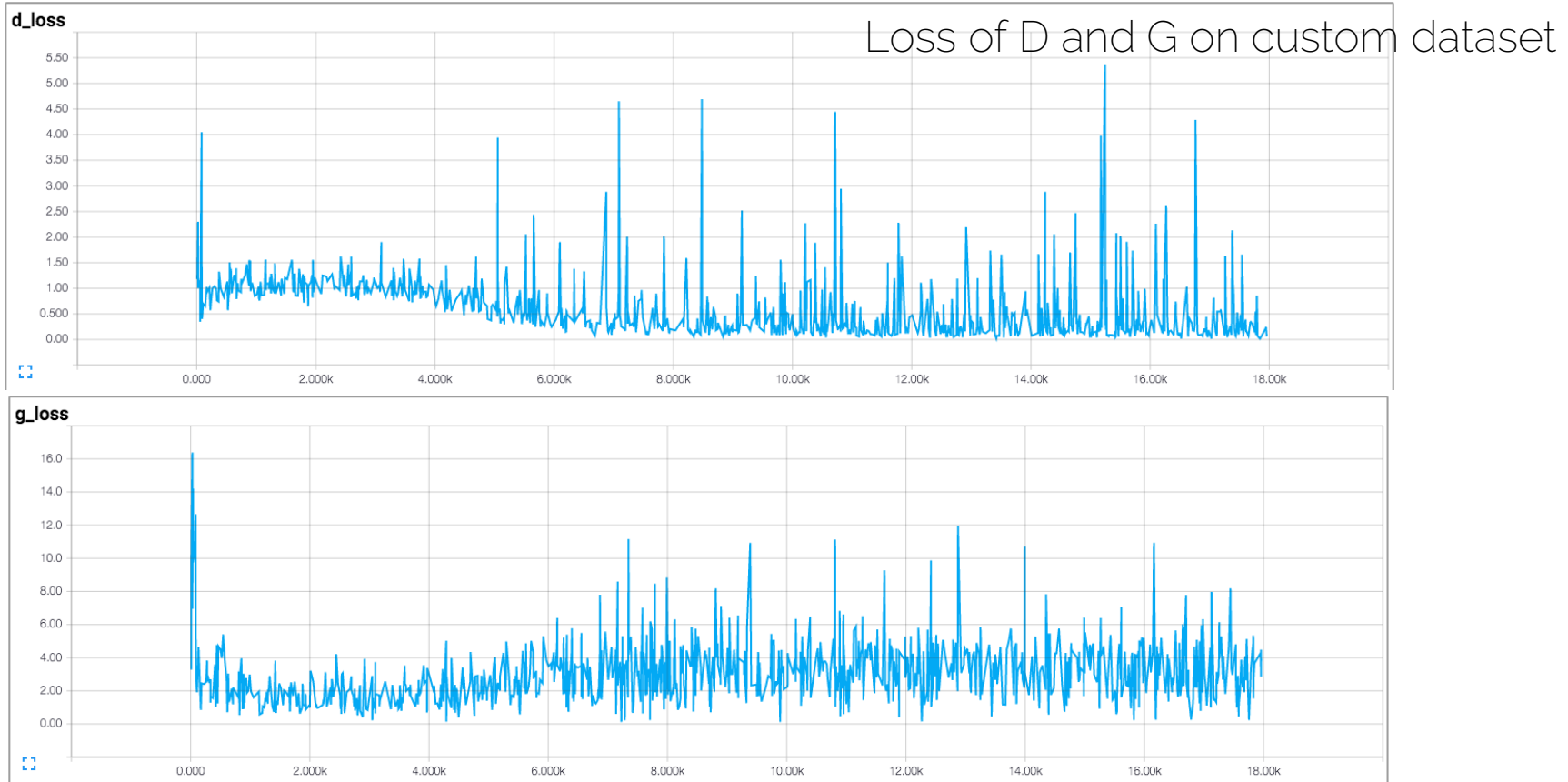


Asian face dataset

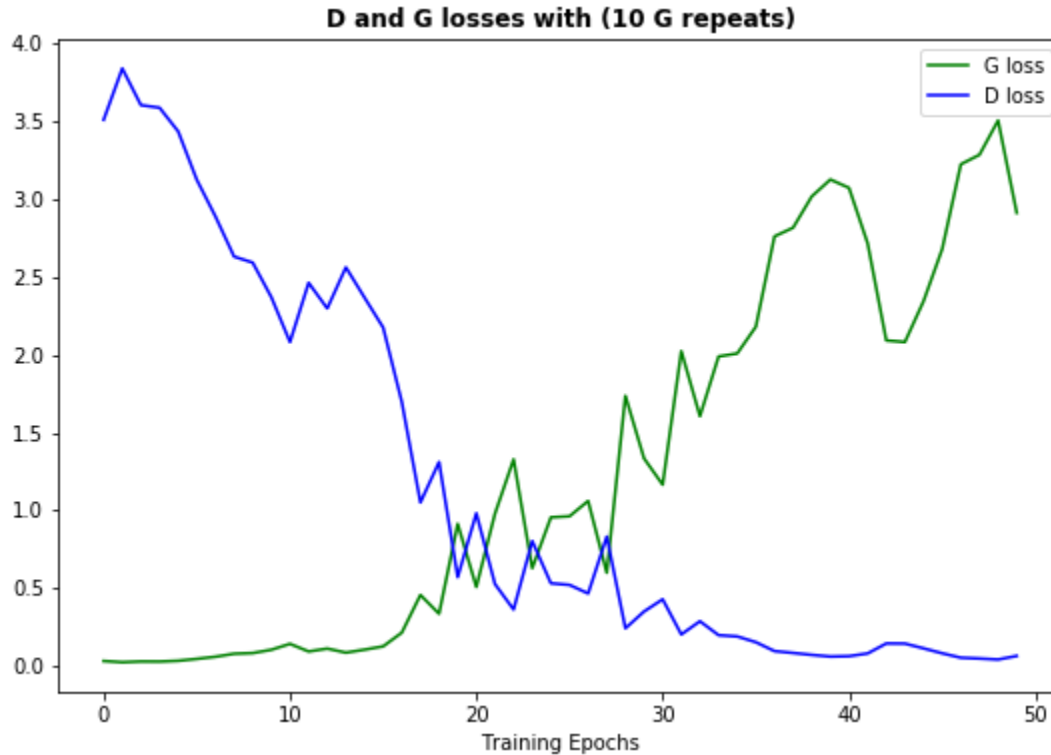
# DCGAN: Results



# DCGAN: Results

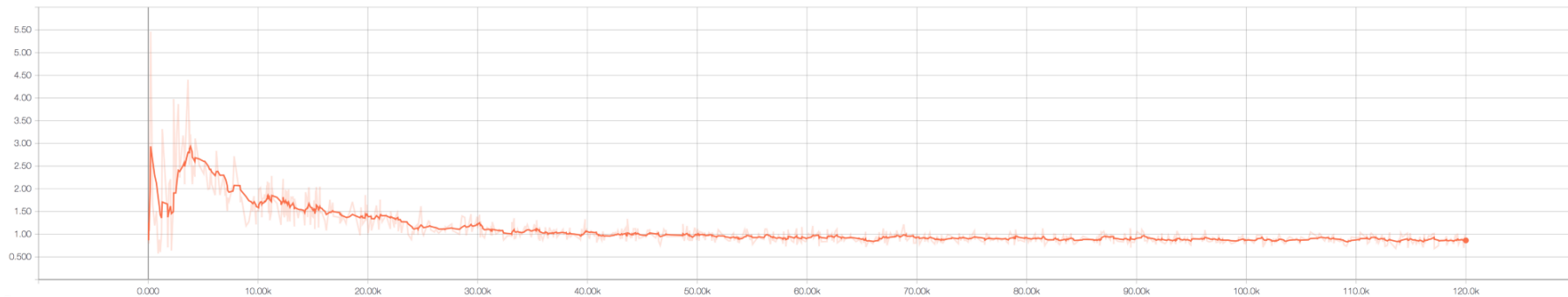


# “Bad” Training Curves

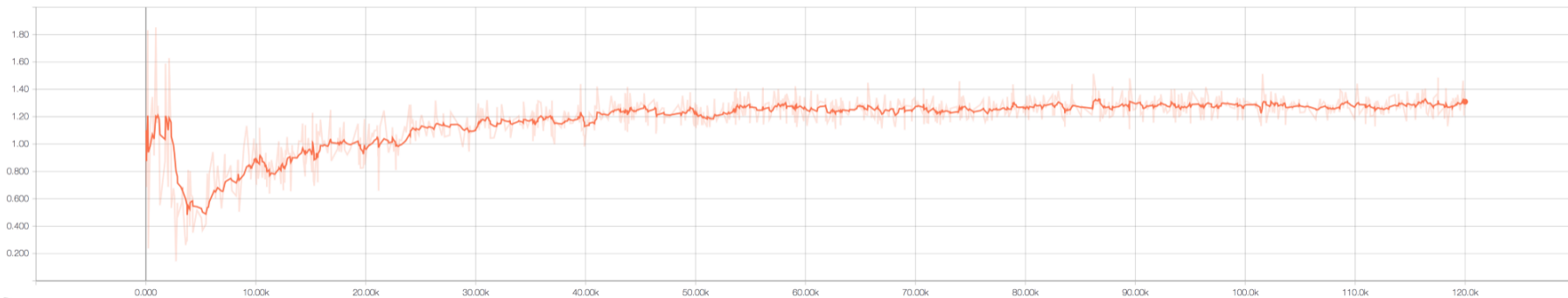


<https://stackoverflow.com/questions/44313306/dcgans-discriminator-getting-too-strong-too-quickly-to-allow-generator-to-learn>

