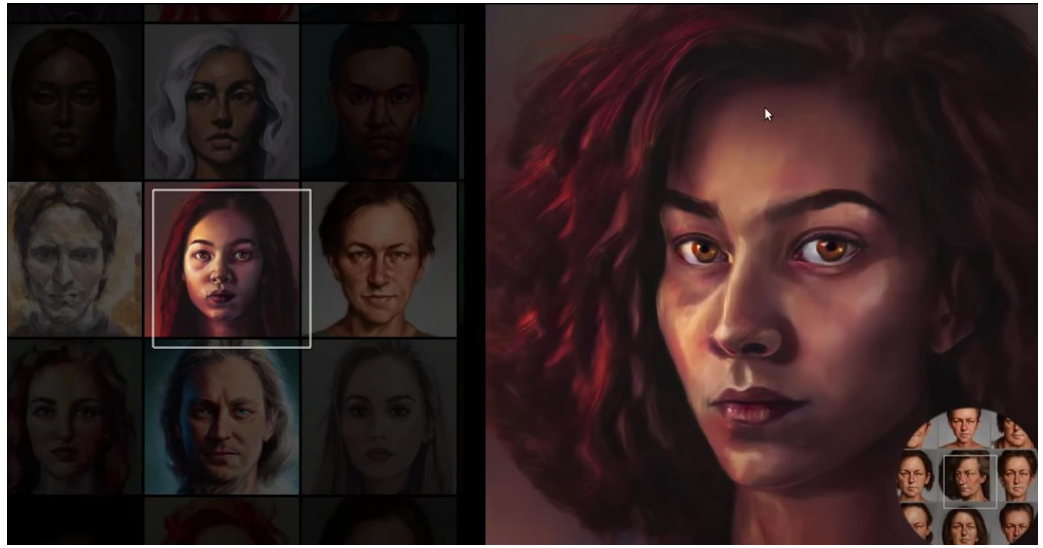
StyleGAN

Advanced Topics in Deep Learning Seminar



Elad Prager

About Me

- M.B.A. Graduate from The Hebrew University
- M.Sc. Student In Computer Science at Reichman University
- Machine Learning Engineer at General Motors
- Lecturer at The Hebrew University Business School

Agenda

- GAN Introduction
- GAN Architecture
- GAN Example
- Training GANs
- GAN Pseudo code
- GAN Results
- GAN Challenges
- StyleGAN Introduction
- Progressive GAN
- StyleGAN Architecture
- Style Mixing
- Style Scaling
- StyleGAN Results
- Conclusion

GAN Introduction

- **GANs Overview:** Generative Adversarial Networks (GANs) are a powerful class of generative models that learns to synthesize data by playing a two-player adversarial game.

Generative Adversarial Nets

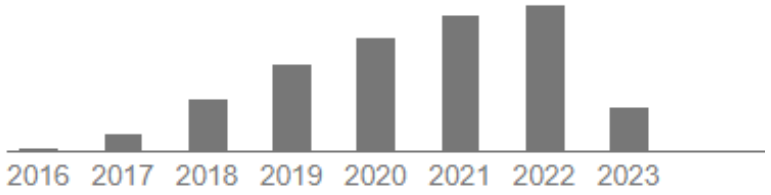
Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

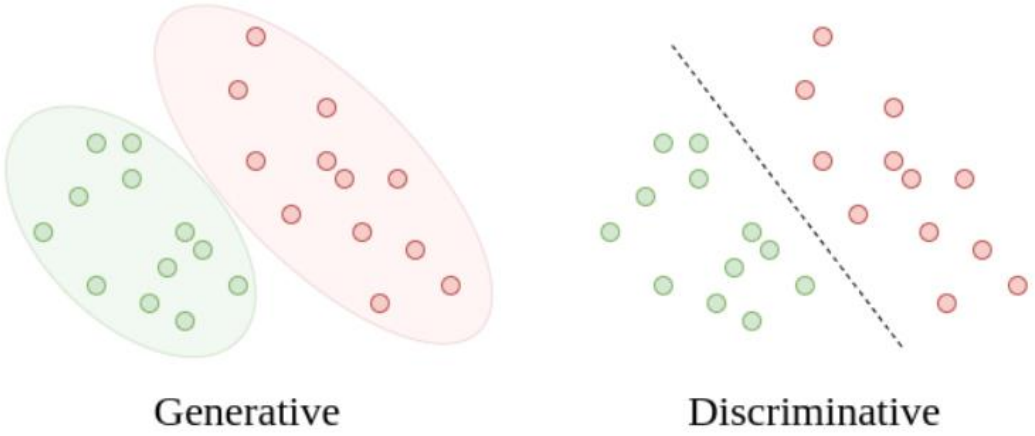
Generative adversarial networks

Authors	Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Ozair, Aaron Courville, Yoshua Bengio
Publication date	2014
Conference	Advances in neural information processing systems
Volume	27
Total citations	Cited by 58666

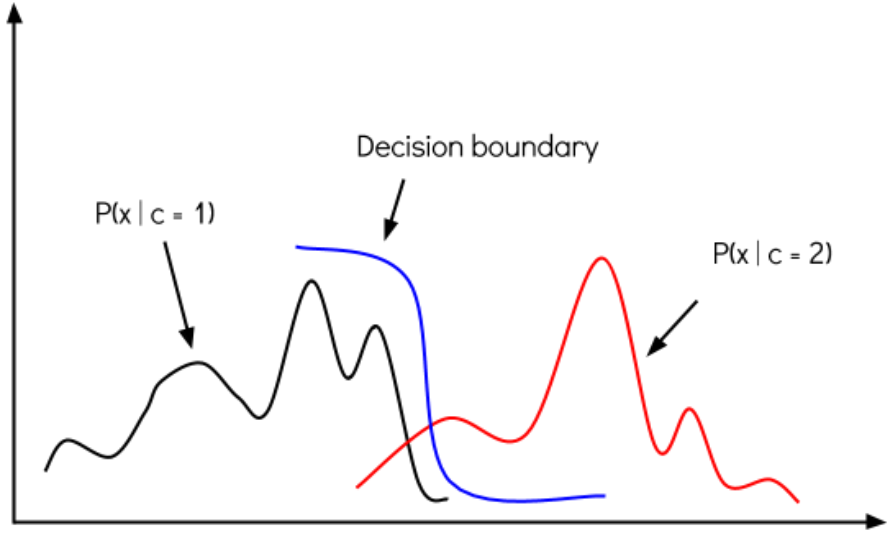


GAN Introduction

- **Discriminative models vs Generative models:** Discriminative models learn the decision boundary between classes estimating P(Y|X), while generative models model the data distribution P(X|Y); GANs belong to the latter category and excel at generating data samples.



1D Example



2D Example

Image Source: https://maximustann.github.io/mach/2015/08/11/generative-vs-discriminative-models/

GAN Introduction

- **Implicit Generative Models:** GANs are implicit generative models, learning to generate samples without explicitly modeling the data distribution (VAE, etc.). The distribution is defined implicitly by taking points from a unit gaussian distribution and mapping them through a deterministic neural network.

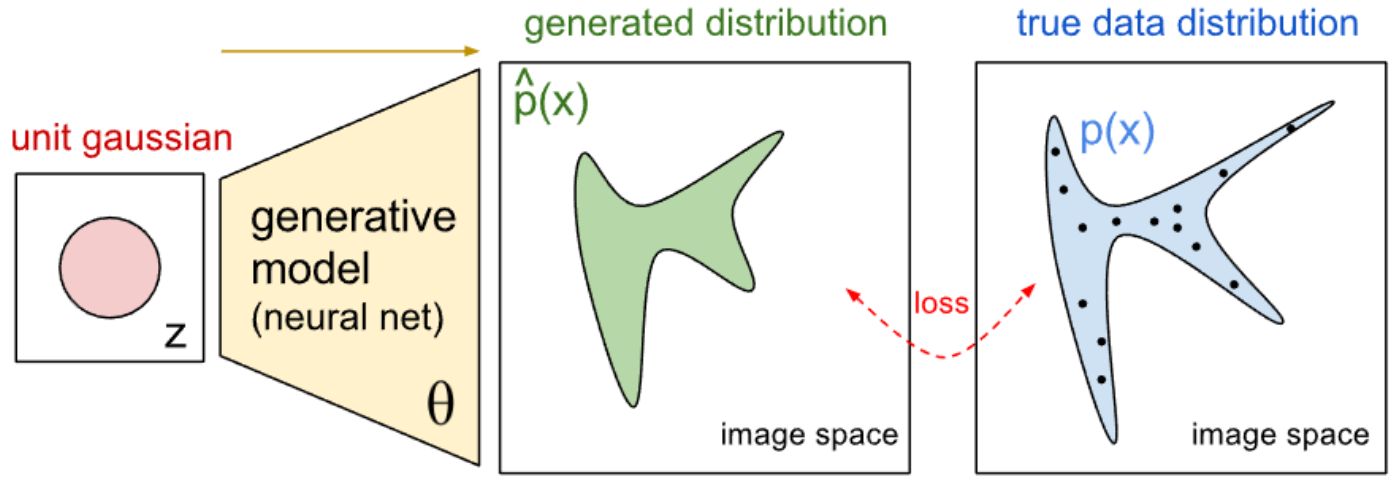


Image Source: https://openai.com/research/generative-models

GAN Architecture

- **Generator and Discriminator:** GANs consist of two main components: a generator network that aims to synthesize realistic samples, and a discriminator network that evaluates the quality of the generated data and attempts to differentiate between real and fake samples.

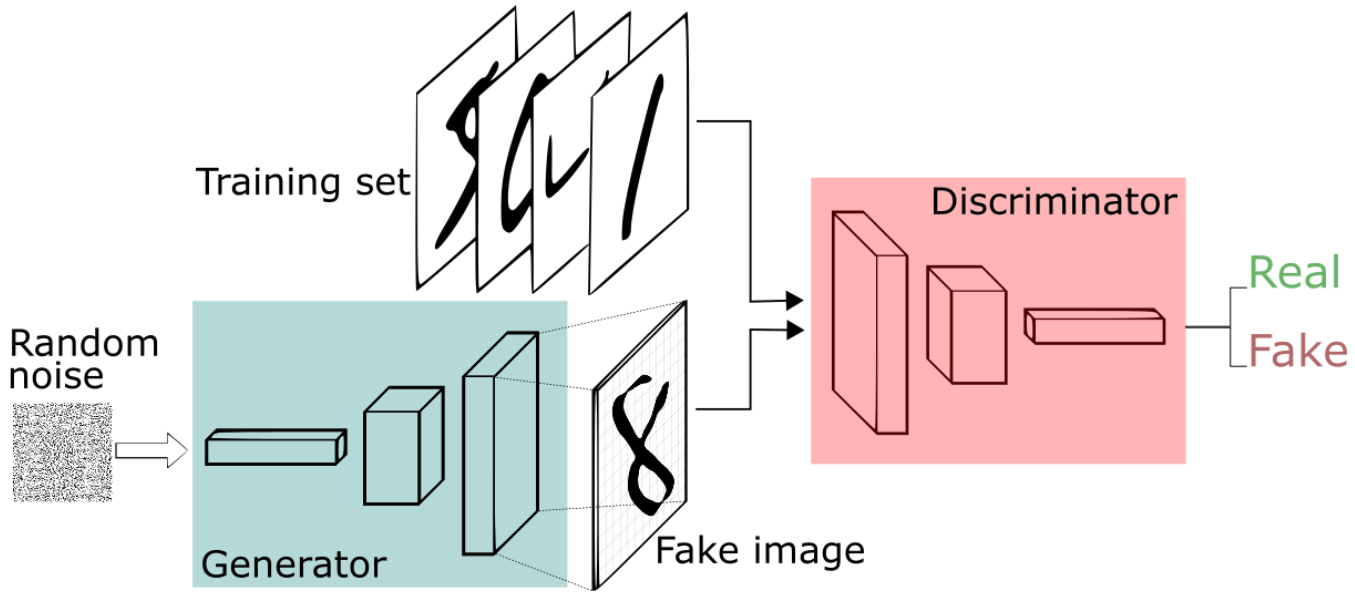
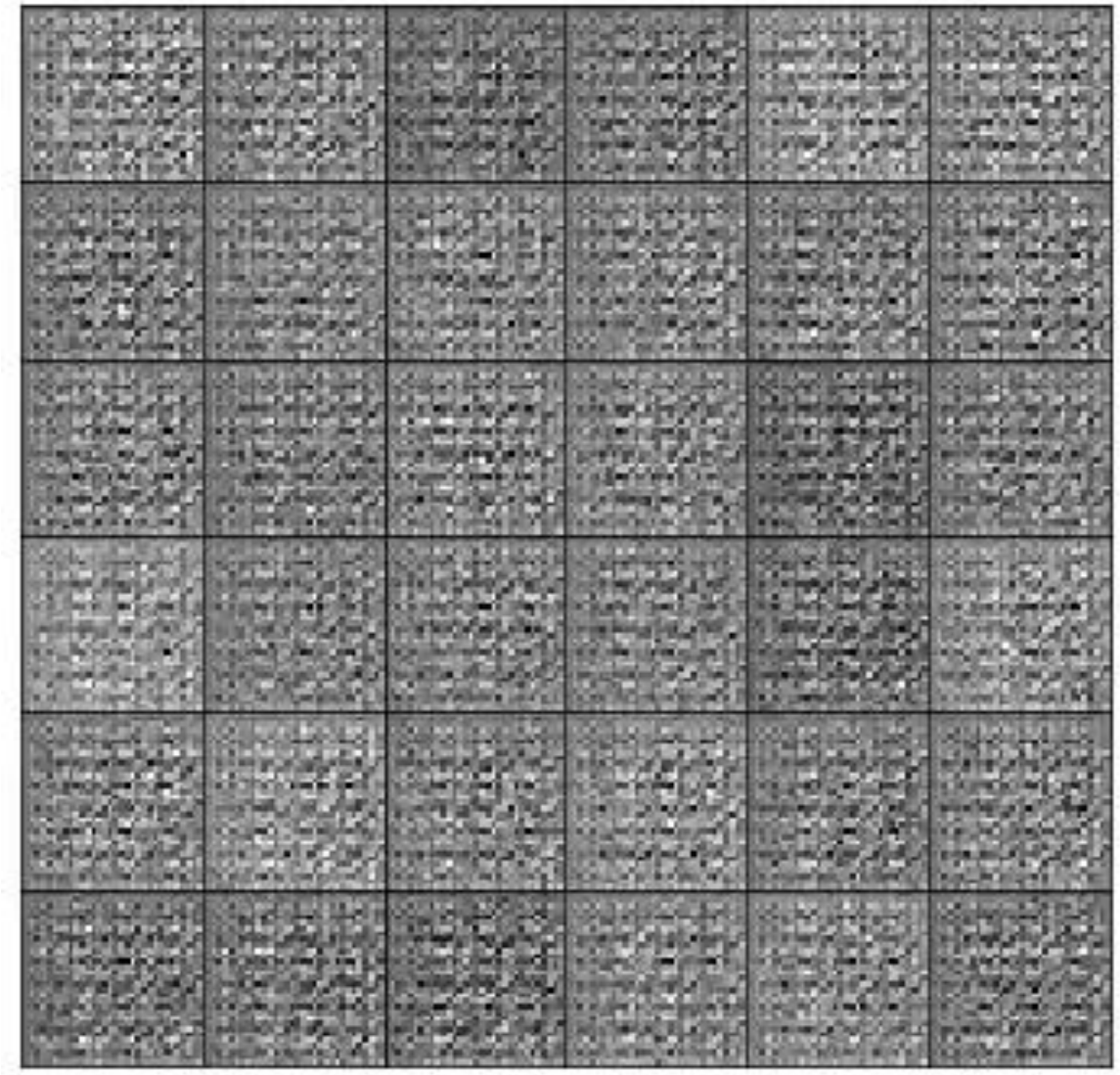


Image Source: https://sthalles.github.io/intro-to-gans/

GAN Examples



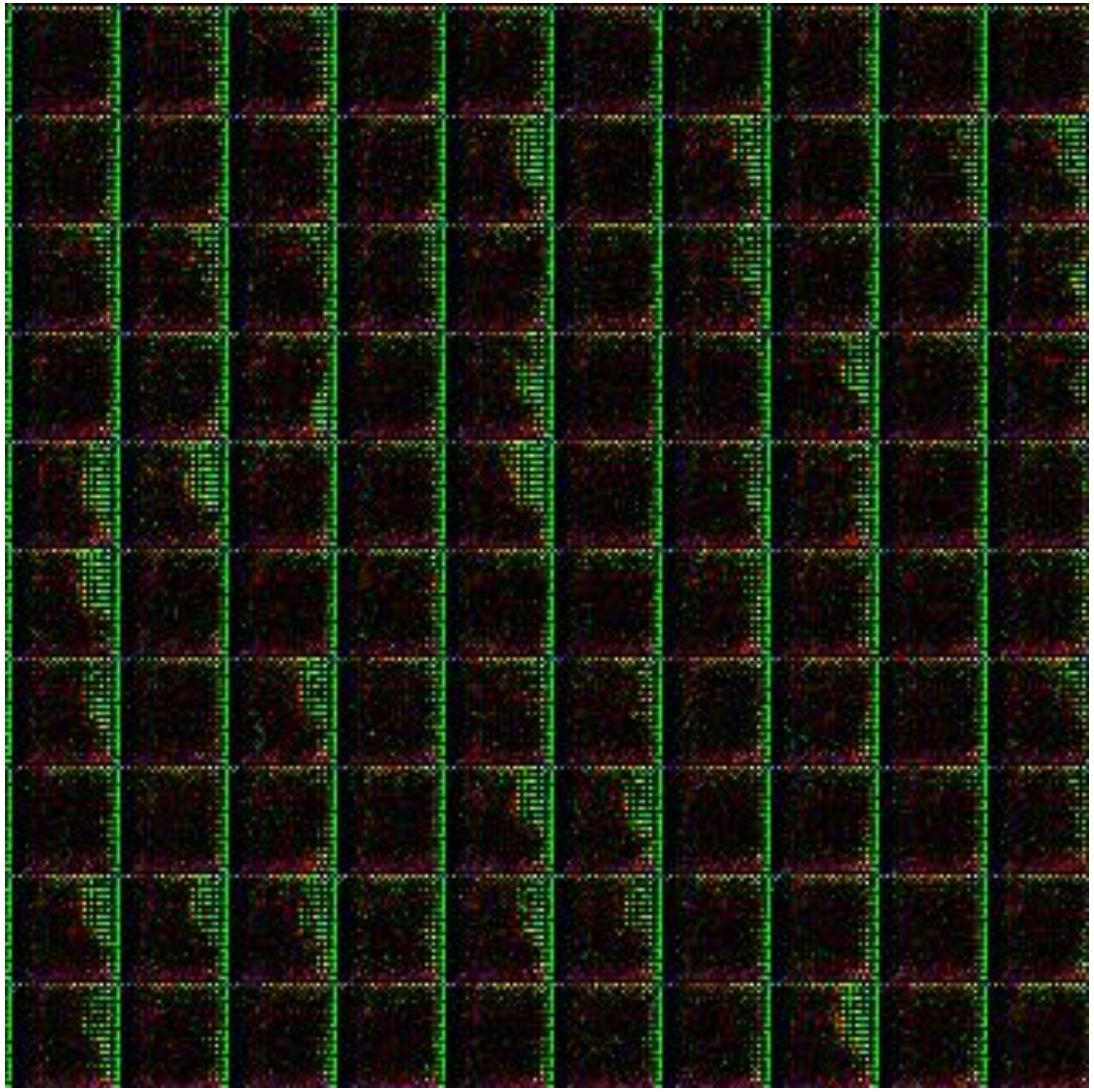


Image Source: https://openai.com/research/generative-models

Training GAN

- **Loss Function:** GANs loss function are formulated as a minimax game, where the generator aims to minimize the ability of the discriminator to correctly identify fake samples, while the discriminator aims to maximize its accuracy. leading to a competitive training dynamic.
- **Update Discriminator:** Push $D(x_{data})\ close\ to\ 1\ and\ D(G(z))$ close to 0
- **Update Generator:** Push $D(G(z))$ close to 1

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

Expectation

x is sampled from real data

Discriminator(real)

z is sampled from N(0,1)

Discriminator(fake)

Training GAN

- **Backpropagation:** GAN training involves an iterative process of updating generator and discriminator weights through backpropagation, ideally leading to convergence where generated samples match the real data distribution.

