Annotated By: Sushant Gautam
Part of 30 Day GAN Paper Reading
https://github.com/sushant097/30-Days-GANs-Paper-Reading

Style GAN

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.

1. Introduction

The resolution and quality of images produced by generative methods — especially generative adversarial networks (GAN) [22] — have seen rapid improvement recently [30, 45, 5]. Yet the generators continue to operate as black boxes, and despite recent efforts [3], the understanding of various aspects of the image synthesis process, e.g., the origin of stochastic features, is still lacking. The properties of the latent space are also poorly understood, and the commonly demonstrated latent space interpolations [13, 52, 37] provide no quantitative way to compare different generators against each other.

Motivated by style transfer literature [27], we re-design the generator architecture in a way that exposes novel ways to control the image synthesis process. Our generator starts from a learned constant input and adjusts the "style" of the image at each convolution layer based on the latent code, therefore directly controlling the strength of image features at different scales. Combined with noise injected directly into the network, this architectural change leads to automatic, unsupervised separation of high-level attributes (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 quantifying these aspects of the generator. Using these metrics, we show that compared to a traditional generator architecture, our generator admits a more linear, less entangled representation of different factors of variation.

Finally, we present a new dataset of human faces (Flickr-Faces-HQ, FFHQ) that offers much higher quality and covers considerably wider variation than existing high-resolution datasets (Appendix A). We have made this dataset publicly available, along with our source code and pre-trained networks.[1] The accompanying video can be found under the same link.

2. Style-based generator

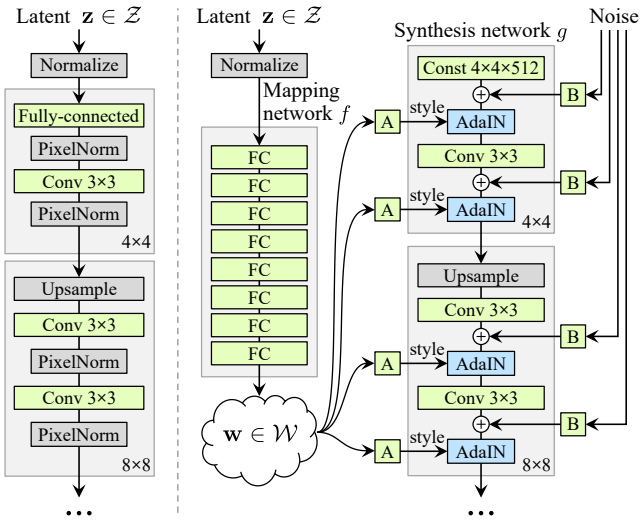
Traditionally the latent code is provided to the generator through an input layer, i.e., the first layer of a feed-forward network (Figure 1a). We depart from this design by omitting the input layer altogether and starting from a learned constant instead (Figure 1b, right). Given a latent code $\mathbf{z}$ in the input latent space $\mathcal{Z}$, a non-linear mapping network $f : \mathcal{Z} \to \mathcal{W}$ first produces $\mathbf{w} \in \mathcal{W}$ (Figure 1b, left). For simplicity, we set the dimensionality of both

[1] https://github.com/NVlabs/stylegan

The key idea author try to generate images of various styles i.e. add some style controlled input to generator.
In traditional architecture(seems like ProGAN) where latent code to generator from input layer, whereas in StyleGAN first they pass latent code to the network which perform non-linear mapping and produce W.

Then, learned affine transformaion specialize w to styles y=(ys,yb) that control AdaIn(Adaptive instance normalization) defined in eqn (1). Also, explicit noise as input to generator.

Hope that is more clear in below highlighted descriptions.



(a) Traditional (b) Style-based generator

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

spaces to 512, and the mapping f is implemented using an 8-layer MLP, a decision we will analyze in Section 4.1. Learned affine transformations then specialize $\mathbf{w}$ to *styles* $\mathbf{y} = (\mathbf{y}_s, \mathbf{y}_b)$ that control adaptive instance normalization (AdaIN) [27, 17, 21, 16] operations after each convolution layer of the synthesis network g. The AdaIN operation is defined as

$$\text{AdaIN}(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}, \quad (1)$$

where each feature map $\mathbf{x}_i$ is normalized separately, and then scaled and biased using the corresponding scalar components from style $\mathbf{y}$. Thus the dimensionality of $\mathbf{y}$ is twice the number of feature maps on that layer.

Comparing our approach to style transfer, we compute the spatially invariant style $\mathbf{y}$ from vector $\mathbf{w}$ instead of an example image. We choose to reuse the word "style" for $\mathbf{y}$ because similar network architectures are already used for feedforward style transfer [27], unsupervised image-to-image translation [28], and domain mixtures [23]. Compared to more general feature transforms [38, 57], AdaIN is particularly well suited for our purposes due to its efficiency and compact representation.

