Unsupervised Learning: Implicit Models (GAN)

Earl Wong

Unsupervised Learning Models (Generative Models)

- Autoregressive (AR)
- Flow
- Variational AutoEncoder (VAE)
- Generative Adversarial Network (GAN)

Regarding Generative Models

- We would like our generative model to produce samples from a learned distribution p_model(x).
- With AR and Flow, the models are explicit.
- With VAEs, the model is explicit, but approximate.
- With GANs, the model is implicit.

Model: GAN

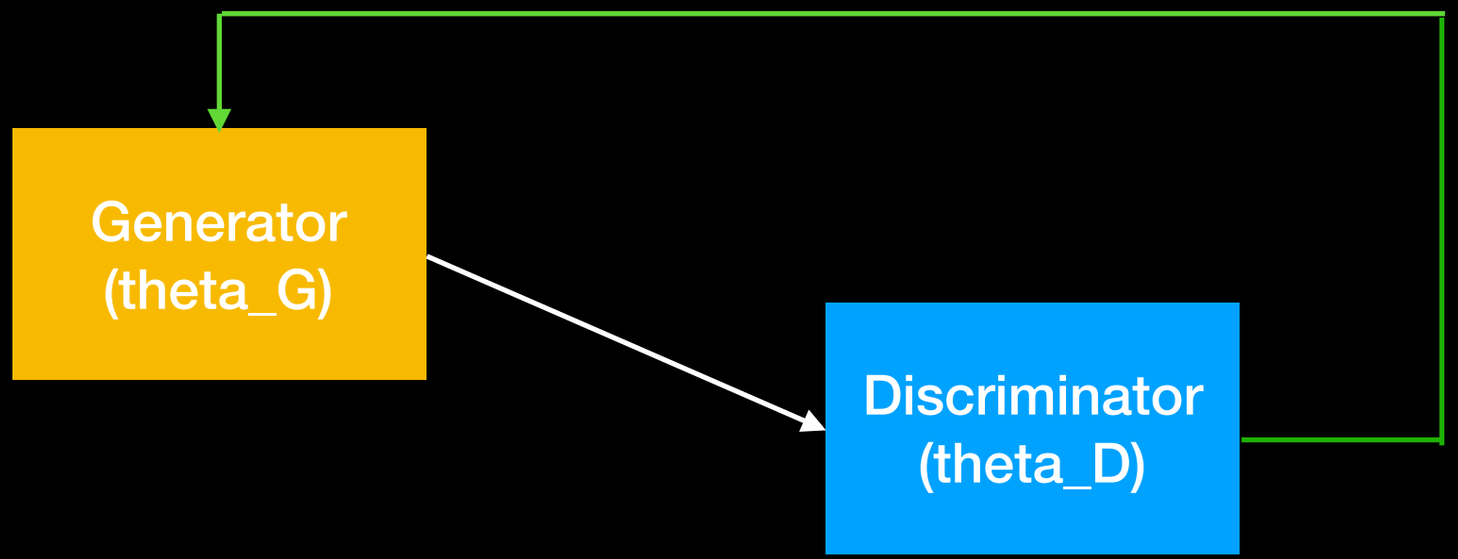
- GANS are implicit generative models.
- Hence, an explicit density function is never computed.
- Because of this, it is difficult to quantitatively evaluate a GAN using traditional methods.
- Instead, the following metrics are used: Inception Score - IS and Frechet Inception Distance - FID.

GAN: Main Concept

- A GAN replicates a minimax game.
- It has two cost functions - one associated with the generator and one associated with the discriminator.
- The objective of the minimax game is to find the the parameters that result in the lowest cost for both the generator and the discriminator.
- As a result, it has two parts / two passes:

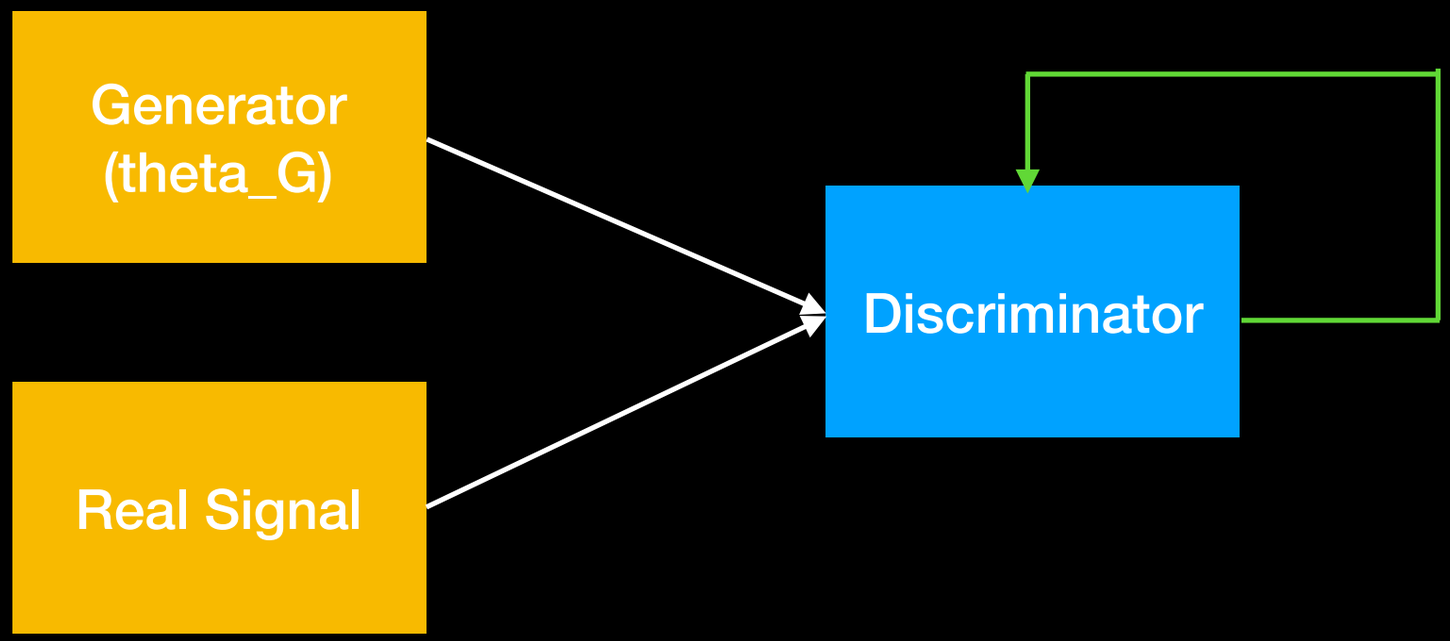
GAN: Pass 1

The generator and discriminator are neural networks.