# “Good” Training Curves

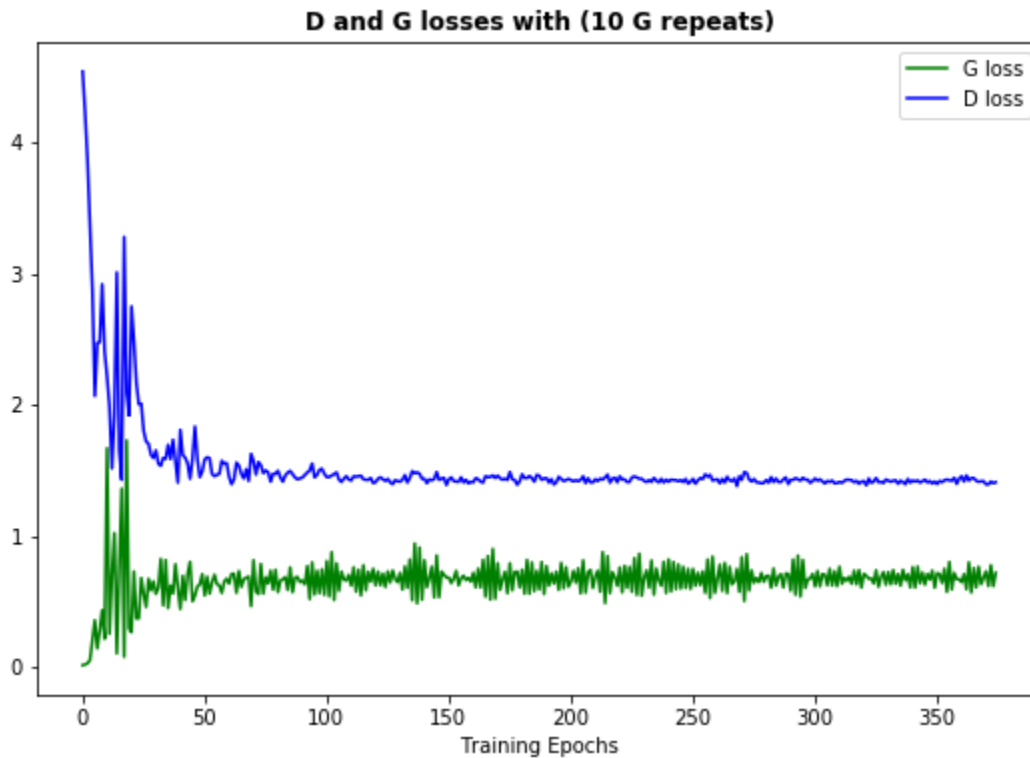


Generator's Error through Time



Discriminator's Error through Time

# “Good” Training Curves





# Training Schedules

- Adaptive schedules

For instance

```
while loss_discriminator > t_d:
```

```
    train discriminator
```

```
while loss_generator > t_g:
```

```
    train generator
```

# Weak vs Strong Discriminator

- Need balance 😊
- Discriminator too weak?
  - No good gradients (cannot get better than teacher...)
- Generator too weak?
  - Discriminator will always be right

# Mode Collapse

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

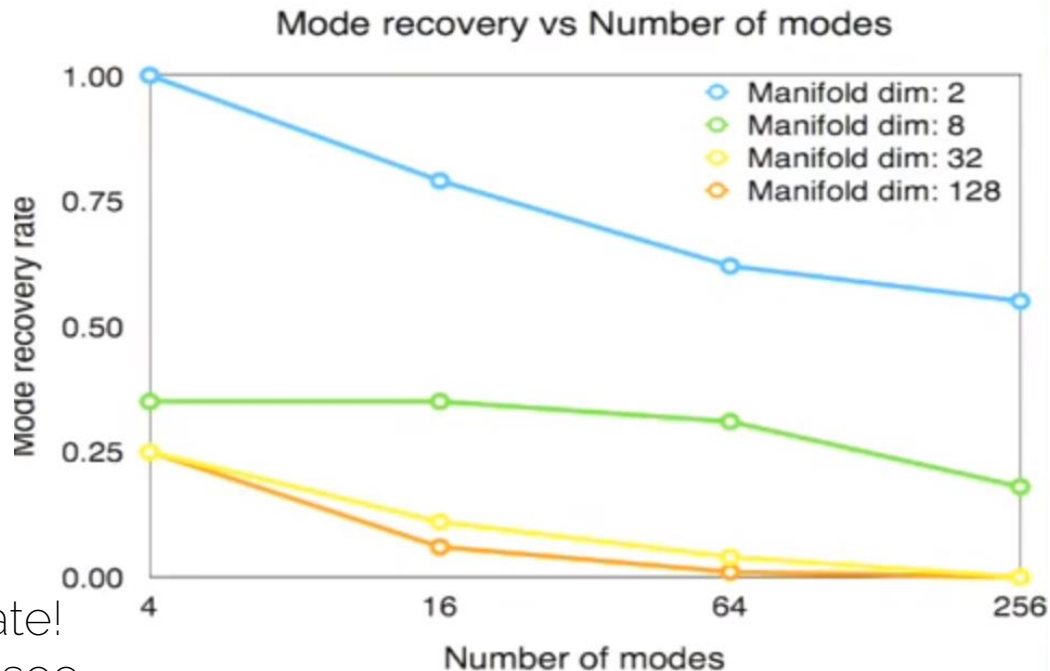
- $D$  in inner loop  $\rightarrow$  convergence to correct dist.
- $G$  in inner loop  $\rightarrow$  easy to convergence to one sample



# Mode Collapse

- Data dim. Fixed (512)
- Performance correlates with # of modes

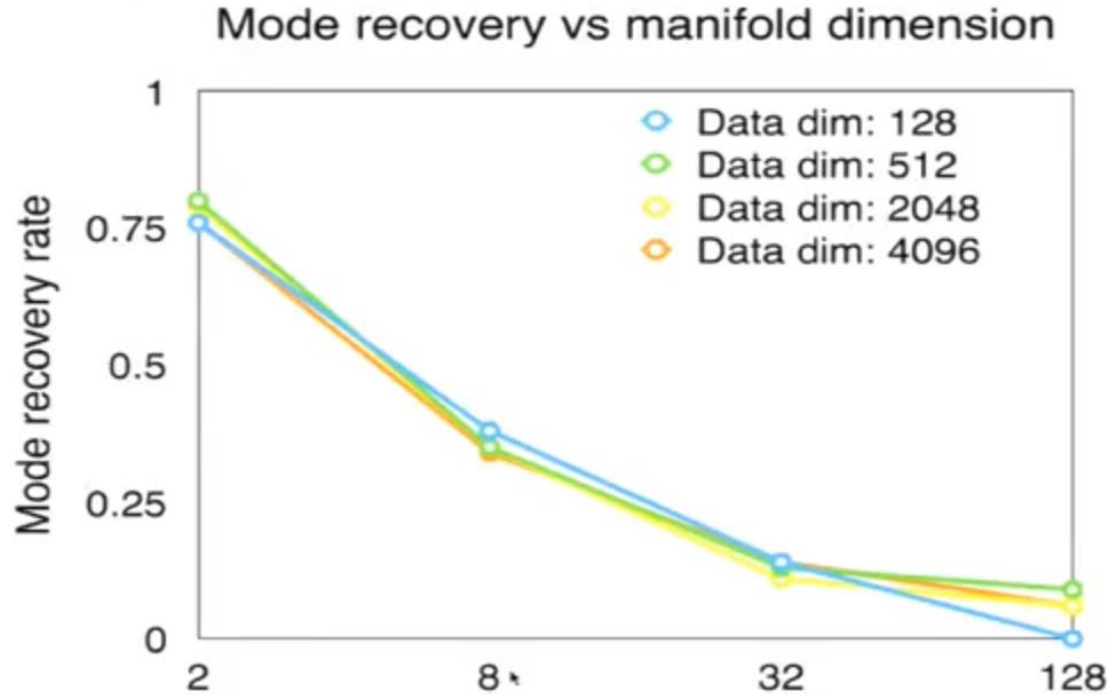
-> More modes, smaller recovery rate!  
-> part of the reason, why we often see GAN-results on specific domains (e.g., faces)



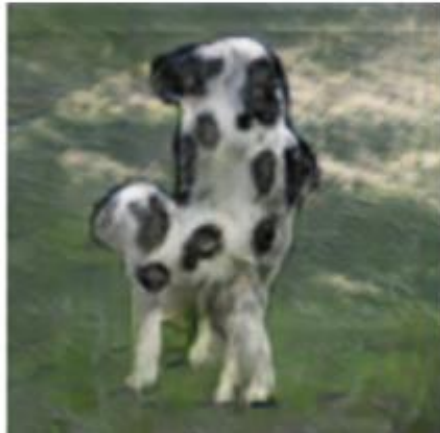
# Mode Collapse

- Performance correlates with dim of manifold

-> Larger latent space, more mode collapse



# Problems with Global Structure



(Goodfellow 2016)

# Problems with Counting



(Goodfellow 2016)

# Evaluation of GAN Performance



# Evaluation of GAN Performance

- Main difficulty of GANs: we don't know how good they are
- People cherry pick results in papers -> some of them will always look good, but how to quantify?
- Do we only memorize or do we generalize?
- GANs are difficult to evaluate! [This et al., ICLR 2016]

# Evaluation of GAN Performance

- Human evaluation:
  - Every  $n$  updates, show a series of predictions
  - Check train curves
  - What does 'look good' mean at the beginning?
    - Need variety!
    - But don't have 'realistic' predictions yet...
  - If it doesn't look good? Go back, try different hyperparameters...

# Evaluation of GAN Performance

- Inception Score (IS)
  - Measures saliency and diversity
  - Train an accurate classifier
  - Train a image generation model (conditional)
  - Check how accurate the classifier can recognize the generated images
  - Makes some assumptions about data distributions...

# Evaluation of GAN Performance

- Inception Score (IS)
  - Saliency: check whether the generated images can be classified with high confidence (i.e., high scores only on a single class)
  - Diversity: check whether we obtain samples from all classes

What if we only have one good image per class?

# Evaluation of GAN Performance

- Frechet Inception Distance (FID)
  - Calculates the feature distance between the real and synthetic distribution (modelled by multivariate Gaussian)
  - Pros:
    - More robust to noise than IS
    - No class concept needs
  - Cons:
    - Still relies on pretrained Inception-V3 model features

# Evaluation of GAN Performance

- Could also look at discriminator
  - If we end up with a strong discriminator, then generator must also be good
  - Use D features, for classification network
  - Only fine-tune last layer
  - If high class accuracy  $\rightarrow$  we have a good D and G

Caveat: doesn't seem widespread in the community

# Next: Making GANs Work in Practice

- Training / Hyperparameters (most important)
- Choice of loss function
- Choice of architecture

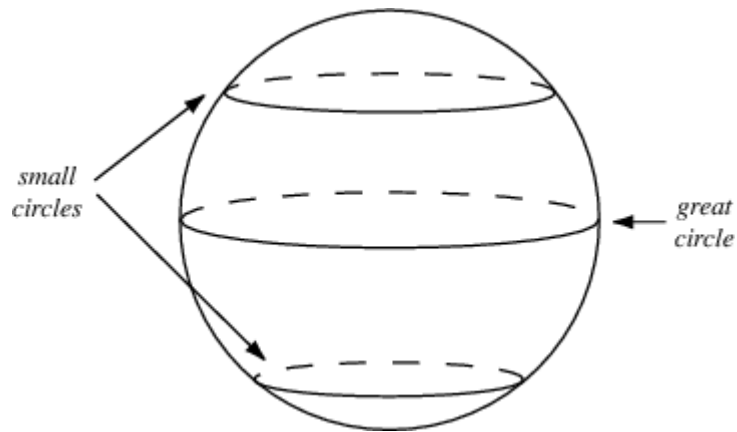
# GAN Hacks: Normalize Inputs

- Normalize the inputs between -1 and 1
- Tanh as the last layer of the generator output
- No-brainer 😊