	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	**5.06**	4.42
F	+ Mixing regularization	5.17	**4.40**

Table 1. Fréchet inception distance (FID) for various generator designs (lower is better). In this paper we calculate the FIDs using 50,000 images drawn randomly from the training set, and report the lowest distance encountered over the course of training.

Finally, we provide our generator with a direct means to generate stochastic detail by introducing explicit *noise inputs*. These are single-channel images consisting of uncorrelated Gaussian noise, and we feed a dedicated noise image to each layer of the synthesis network. The noise image is broadcasted to all feature maps using learned per-feature scaling factors and then added to the output of the corresponding convolution, as illustrated in Figure 1b. The implications of adding the noise inputs are discussed in Sections 3.2 and 3.3.

2.1. Quality of generated images

Before studying the properties of our generator, we demonstrate experimentally that the redesign does not compromise image quality but, in fact, improves it considerably. Table 1 gives Fréchet inception distances (FID) [25] for various generator architectures in CELEBA-HQ [30] and our new FFHQ dataset (Appendix A). Results for other datasets are given in Appendix E. Our baseline configuration (A) is the Progressive GAN setup of Karras et al. [30], from which we inherit the networks and all hyperparameters except where stated otherwise. We first switch to an improved baseline (B) by using bilinear up/downsampling operations [64], longer training, and tuned hyperparameters. A detailed description of training setups and hyperparameters is included in Appendix C. We then improve this new baseline further by adding the mapping network and AdaIN operations (C), and make a surprising observation that the network no longer benefits from feeding the latent code into the first convolution layer. We therefore simplify the architecture by removing the traditional input layer and starting the image synthesis from a learned $4 \times 4 \times 512$ constant tensor (D). We find it quite remarkable that the synthesis network is able to produce meaningful results even though it receives input only through the styles that control the AdaIN operations.

Finally, we introduce the noise inputs (E) that improve the results further, as well as novel *mixing regularization* (F) that decorrelates neighboring styles and enables more fine-grained control over the generated imagery (Section 3.1).

We evaluate our methods using two different loss functions: for CELEBA-HQ we rely on WGAN-GP [24],

Figure 2. Uncurated set of images produced by our style-based generator (config F) with the FFHQ dataset. Here we used a variation of the truncation trick [42, 5, 34] with $\psi = 0.7$ for resolutions $4^2 - 32^2$. Please see the accompanying video for more results.

while FFHQ uses WGAN-GP for configuration A and non-saturating loss [22] with R_1 regularization [44, 51, 14] for configurations B–F. We found these choices to give the best results. Our contributions do not modify the loss function.

We observe that the style-based generator (E) improves FIDs quite significantly over the traditional generator (B), almost 20%, corroborating the large-scale ImageNet measurements made in parallel work [6, 5]. Figure 2 shows an uncurated set of novel images generated from the FFHQ dataset using our generator. As confirmed by the FIDs, the average quality is high, and even accessories such as eyeglasses and hats get successfully synthesized. For this figure, we avoided sampling from the extreme regions of $\mathcal{W}$ using the so-called truncation trick [42, 5, 34] — Appendix B details how the trick can be performed in $\mathcal{W}$ instead of $\mathcal{Z}$. Note that our generator allows applying the truncation selectively to low resolutions only, so that high-resolution details are not affected.

All FIDs in this paper are computed without the truncation trick, and we only use it for illustrative purposes in Figure 2 and the video. All images are generated in 1024^2 resolution.

2.2. Prior art

Much of the work on GAN architectures has focused on improving the discriminator by, e.g., using multiple discriminators [18, 47, 11], multiresolution discrimination [60, 55], or self-attention [63]. The work on generator side has mostly focused on the exact distribution in the input latent space [5] or shaping the input latent space via Gaussian mixture models [4], clustering [48], or encouraging convexity [52].

Recent conditional generators feed the class identifier through a separate embedding network to a large number of layers in the generator [46], while the latent is still provided though the input layer. A few authors have considered feeding parts of the latent code to multiple generator layers [9, 5]. In parallel work, Chen et al. [6] "self modulate" the generator using AdaINs, similarly to our work, but do not consider an intermediate latent space or noise inputs.

3. Properties of the style-based generator

Our generator architecture makes it possible to control the image synthesis via scale-specific modifications to the styles. We can view the mapping network and affine transformations as a way to draw samples for each style from a learned distribution, and the synthesis network as a way to generate a novel image based on a collection of styles. The effects of each style are localized in the network, i.e., modifying a specific subset of the styles can be expected to affect only certain aspects of the image.

To see the reason for this localization, let us consider how the AdaIN operation (Eq. 1) first normalizes each channel to zero mean and unit variance, and only then applies scales and biases based on the style. The new per-channel statistics, as dictated by the style, modify the relative importance of features for the subsequent convolution operation, but they do not depend on the original statistics because of the normalization. Thus each style controls only one convolution before being overridden by the next AdaIN operation.