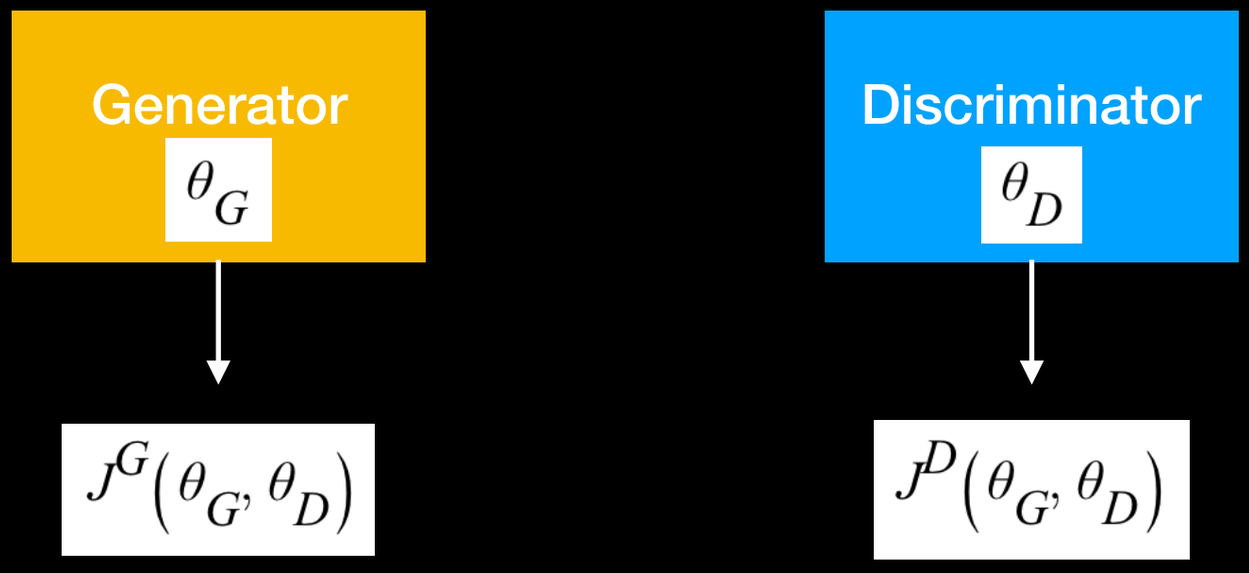
1) The generator sends a fake signal to the discriminator.
2) The discriminator determines whether the signal is real or fake: D(G)
3) This feedback signal is then used to improve the generator.

GAN: Pass 2



1) The generator sends a fake signal to the discriminator.
2) The discriminator determines whether the signal is real or fake: D(G)
3) The discriminator also receives a real signal.
4) The discriminator determines whether the signal is real or fake: D(real)
5) An average loss is computed: [D(Real) + D(G)] / 2.
6) This feedback signal is used to update the discriminator.

GAN: Cost Functions



The generator wishes to minimize its cost (J^G) by generating realistic looking signals. This will result in D(G) = 1.

The discriminator wishes to minimize its cost (J^D) by being able to differentiable between fake and real signals. This will result in D(real) = 1 and D(G) = 0.

These requirements result in a 2 player Nash equilibrium game.

The objective of the game is to find theta_G and theta_D that produce local minima for the respective cost functions.

GAN Objective

The cost associated with the discriminator is a combination of the results of being fed true data (D(real)=1) and fake data (D(G) = 0)

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

The cost associated with the generator is the opposite of the generator.

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

The underlying objective can be summarized by defining the following value function

$$V\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right) = -J^{(D)}\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right)$$

and applying it as shown below. In general, we have a zero sum / minimax game where we wish minimize the outer loop while maximizing the inner loop.

$$\boldsymbol{\theta}^{(G)*} = \arg\min_{\boldsymbol{\theta}^{(G)}} \max_{\boldsymbol{\theta}^{(D)}} V\left(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}\right)$$

GAN Objective

- Using this formulation, it can be shown that resulting optimal discriminator is given by:

$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}$$

- Here, P_r and P_g represent the real and generated data distributions.

- Aside: The above, is equivalent to minimizing the Jensen Shannon divergence:

$$\min_\theta JS\left(P_r \middle|\middle| P_g\right)$$

$$JS\left(P_r \middle|\middle| P_g\right) = KL\left(P_r \middle|\middle| \frac{\left(P_r + P_g\right)}{2}\right) + KL\left(P_g \middle|\middle| \frac{\left(P_r + P_g\right)}{2}\right)$$

GAN Practicalities

- In practice, it is not advisable to compute the optimal discriminator D*, before attempting to minimize the generator loss.

- In addition, it is also ill advised to have a discriminator that is highly superior to the generator.