# GAN Hacks: Sampling

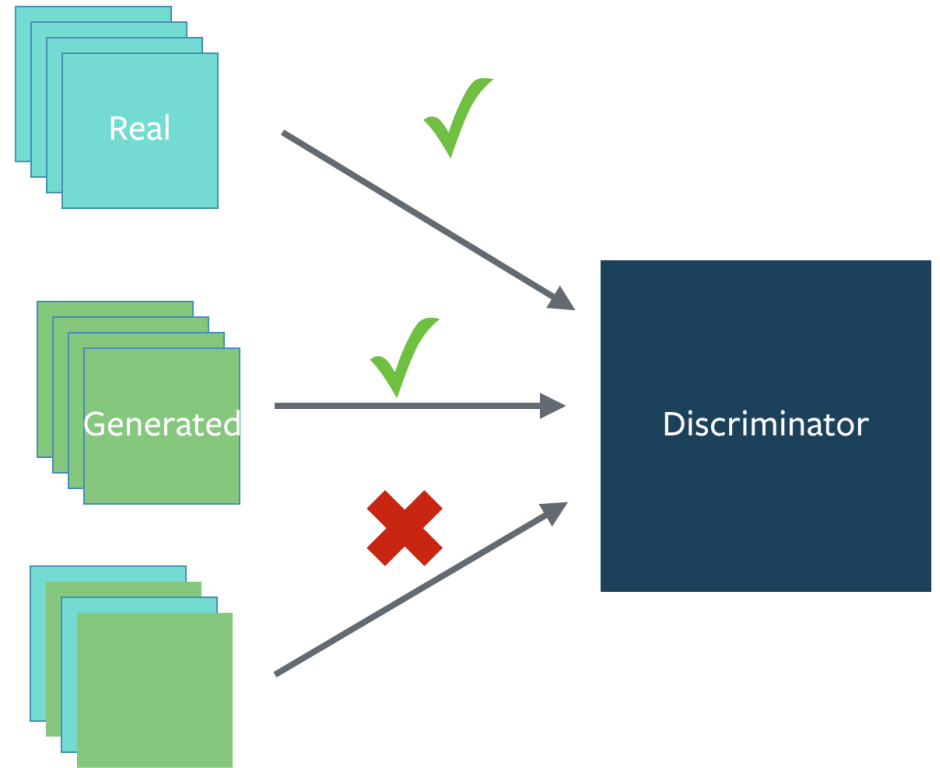
- Use a spherical  $z$
- Don't sample from a uniform distribution
- Sample from a Gaussian Distribution



- When doing interpolations, do the interpolation via a great circle, rather than a straight line from point A to point B
- Tom White's [Sampling Generative Networks](#) ref  
code <https://github.com/dribnet/plat>  
has more details

# GAN Hacks: BatchNorm

- Use Batch Norm
- Construct different mini-batches for real and fake, i.e. each mini-batch needs to contain only all real images or all generated images.



# GAN Hacks: Use ADAM

- See Adam usage [Radford et al. 15]
- SGD for discriminator
- ADAM for generator

# GAN Hacks: One-sided Label Smoothing

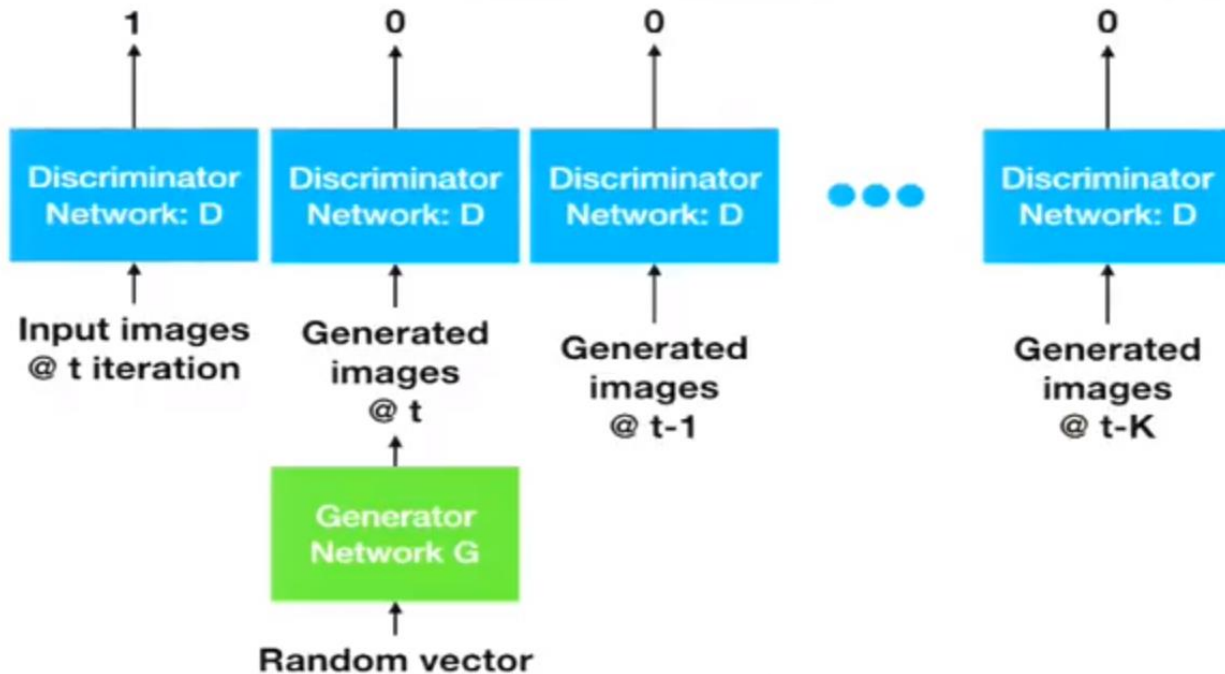
- Prevent discriminator from giving too large gradient signal to generator:

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \lambda \log D(\mathbf{x}) - \frac{1}{2}\mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))$$

Some value smaller than 1; e.g., 0.9

- > reduces confidence; i.e., makes disc. 'weaker'
- > encourages 'extreme samples' (prevents extrapolating)

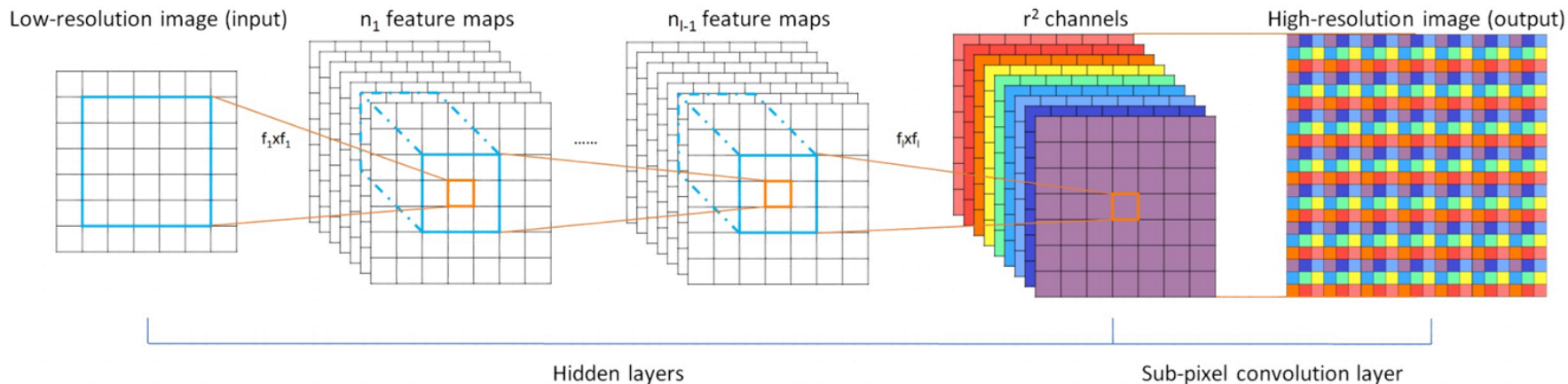
# GAN Hacks: Historical Generator Batches



Help stabilize discriminator training in early stage

# GAN Hacks: Avoid Sparse Gradients

- Stability of GAN game suffers if gradients are sparse
- LeakyReLU -> good in both G and D
- Downsample -> use average pool, conv+stride
- Upsample -> deconv+stride, PixelShuffle



# Exponential Averaging of Weights

- Problem: discriminator is noisy due to SGD
- Rather than taking final result of a GAN, would be biased on last latest iterations (i.e., latest training samples),
  - -> exponential average of weights
  - -> keep second 'vector' of weights that are averaged
    - > almost no cost, average of weights from last n iters

# Other Objective Functions

“heuristic is standard...”

GAN	DISCRIMINATOR LOSS	GENERATOR LOSS
MM GAN	$\mathcal{L}_D^{\text{GAN}} = -\mathbb{E}_{x \sim p_d} [\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{\text{GAN}} = \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$
NS GAN	$\mathcal{L}_D^{\text{NSGAN}} = -\mathbb{E}_{x \sim p_d} [\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{\text{NSGAN}} = -\mathbb{E}_{\hat{x} \sim p_g} [\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_D^{\text{WGAN}} = -\mathbb{E}_{x \sim p_d} [D(x)] + \mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$	$\mathcal{L}_G^{\text{WGAN}} = -\mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$
WGAN GP	$\mathcal{L}_D^{\text{WGANGP}} = \mathcal{L}_D^{\text{WGAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_g} [(\ \nabla D(\alpha x + (1 - \alpha \hat{x}))\ _2 - 1)^2]$	$\mathcal{L}_G^{\text{WGANGP}} = -\mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$
LS GAN	$\mathcal{L}_D^{\text{LSGAN}} = -\mathbb{E}_{x \sim p_d} [(D(x) - 1)^2] + \mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})^2]$	$\mathcal{L}_G^{\text{LSGAN}} = -\mathbb{E}_{\hat{x} \sim p_g} [(D(\hat{x}) - 1)^2]$
DRAGAN	$\mathcal{L}_D^{\text{DRAGAN}} = \mathcal{L}_D^{\text{GAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_d + \mathcal{N}(0, c)} [(\ \nabla D(\hat{x})\ _2 - 1)^2]$	$\mathcal{L}_G^{\text{DRAGAN}} = \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$
BEGAN	$\mathcal{L}_D^{\text{BEGAN}} = \mathbb{E}_{x \sim p_d} [\ x - \text{AE}(x)\ _1] - k_t \mathbb{E}_{\hat{x} \sim p_g} [\ \hat{x} - \text{AE}(\hat{x})\ _1]$	$\mathcal{L}_G^{\text{BEGAN}} = \mathbb{E}_{\hat{x} \sim p_g} [\ \hat{x} - \text{AE}(\hat{x})\ _1]$



# OtherObjective Functions

“heuristic is standard...”