3.1. Style mixing

To further encourage the styles to localize, we employ *mixing regularization*, where a given percentage of images are generated using two random latent codes instead of one during training. When generating such an image, we simply switch from one latent code to another — an operation we refer to as *style mixing* — at a randomly selected point in the synthesis network. To be specific, we run two latent codes $\mathbf{z}_1, \mathbf{z}_2$ through the mapping network, and have the corresponding $\mathbf{w}_1, \mathbf{w}_2$ control the styles so that $\mathbf{w}_1$ applies before the crossover point and $\mathbf{w}_2$ after it. This regularization technique prevents the network from assuming that adjacent styles are correlated.

Table 2 shows how enabling mixing regularization dur-

See below summary for related explanation

Figure 3. Two sets of images were generated from their respective latent codes (sources A and B); the rest of the images were generated by copying a specified subset of styles from source B and taking the rest from source A. Copying the styles corresponding to coarse spatial resolutions ($4^2 - 8^2$) brings high-level aspects such as pose, general hair style, face shape, and eyeglasses from source B, while all colors (eyes, hair, lighting) and finer facial features resemble A. If we instead copy the styles of middle resolutions ($16^2 - 32^2$) from B, we inherit smaller scale facial features, hair style, eyes open/closed from B, while the pose, general face shape, and eyeglasses from A are preserved. Finally, copying the fine styles ($64^2 - 1024^2$) from B brings mainly the color scheme and microstructure.

	Mixing regularization	Number of latents during testing 1	2	3	4
E	0%	4.42	8.22	12.88	17.41
	50%	4.41	6.10	8.71	11.61
F	90%	**4.40**	**5.11**	6.88	9.03
	100%	4.83	5.17	**6.63**	**8.40**

Table 2. FIDs in FFHQ for networks trained by enabling the mixing regularization for different percentage of training examples. Here we stress test the trained networks by randomizing 1 . . . 4 latents and the crossover points between them. Mixing regularization improves the tolerance to these adverse operations significantly. Labels E and F refer to the configurations in Table 1.

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

Figure 4. Examples of stochastic variation. (a) Two generated images. (b) Zoom-in with different realizations of input noise. While the overall appearance is almost identical, individual hairs are placed very differently. (c) Standard deviation of each pixel over 100 different realizations, highlighting which parts of the images are affected by the noise. The main areas are the hair, silhouettes, and parts of background, but there is also interesting stochastic variation in the eye reflections. Global aspects such as identity and pose are unaffected by stochastic variation.

ing training improves the localization considerably, indicated by improved FIDs in scenarios where multiple latents are mixed at test time. Figure 3 presents examples of images synthesized by mixing two latent codes at various scales. We can see that each subset of styles controls meaningful high-level attributes of the image.

3.2. Stochastic variation

There are many aspects in human portraits that can be regarded as stochastic, such as the exact placement of hairs, stubble, freckles, or skin pores. Any of these can be randomized without affecting our perception of the image as long as they follow the correct distribution.

Let us consider how a traditional generator implements stochastic variation. Given that the only input to the network is through the input layer, the network needs to invent a way to generate spatially-varying pseudorandom numbers

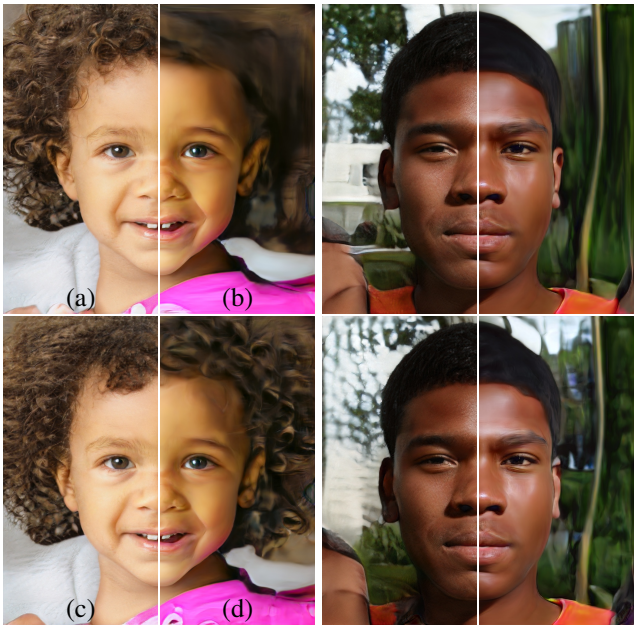
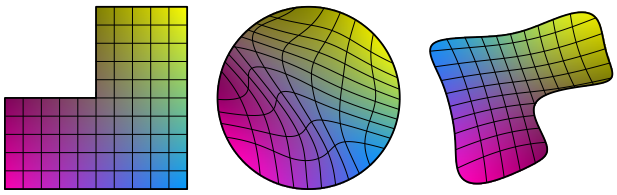