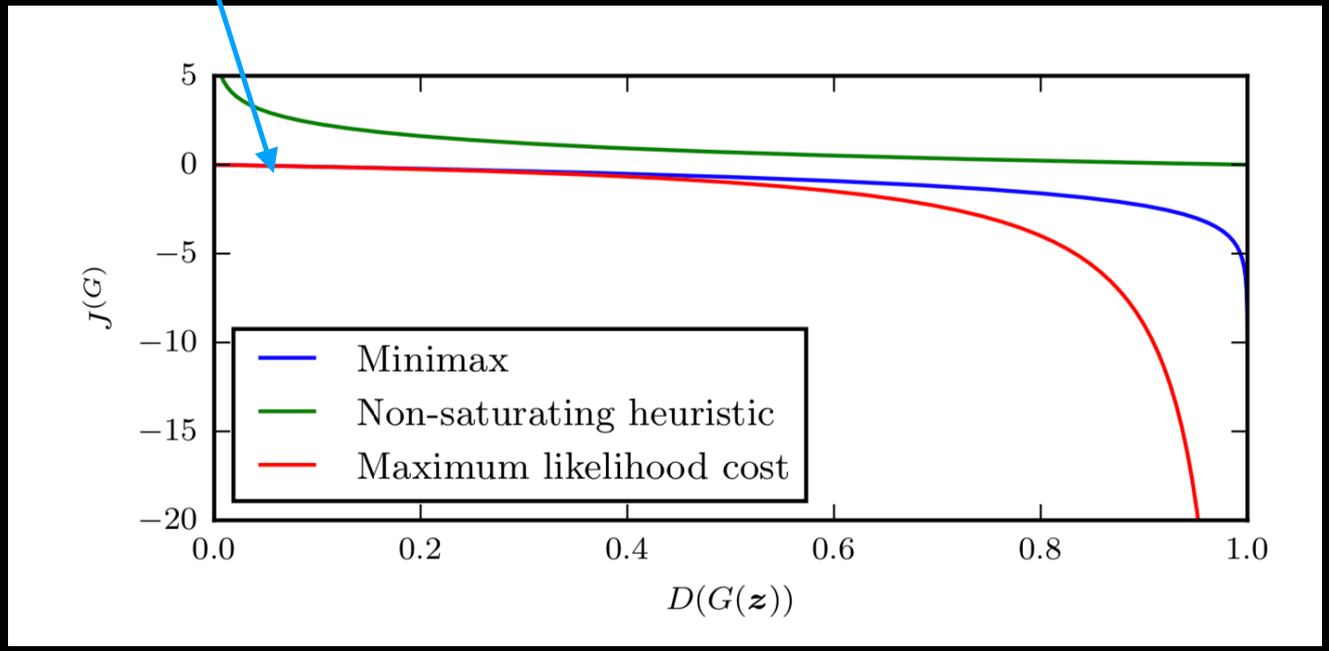
- In these scenarios, the discriminator cannot send a meaningful signal to the generator during training.

- i.e. When the quality of the generated signal G is low (at the onset of training) and the discriminator is very efficient at screening out the fake discriminator signal (i.e. D(G) = “small”), no meaningful signal is sent to the generator by the discriminator.

Pictorially

$$\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(z)\right)\right)$$

D(G) ~ 0 when the generator is weak and the discriminator is strong. Hence, little to no signal is being sent by the discriminator to help train the generator.



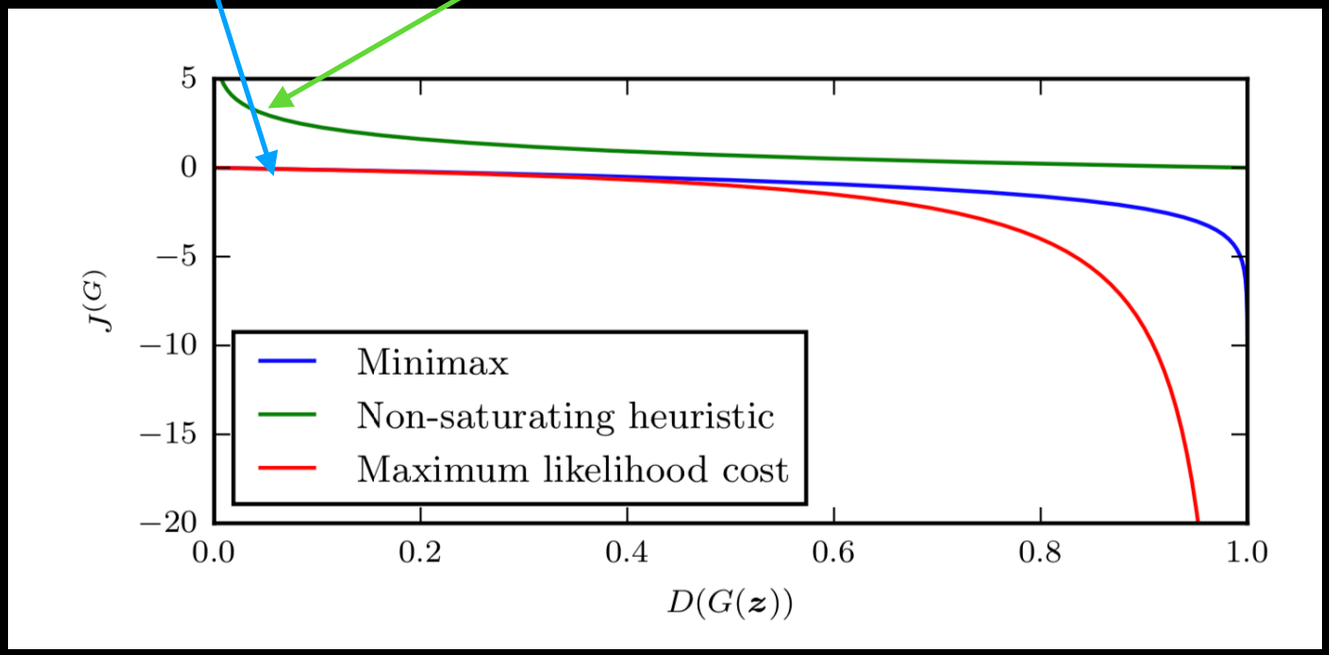
GAN Practicalities

$$\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log\left(1 - D\left(G(z)\right)\right)$$

To remedy this problem, Goodfellow proposed the following modification,

$$-\frac{1}{2}\mathbb{E}_{\boldsymbol{z}} \log D(G(\boldsymbol{z}))$$

As a result, a meaningful signal is now sent from the discriminator to help train the generator.