GAN	DISCRIMINATOR LOSS	GENERATOR LOSS
MM GAN	$\mathcal{L}_D^{\text{GAN}} = -\mathbb{E}_{x \sim p_d} [\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{\text{GAN}} = \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$
NS GAN	$\mathcal{L}_D^{\text{NSGAN}} = -\mathbb{E}_{x \sim p_d} [\log(D(x))] - \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$	$\mathcal{L}_G^{\text{NSGAN}} = -\mathbb{E}_{\hat{x} \sim p_g} [\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_D^{\text{WGAN}} = -\mathbb{E}_{x \sim p_d} [D(x)] + \mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$	$\mathcal{L}_G^{\text{WGAN}} = -\mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$
WGAN GP	$\mathcal{L}_D^{\text{WGANGP}} = \mathcal{L}_D^{\text{WGAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_g} [(\ \nabla D(\alpha x + (1 - \alpha \hat{x}))\ _2 - 1)^2]$	$\mathcal{L}_G^{\text{WGANGP}} = -\mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})]$
LS GAN	$\mathcal{L}_D^{\text{LSGAN}} = -\mathbb{E}_{x \sim p_d} [(D(x) - 1)^2] + \mathbb{E}_{\hat{x} \sim p_g} [D(\hat{x})^2]$	$\mathcal{L}_G^{\text{LSGAN}} = -\mathbb{E}_{\hat{x} \sim p_g} [(D(\hat{x} - 1))^2]$
DRAGAN	$\mathcal{L}_D^{\text{DRAGAN}} = \mathcal{L}_D^{\text{GAN}} + \lambda \mathbb{E}_{\hat{x} \sim p_d + \mathcal{N}(0, c)} [(\ \nabla D(\hat{x})\ _2 - 1)^2]$	$\mathcal{L}_G^{\text{DRAGAN}} = \mathbb{E}_{\hat{x} \sim p_g} [\log(1 - D(\hat{x}))]$
BEGAN	$\mathcal{L}_D^{\text{BEGAN}} = \mathbb{E}_{x \sim p_d} [\ x - \text{AE}(x)\ _1] - k_t \mathbb{E}_{\hat{x} \sim p_g} [\ \hat{x} - \text{AE}(\hat{x})\ _1]$	$\mathcal{L}_G^{\text{BEGAN}} = \mathbb{E}_{\hat{x} \sim p_g} [\ \hat{x} - \text{AE}(\hat{x})\ _1]$

The loss function alone will not make it suddenly work!

# GAN Losses: EBGAN

- Discriminator is AE (Energy-based GAN)
- a good autoencoder: we want the reconstruction cost  $D(x)$  for real images to be low.
- a good critic: we want to penalize the discriminator if the reconstruction error for generated images drops below a value  $m$ .

$$\mathcal{L}_D(x, z) = D(x) + [m - D(G(z))]^+$$

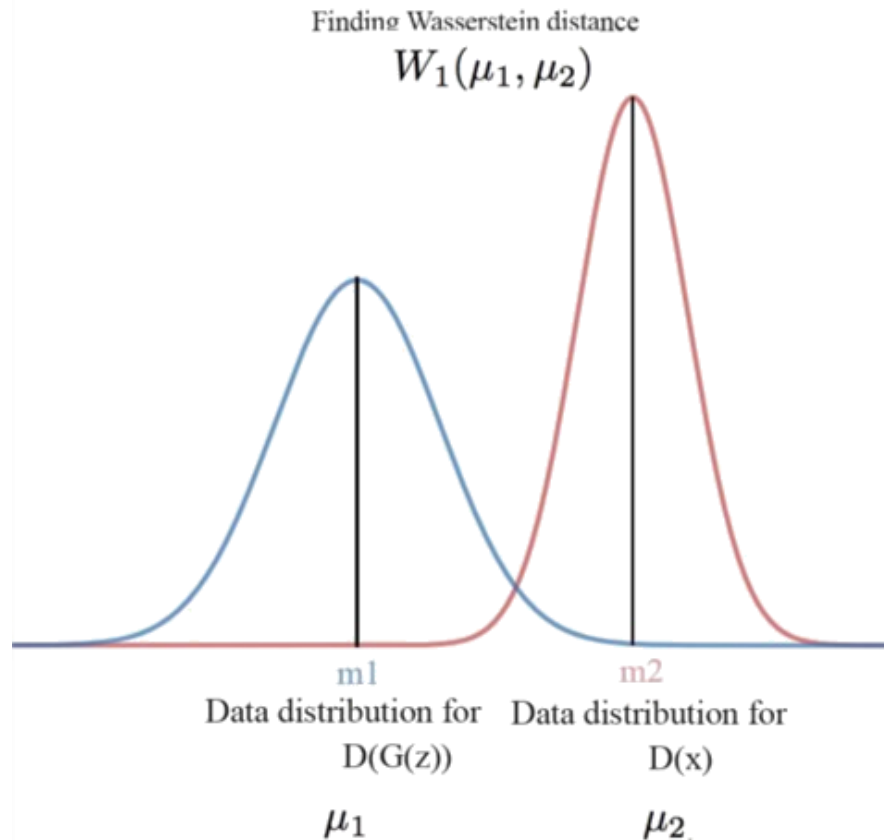
$$\mathcal{L}_G(z) = D(G(z))$$

$$D(x) = ||Dec(Enc(x)) - x||$$

$$\text{where } [u]^+ = \max(0, u)$$

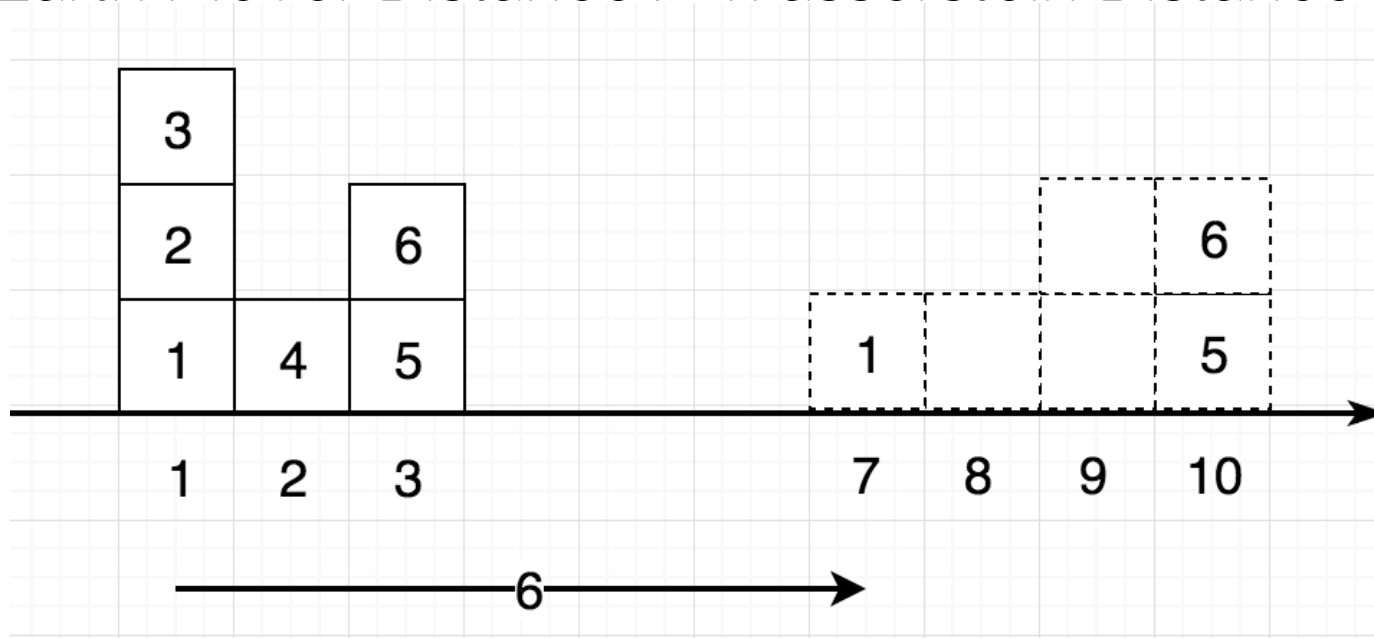
# GAN Losses: BEGAN

- Similar to EBGAN
- Instead of reconstruction loss, measure difference in data distribution of real and generated images



# GAN Losses: WGAN

- Earth Mover Distance / Wasserstein Distance



Minimum amount of work to move earth from  $p(x)$  to  $q(x)$

# GAN Losses: WGAN

- Formulate EMD via it's dual:

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r} [f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta} [f(x)]$$

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|.$$

1-Lipschitz function: upper bound between densities

# GAN Losses: WGAN

$$|f(x_1) - f(x_2)| \leq |x_1 - x_2|.$$

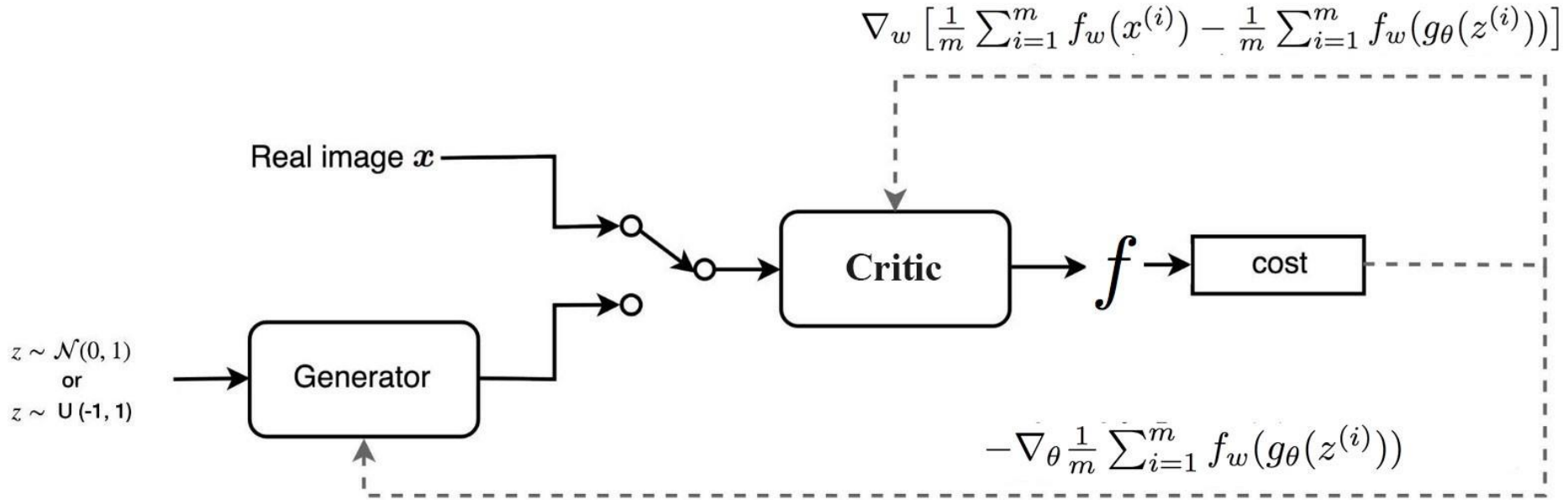
$f$  is a critic function, defined by a neural network

->  $f$  needs to be 1-Lipschitz; WGAN restricts max weight value in  $f$ ;  
weights of the discriminator must be within a certain range controlled by hyperparameters  $c$

$$w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$$

$$w \leftarrow \text{clip}(w, -c, c)$$

# GAN Losses: WGAN



# GAN Losses: WGAN

	<b>Discriminator/Critic</b>	<b>Generator</b>
<b>GAN</b>	$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[ \log D(\mathbf{x}^{(i)}) + \log(1 - D(G(\mathbf{z}^{(i)}))) \right]$	$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m -\log(D(G(\mathbf{z}^{(i)})))$
<b>WGAN</b>	$\nabla_w \frac{1}{m} \sum_{i=1}^m [f(\mathbf{x}^{(i)}) - f(G(\mathbf{z}^{(i)}))] ]$	$\nabla_{\theta} \frac{1}{m} \sum_{i=1}^m -f(G(\mathbf{z}^{(i)}))$



# GAN Losses: WGAN

---

**Algorithm 1** WGAN, our proposed algorithm. All experiments in the paper used the default values  $\alpha = 0.00005$ ,  $c = 0.01$ ,  $m = 64$ ,  $n_{\text{critic}} = 5$ .