Figure 5. Effect of noise inputs at different layers of our generator. (a) Noise is applied to all layers. (b) No noise. (c) Noise in fine layers only (64^2 – 1024^2). (d) Noise in coarse layers only (4^2 – 32^2). We can see that the artificial omission of noise leads to featureless "painterly" look. Coarse noise causes large-scale curling of hair and appearance of larger background features, while the fine noise brings out the finer curls of hair, finer background detail, and skin pores.

from earlier activations whenever they are needed. This consumes network capacity and hiding the periodicity of generated signal is difficult — and not always successful, as evidenced by commonly seen repetitive patterns in generated images. Our architecture sidesteps these issues altogether by adding per-pixel noise after each convolution.

Figure 4 shows stochastic realizations of the same underlying image, produced using our generator with different noise realizations. We can see that the noise affects only the stochastic aspects, leaving the overall composition and high-level aspects such as identity intact. Figure 5 further illustrates the effect of applying stochastic variation to different subsets of layers. Since these effects are best seen in animation, please consult the accompanying video for a demonstration of how changing the noise input of one layer leads to stochastic variation at a matching scale.

We find it interesting that the effect of noise appears tightly localized in the network. We hypothesize that at any point in the generator, there is pressure to introduce new content as soon as possible, and the easiest way for our network to create stochastic variation is to rely on the noise provided. A fresh set of noise is available for every layer, and thus there is no incentive to generate the stochastic effects from earlier activations, leading to a localized effect.



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

Figure 6. Illustrative example with two factors of variation (image features, e.g., masculinity and hair length). (a) An example training set where some combination (e.g., long haired males) is missing. (b) This forces the mapping from $\mathcal{Z}$ to image features to become curved so that the forbidden combination disappears in $\mathcal{Z}$ to prevent the sampling of invalid combinations. (c) The learned mapping from $\mathcal{Z}$ to $\mathcal{W}$ is able to "undo" much of the warping.

3.3. Separation of global effects from stochasticity

The previous sections as well as the accompanying video demonstrate that while changes to the style have global effects (changing pose, identity, etc.), the noise affects only inconsequential stochastic variation (differently combed hair, beard, etc.). This observation is in line with style transfer literature, where it has been established that spatially invariant statistics (Gram matrix, channel-wise mean, variance, etc.) reliably encode the style of an image [20, 39] while spatially varying features encode a specific instance.

In our style-based generator, the style affects the entire image because complete feature maps are scaled and biased with the same values. Therefore, global effects such as pose, lighting, or background style can be controlled coherently. Meanwhile, the noise is added independently to each pixel and is thus ideally suited for controlling stochastic variation. If the network tried to control, e.g., pose using the noise, that would lead to spatially inconsistent decisions that would then be penalized by the discriminator. Thus the network learns to use the global and local channels appropriately, without explicit guidance.

4. Disentanglement studies

There are various definitions for disentanglement [54, 50, 2, 7, 19], but a common goal is a latent space that consists of linear subspaces, each of which controls one factor of variation. However, the sampling probability of each combination of factors in $\mathcal{Z}$ needs to match the corresponding density in the training data. As illustrated in Figure 6, this precludes the factors from being fully disentangled with typical datasets and input latent distributions.[2]

A major benefit of our generator architecture is that the intermediate latent space $\mathcal{W}$ does not have to support sampling according to any *fixed* distribution; its sampling density is induced by the *learned* piecewise continuous mapping $f(\mathbf{z})$. This mapping can be adapted to "unwarp" $\mathcal{W}$ so that the factors of variation become more linear. We posit that there is pressure for the generator to do so, as it should be easier to generate realistic images based on a disentangled representation than based on an entangled representation. As such, we expect the training to yield a less entangled $\mathcal{W}$ in an unsupervised setting, i.e., when the factors of variation are not known in advance [10, 35, 49, 8, 26, 32, 7].

Unfortunately the metrics recently proposed for quantifying disentanglement [26, 32, 7, 19] require an encoder network that maps input images to latent codes. These metrics are ill-suited for our purposes since our baseline GAN lacks such an encoder. While it is possible to add an extra network for this purpose [8, 12, 15], we want to avoid investing effort into a component that is not a part of the actual solution. To this end, we describe two new ways of quantifying disentanglement, neither of which requires an encoder or known factors of variation, and are therefore computable for any image dataset and generator.

4.1. Perceptual path length

As noted by Laine [37], interpolation of latent-space vectors may yield surprisingly non-linear changes in the image. For example, features that are absent in either endpoint may appear in the middle of a linear interpolation path. This is a sign that the latent space is entangled and the factors of variation are not properly separated. To quantify this effect, we can measure how drastic changes the image undergoes as we perform interpolation in the latent space. Intuitively, a less curved latent space should result in perceptually smoother transition than a highly curved latent space.