GAN Practicalities

- Yet, even with this altered cost function, is was noted by Arjovsky, that GAN training still did not proceed well.
- Specifically, “the updates tended to get worse, and the optimization becomes unstable”.
- This issue was eventually addressed, with the introduction of the Wasserstein GAN.
- In the meantime, researchers used the following heuristic during training, hoping for the best.

How Do We Train the GAN?

- Apply an “alternating gradients” technique to update the two networks (= baby sitting the training). Specifically:

$$D_{weightNew} = D_{weightPast} + \alpha_D \cdot \frac{dV}{dD}$$

$$G_{weightNew} = G_{weightPast} + \alpha_G \cdot \frac{dV}{dG}$$

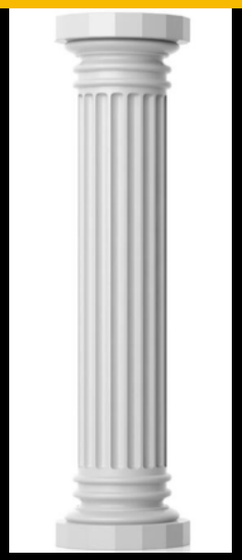
- Ideally, we want both networks to acquire knowledge at approximately the same rate and be “equally” smart or dumb.

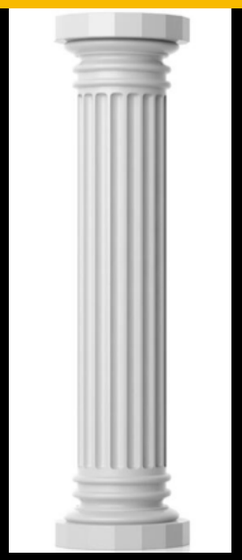
- In practice, if one network becomes significantly better than the other, additional iterations can be applied to the underperforming network.

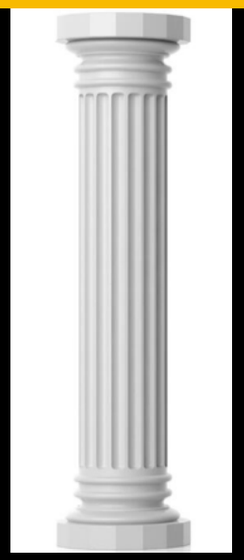
Unlabeled Dataset

Algorithm

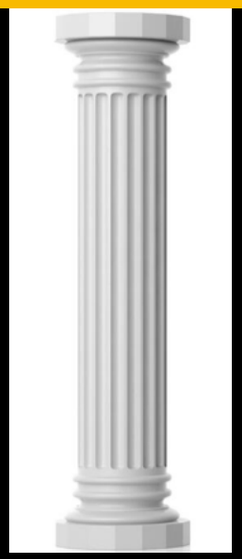
Loss Function







Unlabeled Dataset

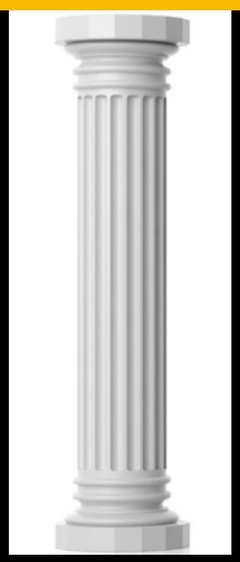


Unlabeled Dataset

DC GAN	Wasserstein GAN-GP	Spectral Normalization	Big GAN	Style GAN
ImageNet (32x32)	LSUN Bedrooms	ImageNet (128x128)	ImageNet (512x512)	FFHQ (1024x1024)
LSUN Bedrooms		CIFAR-10	JFT-300M	
Faces		STL-10		

LSUN Bedrooms = 3M bedroom images
ImageNet = 1M images
JFT-300M = 292M images
FFHQ = 70K very high quality face images

Algorithm



Progression

- DCGAN (11/15)
- Wasserstein GAN (6/17)
- Wasserstein GAN Gradient Penalty (3/17)
- Spectral Norm GAN (2/18)
- SelfAttentionGAN (5/18)
- BigGAN (9/18)
- StyleGAN (12/18), StyleGAN2 (12/19)
- Other: ConditionalGAN (11/14)
- Other: CycleGAN (3/17)