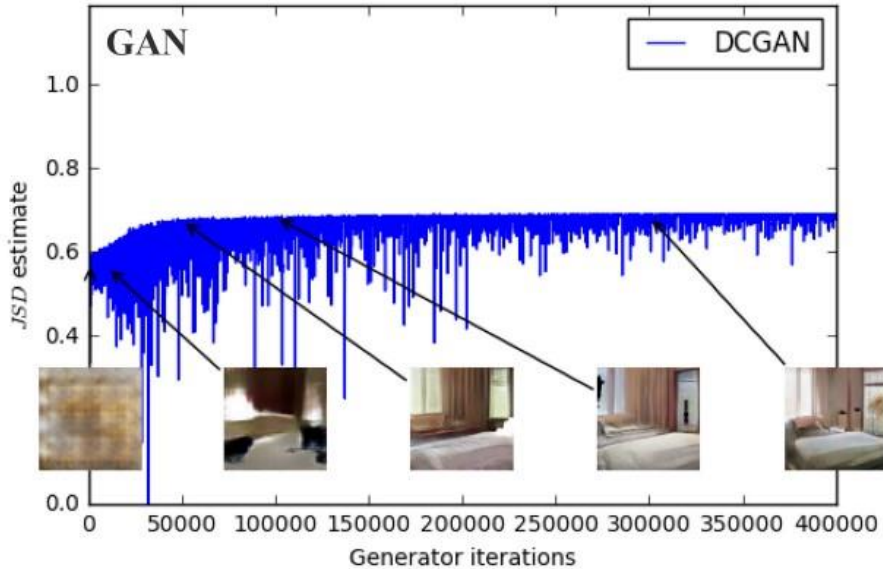
---

**Require:** :  $\alpha$ , the learning rate.  $c$ , the clipping parameter.  $m$ , the batch size.  $n_{\text{critic}}$ , the number of iterations of the critic per generator iteration.

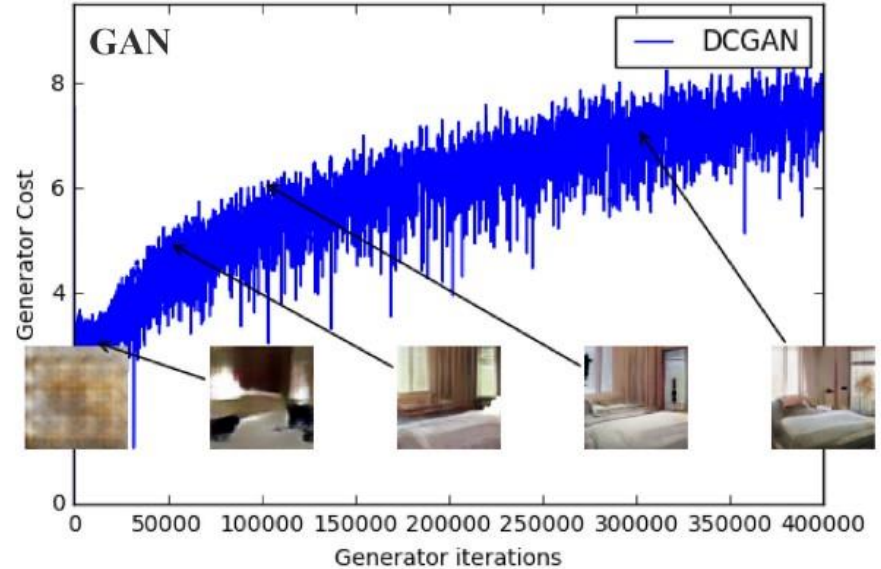
**Require:** :  $w_0$ , initial critic parameters.  $\theta_0$ , initial generator's parameters.

```
1: while  $\theta$  has not converged do
2:   for  $t = 0, \dots, n_{\text{critic}}$  do
3:     Sample  $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$  a batch from the real data.
4:     Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
5:      $g_w \leftarrow \nabla_w \left[ \frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)})) \right]$ 
6:      $w \leftarrow w + \alpha \cdot \text{RMSPProp}(w, g_w)$ 
7:      $w \leftarrow \text{clip}(w, -c, c)$ 
8:   end for
9:   Sample  $\{z^{(i)}\}_{i=1}^m \sim p(z)$  a batch of prior samples.
10:   $g_\theta \leftarrow -\nabla_\theta \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$ 
11:   $\theta \leftarrow \theta - \alpha \cdot \text{RMSPProp}(\theta, g_\theta)$ 
12: end while
```

# GAN Losses: WGAN

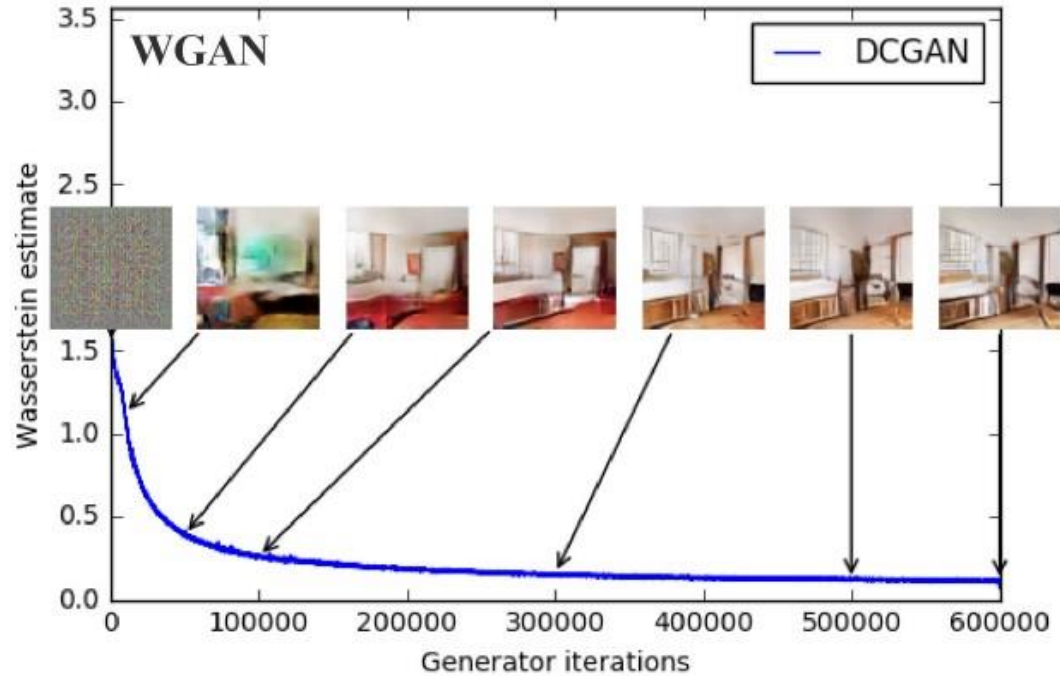


$$\frac{1}{m} \sum_{i=1}^m \log(1 - D(G(z^{(i)})))$$



$$\frac{1}{m} \sum_{i=1}^m -\log(D(G(z^{(i)})))$$

# GAN Losses: WGAN



$$\frac{1}{m} \sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^m f_w(g_\theta(z^{(i)}))$$

# GAN Losses: WGAN

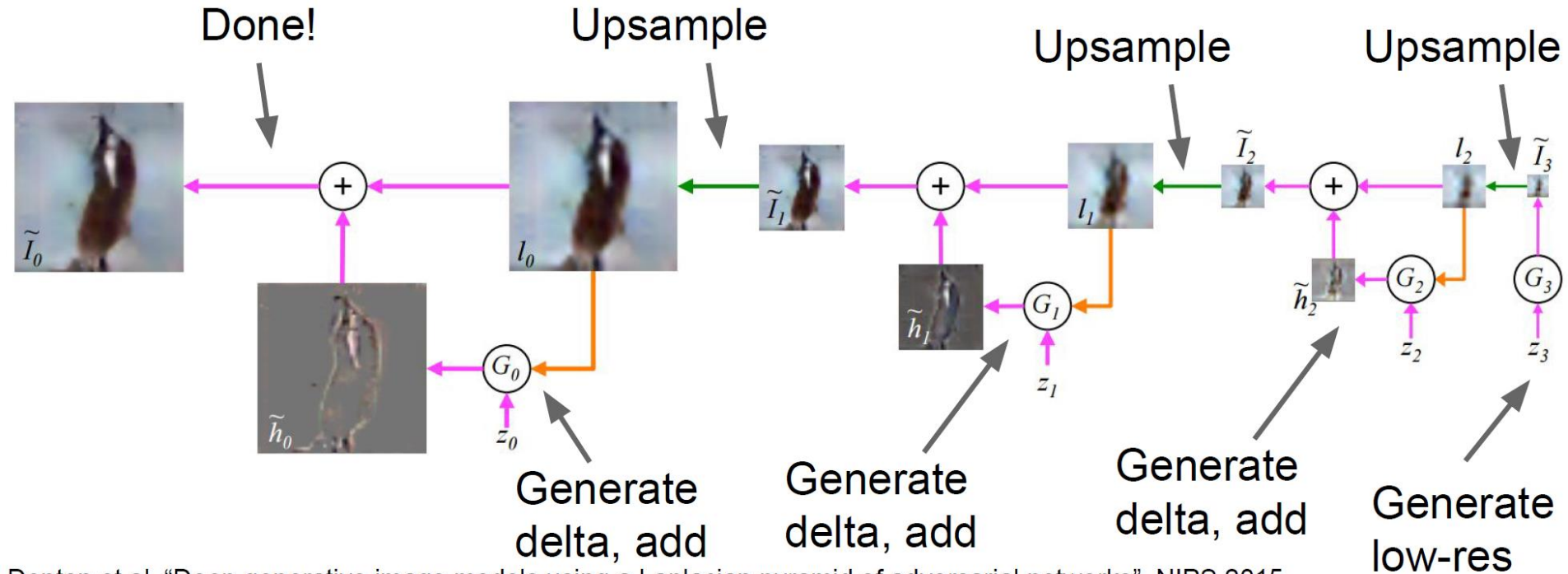
- + mitigates mode collapse
- + generator still learns when critic performs well
- + actual convergence
  