As a basis for our metric, we use a perceptually-based pairwise image distance [65] that is calculated as a weighted difference between two VGG16 [58] embeddings, where the weights are fit so that the metric agrees with human perceptual similarity judgments. If we subdivide a latent space interpolation path into linear segments, we can define the total perceptual length of this segmented path as the sum of perceptual differences over each segment, as reported by the image distance metric. A natural definition for the perceptual path length would be the limit of this sum under infinitely fine subdivision, but in practice we approximate it using a small subdivision epsilon $\epsilon = 10^{-4}$. The average perceptual path length in latent space $\mathcal{Z}$, over all possible endpoints, is therefore

$$l_{\mathcal{Z}} = \mathbb{E}\Big[\frac{1}{\epsilon^2} d\big(G(\mathrm{slerp}(\mathbf{z}_1, \mathbf{z}_2;\, t)),\; G(\mathrm{slerp}(\mathbf{z}_1, \mathbf{z}_2;\, t+\epsilon))\big)\Big], \tag{2}$$

where $\mathbf{z}_1, \mathbf{z}_2 \sim P(\mathbf{z}), t \sim U(0,1)$, G is the generator (i.e., $g \circ f$ for style-based networks), and $d(\cdot,\cdot)$ evaluates the per-

[2] The few artificial datasets designed for disentanglement studies (e.g., [43, 19]) tabulate all combinations of predetermined factors of variation with uniform frequency, thus hiding the problem.

	Method	Path length full	Path length end	Separability
B	Traditional generator $\mathcal{Z}$	412.0	415.3	10.78
D	Style-based generator $\mathcal{W}$	446.2	376.6	3.61
E	+ Add noise inputs $\mathcal{W}$	**200.5**	**160.6**	3.54
	+ Mixing 50% $\mathcal{W}$	231.5	182.1	**3.51**
F	+ Mixing 90% $\mathcal{W}$	234.0	195.9	3.79

Table 3. Perceptual path lengths and separability scores for various generator architectures in FFHQ (lower is better). We perform the measurements in $\mathcal{Z}$ for the traditional network, and in $\mathcal{W}$ for style-based ones. Making the network resistant to style mixing appears to distort the intermediate latent space $\mathcal{W}$ somewhat. We hypothesize that mixing makes it more difficult for $\mathcal{W}$ to efficiently encode factors of variation that span multiple scales.

ceptual distance between the resulting images. Here slerp denotes spherical interpolation [56], which is the most appropriate way of interpolating in our normalized input latent space [61]. To concentrate on the facial features instead of background, we crop the generated images to contain only the face prior to evaluating the pairwise image metric. As the metric d is quadratic [65], we divide by ϵ^2. We compute the expectation by taking 100,000 samples.

Computing the average perceptual path length in $\mathcal{W}$ is carried out in a similar fashion:

$$l_{\mathcal{W}} = \mathbb{E}\Big[\frac{1}{\epsilon^2} d\big(g(\mathrm{lerp}(f(\mathbf{z}_1), f(\mathbf{z}_2); t)), g(\mathrm{lerp}(f(\mathbf{z}_1), f(\mathbf{z}_2); t+\epsilon))\big)\Big], \quad (3)$$

where the only difference is that interpolation happens in $\mathcal{W}$ space. Because vectors in $\mathcal{W}$ are not normalized in any fashion, we use linear interpolation (lerp).

Table 3 shows that this full-path length is substantially shorter for our style-based generator with noise inputs, indicating that $\mathcal{W}$ is perceptually more linear than $\mathcal{Z}$. Yet, this measurement is in fact slightly biased in favor of the input latent space $\mathcal{Z}$. If $\mathcal{W}$ is indeed a disentangled and "flattened" mapping of $\mathcal{Z}$, it may contain regions that are not on the input manifold — and are thus badly reconstructed by the generator — even between points that are mapped from the input manifold, whereas the input latent space $\mathcal{Z}$ has no such regions by definition. It is therefore to be expected that if we restrict our measure to path endpoints, i.e., $t \in \{0, 1\}$, we should obtain a smaller $l_{\mathcal{W}}$ while $l_{\mathcal{Z}}$ is not affected. This is indeed what we observe in Table 3.

Table 4 shows how path lengths are affected by the mapping network. We see that both traditional and style-based generators benefit from having a mapping network, and additional depth generally improves the perceptual path length as well as FIDs. It is interesting that while $l_{\mathcal{W}}$ improves in the traditional generator, $l_{\mathcal{Z}}$ becomes considerably worse, illustrating our claim that the input latent space can indeed be arbitrarily entangled in GANs.