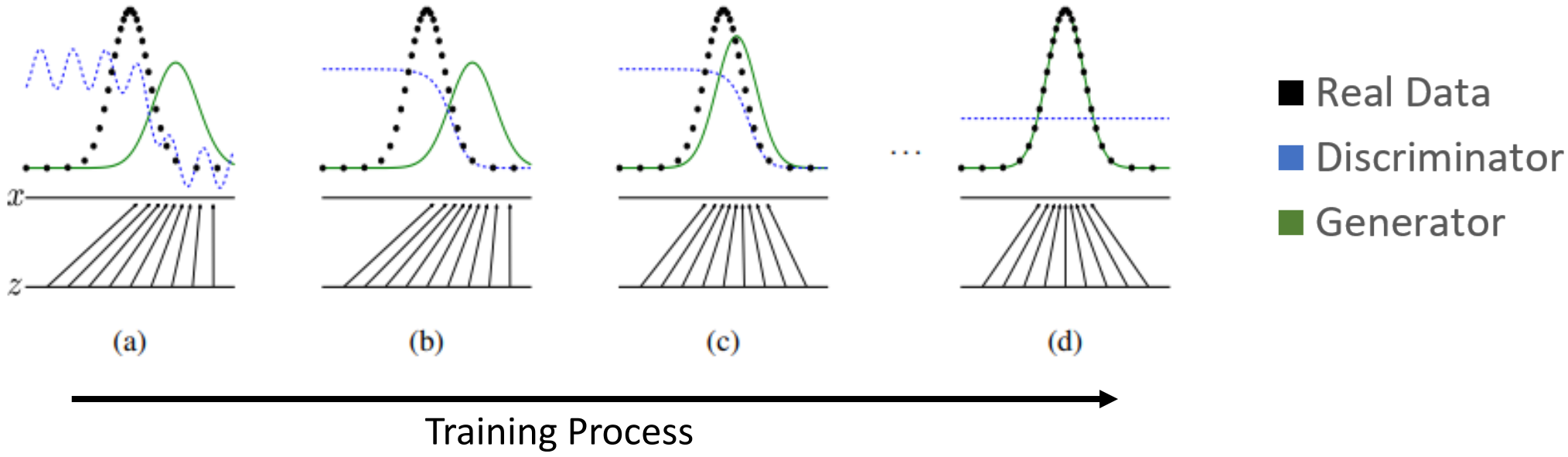


Image Source: https://arxiv.org/pdf/1406.2661.pdf

GAN Pseudo code

for number of training iterations **do**

for k steps **do**

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

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

end for

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

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

end for

GAN Results

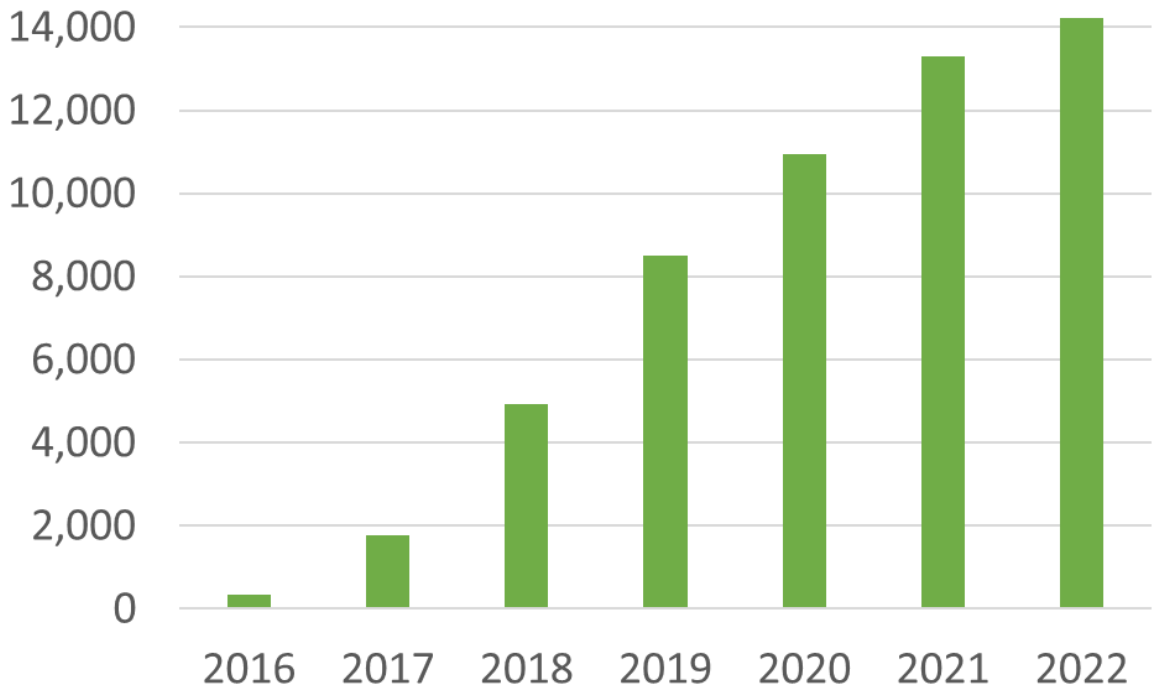
- **Original Results:** The original GAN paper demonstrated impressive results in generating realistic images, showing the potential of GANs as powerful generative models for various domains.

Nearest real image for sample to the left

Image Source: https://arxiv.org/pdf/1406.2661.pdf

GAN Results

- **GAN Buzz:** GANs have generated significant excitement in the research community due to their capabilities in generating high-quality samples and their potential applications across multiple domains.



GAN Paper Citations Per Year

Image Source: View article (google.ca)

GAN Results

- **Latent Space Arithmetic:** GANs learn meaningful latent spaces, enabling latent space arithmetic for image manipulation, such as combining or modifying attributes of generated samples.

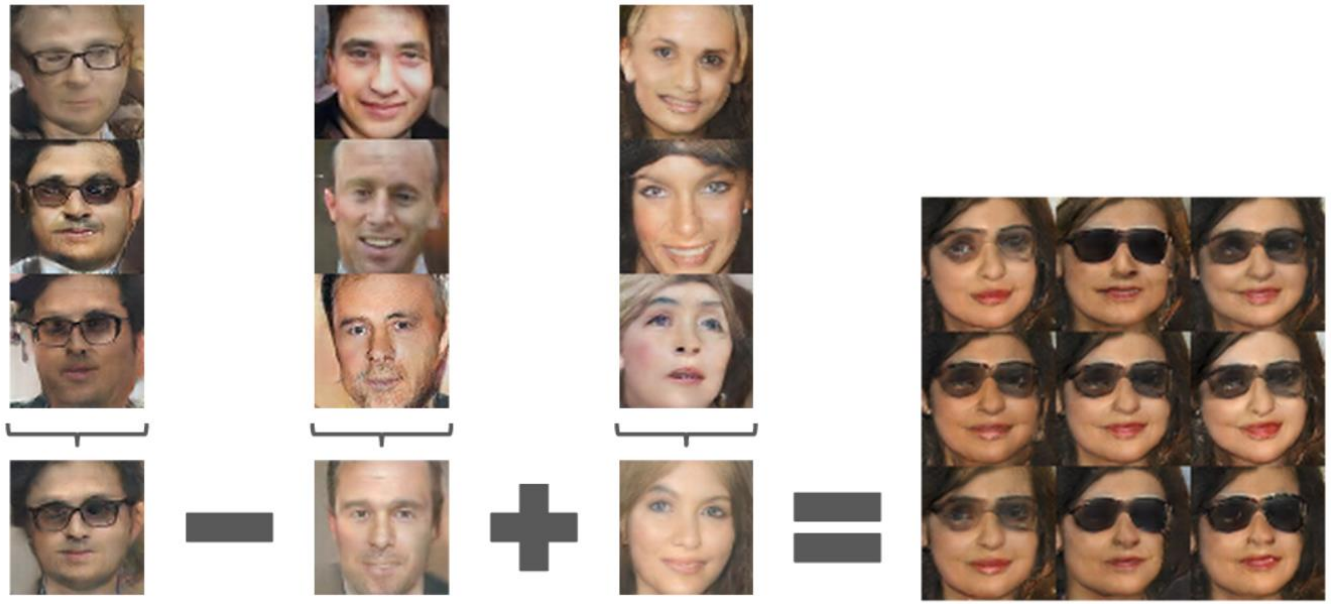
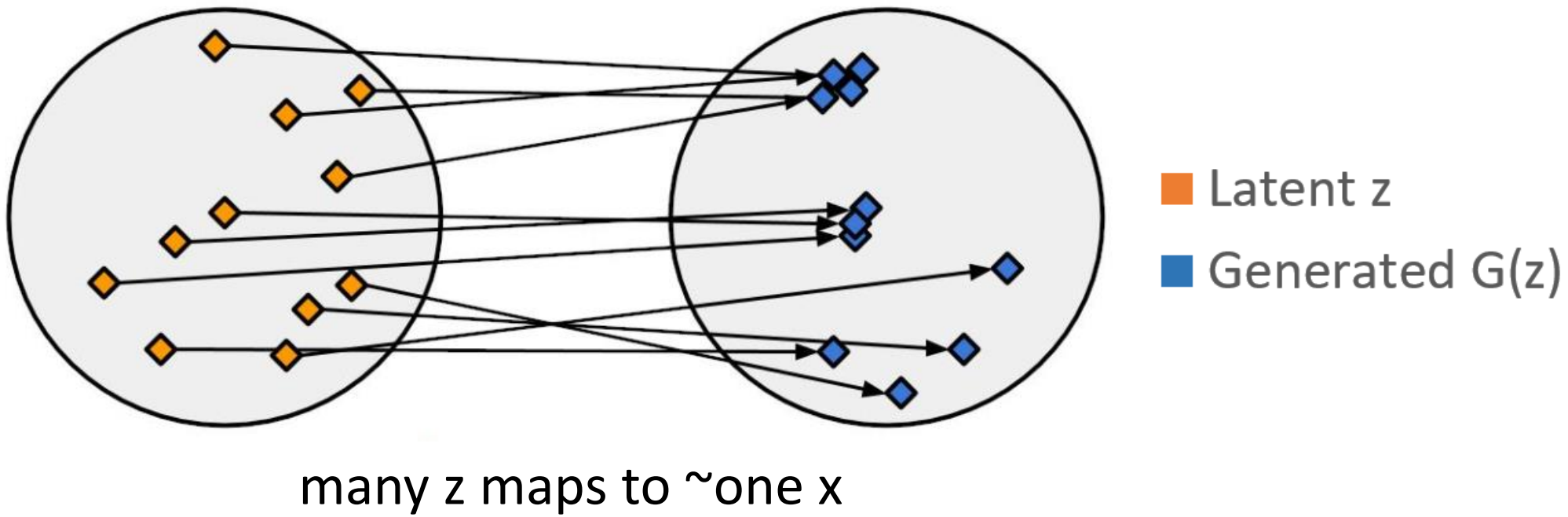


Image Source: https://arxiv.org/pdf/1511.06434.pdf

GAN Challenges

- **Mode Collapse:** GAN training may suffer from mode collapse, where the generator only produces a limited variety of samples.

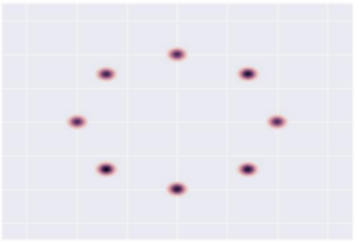


many z maps to ~one x

Image Source: https://twitter.com/ajmooch/status/1182018330070306816/photo/1

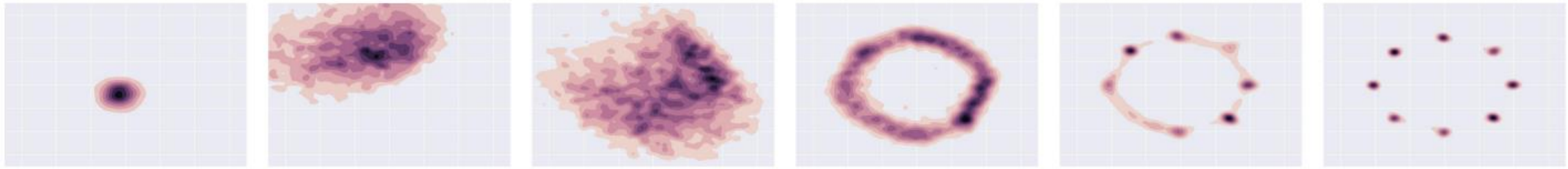
GAN Challenges

P_{data}

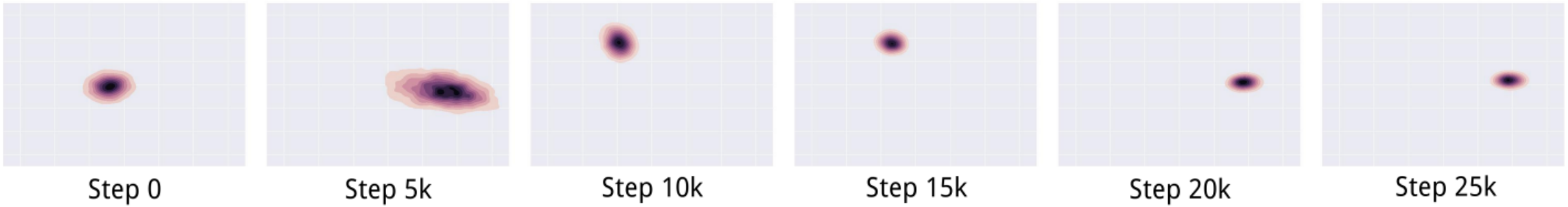


target

Without mode collapse:

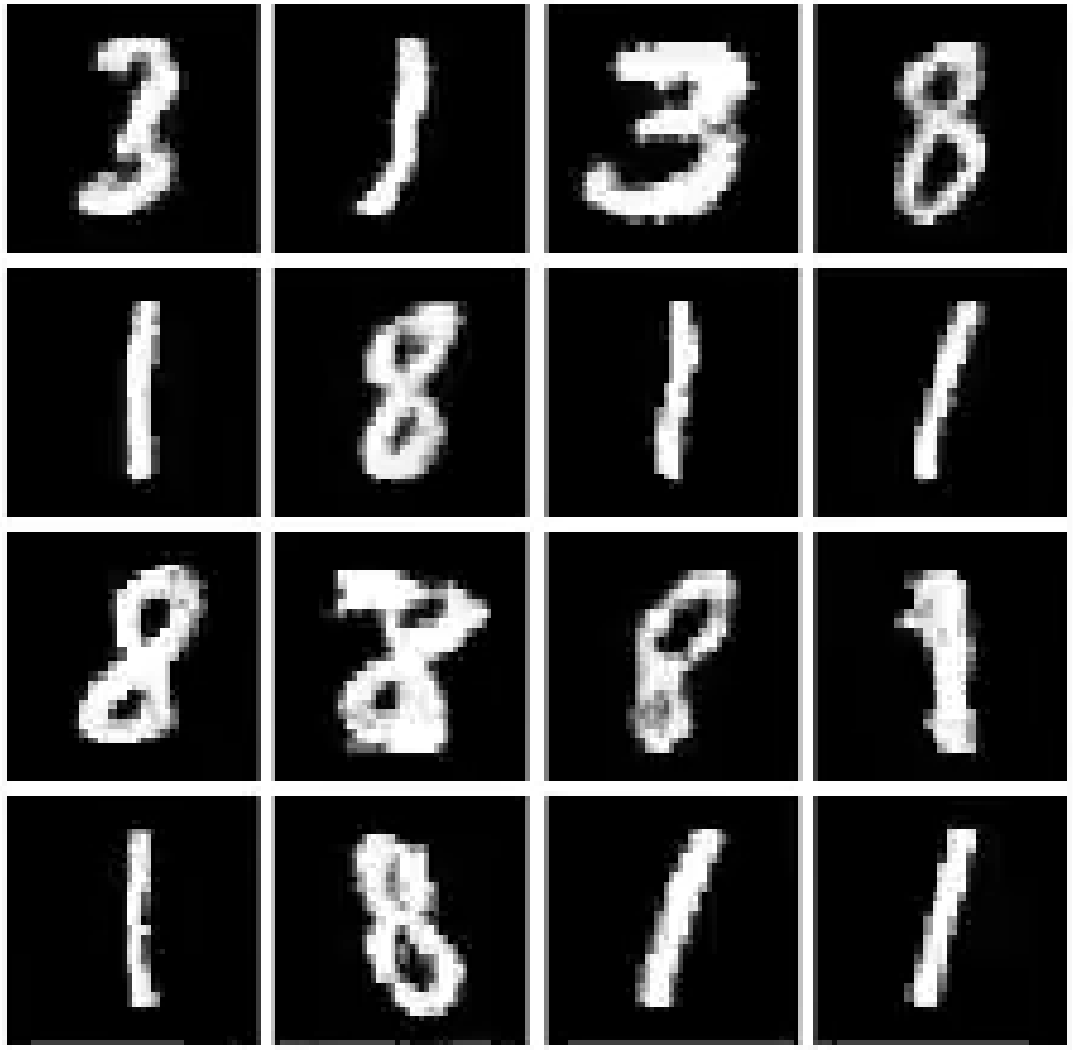


With mode collapse:



Video Source: https://arxiv.org/pdf/1611.02163.pdf

GAN Challenges



Mode Collapse on MNIST Training

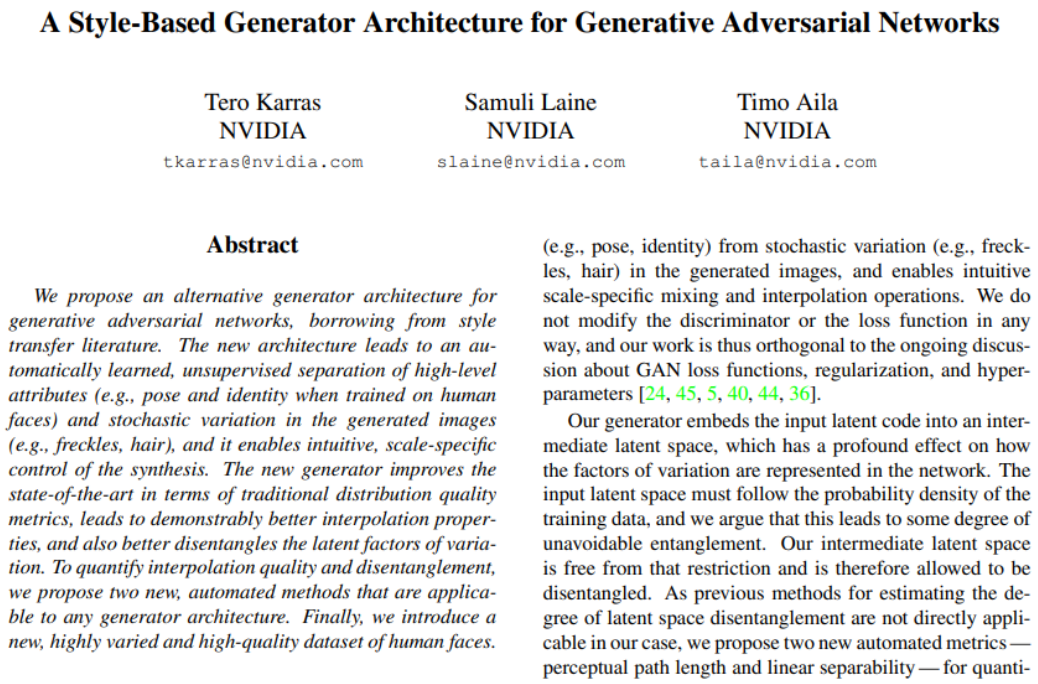
Video Source: https://www.youtube.com/watch?v=ktxhiKhWoEE

GAN Challenges

- **Training Stability:** Training GANs can be unstable and sensitive to hyperparameters, requiring careful tuning and experimentation.
- **Traditional GAN Challenges:** Traditional GAN architectures struggle with disentangled feature representation, limited control over style, and synthesis of high-resolution images; StyleGAN aims to overcome these challenges.

StyleGAN Introduction

- **StyleGAN Overview:** StyleGAN introduces a novel generator architecture focusing on style manipulation and feature disentanglement; it separates the generator into a mapping network and a synthesis network for better control and quality.



A Style-Based Generator Architecture for Generative Adversarial Networks

Tero Karras NVIDIA tkarras@nvidia.com

Samuli Laine NVIDIA slaine@nvidia.com

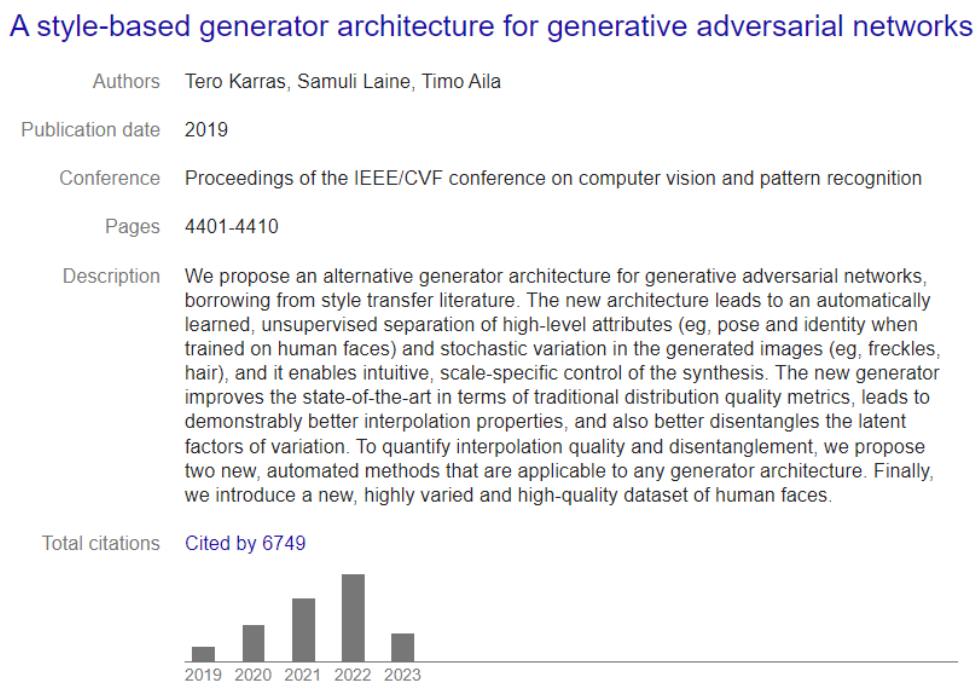
Timo Aila NVIDIA taila@nvidia.com

Abstract

We propose an alternative generator architecture for generative adversarial networks, borrowing from style transfer literature. The new architecture leads to an automatically learned, unsupervised separation of high-level attributes (e.g., pose and identity when trained on human faces) and stochastic variation in the generated images (e.g., freckles, hair), and it enables intuitive, scale-specific control of the synthesis. The new generator improves the state-of-the-art in terms of traditional distribution quality metrics, leads to demonstrably better interpolation properties, and also better disentangles the latent factors of variation. To quantify interpolation quality and disentanglement, we propose two new, automated methods that are applicable to any generator architecture. Finally, we introduce a new, highly varied and high-quality dataset of human faces.

(e.g., pose, identity) from stochastic variation (e.g., freckles, hair) in the generated images, and enables intuitive scale-specific mixing and interpolation operations. We do not modify the discriminator or the loss function in any way, and our work is thus orthogonal to the ongoing discussion about GAN loss functions, regularization, and hyperparameters [24, 45, 5, 40, 44, 36].

Our generator embeds the input latent code into an intermediate latent space, which has a profound effect on how the factors of variation are represented in the network. The input latent space must follow the probability density of the training data, and we argue that this leads to some degree of unavoidable entanglement. Our intermediate latent space is free from that restriction and is therefore allowed to be disentangled. As previous methods for estimating the degree of latent space disentanglement are not directly applicable in our case, we propose two new automated metrics — perceptual path length and linear separability — for quanti-

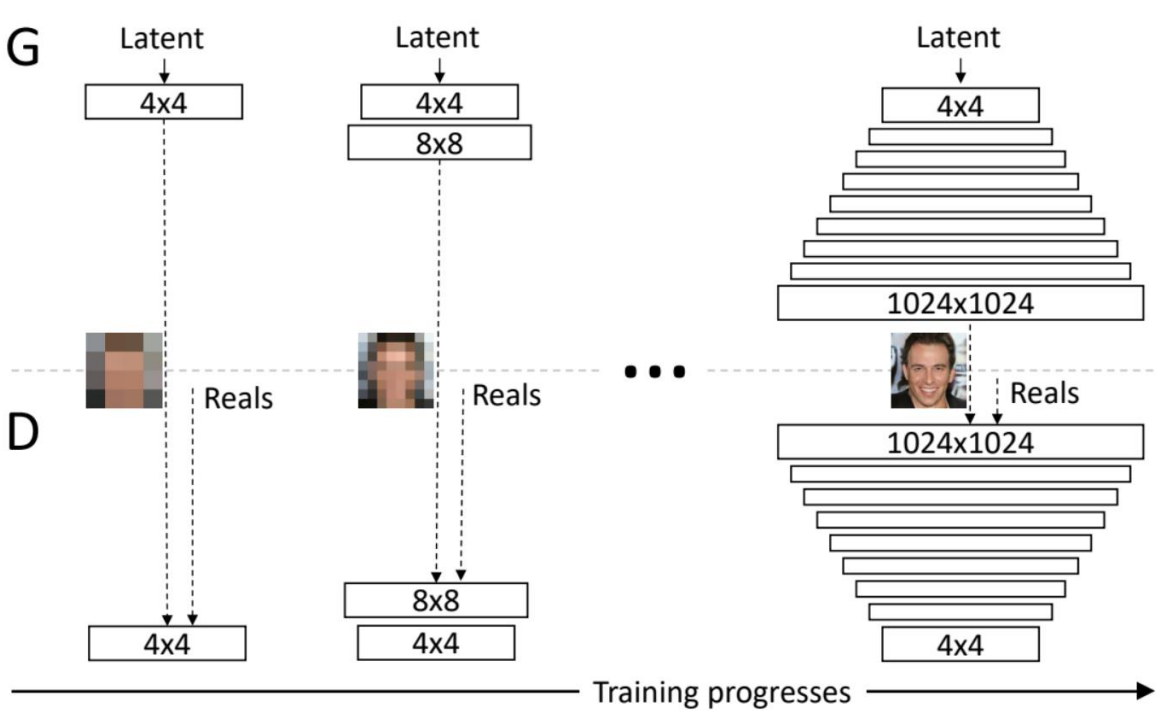


A style-based generator architecture for generative adversarial networks

Authors	Tero Karras, Samuli Laine, Timo Aila
Publication date	2019
Conference	Proceedings of the IEEE/CVF conference on computer vision and pattern recognition
Pages	4401-4410
Description	We propose an alternative generator architecture for generative adversarial networks, borrowing from style transfer literature. The new architecture leads to an automatically learned, unsupervised separation of high-level attributes (eg, pose and identity when trained on human faces) and stochastic variation in the generated images (eg, freckles, hair), and it enables intuitive, scale-specific control of the synthesis. The new generator improves the state-of-the-art in terms of traditional distribution quality metrics, leads to demonstrably better interpolation properties, and also better disentangles the latent factors of variation. To quantify interpolation quality and disentanglement, we propose two new, automated methods that are applicable to any generator architecture. Finally, we introduce a new, highly varied and high-quality dataset of human faces.
Total citations	Cited by 6749

Progressive Growing GAN

- **Progressive Growing GAN:** StyleGAN builds upon progressive growing techniques, which involve incrementally increasing the resolution of generated images for improved efficiency, training stability and image quality.



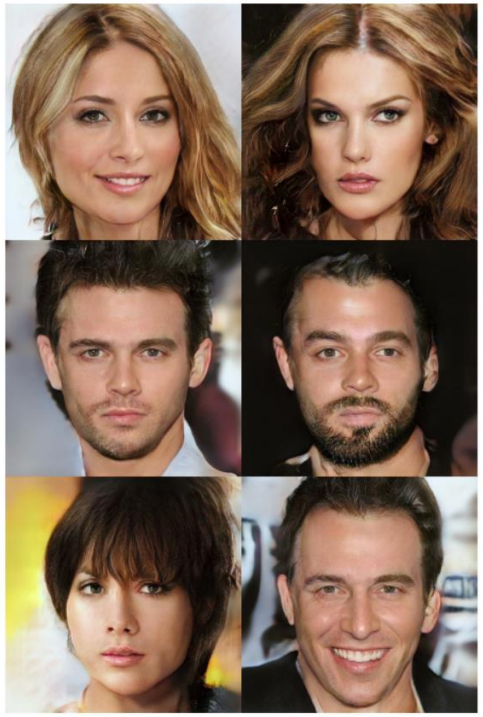
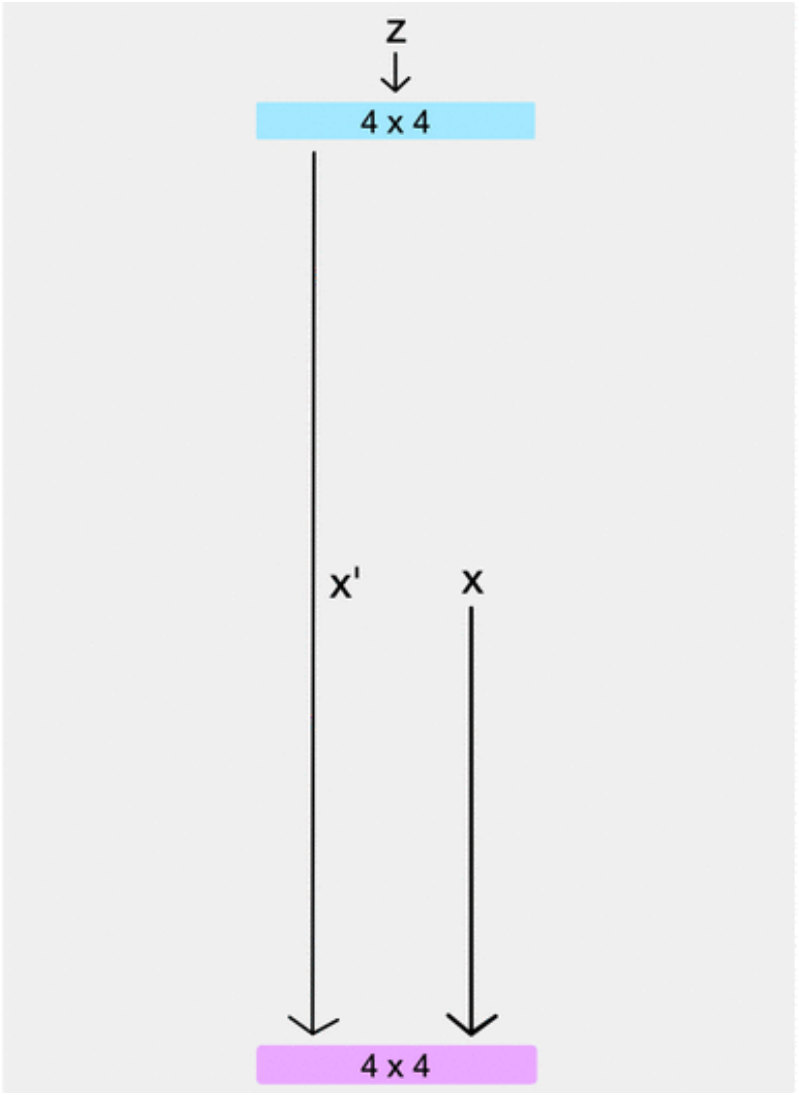
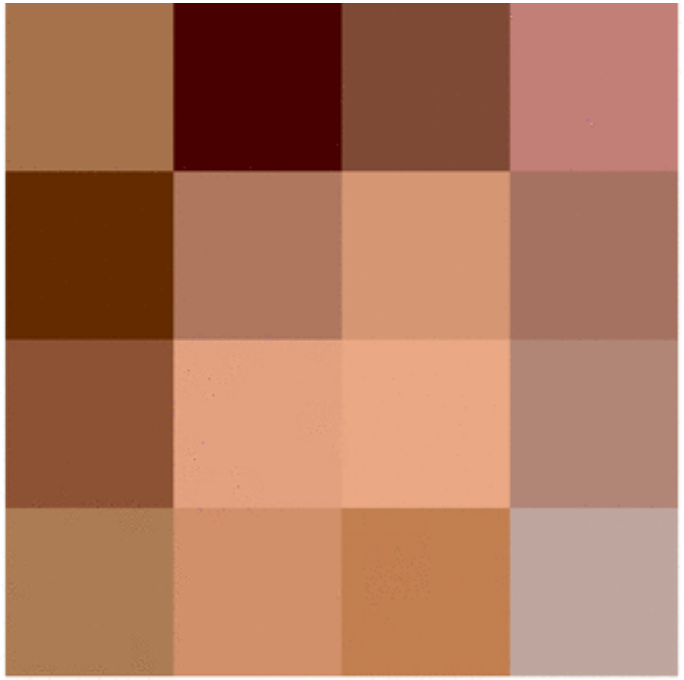


Image Source: https://arxiv.org/pdf/1710.10196.pdf

Progressive Growing GAN





Training time: 0 days
4x4 resolution

Generator
Discriminator

z = random code
x = real image
x' = generated image

ProGAN in Action

Image Source: https://towardsdatascience.com/progan-how-nvidia-generated-images-of-unprecedented-quality-51c98ec2cbd2

Progressive Growing GAN

- **Efficiency:** For a given amount of training time, ProGAN (green) was able to train on many more images than a traditional GAN (blue).
- **Early Training:** The difference is most extreme in early training, since this is when the progressively grown network is smallest.