- Enforcing Lipschitz constraint is difficult
- Weight clipping is “terrible”
  - -> too high: takes long time to reach limit; slow training
  - > too small: vanishing gradients when layers are big

# GAN Losses

- Many more variations!!!
- High-level understanding: “loss” is a meta loss to train the actual loss (i.e.,  $D$ ) to provide gradients for  $G$
- Always start simple: if things don't converge, don't randomly shuffle loss around; always try easy things first (AE, VAE, 'simple heuristic' GAN)

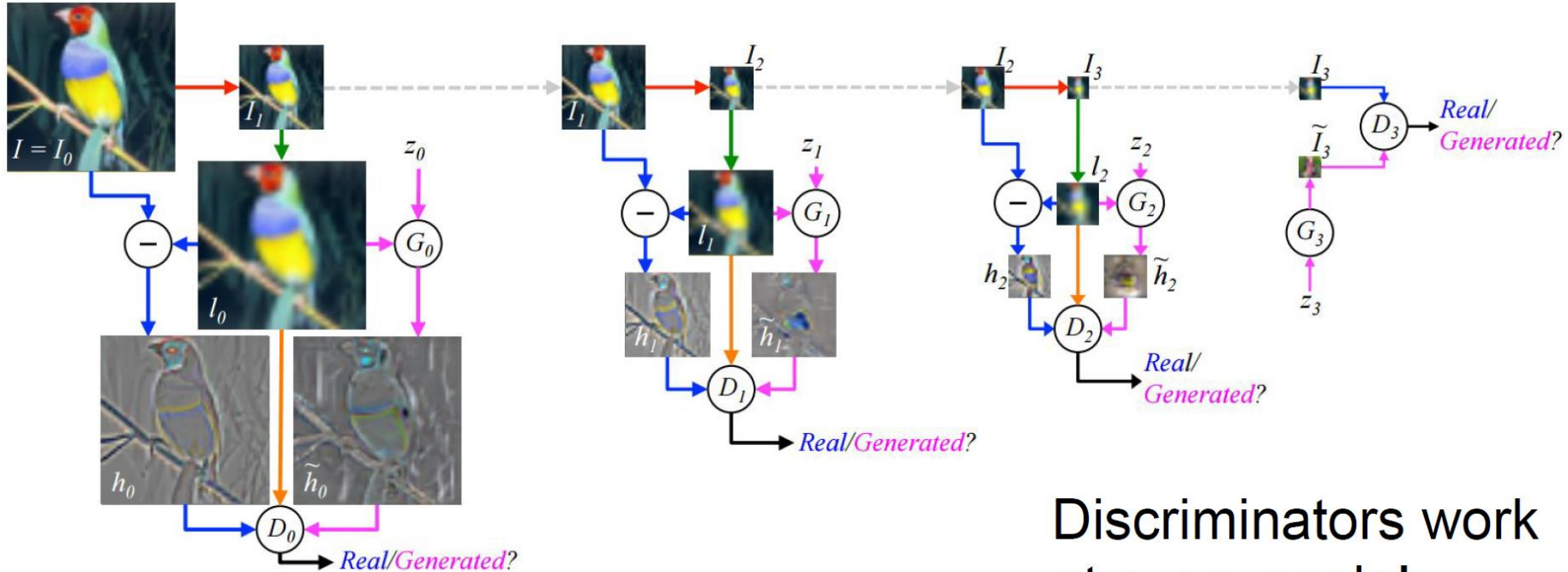
# GAN Architectures

# Multiscale GANs



Denton et al, "Deep generative image models using a Laplacian pyramid of adversarial networks", NIPS 2015

# Multiscale GANs

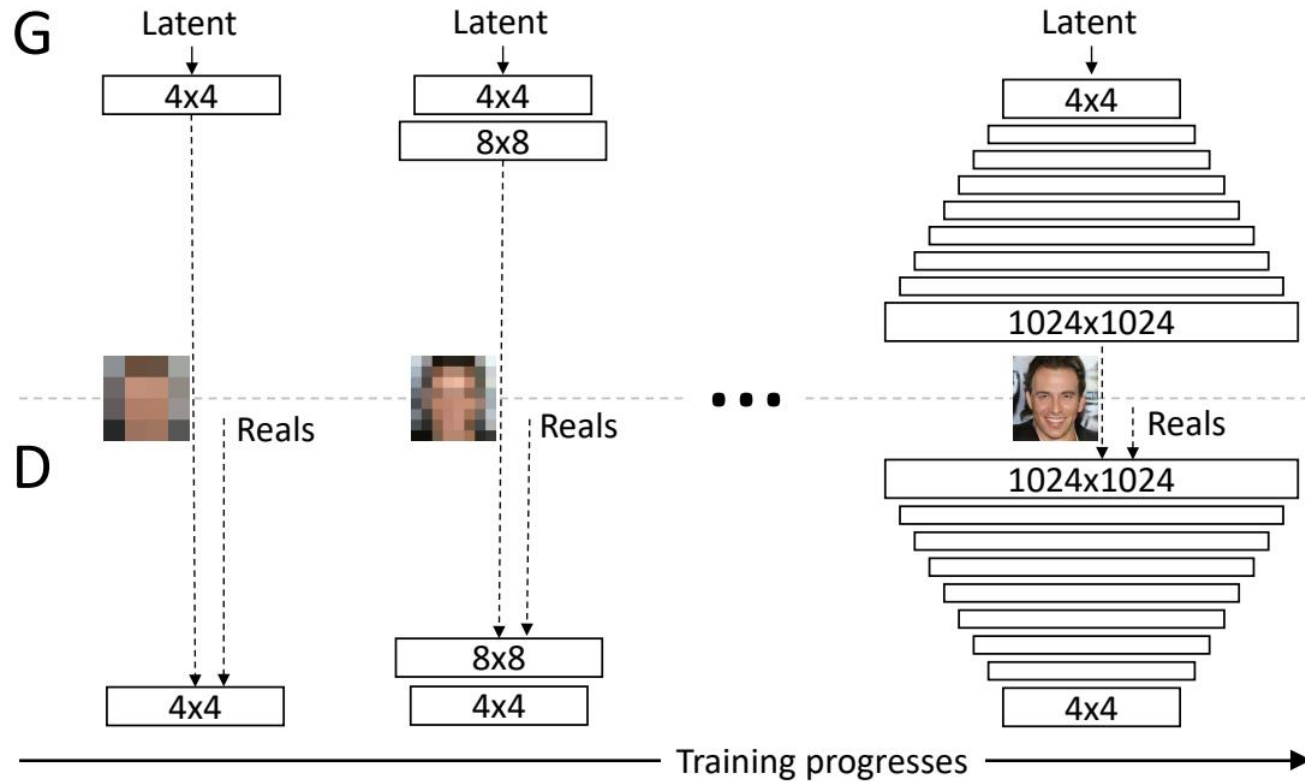


Discriminators work at every scale!

Denton et al, NIPS 2015

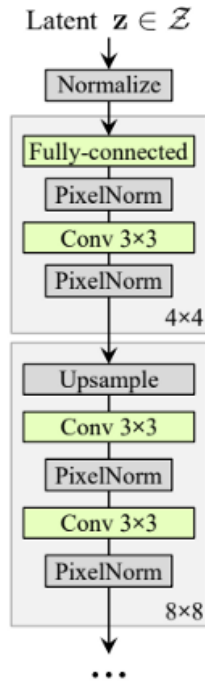


# Progressive Growing GANs



# StyleGAN[x] Architectures

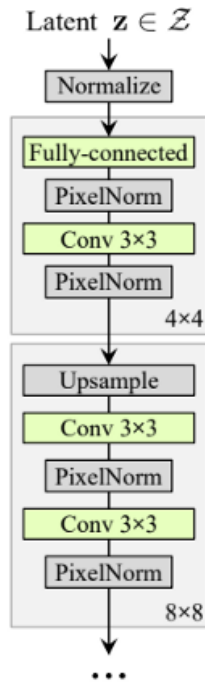
# StyleGAN Architectures



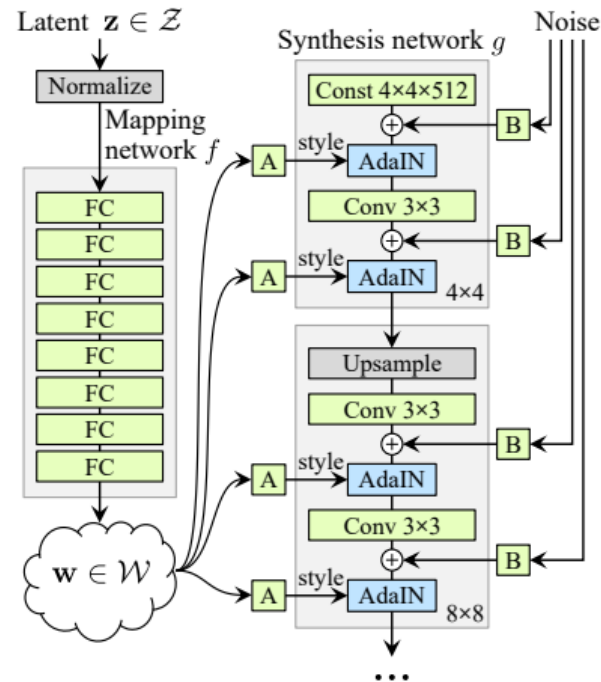
Progressive GAN

Method	CelebA-HQ	FFHQ
A Baseline Progressive GAN [30]	7.79	8.04
B + Tuning (incl. bilinear up/down)	6.11	5.25
C + Add mapping and styles	5.34	4.85
D + Remove traditional input	5.07	4.88
E + Add noise inputs	<b>5.06</b>	4.42
F + Mixing regularization	5.17	<b>4.40</b>