Notes

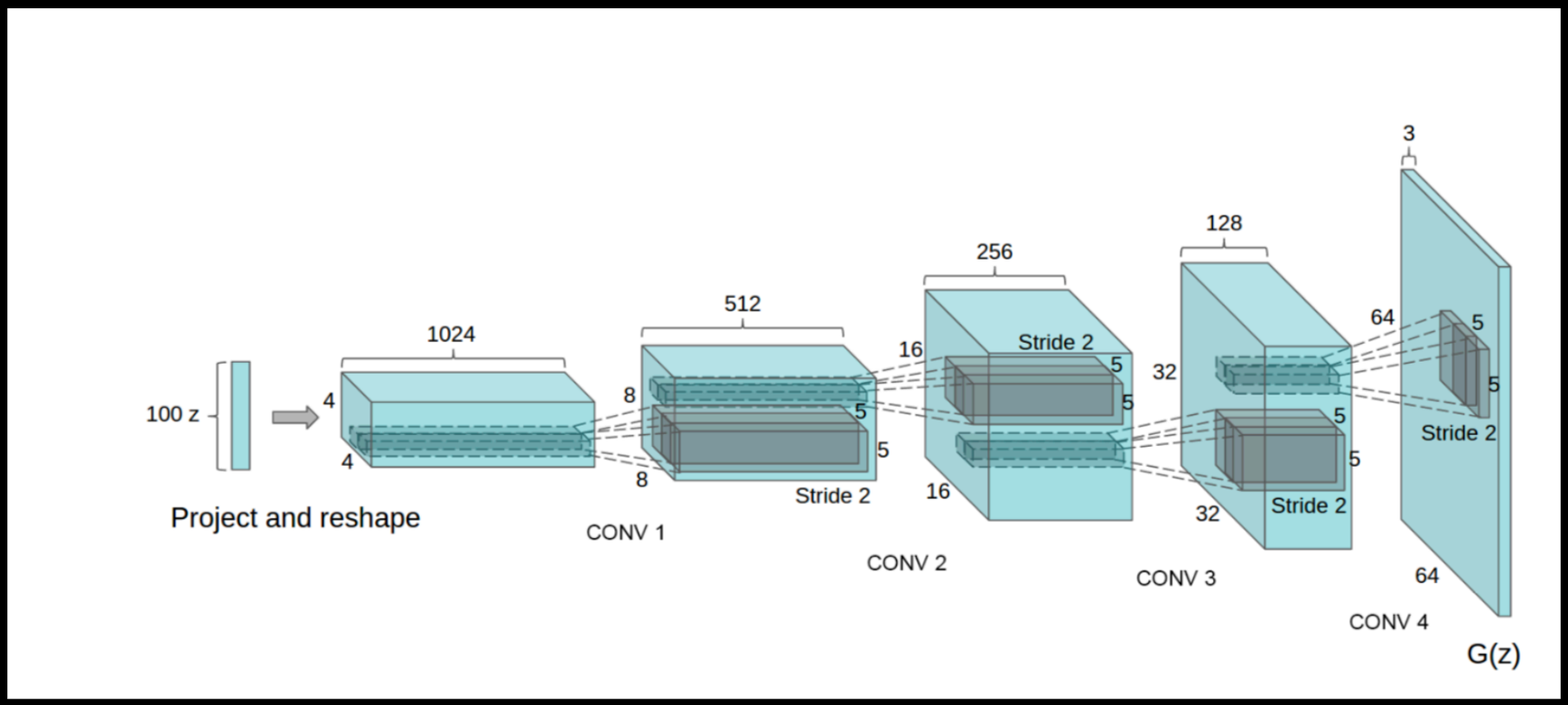
- A large body of GAN research involves improving the training stability.
- This has been achieved by directly modifying the existing weights in a given architecture (weight clipping, spectral normalization), or, indirectly modifying the weights using an added regularization term (orthogonal regularization).
- ProGAN achieved improved fidelity by using progressive training.
- However, BigGAN showed that similar results could be achieved, without the need for progressive training.

Notes

- Improved results have been consistently demonstrated, by using larger networks.
- However, training instability makes training larger networks challenging.
- Early stopping, etc. is an indication of the brittleness of the current “state of the art”.
- Given this brittleness, it is clear that there is currently a substantial lag in theory.

DCGAN

Deep CNN Generator



The goal of this paper was to perform unsupervised learning using GANs.

The authors employed various architectural modifications to successfully train the generator and the discriminator: striding, batchnorm, no fully connected layers, Relu's.

The authors demonstrated that their generator had learned object representations (LSUN - bedroom windows)

The authors demonstrated successful classification results using the learned discriminator.

The authors demonstrated the vector arithmetic property of GAN generators.

Wasserstein GAN

- The authors were looking for a way to improve the stability of GAN training.

- They examined different ways to measure the distance between a true data distribution and a model distribution: Total Variation (TV), Kullback Leibler (KL), Jensen Shannon (JS), Earth Mover (EM)

- To differentiate between the approaches, the authors examined the sequence convergence properties associated with the different measures.

- It was observed that EM (Wasserstein -1) had the weakest sequence convergence properties.

- Since this was deemed highly beneficial, the EM distance measure was incorporated into the GAN cost function.

Wasserstein GAN

The original minimax problem then changed from:

$$\min_G \max_D \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}[\log(D(\boldsymbol{x}))] + \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}[\log(1 - D(\tilde{\boldsymbol{x}}))]$$

To:

$$\min_G \max_{D\in\mathcal{D}} \mathop{\mathbb{E}}_{\boldsymbol{x}\sim\mathbb{P}_r}\big[D(\boldsymbol{x})\big] - \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}\sim\mathbb{P}_g}\big[D(\tilde{\boldsymbol{x}}))\big]$$

The EM distance measured the minimum cost of transporting mass from x to y, in order to change a distribution from P_r(x) to P_g(y).

In the above formulation, script D restricted the possible space of allowable discriminators (now called critics) to 1-Lipschitz functions.

In practice, this restriction was (initially) implemented using a simple weight clipping function [w <- clip(w, -c, c)], thereby restricting the output growth to be linear.

Using this modification, training stability improved and mode collapse became less frequent.

Wasserstein GAN GP

However, using weight clipping, the critics for deep Wasserstein GANs often failed to converge.

The authors concluded that this was because “clipping caused the higher order moments of the data distribution to be ignored”.

To remedy this, the authors proposed to implement the 1-Lipschitz constraint by adding a new loss function / penalty term instead:

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}}.$$

The details of the gradient penalty term involved creating a new image consisting of a blend between a generated image and a real image, and computing a gradient penalty using this new image.

for $i = 1, ..., m$ **do**
  Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$
  $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
  $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
  $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
end for

.

Spectral Normalization

- Another technique for improving the stability of GAN training was spectral normalization (SN).

  Review: The norm of a matrix A determines the "size" of the matrix

- 1 norm = $\|A\|_1 = \max_j \left(\sum_i |a_{ij}| \right)$.
- Infinity norm of a matrix = $\|A\|_\infty = \max_i \left(\sum_j |a_{ij}| \right)$
- 2 norm of a matrix (SN) = $||A||_2 = \sqrt{\max\left(eig\left(A^T \cdot A\right)\right)}$

Spectral Normalization

- Initially, spectral normalization was applied to the discriminator of a GAN (SNGAN).

- Here, the weight matrix W in every layer of the discriminator was normalized by the spectral norm associated with W.

- This enforced Lipschitz continuity on the discriminator. (Later, SN was also applied to the generator.)