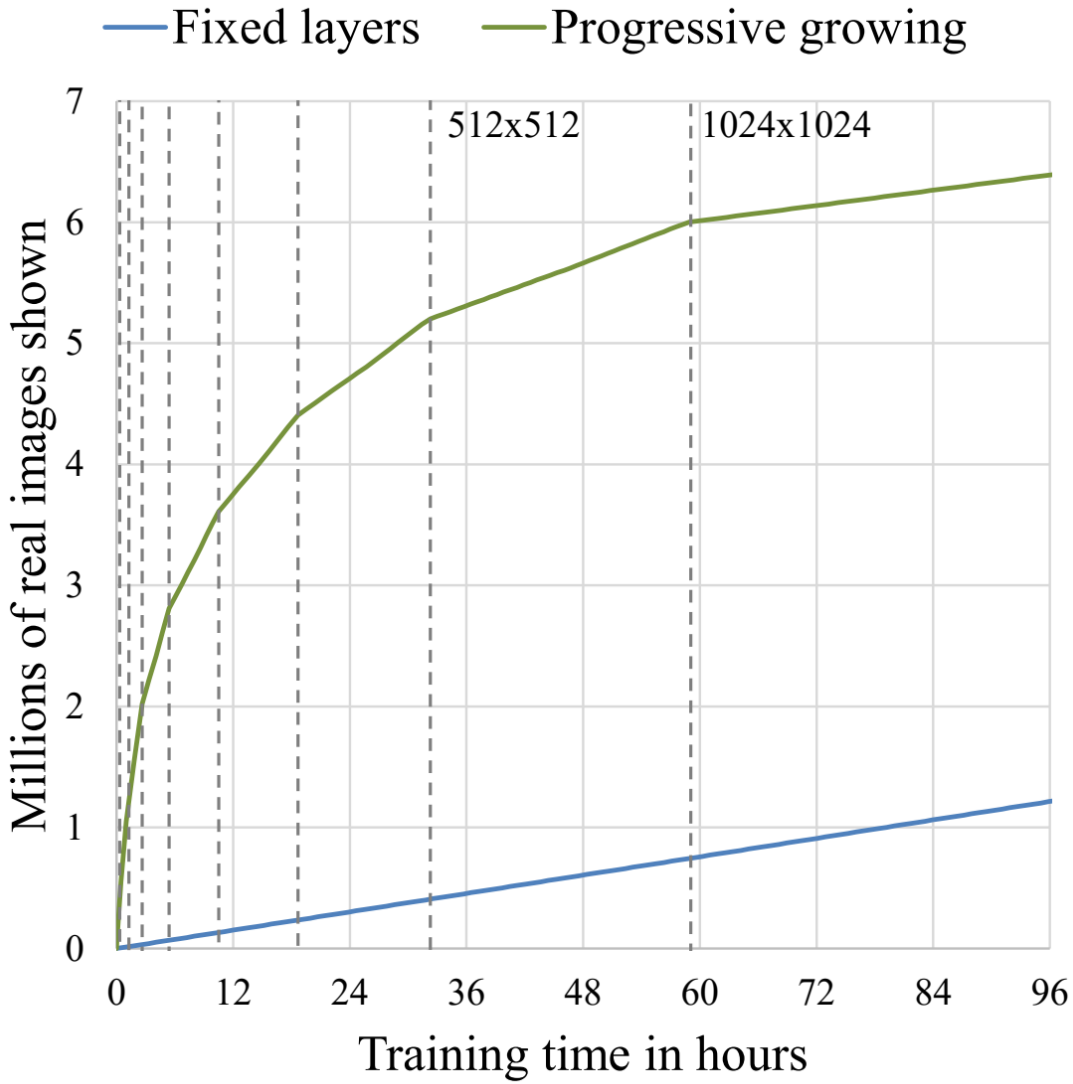
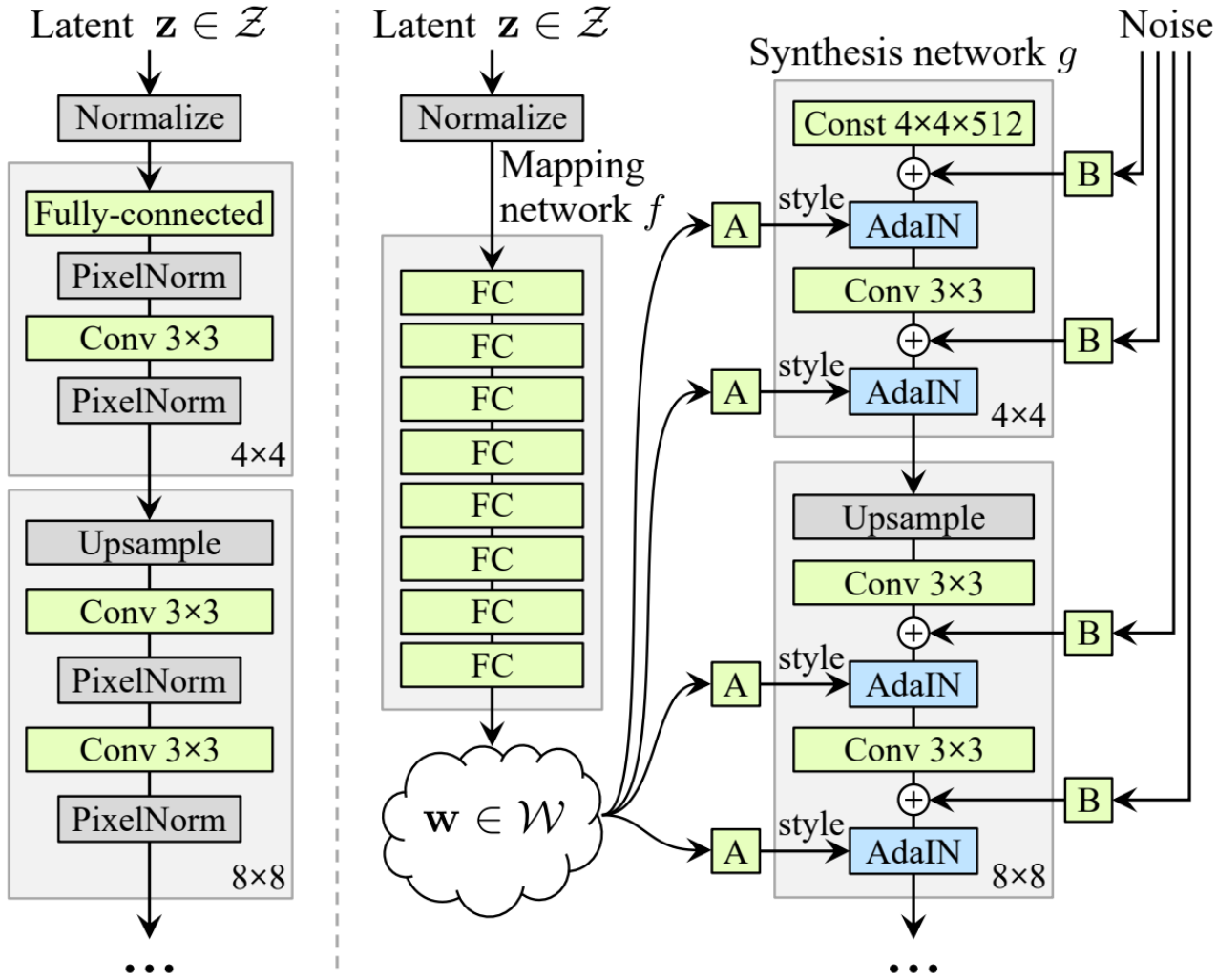


Image Source: https://arxiv.org/pdf/1710.10196.pdf

Style GAN vs. Traditional GAN

- **Framework:** StyleGAN is build on top of the Progressive GAN, implementing a style-based generator architecture, which includes a mapping and synthesis network for better control and versatility.



(a) Traditional

(b) Style-based generator

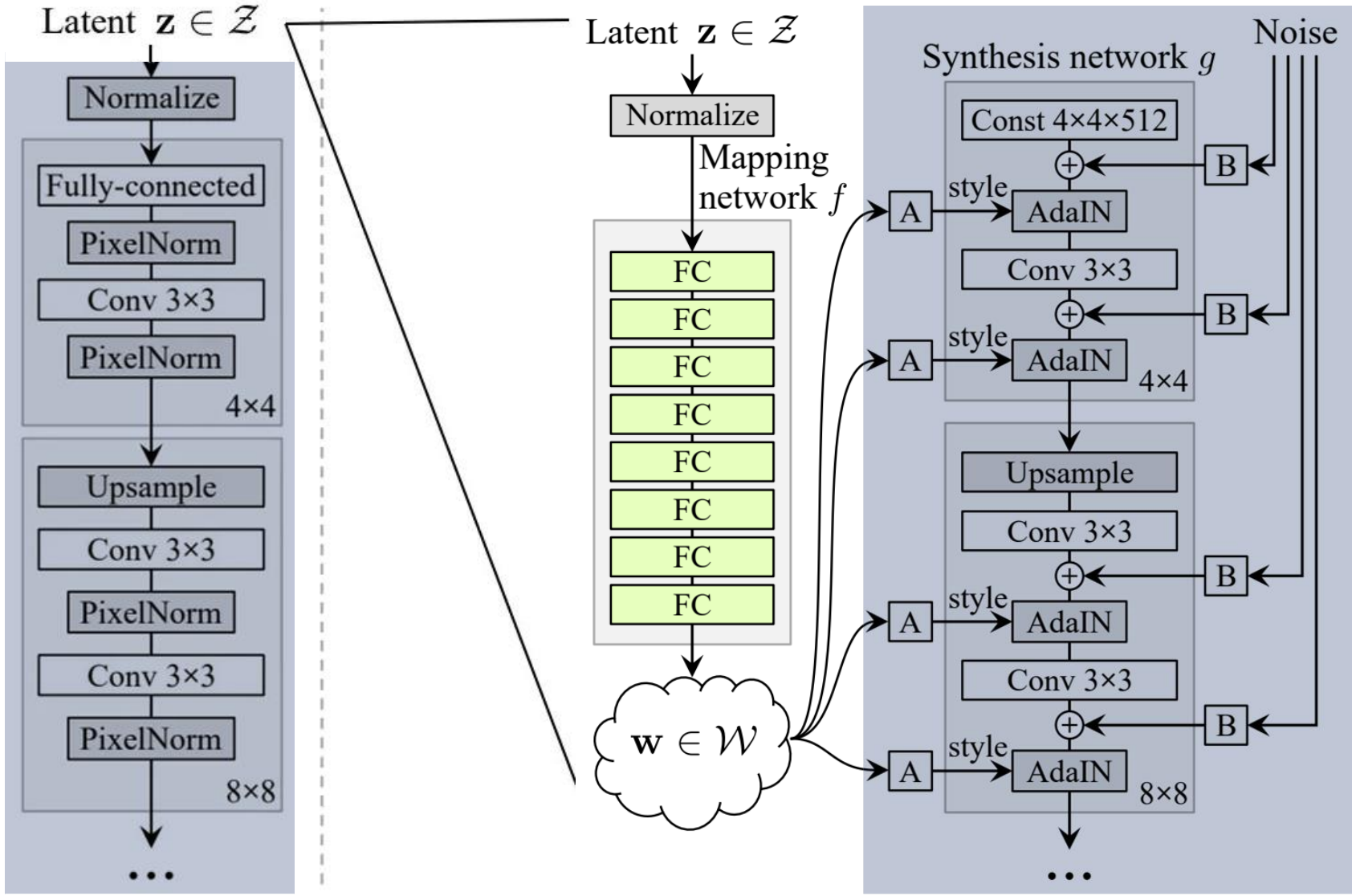
Image Source: https://arxiv.org/pdf/1812.04948.pdf

StyleGAN Architecture

- **Mapping Network:** Transforms input latent codes into an intermediate latent space, enabling control over style; this network consists of eight fully connected layers.
- **Synthesis Network:** Generates images from the intermediate latent space, combining style and content information; the network uses AdaIN operations and noise injection for improved control and variation.

Mapping Network

Traditional GAN vs
Style GAN Latent code

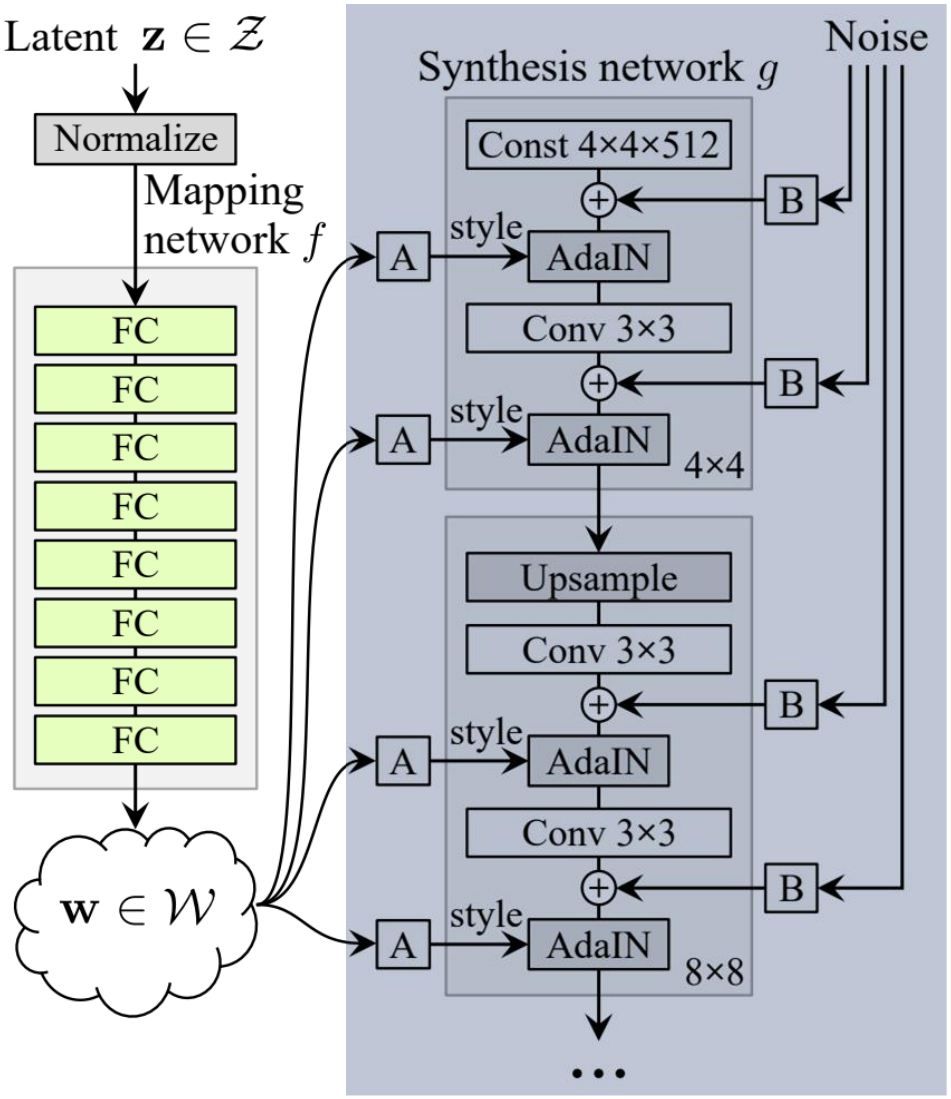


Traditional GAN

StyleGAN

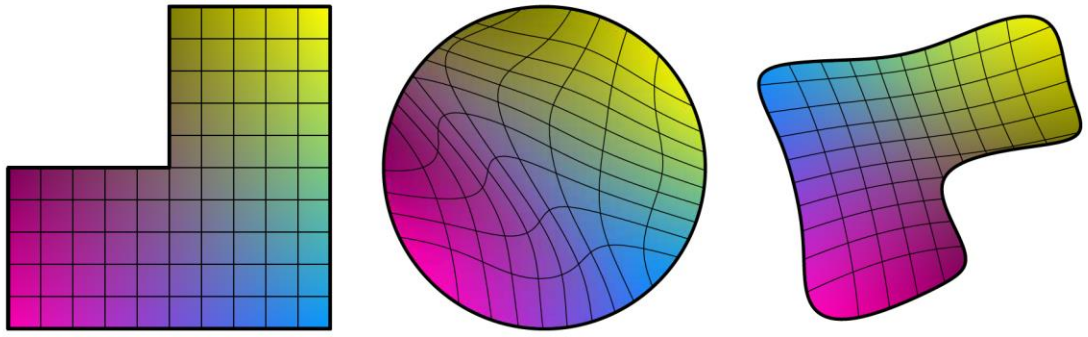
Mapping Network

- **Traditional Gan:** As the network progresses deeper, z's influence reduces
- **StyleGAN:** Introduce non-linear mapping network $f : z \rightarrow w$, which fed to each layer
- **Layers:** f is implemented using an 8-layer MLP
- **Dimensionality:** Set the dimensionality of both spaces (z and w) to 512



Mapping Network

- **Why Mapping?** Take for example generating portraits of military personnel with two factors: masculinity and hair length. In the training set, there's a missing category: male soldiers with long hair.
- **Intermediate Space:** Latent mapping transforms input latent codes into an intermediate latent space, which avoid feature combinations that do not found in the training dataset (such as male soldiers with long hair).



(a) Distribution of features in training set

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

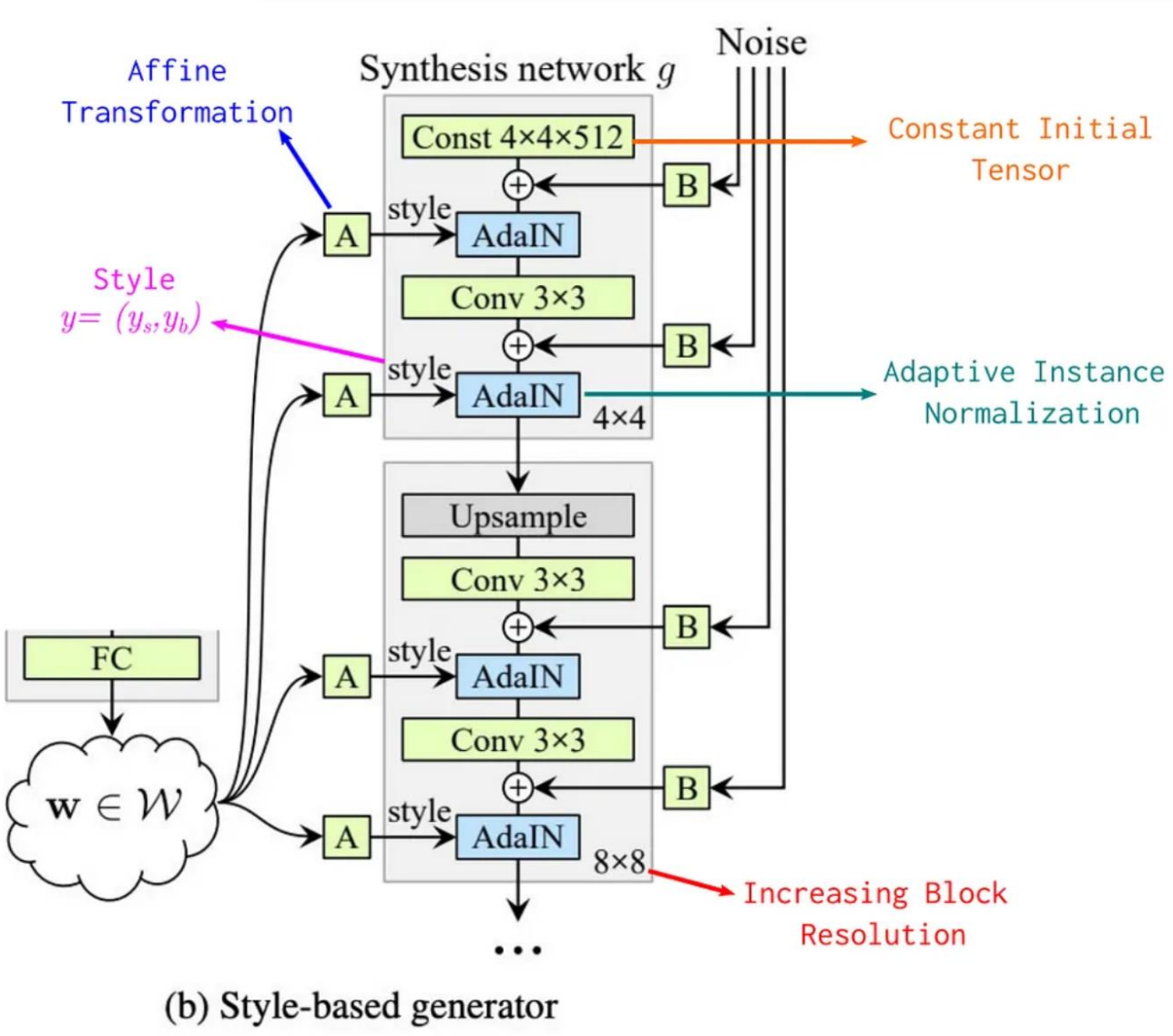
(c) Mapping from $\mathcal{W}$ to features

Image Source: https://arxiv.org/pdf/1812.04948.pdf

Disentanglement

- **Entanglement:** mixing or interdependence of different features or characteristics within a dataset, making it difficult to control them independently.
- **Traditional GAN entanglement:** Normal or uniform distributions, do not represent the real distribution. Thus, the latent space must include entangled factors.
- **Disentanglement:** Separating different features or characteristics in a dataset and learning a representation where each dimension captures an independent factor of the data so that they can be controlled independently.