# StyleGAN(x) Architectures

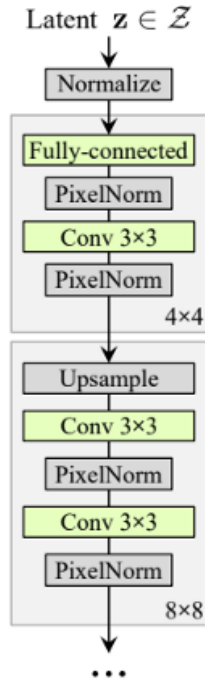


Progressive GAN

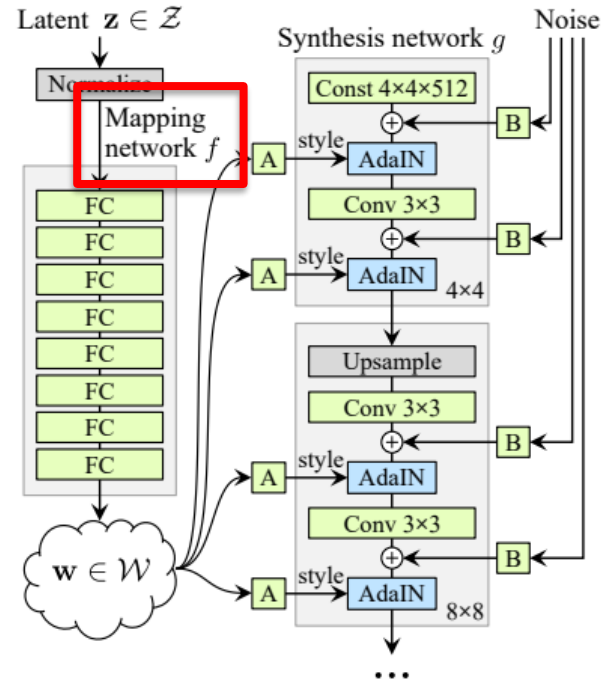


StyleGAN

# StyleGAN – Mapping Network

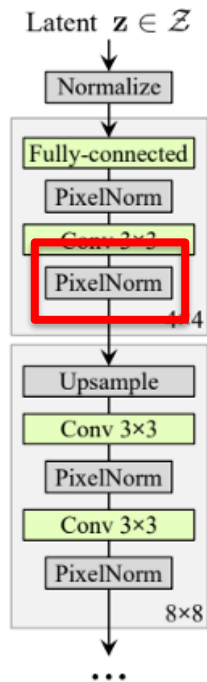


Progressive GAN

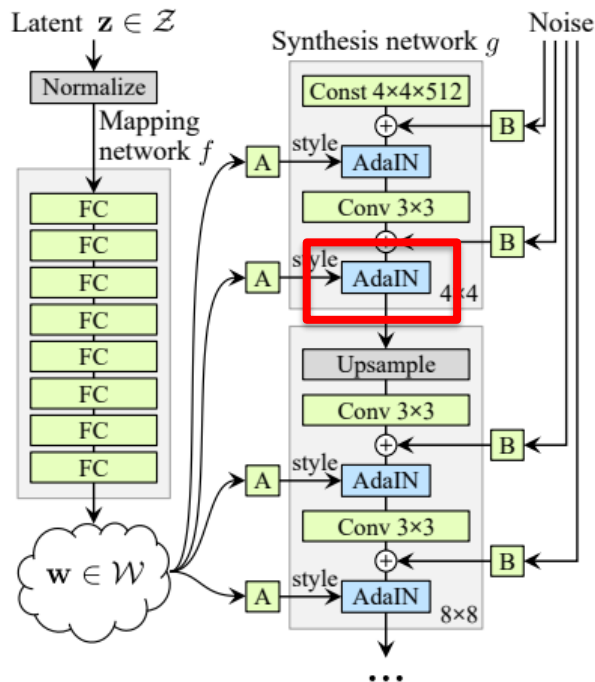


StyleGAN

# StyleGAN – Style Normalization

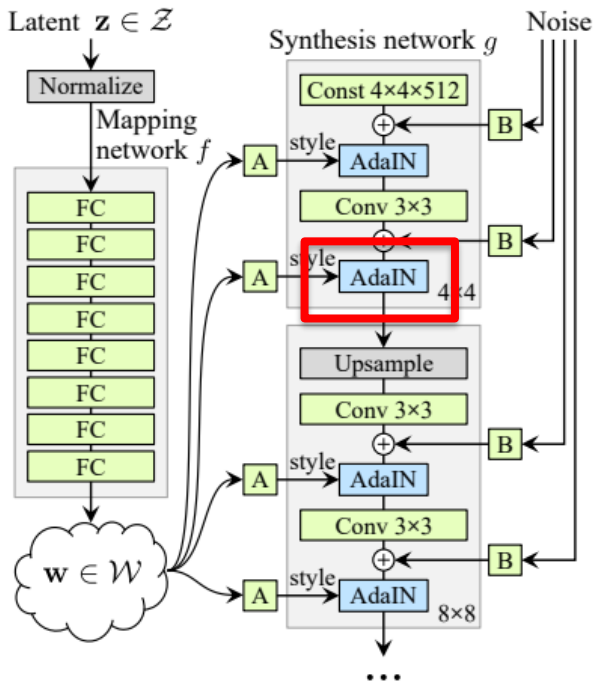


Progressive GAN



StyleGAN

# StyleGAN – Style Normalization

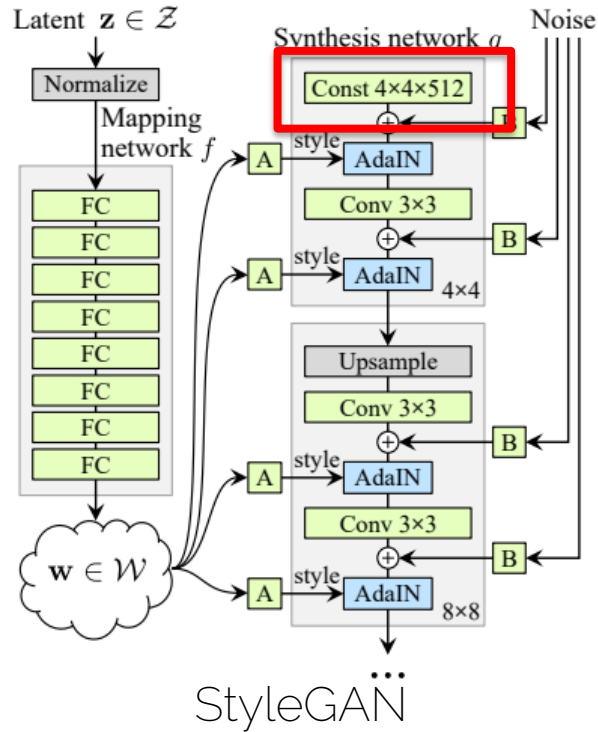


StyleGAN

$$\text{AdaIN}(x, y) = \sigma(y) \left( \frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y)$$

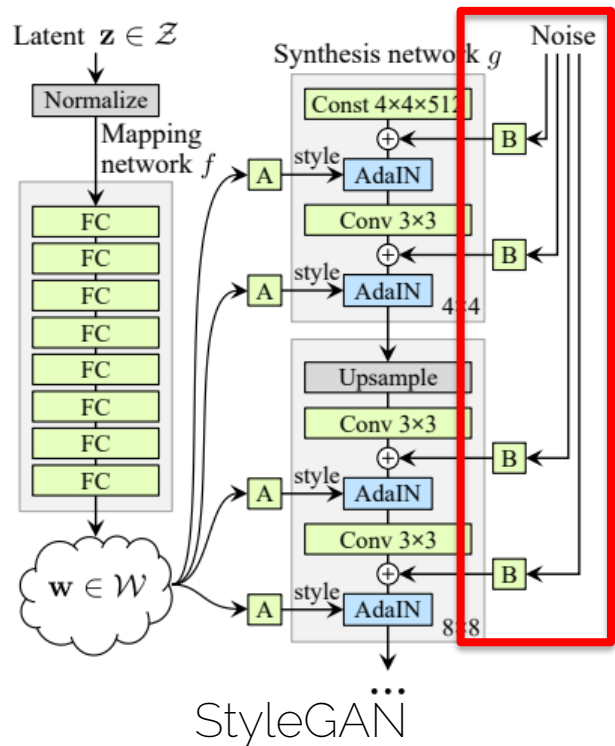
1.  $x$ : the activation from the previous layer
2.  $y$ : the style features (e.g. extracted from CNN) of your target style image
3. No trainable variables – mean and var directly calculated