Spectral Normalization

- Earlier, weight normalization was applied to every row in the weight matrix W (associated with every layer of the discriminator).

- Here, normalization was achieved using the 2 norm (for vectors).

- This was known as WNGAN.

- In general, spectral normalization permitted larger learning rates to be used, and was shown to yield better results than WGAN, WGAN GP and WNGAN.

Self Attention GAN

- Please refer to a future presentation, dedicated solely to the topic of Self Attention.

BigGAN

- This paper produced the best generated images to date (image fidelity and image resolution @512x512) without the use of progressive training.

- Here, the authors applied orthogonal regularization to generator training.

- Orthogonal regularization “encouraged” the weight matrices W to be diagonal: $$R_\beta(W) = \beta \|W^\top W - I\|_{\mathrm{F}}^2$$

- In practice, the above relation was slightly modified, in order to achieve the desired results: $$R_\beta(W) = \beta \|W^\top W \odot (\mathbf{1} - I)\|_{\mathrm{F}}^2$$

1 denotes the matrix of all 1’s

BigGAN

- With orthogonal regularization, the authors were then able to employ the truncation trick.

- The truncation trick allowed the generation of higher fidelity images, by restricting the range of z-vectors sampled from p(z).

Other observations made under BigGAN included:

- By scaling to larger batch sizes and larger models, image fidelity improved.

- By scaling the training data from ImageNet to JFT-300M, improved training stability was observed.

BigGAN

- However, as the network size increased, mode collapse and training stability worsened. (i.e. Previous stable settings now produced unstable results.)

- In general, many experiments (across the board) were performed in BigGAN to produce the new results.

- These experiments involved investigating various generator and discriminator architectures (skip connections for the latent vector / hierarchical latent spaces, different output resolutions and depths of networks, etc.) as well as the return of “babysitting the training”.

- This indicated that a high degree of brittleness was associated with the improved results.

Ideal Results

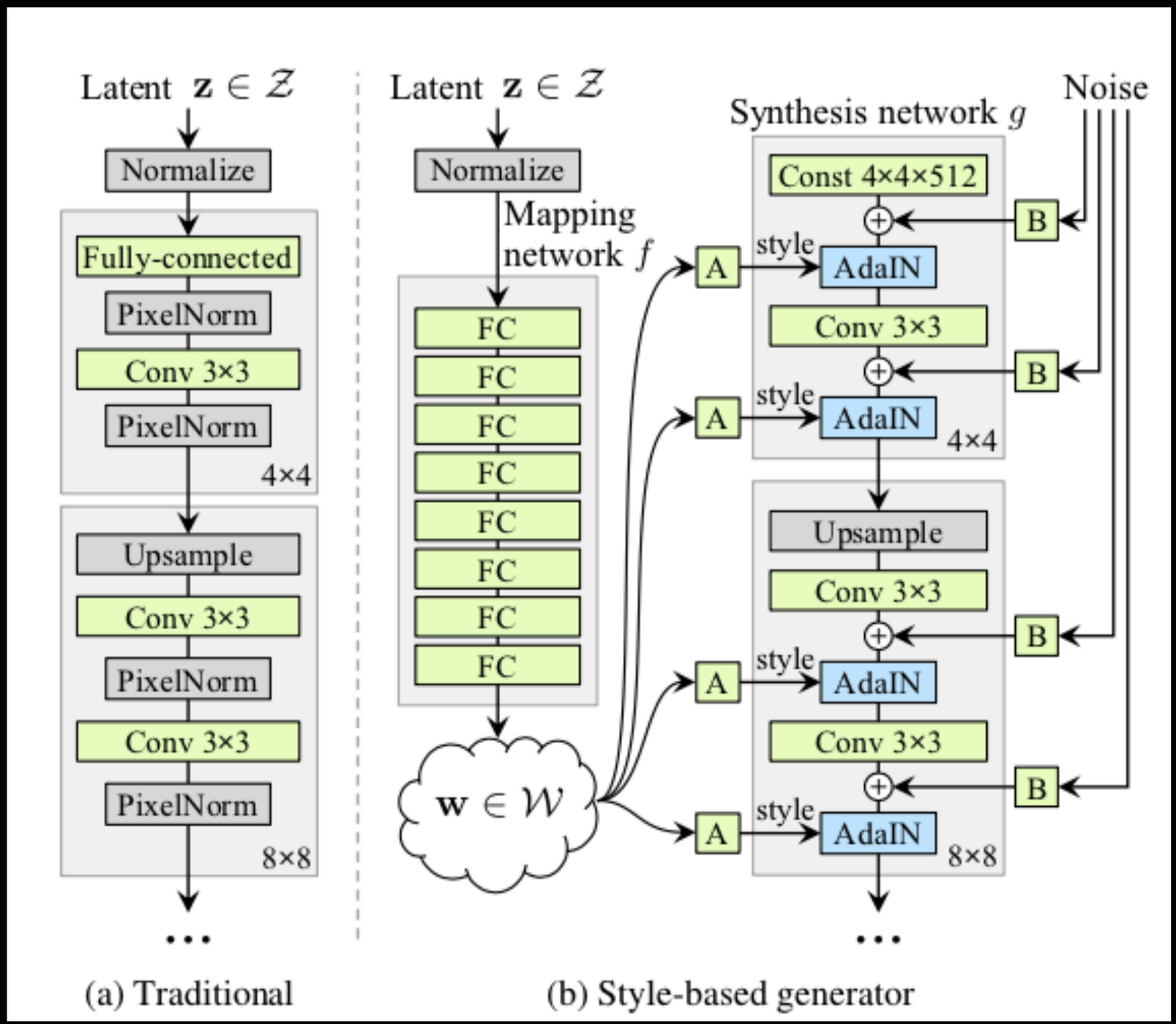
Imperfect Failure Cases

BigGAN (512 x 512) image results

StyleGAN

- In StyleGAN, a major architectural change was introduced into the generator.
- In the past, a z vector was consumed at the input of the generator.
- Now, the z vector underwent two transformations: MLP (for disentanglement*) and various affine transformations (for feature selection).
- The resulting output was then used to independently control every layer of the generator.
- *In the past, disentanglement was also achieved by maximizing mutual information. (See InfoGAN.)