Synthesis Network

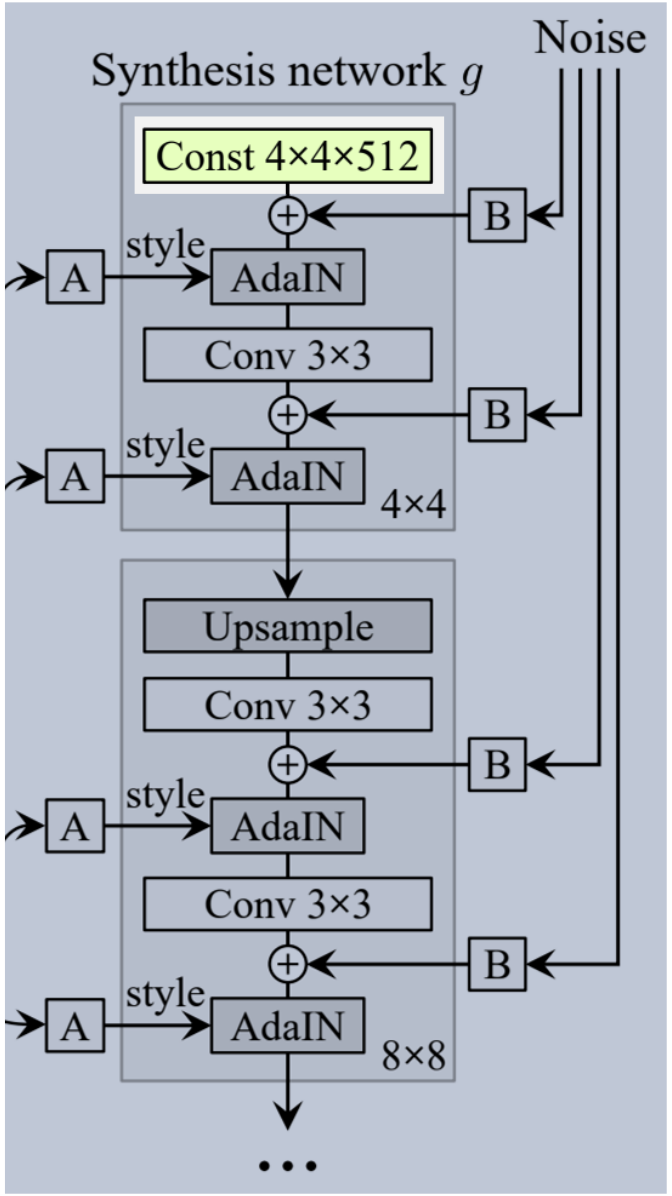


Synthesis Network Detailed

Image Source: [Review] Image2StyleGAN — Embedding An Image Into StyleGAN. | by Oscar Guarnizo | Medium

Constant Input Tensor

- **Constant Input Tensor:** StyleGAN uses a constant input tensor instead of traditional noise vectors
- **Improved Disentanglement:** The constant tensor provides the base content structure, while the style vectors and AdaIN operations handle style variations, which encouraged the network to disentangle content and style



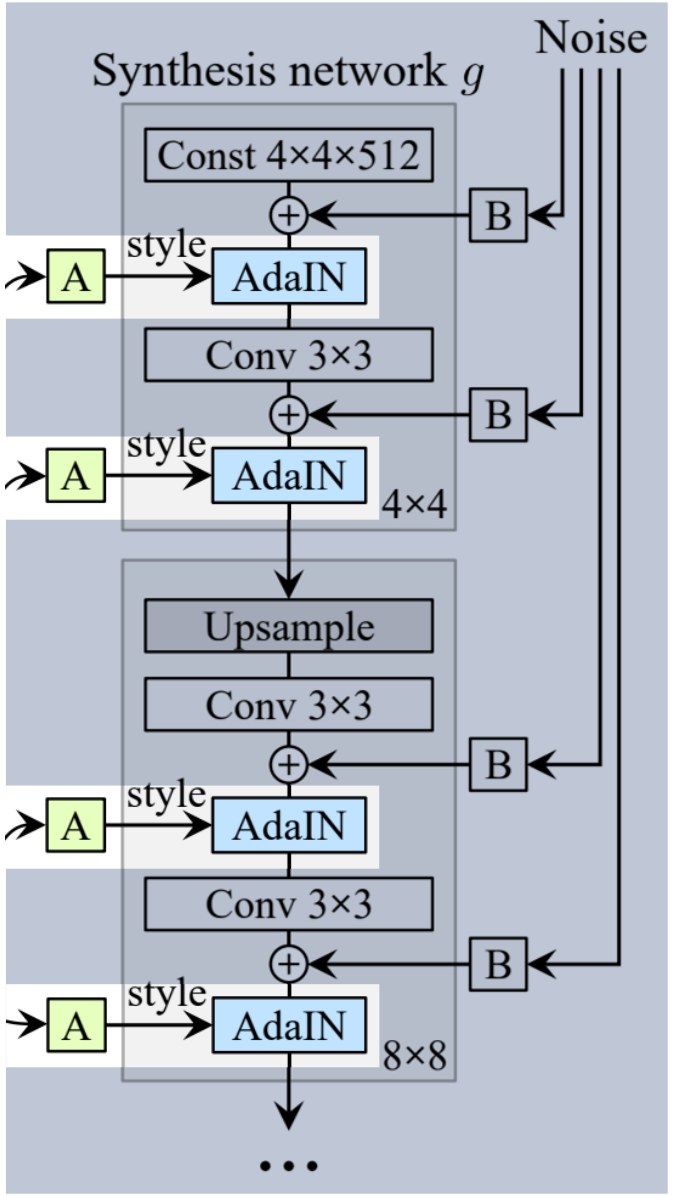
AdaIN

- **Affine Transformation:** For each feature layer we learn an affine operation A that transforms w into the style information $\begin{pmatrix} y_{s,i} \\ y_{b,i} \end{pmatrix} = A_i w$
- **Adaptive Instance Normalization:** This technique used in the synthesis network to incorporate style information into the generated image. It normalizes feature maps and rescales them using style-related affine transformations

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

Scale

bias



AdaIN

- **Dimensions: z= 512x1, w=512x1, A=2xC, AdaIN=CxHxW**

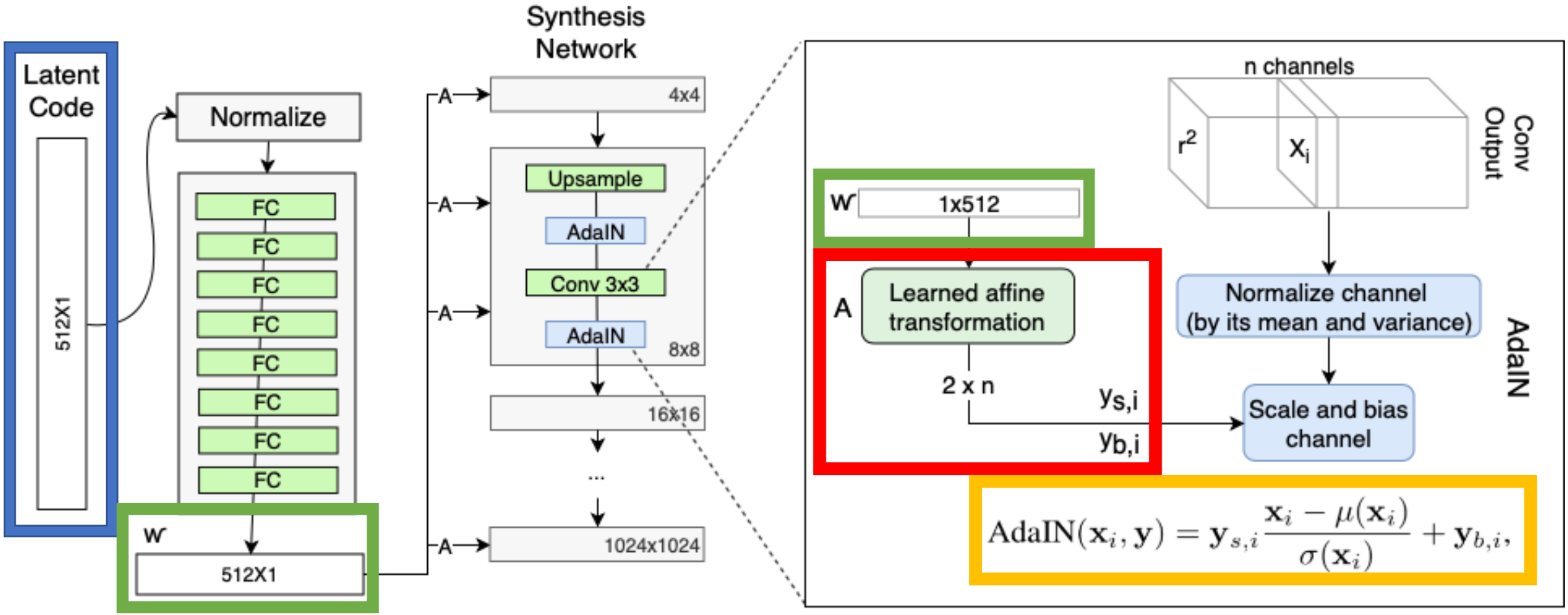


Image Source: http://people.ee.duke.edu/~lcarin/Qian7.19.2019.pdf

Stochastic Noise

- **Noise Injection:** StyleGAN introduces noise directly into the feature maps of the synthesis network, allowing for better control over stochastic variation
  Noise Scaling: At every spatial layer, a Gaussian noise matrix is incorporated into the feature maps, and the mapping B learns distinct scaling factors for each feature map.
- **Disentanglement:** This further helping disentangle high-frequency details from the global structure.

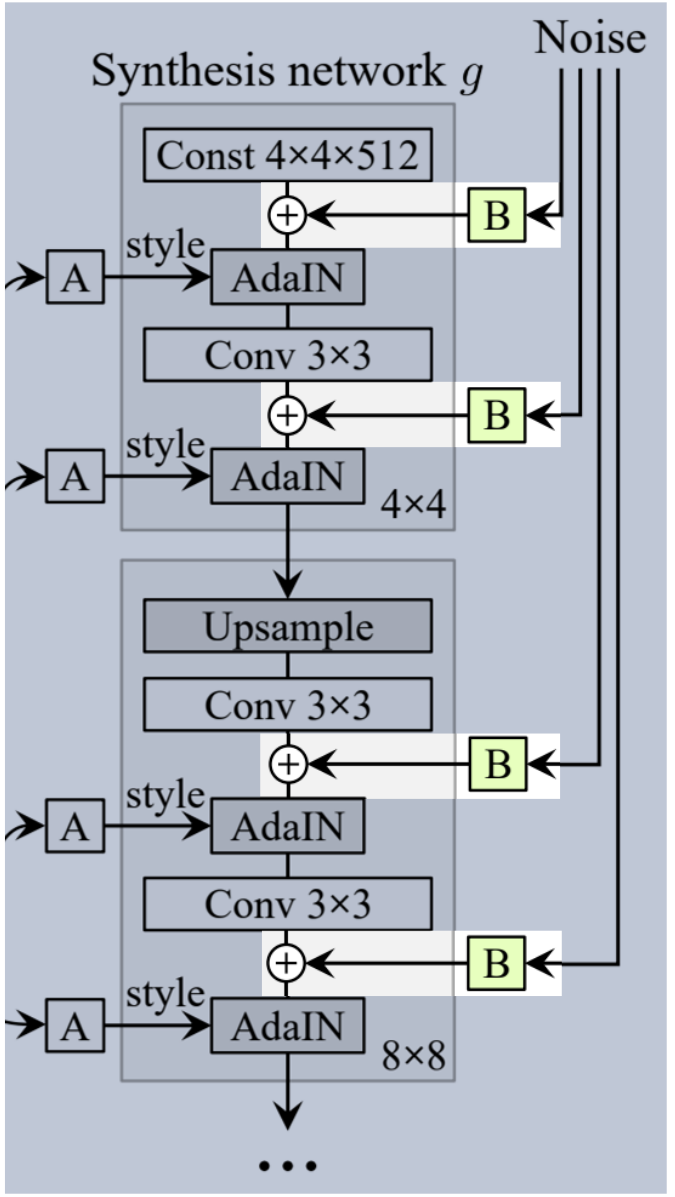


Image Source: https://arxiv.org/pdf/1812.04948.pdf

Stochastic Noise

- **Noise injection step-by-step breakdown:**

  (a) Sample HxW Gaussian noise for each layer.

  (b) Learn channel-specific scaling factors.

  (c) Multiply noise map by scaling factors.

  (d) Add scaled noise element-wise to feature maps.

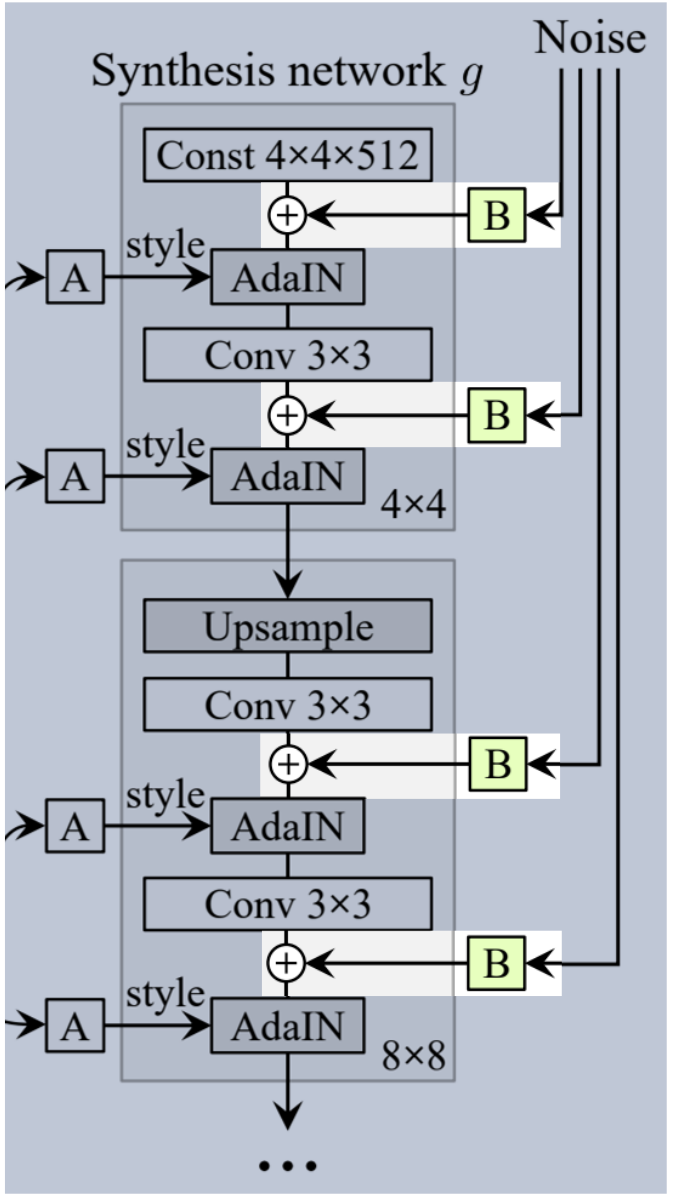


Image Source: https://arxiv.org/pdf/1812.04948.pdf

Stochastic Noise

(a) Generated image (b) Stochastic variation (c) Standard deviation

Image Source: https://arxiv.org/pdf/1812.04948.pdf

Stochastic
Variation Example

Stochastic Noise

- **Effect of noise:** Stochastic noise inputs at different layers of the generator.

  (a) Noise is applied to all layers.

  (b) No noise.