# StyleGAN – Constant Input

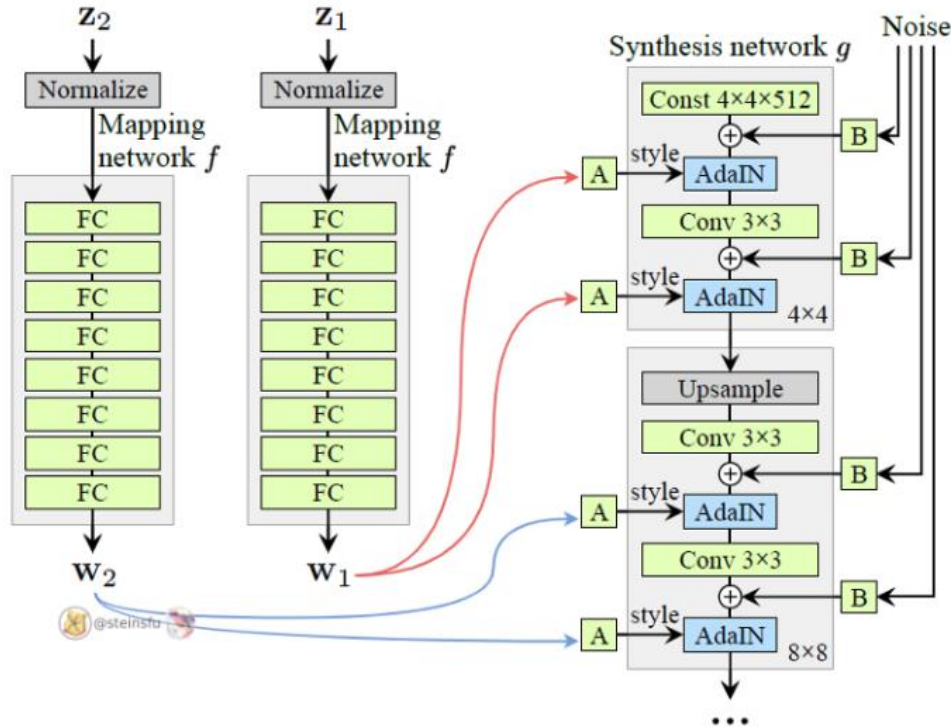




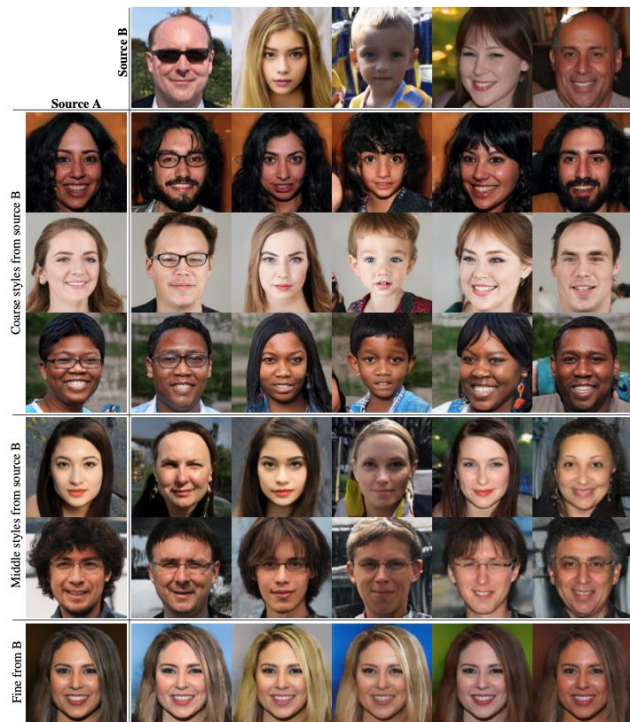
# StyleGAN – Style Normalization



# StyleGAN – Mixing Regularization



# StyleGAN



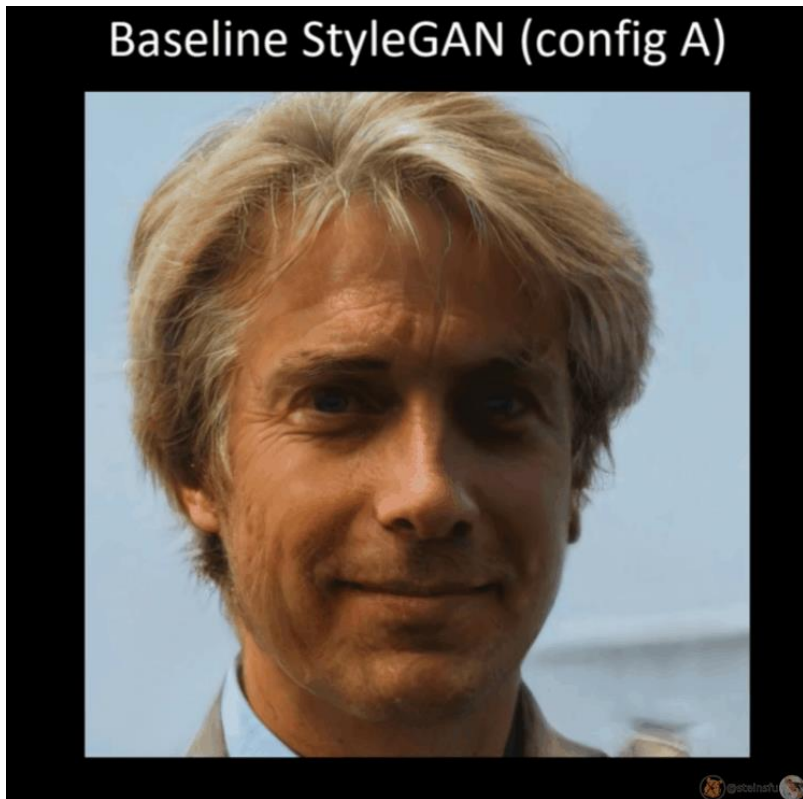
# StyleGAN2 – No Droplets



Normalization

$$AdaIN(x, y) = \sigma(y) \left( \frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y)$$

Modulation

Without Normalization, the droplet artifact disappear



 @steinsfu 

A → AdaIN

$$AdaIN(x_i, y) = y_{s,i} \frac{x_i - \mu(x_i)}{\sigma(x_i)} + y_{b,i}$$

Redraw ↓

Norm mean/std

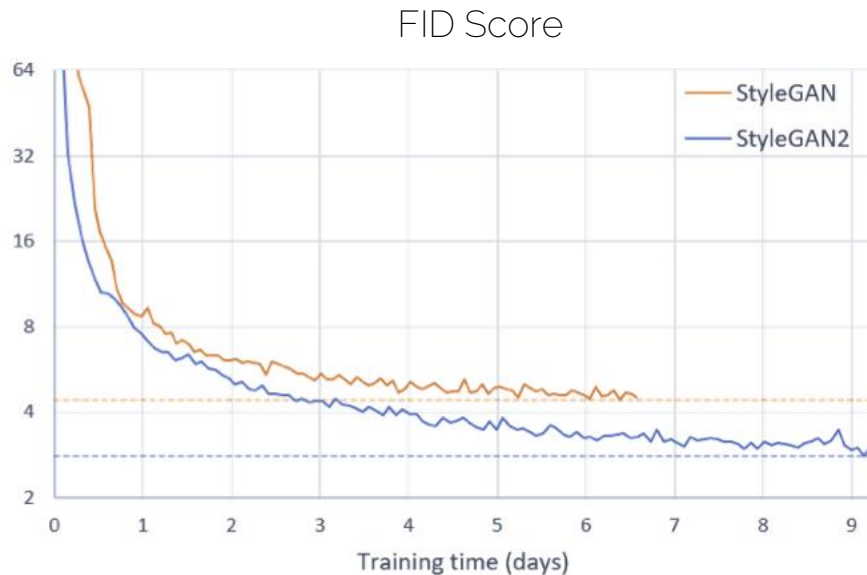
$$Norm(x_i) = \frac{x_i - \mu(x_i)}{\sigma(x_i)}$$

A → Mod mean/std

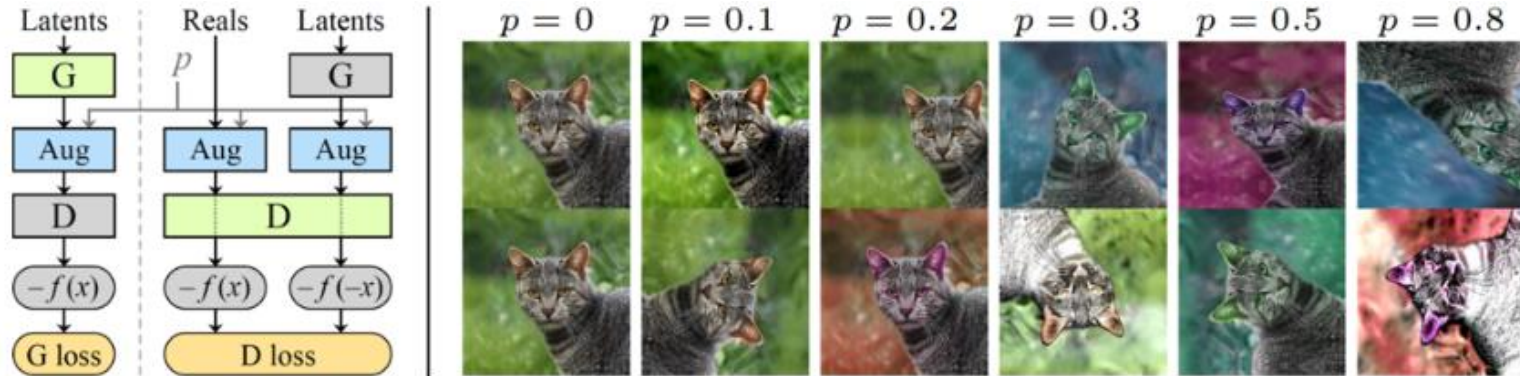
$$Mod(o_i, y) = y_{s,i} \cdot o_i + y_{b,i}$$

# StyleGAN2 – Additional Changes

- Remove redundant operations
- Noise added outside of style area
- Normalization and modulation only applied on standard deviation
- Modulation and Convolution combined in single operation
- Training strategy changes, see: <https://github.com/NVLabs/stylegan2>



# StyleGAN2-ADA – Limited Data



The Discriminator latents are directly augmented with probability  $p$ .

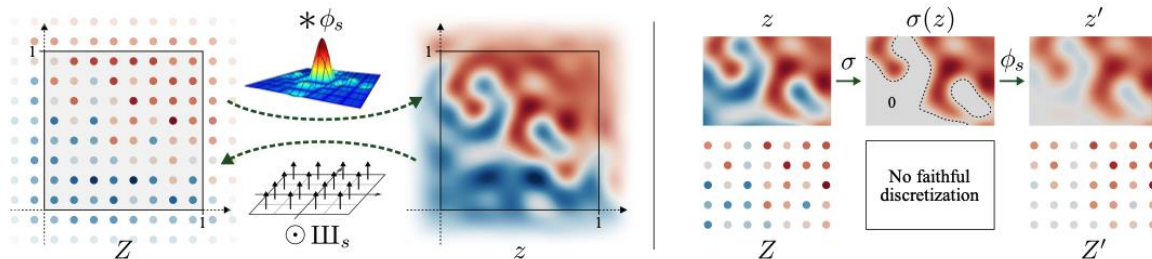
Better for limited Data

No manual augmentation

# StyleGAN3 – No Aliasing

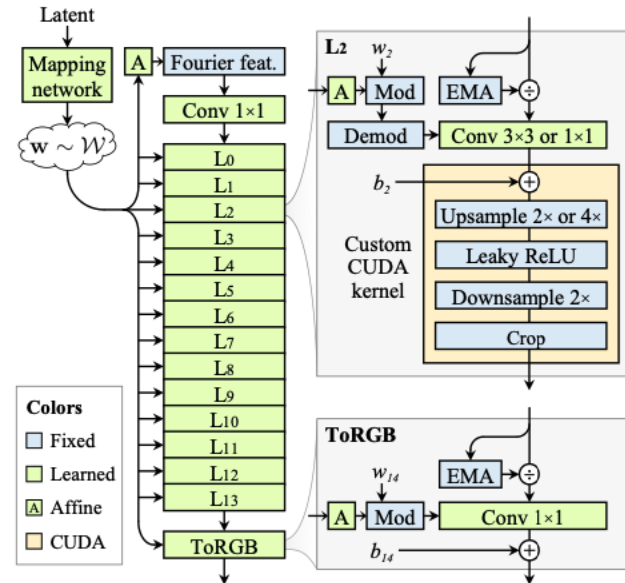


# StyleGAN3 – No Aliasing



Most Important differences:

- Input constant replaced with continuous Fourier feature
- Remove per pixel noise – no positional references
- Smaller mapping network depth
- Better upsampling with updated approximations of the Fourier low pass filter





# Reading Homework

- GANs [Goodfellow et al. 2014] Generative adversarial networks
  - <https://arxiv.org/abs/1406.2661>
- [Radford et al. 2015] Unsupervised representation learning with deep convolutional generative adversarial networks
  - <https://arxiv.org/abs/1511.06434>
- [Karras et al. 19] A style-based generator architecture for generative adversarial networks.
  - <https://arxiv.org/abs/1812.04948>

Thanks for watching!