StyleGAN



StyleGAN proposed a major architectural shift to the generator component.

In the traditional generator (similar to VAE's), the latent vector z was fed in at the top of the generating network.

Now, a constant vector was fed in at the top.

Instead, the latent vector z was fed into a MLP, generating a new vector, W.

Information was then extracted from W (via affine transform) and applied to the various feature maps.

StyleGAN

- What happened in the AdaIN block?

- AdaIN normalized every feature map in the generator by its spatial mean and spatial variance (i.e. Instance normalization)

- These feature maps were then independently scaled and shifted, thereby changing the style components of the image.

- The scale and shift parameters were computed by applying different affine transforms to the disentangled W vector.

- Finally, control of the finer grain stochastic details of an image were also possible, by adding a scaled (B) noise vector to every layer.

StyleGAN

Hence, the generator network learns how to effective disentangle styles, by back propagating through the generator and learning the appropriate MLP weights needed to generate a disentangled w vector.

By selecting different dis-entangled latent codes and applying them to different feature maps using an affine transform, the style of an image can then be changed in a controlled fashion.

StyleGAN

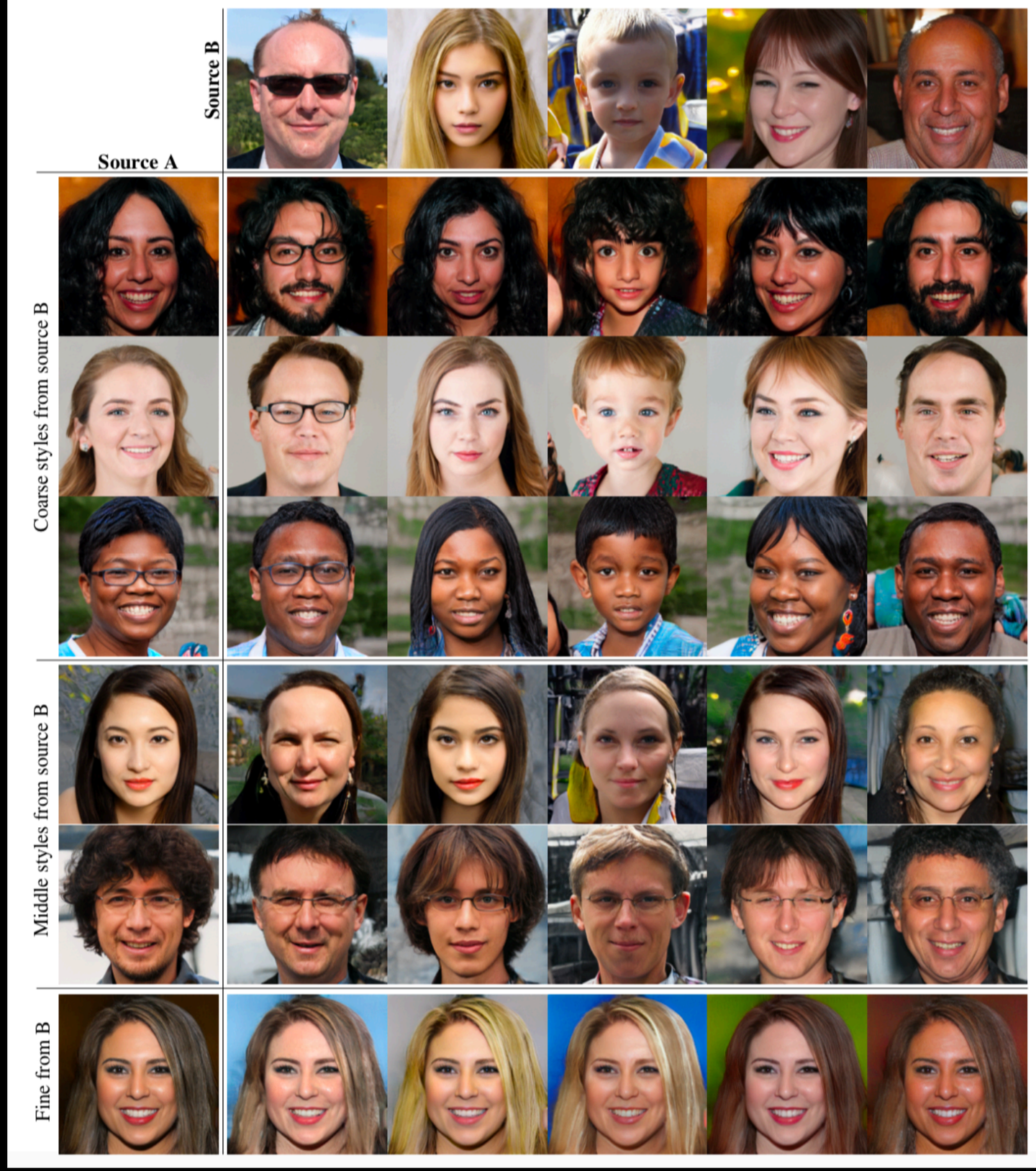
- Using the GAN framework, the discriminator sends a training signal to the generator.

- In order to generate a realistic image, the training signal alters the weights of the MLP, the weights associated with various affine transformations and the weights for a stochastic noise vector.