  (c) Noise in fine layers only

  (d) Noise in coarse layers only

  Noise in all layers resulted in a more realistic outcome

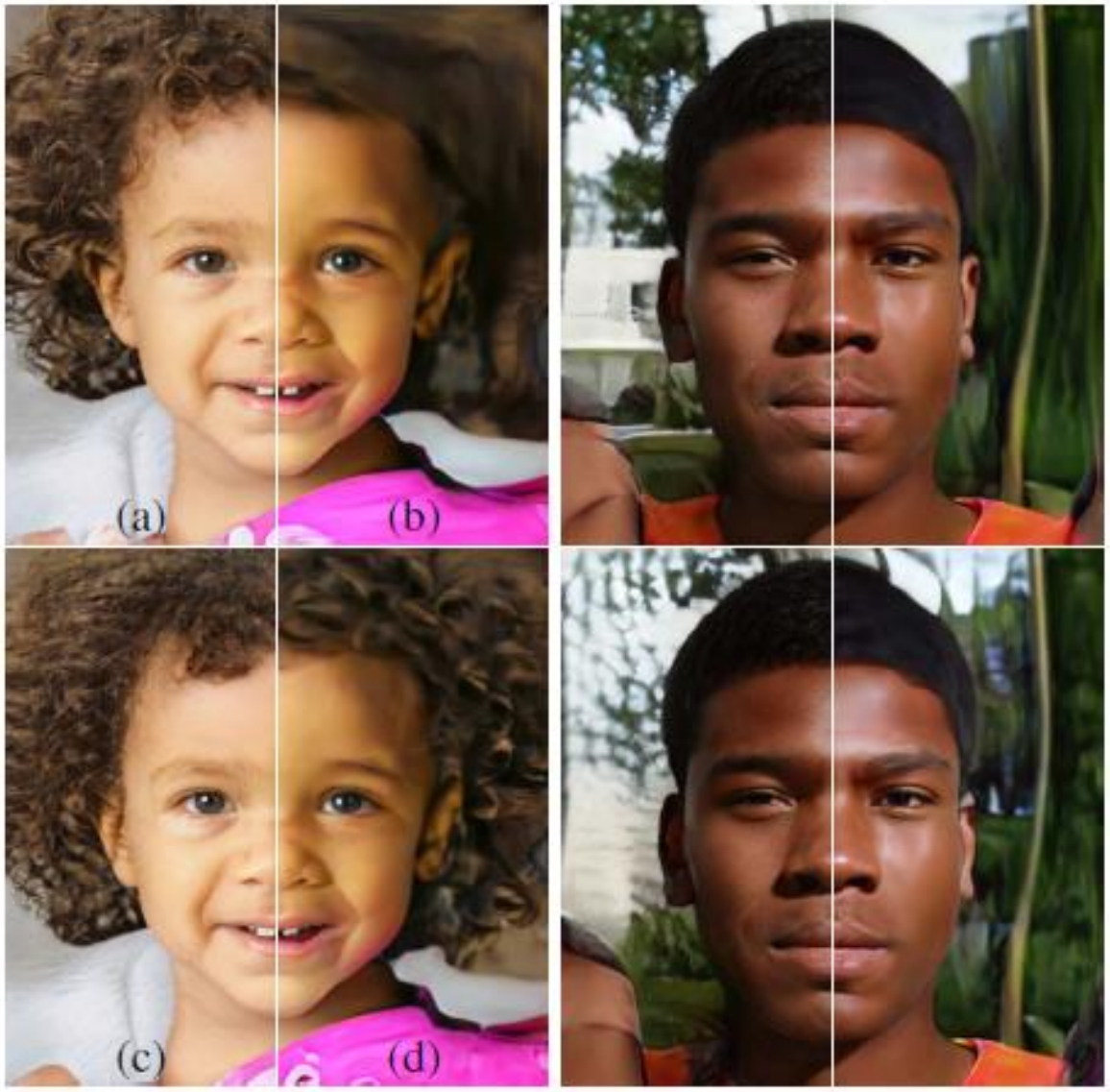


Image Source: https://arxiv.org/pdf/1812.04948.pdf

Style Mixing

- **Style Mixing Technique:** Combining multiple styles within a single generated image. By sampling multiple latent codes and applying them to different layers of the synthesis network, the generator can produce images with a mixture of characteristics from multiple sources
- **Image Characteristics Control:** StyleGAN allows for manipulation of global, intermediate, and local image characteristics, resulting in a diverse range of generated image styles.

Image Source: https://arxiv.org/pdf/1812.04948.pdf

Style Mixing

- Coarse styles: pose, hair, face shape.
- Middle styles: facial features, eyes.
- Fine styles: color scheme.

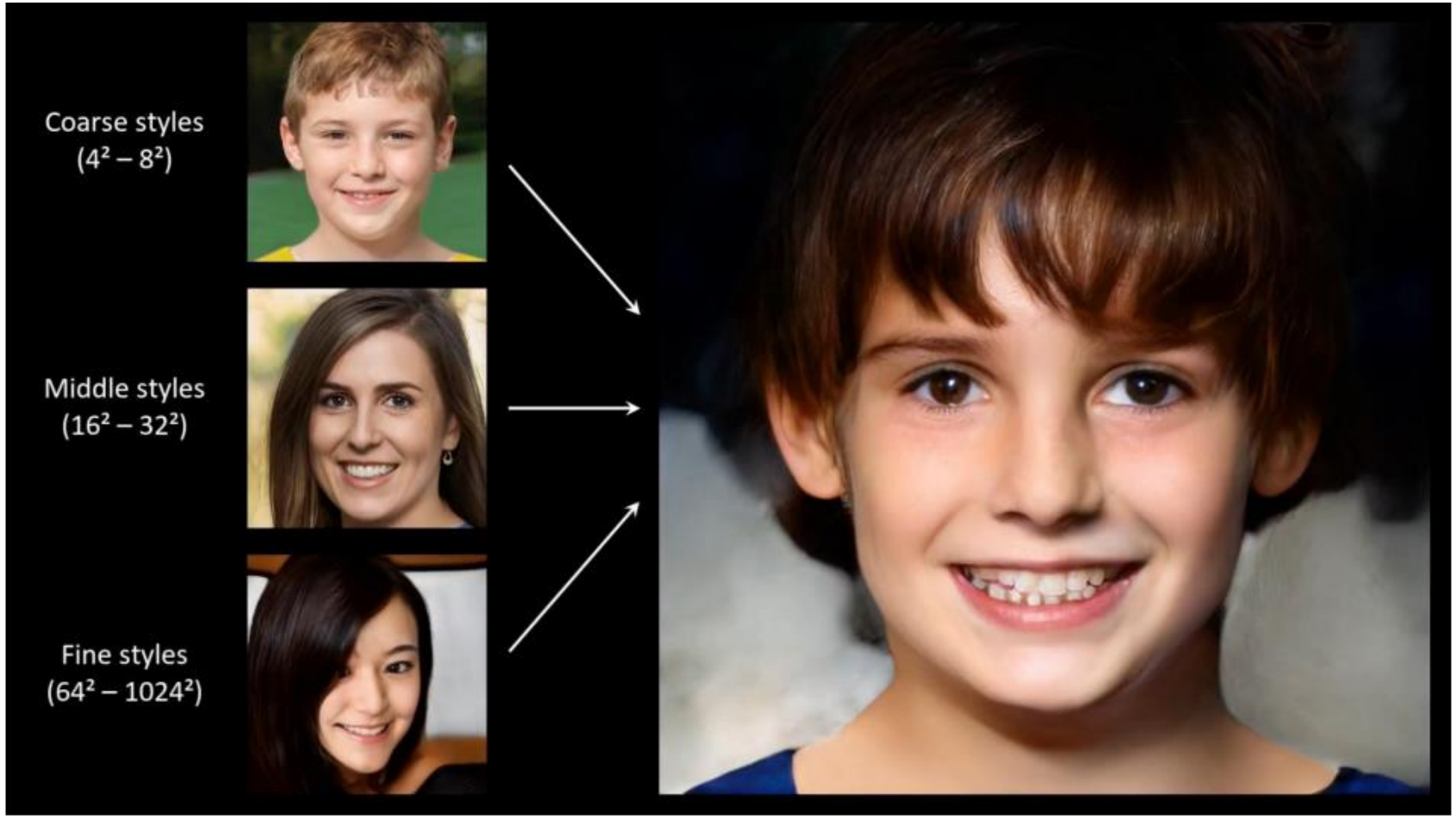


Image Source: https://dvl.in.tum.de/slides/adl4cv-ss20/7-Videos-Autoreg.pdf

Style Mixing

- Coarse styles: pose, hair, face shape.
- Middle styles: facial features, eyes.
- Fine styles: color scheme.

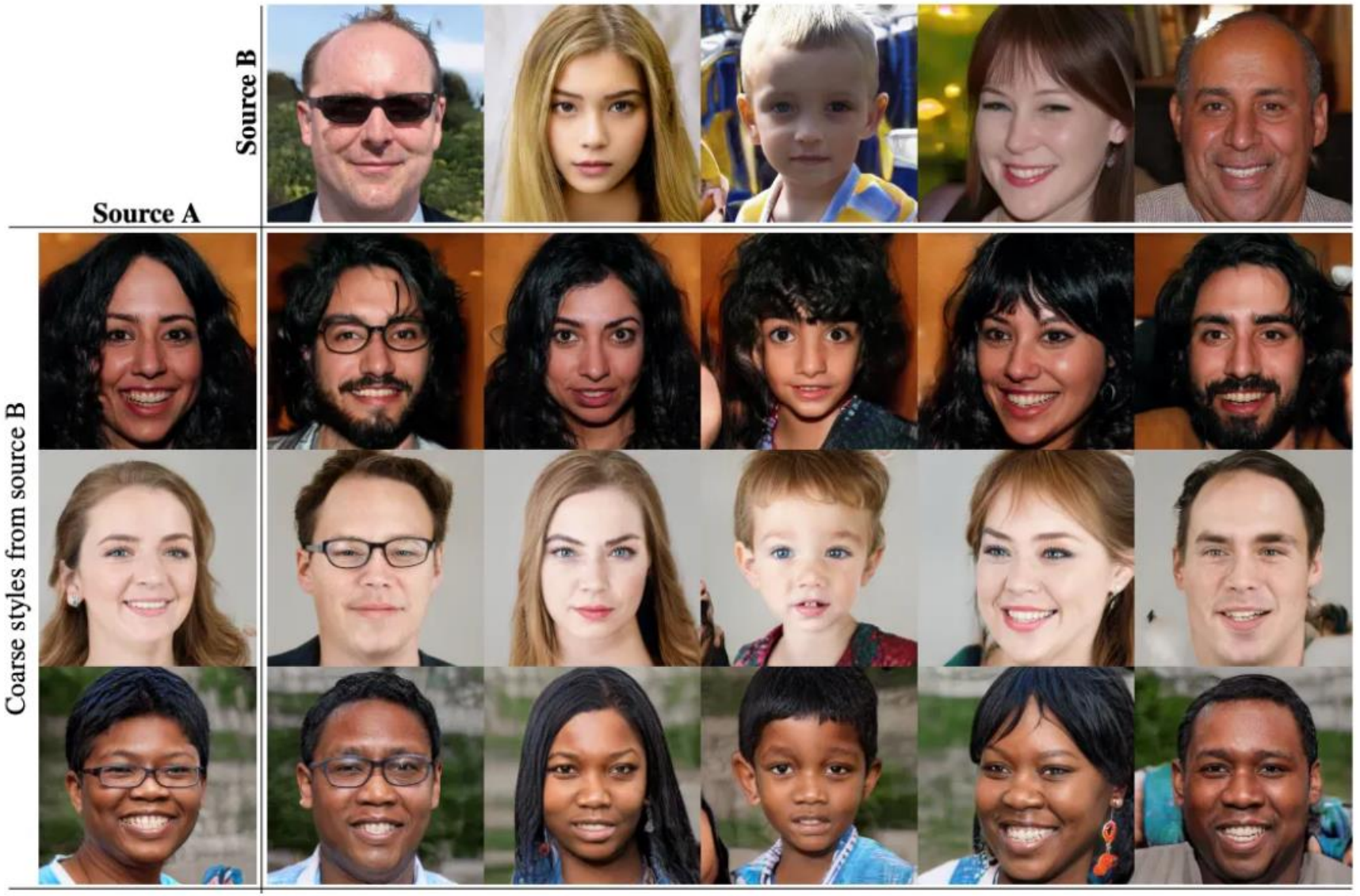


Image Source: https://arxiv.org/pdf/1812.04948.pdf

Style Mixing

- Coarse styles: pose, hair, face shape.
- Middle styles: facial features, eyes.
- Fine styles: color scheme.

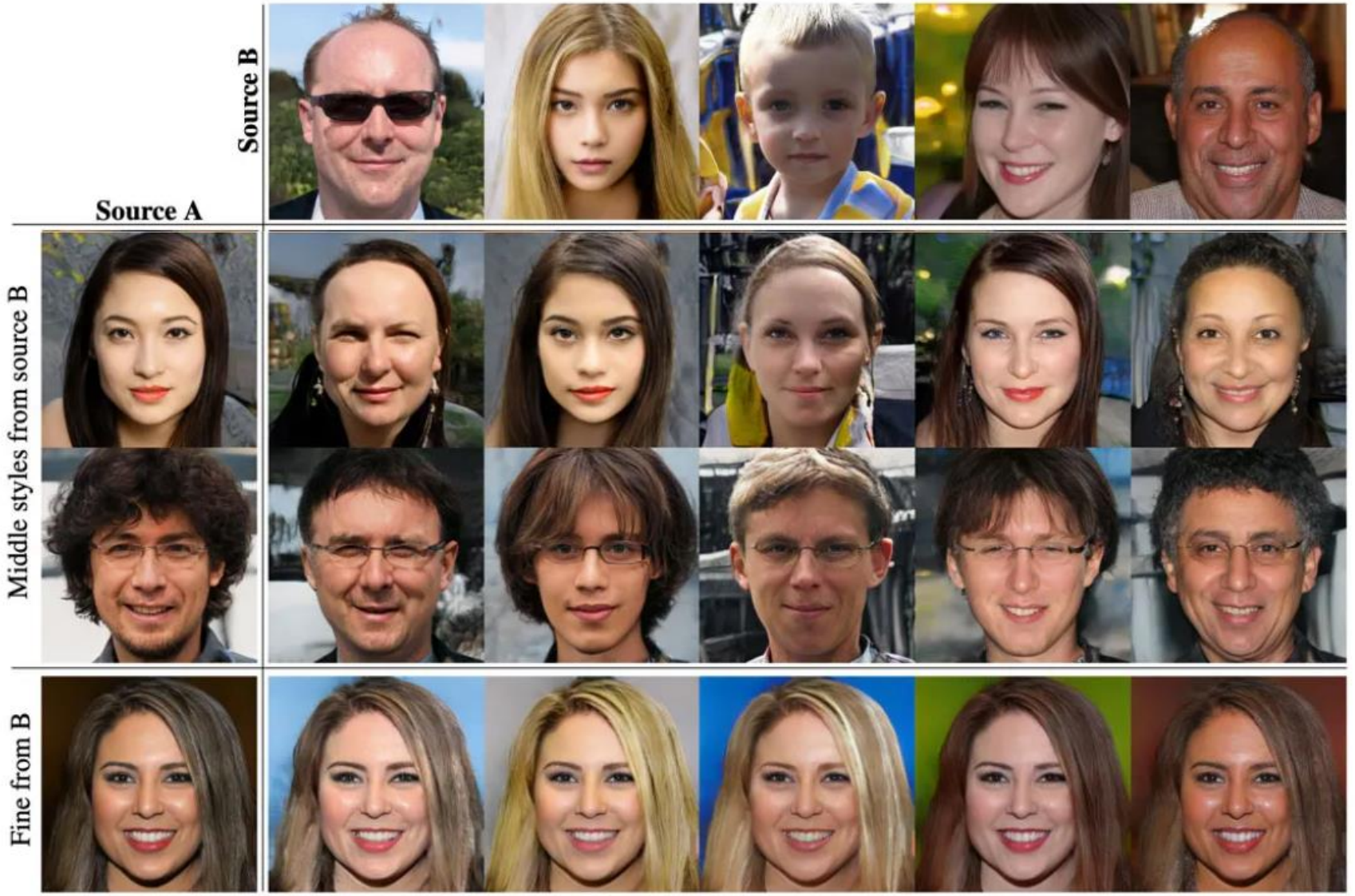


Image Source: https://arxiv.org/pdf/1812.04948.pdf

Style Scaling

- **Style intensity :** StyleGAN offers the ability to scale or adjust the intensity of style features within generated images.

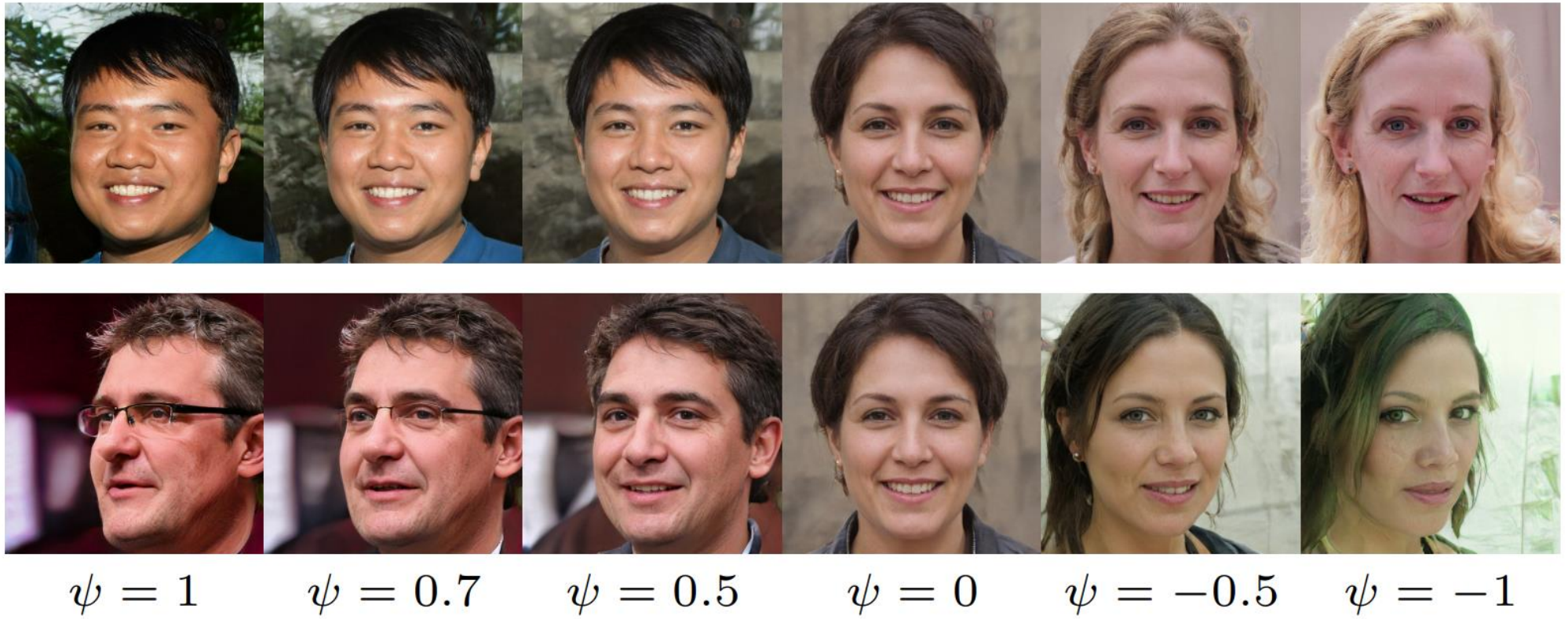


Image Source: https://arxiv.org/pdf/1812.04948.pdf

Style Scaling

- **How to Scale:** By modifying the weights associated with the AdaIN layers in the synthesis network, users can control the prominence of certain style characteristics
- **Mean face:** For all sampled images we compute the mean style $\overline{w}$, representing the "mean face" across all images
- **Style Deviation:** Style-scaling adjusts the style deviation about the mean

$$w' = \overline{w} + \psi(x - \overline{w}), \qquad where: \ \overline{w} = E_{z \sim p(z)}[f(z)]$$