	Method	FID	Path length full	Path length end	Separability
B	Traditional 0 $\mathcal{Z}$	5.25	412.0	415.3	10.78
	Traditional 8 $\mathcal{Z}$	4.87	896.2	902.0	170.29
	Traditional 8 $\mathcal{W}$	4.87	324.5	212.2	6.52
	Style-based 0 $\mathcal{Z}$	5.06	283.5	285.5	9.88
	Style-based 1 $\mathcal{W}$	4.60	219.9	209.4	6.81
	Style-based 2 $\mathcal{W}$	4.43	**217.8**	199.9	6.25
F	Style-based 8 $\mathcal{W}$	**4.40**	234.0	**195.9**	**3.79**

Table 4. The effect of a mapping network in FFHQ. The number in method name indicates the depth of the mapping network. We see that FID, separability, and path length all benefit from having a mapping network, and this holds for both style-based and traditional generator architectures. Furthermore, a deeper mapping network generally performs better than a shallow one.

4.2. Linear separability

If a latent space is sufficiently disentangled, it should be possible to find direction vectors that consistently correspond to individual factors of variation. We propose another metric that quantifies this effect by measuring how well the latent-space points can be separated into two distinct sets via a linear hyperplane, so that each set corresponds to a specific binary attribute of the image.

In order to label the generated images, we train auxiliary classification networks for a number of binary attributes, e.g., to distinguish male and female faces. In our tests, the classifiers had the same architecture as the discriminator we use (i.e., same as in [30]), and were trained using the CELEBA-HQ dataset that retains the 40 attributes available in the original CelebA dataset. To measure the separability of one attribute, we generate 200,000 images with $\mathbf{z} \sim P(\mathbf{z})$ and classify them using the auxiliary classification network. We then sort the samples according to classifier confidence and remove the least confident half, yielding 100,000 labeled latent-space vectors.

For each attribute, we fit a linear SVM to predict the label based on the latent-space point — $\mathbf{z}$ for traditional and $\mathbf{w}$ for style-based — and classify the points by this plane. We then compute the conditional entropy $\mathrm{H}(Y|X)$ where X are the classes predicted by the SVM and Y are the classes determined by the pre-trained classifier. This tells how much additional information is required to determine the true class of a sample, given that we know on which side of the hyperplane it lies. A low value suggests consistent latent space directions for the corresponding factor(s) of variation.

We calculate the final separability score as $\exp(\sum_i \mathrm{H}(Y_i|X_i))$, where i enumerates the 40 attributes. Similar to the inception score [53], the exponentiation brings the values from logarithmic to linear domain so that they are easier to compare.

Tables 3 and 4 show that $\mathcal{W}$ is consistently better separable than $\mathcal{Z}$, suggesting a less entangled representation.

Figure 7. The FFHQ dataset offers a lot of variety in terms of age, ethnicity, viewpoint, lighting, and image background.

Furthermore, increasing the depth of the mapping network improves both image quality and separability in $\mathcal{W}$, which is in line with the hypothesis that the synthesis network inherently favors a disentangled input representation. Interestingly, adding a mapping network in front of a traditional generator results in severe loss of separability in $\mathcal{Z}$ but improves the situation in the intermediate latent space $\mathcal{W}$, and the FID improves as well. This shows that even the traditional generator architecture performs better when we introduce an intermediate latent space that does not have to follow the distribution of the training data.

5. Conclusion

Based on both our results and parallel work by Chen et al. [6], it is becoming clear that the traditional GAN generator architecture is in every way inferior to a style-based design. This is true in terms of established quality metrics, and we further believe that our investigations to the separation of high-level attributes and stochastic effects, as well as the linearity of the intermediate latent space will prove fruitful in improving the understanding and controllability of GAN synthesis.

We note that our average path length metric could easily be used as a regularizer during training, and perhaps some variant of the linear separability metric could act as one, too. In general, we expect that methods for directly shaping the intermediate latent space during training will provide interesting avenues for future work.

6. Acknowledgements

We thank Jaakko Lehtinen, David Luebke, and Tuomas Kynkäänniemi for in-depth discussions and helpful comments; Janne Hellsten, Tero Kuosmanen, and Pekka Jänis for compute infrastructure and help with the code release.

A. The FFHQ dataset

We have collected a new dataset of human faces, Flickr-Faces-HQ (FFHQ), consisting of 70,000 high-quality images at 1024^2 resolution (Figure 7). The dataset includes vastly more variation than CELEBA-HQ [30] in terms of age, ethnicity and image background, and also has much better coverage of accessories such as eyeglasses, sunglasses, hats, etc. The images were crawled from Flickr (thus inheriting all the biases of that website) and automatically aligned [31] and cropped. Only images under permissive licenses were collected. Various automatic filters were used to prune the set, and finally Mechanical Turk allowed us to remove the occasional statues, paintings, or photos of photos. We have made the dataset publicly available at `https://github.com/NVlabs/ffhq-dataset`