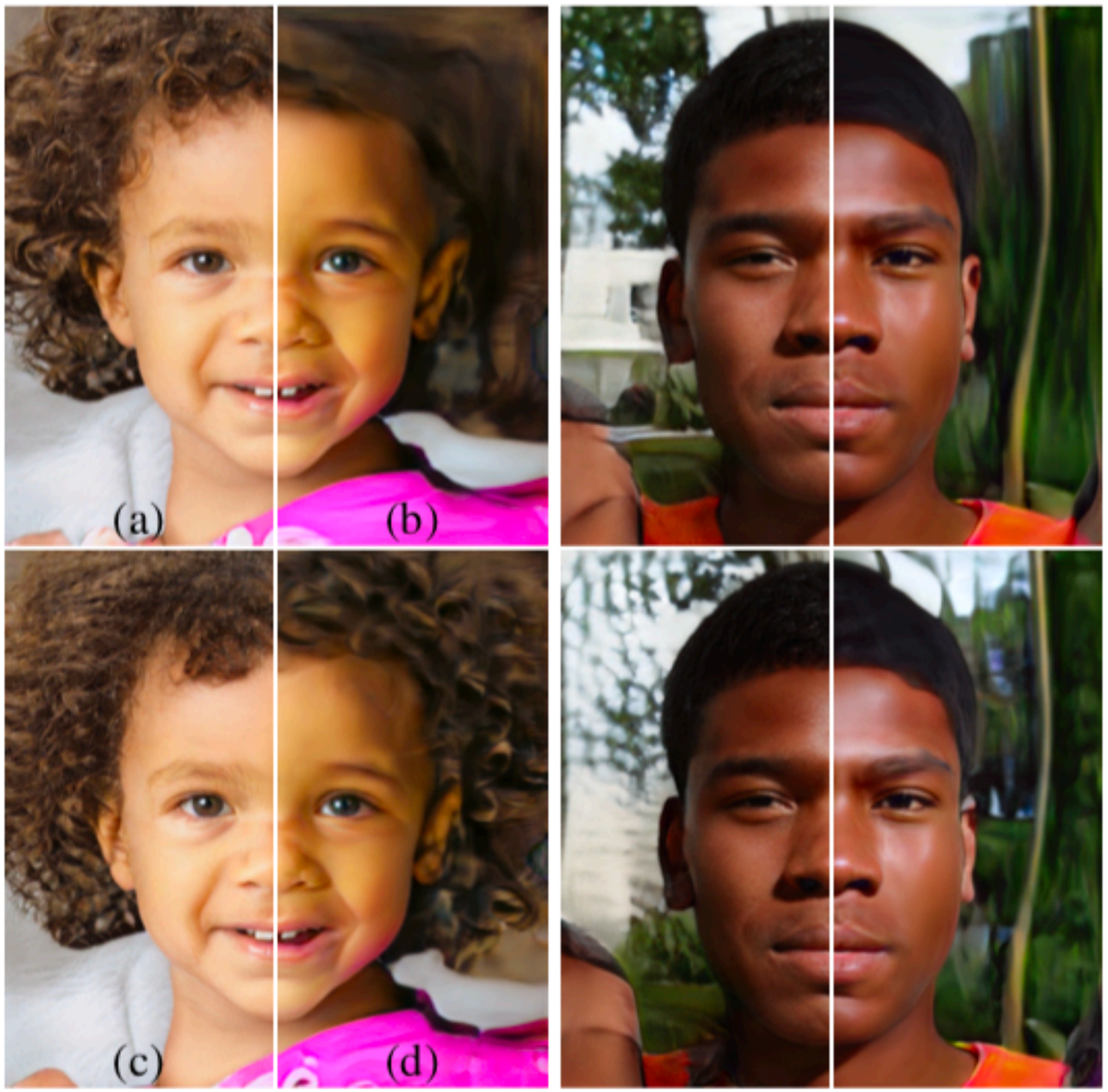
- In the process, different style features are applied to different generator layers, via computed shift and scale parameters.

- The computed shift and scale parameters result from applying various affine transforms, to a disentangled W vector.

- In the eyes of the affine transforms, the W vector contains useful disentangled information, since the MLP has learned how to create such a vector from training.

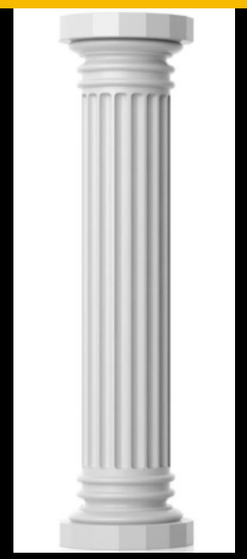


StyleGAN (1024 x 1024) image results
Style features were controlled at various levels in the generator using AdaIn.



Stochastic features were controlled at various levels in the generator by adding scaled versions of a fixed noise vector to the input of the layer.
a) Noise is applied to all layers. b) No noise is applied. c) Noise is applied to the later fine layers. d) Noise is applied to the early coarse layers.

Loss Function



Loss Function

DC GAN	Wasserstein GAN-GP	Spectral Normalization	Big GAN	Style GAN
Adversarial Training	Critic Based Adversarial Training + Lipschitz Constraint/ Gradient Penalty	Adversarial Training with Spectral Norm constraint on the weight matrices	Critic Based? Adversarial Training + Lipschitz Contraint/ Gradient Penalty + R Regularization	Critic Based Adversarial Training + Lipschitz Contraint/ Gradient Penalty + R Regularization

R Regularization = Orthogonal Regularization on the Weight Matrix

$$\|W^T * W - I\|$$

References

- Generative Adversarial Networks, Goodfellow, et. al., arXiv June 2014
- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, Radford, et. al., arXiv November 2015
- Improved Techniques for Training GANS, Salisman,et. al., arXiv June 2016
- Towards Principled Methods for Training GANS, Arjovsky and Bottou, arXiv January 2017
- Wasserstein GAN, Arjovsky, et. al., June 2017
- Improved Training of Wasserstein GAN, Gulrajani, et. al., arXiv March 2017
- InfoGAN, Chen, et. al., arXiv June 2016
- Spectral Normalization for Generative Adversarial Networks, Miyato, et. al., arXiv February 2018
- Self Attention GAN, Zhang, et. l., arXiv May 2018
- Large Scale GAN Training for High Fidelity Natural Image Synthesis, Brock, et. al., arXiv September 2018
- A Style Based Generator Architecture for Generative Adversarial Networks, Karras, et. al., arXiv December 2018