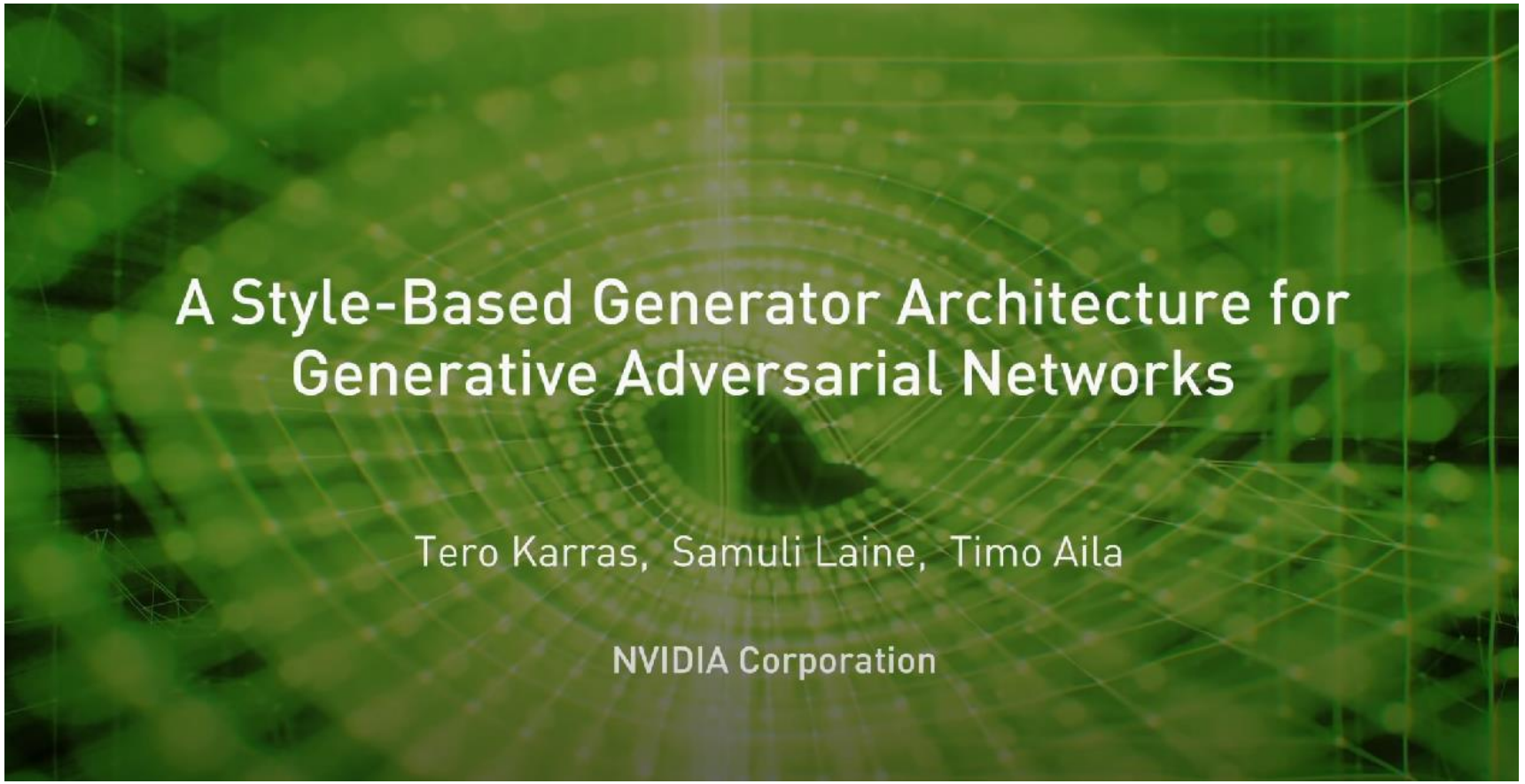
“Mean face”

StyleGAN Results

- **StyleGAN Superiority:** StyleGAN achieves top performance on CelebA-HQ and FFHQ datasets, surpassing PGGAN and other GANs in image quality and control. Its capabilities include diverse human face generation and superior feature disentanglement.

Image Source: https://arxiv.org/pdf/1812.04948.pdf

StyleGAN Video



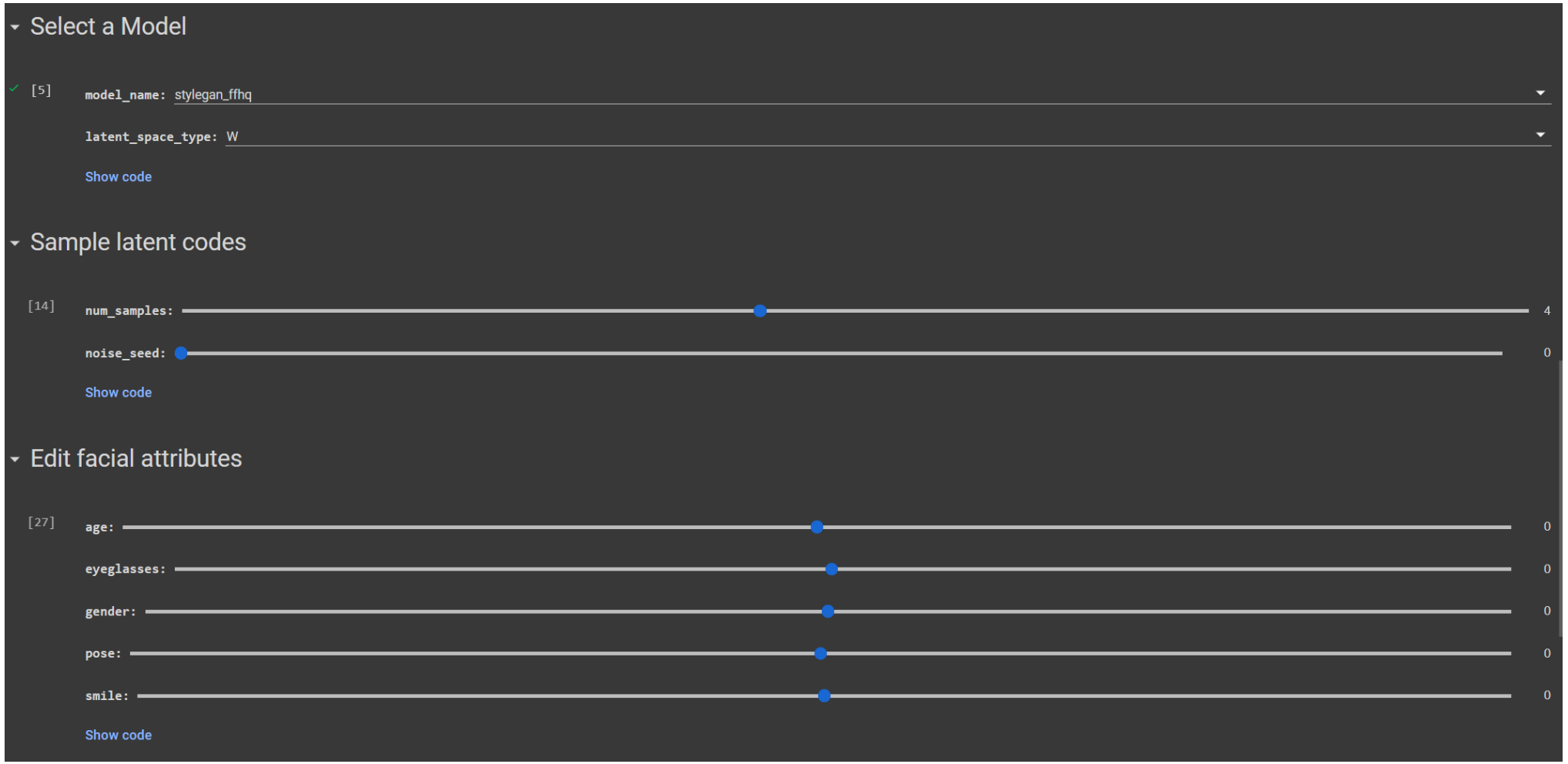
A Style-Based Generator Architecture for Generative Adversarial Networks - YouTube

(24) Synthesizing High-Resolution Images with StyleGAN2 - YouTube

Conclusion

- **Key Contributions:** StyleGAN introduces a groundbreaking generator architecture that improves control, disentanglement, and quality in GAN-based image synthesis.
- **Impact on Image Synthesis:** StyleGAN has a significant impact on the field of image synthesis, paving the way for further research and improvements in GAN architectures.

Implementation



[InterFaceGAN - Colaboratory (google.com)]

Image2StyleGAN++ Introduction

- **Image2StyleGAN++ Overview:** Image2StyleGAN++ is an extension of StyleGAN that allows for the efficient and robust manipulation of real images by inverting them into StyleGAN's latent space.

Image2StyleGAN++: How to Edit the Embedded Images?

Rameen Abdal
KAUST
rameen.abdal@kaust.edu.sa

Yipeng Qin
Cardiff University
qiny16@cardiff.ac.uk

Peter Wonka
KAUST
pwonka@gmail.com

(a) (b) (c) (d)

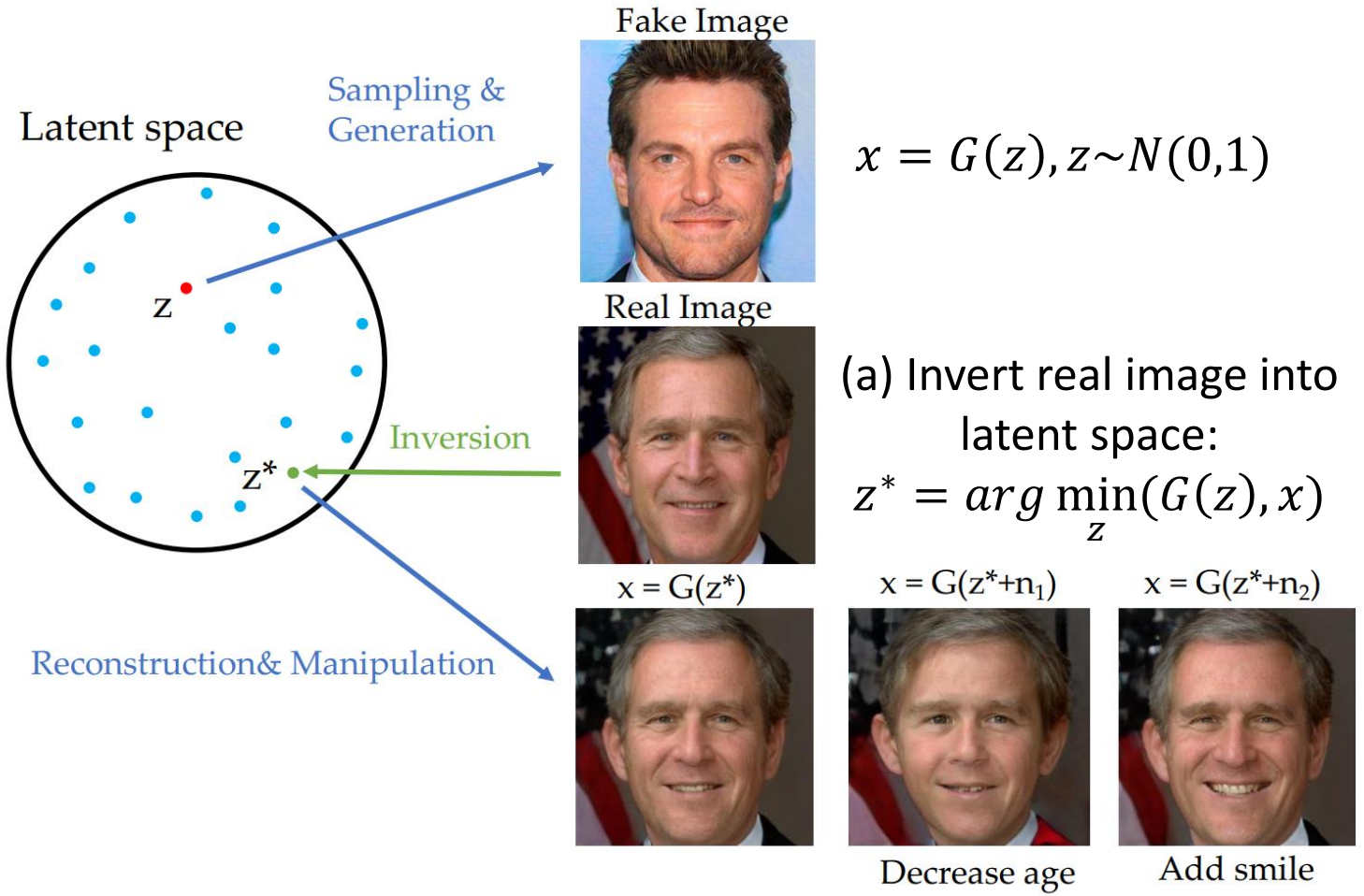
Figure 1: (a) and (b): input images; (c): the "two-face" generated by naively copying the left half from (a) and the right half from (b); (d): the "two-face" generated by our Image2StyleGAN++ framework.

Image2StyleGAN++: How to Edit the Embedded Images?

Authors	Rameen Abdal, Yipeng Qin, Peter Wonka
Publication date	2020
Conference	Proceedings of the IEEE/CVF CVPR Conference on Computer Vision and Pattern Recognition 2020
Pages	8296-8305
Description	We propose Image2StyleGAN++, a flexible image editing framework with many applications. Our framework extends the recent Image2StyleGAN in three ways. First, we introduce noise optimization as a complement to the W+ latent space embedding. Our noise optimization can restore high frequency features in images and thus significantly improves the quality of reconstructed images, eg a big increase of PSNR from 20 dB to 45 dB. Second, we extend the global W+ latent space embedding to enable local embeddings. Third, we combine embedding with activation tensor manipulation to perform high quality local edits along with global semantic edits on images. Such edits motivate various high quality image editing applications, eg image reconstruction, image inpainting, image crossover, local style transfer, image editing using scribbles, and attribute level feature transfer. Examples of the edited images are shown across the paper for visual inspection.
Total citations	Cited by 359

Inversion Challenge

- **Inversion Problem:** Finding a latent code z that, when input into StyleGAN, reconstructs a given real image x. (In our case w+)



$$x = G(z), z \sim N(0,1)$$

(a) Invert real image into latent space:

$$z^* = arg\min_z (G(z), x)$$

(b) Manipulate the inverted image in the latent space

Image Source: https://arxiv.org/pdf/2101.05278.pdf

Image2StyleGAN++ Inversion

- **Extended latent spaces Z+ and W+:**

  - **W+** is obtained by concatenating 18 different 512-dimensional w vectors, one for each layer of the StyleGAN architecture that can receive input via AdaIN.

  - **Z+** is the concatenation of the latent codes that correspond to the optimized 18 w vectors.

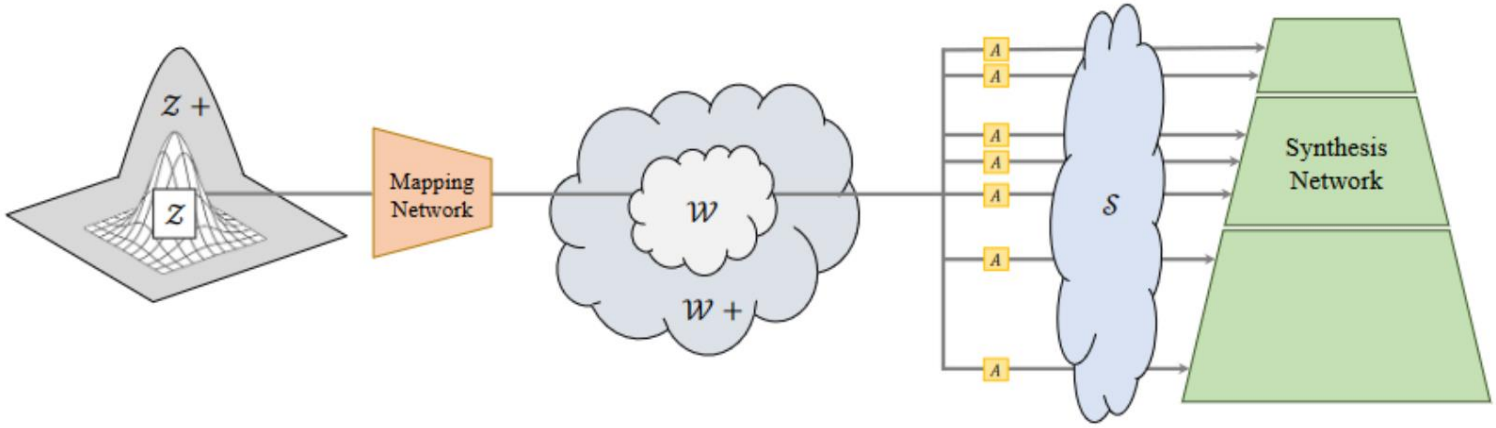


Image Source: https://arxiv.org/pdf/2212.09102.pdf

Image2StyleGAN++ Inversion

- **Why extended spaces?** The original latent space (z) and the intermediate latent space (w) may not provide enough flexibility to fully capture the characteristics of the input images during inversion.
- **Better inversion:** By extending the latent spaces and allowing the model to optimize w vectors individually for each layer, the inversion process becomes more powerful and can better capture the intricate details of the input images