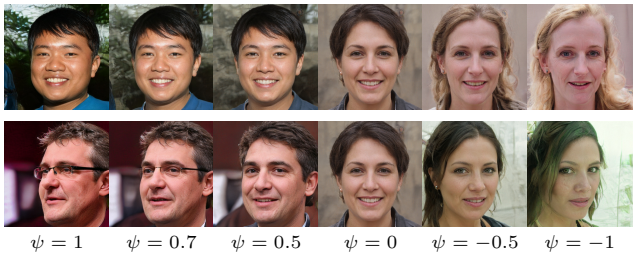


Figure 8. The effect of truncation trick as a function of style scale ψ. When we fade $\psi \to 0$, all faces converge to the "mean" face of FFHQ. This face is similar for all trained networks, and the interpolation towards it never seems to cause artifacts. By applying negative scaling to styles, we get the corresponding opposite or "anti-face". It is interesting that various high-level attributes often flip between the opposites, including viewpoint, glasses, age, coloring, hair length, and often gender.

B. Truncation trick in $\mathcal{W}$

If we consider the distribution of training data, it is clear that areas of low density are poorly represented and thus likely to be difficult for the generator to learn. This is a significant open problem in all generative modeling techniques. However, it is known that drawing latent vectors from a truncated [42, 5] or otherwise shrunk [34] sampling space tends to improve average image quality, although some amount of variation is lost.

We can follow a similar strategy. To begin, we compute the center of mass of $\mathcal{W}$ as $\bar{\mathbf{w}} = \mathbb{E}_{\mathbf{z} \sim P(\mathbf{z})}[f(\mathbf{z})]$. In case of FFHQ this point represents a sort of an average face (Figure 8, $\psi = 0$). We can then scale the deviation of a given $\mathbf{w}$ from the center as $\mathbf{w}' = \bar{\mathbf{w}} + \psi(\mathbf{w} - \bar{\mathbf{w}})$, where $\psi < 1$. While Brock et al. [5] observe that only a subset of networks is amenable to such truncation even when orthogonal regularization is used, truncation in $\mathcal{W}$ space seems to work reliably even without changes to the loss function.

C. Hyperparameters and training details

We build upon the official TensorFlow [1] implementation of Progressive GANs by Karras et al. [30], from which we inherit most of the training details.[3] This original setup corresponds to configuration A in Table 1. In particular, we use the same discriminator architecture, resolution-dependent minibatch sizes, Adam [33] hyperparameters, and exponential moving average of the generator. We enable mirror augmentation for CelebA-HQ and FFHQ, but disable it for LSUN. Our training time is approximately one week on an NVIDIA DGX-1 with 8 Tesla V100 GPUs.

For our improved baseline (B in Table 1), we make several modifications to improve the overall result quality. We replace the nearest-neighbor up/downsampling in both networks with bilinear sampling, which we implement by low-pass filtering the activations with a separable 2^{nd} order binomial filter after each upsampling layer and before each downsampling layer [64]. We implement progressive growing the same way as Karras et al. [30], but we start from 8^2 images instead of 4^2. For the FFHQ dataset, we switch from WGAN-GP to the non-saturating loss [22] with R_1 regularization [44] using $\gamma = 10$. With R_1 we found that the FID scores keep decreasing for considerably longer than with WGAN-GP, and we thus increase the training time from 12M to 25M images. We use the same learning rates as Karras et al. [30] for FFHQ, but we found that setting the learning rate to 0.002 instead of 0.003 for 512^2 and 1024^2 leads to better stability with CelebA-HQ.

For our style-based generator (F in Table 1), we use leaky ReLU [41] with $\alpha = 0.2$ and equalized learning rate [30] for all layers. We use the same feature map counts in our convolution layers as Karras et al. [30]. Our mapping network consists of 8 fully-connected layers, and the dimensionality of all input and output activations — including $\mathbf{z}$ and $\mathbf{w}$ — is 512. We found that increasing the depth of the mapping network tends to make the training unstable with high learning rates. We thus reduce the learning rate by two orders of magnitude for the mapping network, i.e., $\lambda' = 0.01 \cdot \lambda$. We initialize all weights of the convolutional, fully-connected, and affine transform layers using $\mathcal{N}(0, 1)$. The constant input in synthesis network is initialized to one. The biases and noise scaling factors are initialized to zero, except for the biases associated with $\mathbf{y}_s$ that we initialize to one.

The classifiers used by our separability metric (Section 4.2) have the same architecture as our discriminator except that minibatch standard deviation [30] is disabled. We use the learning rate of 10^{-3}, minibatch size of 8, Adam optimizer, and training length of 150,000 images. The classifiers are trained independently of generators, and the same 40 classifiers, one for each CelebA attribute, are used for measuring the separability metric for all generators. We will release the pre-trained classifier networks so that our measurements can be reproduced.

We do not use batch normalization [29], spectral normalization [45], attention mechanisms [63], dropout [59], or pixelwise feature vector normalization [30] in our networks.