Image2StyleGAN++ Inversion

- **Inversion Stages:**

  (a) Optimize w+: Find an optimal intermediate latent code w+

  (b) Optimize noise: Find an optimal noise map

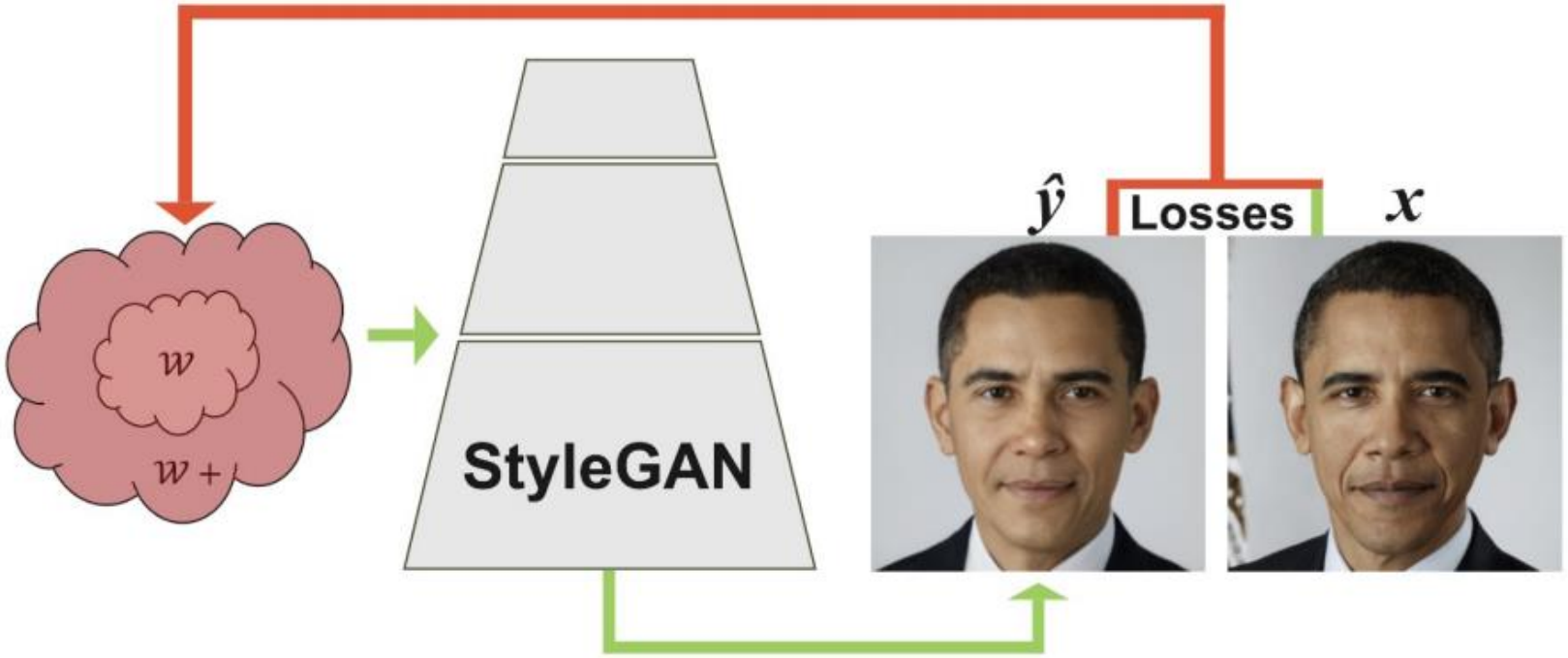


Image Source: https://arxiv.org/pdf/2212.09102.pdf

Image2StyleGAN++ Inversion

- **Optimize the latent code (w+):** For a given real image, find the best latent code (w+) that minimizes the difference between the real image and the image generated by the StyleGAN network using the latent code.
- **Noise fixed:** During this step, the noise maps are kept fixed.

- **Input:** Real image (x)
- **Output:** Latent code (w+)

Image2StyleGAN++ Inversion

- **Optimization Procedure step-by-step breakdown:**

  (a) Initialize W+ (random or estimated).

  (b) For each layer, optimize with backpropagation (min. the inversion loss).

  (c) Minimize inversion loss between real and generated images.

  (d) Update the 18 w vectors separately, noise maps fixed.

  (e) Iterate until convergence or max iterations.

Image2StyleGAN++ Inversion

- **Optimization Pseudo code:**

Input: images $x, y \in \mathbb{R}^{n \times m \times 3}$; masks M_s, M_m, M_p; a pre-trained generator $G(\cdot, \cdot)$; gradient-based optimizer F'.

Output: the embedded code (w, n)

Initialize() the code $(w, n) = (w', n')$;

while *not converged* **do**

  $Loss \leftarrow L(x, y, M_s, M_m, M_p)$;

  $(w, n) \leftarrow (w, n) - \eta F'(\nabla_{w,n} L, w, n)$;

end

Image2StyleGAN++ Inversion

- **Optimize noise maps:** After obtaining the optimal latent code (w) in the first step, further refine the reconstruction by optimizing the noise maps
- **Latent Code Fixed:** During this step, we're keeping the latent code (w+) fixed
- **Why optimize noise separately?** By separately optimizing the noise maps, you can fine-tune these local features without affecting the global structure or style information already captured by the optimal w.

- **Input:** Real image (x), latent code (w+), initial noise maps
- **Output:** Optimized noise maps

Image2StyleGAN++ Inversion

Original | embedded in W+ Space | embedded in W+ Space and Noise space

Original | embedded in W+ Space

Image Source: https://arxiv.org/pdf/1911.11544.pdf

Image2StyleGAN++ Inversion

- **Enhanced optimization:** Image2StyleGAN++ employs an improved optimization strategy that includes alternating between optimizing the latent code (w+) and the noise maps.
- **Why alternating?** This alternating process allows for a more thorough exploration of the optimization landscape and ensures that the final reconstructed image is of high quality.

Inversion Loss Functions

- **Loss Functions:** Image2StyleGAN++ uses multiple loss functions during the inversion process: (a) Pixel Loss, (b) Perceptual Loss, (c) Style Loss

Combined Loss: The combined loss is a weighted sum of the three individual losses.

$$
\begin{aligned}
L = & \lambda_s L_{style}(M_s, G(w,n), y) \\
& + \frac{\lambda_{mse_1}}{N} \| M_m \odot (G(w,n) - x) \|_2^2 \\
& + \frac{\lambda_{mse_2}}{N} \| (1 - M_m) \odot (G(w,n) - y) \|_2^2 \\
& + \lambda_p L_{percept}(M_p, G(w,n), x)
\end{aligned}
$$

Inversion Loss Functions

- **(a) Mm (Pixel Loss Mask):** Focuses pixel-wise MSE loss on specific regions. Directs optimization towards important areas in the image.
- **(b) Ms (Style Loss Mask):** Controls which spatial locations contribute to the style loss. Guides optimization towards desired styles or textures.
- **(c) Mp (Perceptual Loss Mask):** Applied to VGG feature differences to control spatial location contributions. Focuses optimization on semantically important regions.

Inversion Loss Functions

- **Why Masks?** Masks help direct the optimization process and improve reconstruction performance by focusing on important areas in the input image.
- **How to determine?** Use domain-specific knowledge, Identify regions with specific styles or textures of interest, Use attention mechanisms or semantic segmentation.

Inversion Loss Functions

- **Pixel-wise Loss:** Measures the difference between the input image (x) and the generated image (G(w)) on a per-pixel basis. It encourages the generated image to be visually like the input image.
- **Comparison**: Directly compares the pixel values of the generated image and the input image.

$$\frac{\lambda_{mse_1}}{N}\|M_m \odot (G(w,n) - x)\|_2^2 + \frac{\lambda_{mse_2}}{N}\|(1 - M_m) \odot (G(w,n) - y)\|_2^2$$

Inversion Loss Functions

- **Perceptual Loss:** Uses pre-trained VGG network features to measure the semantic content difference between the input image (x) and the generated image (G(w+)). It encourages the generated image to maintain the high-level content structure of the input image.
- **Comparison:** Extracts features from multiple layers of the VGG network for both the generated and input images, and compares the features using a distance metric (e.g., L2 or L1 distance).

$$\lambda_p L_{percept}(M_p, G(w, n), x)$$

Inversion Loss Functions

- **Style Loss:** Encourages the generated image to have the same style as the input image. It measures the difference between the Gram matrices of the feature maps extracted from different layers of a pre-trained VGG network for both the real and generated images.
- **Comparison:** Computes the Gram matrix for each layer's features in the VGG network for both the generated and input images, and compares the Gram matrices using a distance metric (e.g., L2 or L1 distance).

$$\lambda_s L_{style}(M_s, G(w, n), y)$$

Inversion Loss Functions

- **Compute:** To compute the Gram matrix for an image:

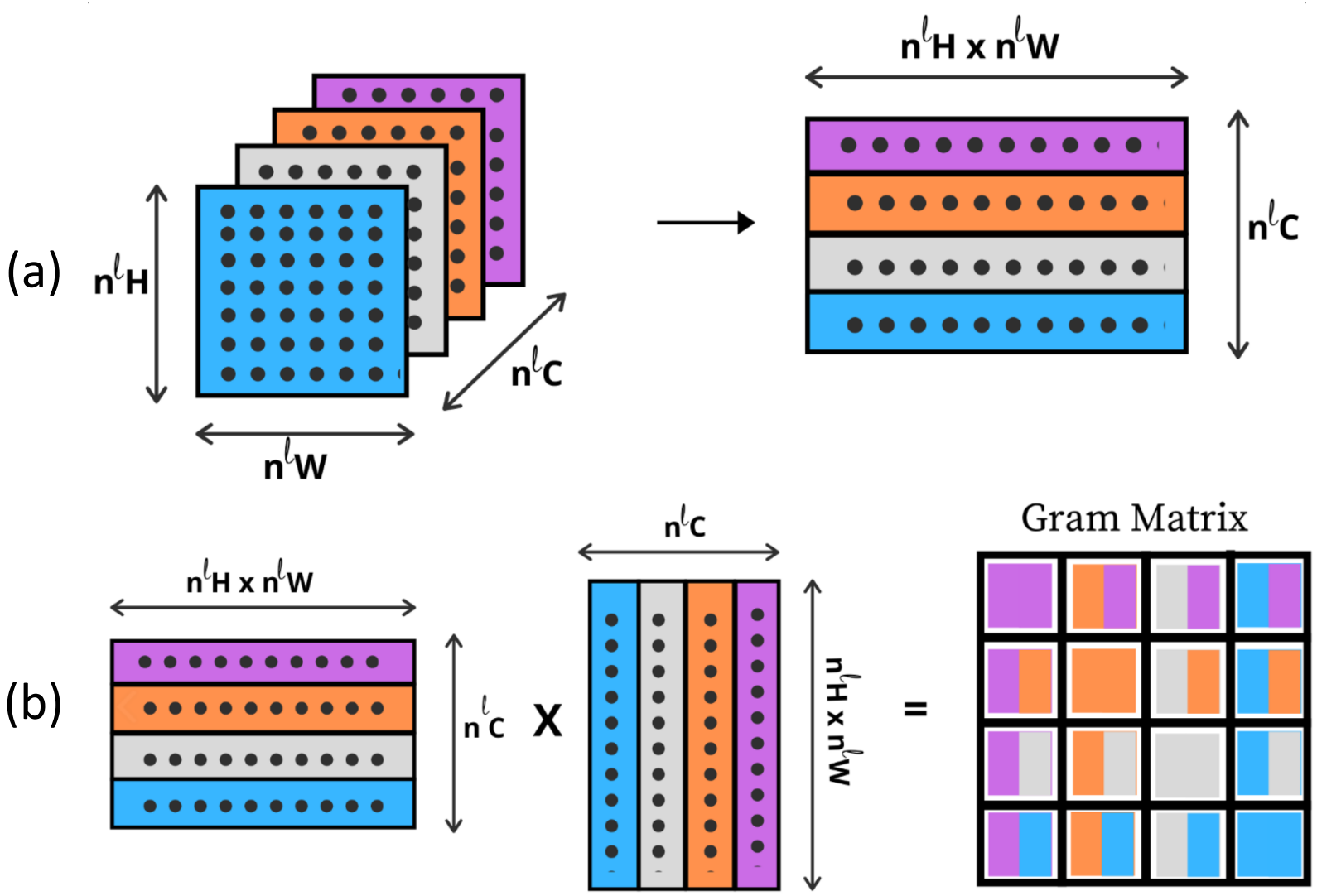
  (a) Pass the image through the VGG network and obtain its feature maps at specific layers.

  (b) Flatten each feature map into a 2D matrix (rows: channels, columns: flattened spatial dimensions).

  (c) Compute the inner product between each pair of feature channels by multiplying the 2D matrix with its transpose: $G = A * A^T$.

- **Correlation:** The resulting Gram matrix has entries (i, j) representing the correlation between the i-th and j-th feature channels.

Inversion Loss Functions



Gram Matrix Example

Image Source: https://aigong.tistory.com/360

Inversion Loss Functions

- **Measure:** To measure closeness between Gram matrices for input and generated images:

  (a) Compute Gram matrices for both images at specific layers.

  (b) Calculate the squared Frobenius norm (sum of squared differences) between corresponding Gram matrices for each layer.

  (c) Weight and sum squared Frobenius norms from different layers to obtain the overall style loss.

- **Why style loss?** Minimizing style loss encourages the generated image to have similar style features as the input image.

Image2StyleGAN++ Applications

- **Image Editing:** Image2StyleGAN++ allows users to edit real images using the powerful disentanglement and manipulation properties of StyleGAN.

Image Source: https://arxiv.org/pdf/1904.03189.pdf

Image2StyleGAN++ Applications

- **Face Swapping:** Replacing the facial features of one person with those of another while maintaining the original pose and lighting conditions.

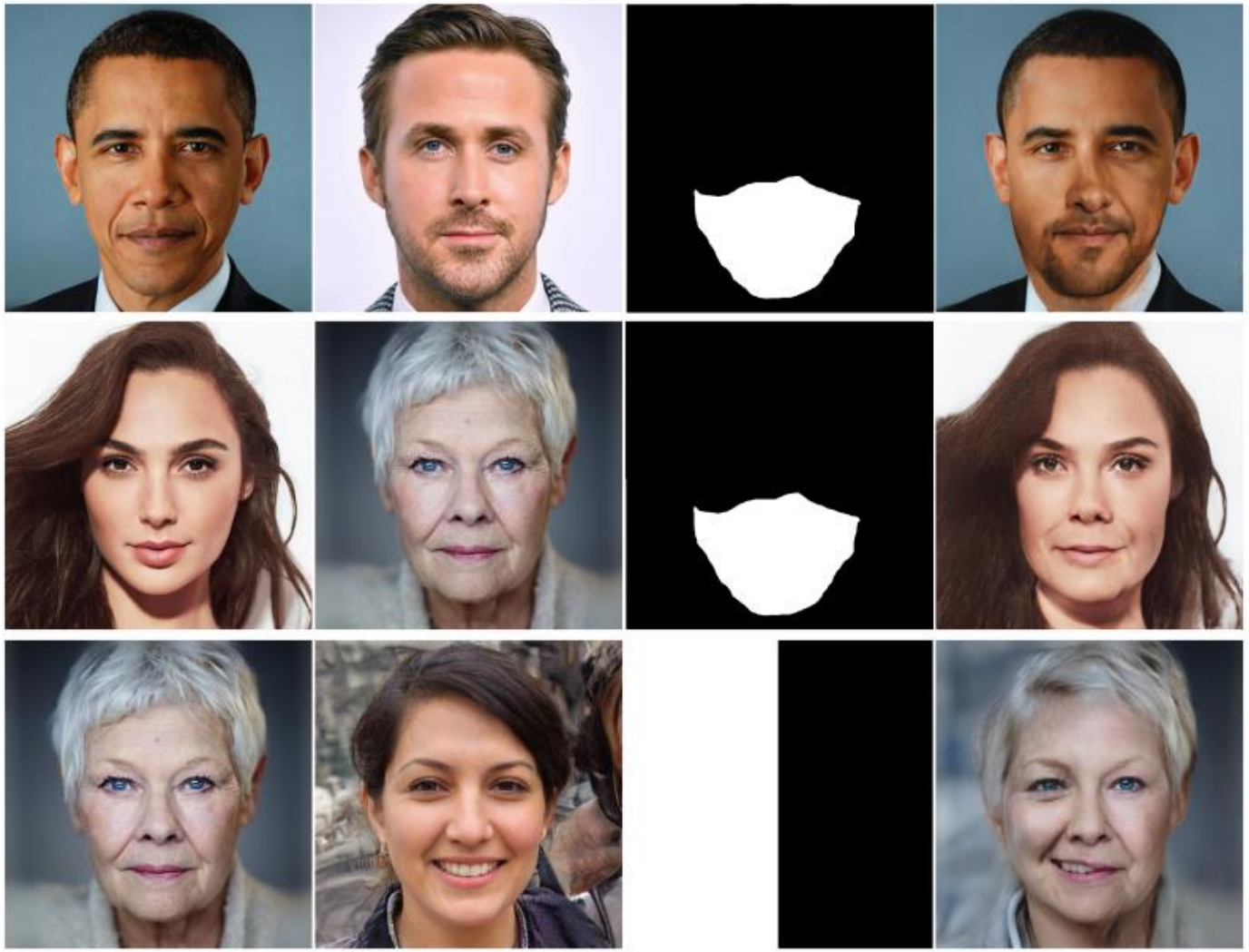


Image Source: https://arxiv.org/pdf/1904.03189.pdf

Image2StyleGAN++ Applications

- **Attribute Manipulation:** Adjusting facial attributes like age, hairstyle, or expression using StyleGAN's latent space arithmetic.

Image Source: https://arxiv.org/pdf/1904.03189.pdf

Image2StyleGAN++ Applications

- **Style Mixing:** Combining different styles of multiple images while preserving the content of the target image.

Image Source: https://arxiv.org/pdf/1904.03189.pdf

Image2StyleGAN++ Applications

- **Image Morphing:** Generating a smooth transition between two real images by interpolating their latent codes.

Image Source: https://arxiv.org/pdf/1904.03189.pdf

Image2StyleGAN++ Results

- **Image2StyleGAN++ Performance:** The method demonstrates state-of-the-art performance in inverting real images into StyleGAN's latent space, allowing for accurate image reconstruction and diverse image manipulation capabilities.
- **Comparison to Prior Techniques:** Image2StyleGAN++ outperforms previous inversion methods in terms of image reconstruction quality and computational efficiency.

Conclusion

- **Key Contributions:** Image2StyleGAN++ extends the capabilities of StyleGAN, enabling efficient and powerful manipulation of real images by inverting them into the latent space.
- **Impact on Image Editing:** The method has the potential to revolutionize image editing by leveraging the disentanglement properties and versatility of StyleGAN for real-image manipulation.