[3] https://github.com/tkarras/progressive_growing_of_gans

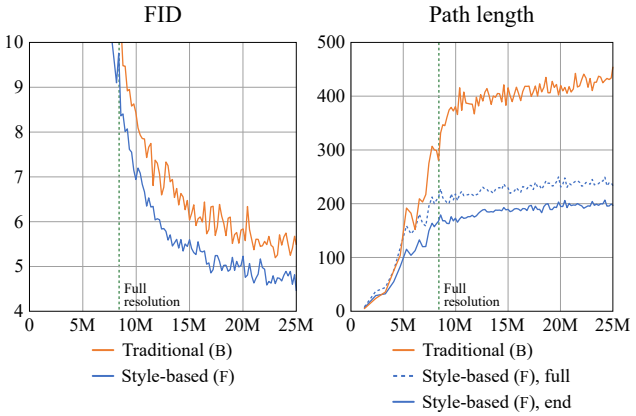


Figure 9. FID and perceptual path length metrics over the course of training in our configurations B and F using the FFHQ dataset. Horizontal axis denotes the number of training images seen by the discriminator. The dashed vertical line at 8.4M images marks the point when training has progressed to full 1024^2 resolution. On the right, we show only one curve for the traditional generator's path length measurements, because there is no discernible difference between full-path and endpoint sampling in $\mathcal{Z}$.

D. Training convergence

Figure 9 shows how the FID and perceptual path length metrics evolve during the training of our configurations B and F with the FFHQ dataset. With R_1 regularization active in both configurations, FID continues to slowly decrease as the training progresses, motivating our choice to increase the training time from 12M images to 25M images. Even when the training has reached the full 1024^2 resolution, the slowly rising path lengths indicate that the improvements in FID come at the cost of a more entangled representation. Considering future work, it is an interesting question whether this is unavoidable, or if it were possible to encourage shorter path lengths without compromising the convergence of FID.

E. Other datasets

Figures 10, 11, and 12 show an uncurated set of results for LSUN [62] BEDROOM, CARS, and CATS, respectively. In these images we used the truncation trick from Appendix Bwith $\psi = 0.7$ for resolutions $4^2 - 32^2$. The accompanying video provides results for style mixing and stochastic variation tests. As can be seen therein, in case of

Figure 10. Uncurated set of images produced by our style-based generator (config F) with the LSUN BEDROOM dataset at 256^2. FID computed for 50K images was 2.65.

Figure 11. Uncurated set of images produced by our style-based generator (config F) with the LSUN CAR dataset at 512×384. FID computed for 50K images was 3.27.

BEDROOM the coarse styles basically control the viewpoint of the camera, middle styles select the particular furniture, and fine styles deal with colors and smaller details of materials. In CARS the effects are roughly similar. Stochastic variation affects primarily the fabrics in BEDROOM, backgrounds and headlamps in CARS, and fur, background, and interestingly, the positioning of paws in CATS. Somewhat surprisingly the wheels of a car never seem to rotate based on stochastic inputs.

These datasets were trained using the same setup as FFHQ for the duration of 70M images for BEDROOM and CATS, and 46M for CARS. We suspect that the results for BEDROOM are starting to approach the limits of the training data, as in many images the most objectionable issues are the severe compression artifacts that have been inherited from the low-quality training data. CARS has much higher quality training data that also allows higher spatial resolution (512×384 instead of 256^2), and CATS continues to be a difficult dataset due to the high intrinsic variation in poses, zoom levels, and backgrounds.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: a system for large-scale machine learning. In *Proc. 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16, pages 265–283, 2016. 9

[2] A. Achille and S. Soatto. On the emergence of invariance and disentangling in deep representations. *CoRR*, abs/1706.01350, 2017. 6

[3] D. Bau, J. Zhu, H. Strobelt, B. Zhou, J. B. Tenenbaum, W. T. Freeman, and A. Torralba. GAN dissection: Visualizing and understanding generative adversarial networks. In *Proc. ICLR*, 2019. 1

[4] M. Ben-Yosef and D. Weinshall. Gaussian mixture generative adversarial networks for diverse datasets, and the unsupervised clustering of images. *CoRR*, abs/1808.10356, 2018. 3

[5] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018. 1, 3, 8

Figure 12. Uncurated set of images produced by our style-based generator (config F) with the LSUN CAT dataset at 256^2. FID computed for 50K images was 8.53.

[6] T. Chen, M. Lucic, N. Houlsby, and S. Gelly. On self modulation for generative adversarial networks. *CoRR*, abs/1810.01365, 2018. 3, 8

[7] T. Q. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud. Isolating sources of disentanglement in variational autoencoders. *CoRR*, abs/1802.04942, 2018. 6

[8] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. InfoGAN: interpretable representation learning by information maximizing generative adversarial nets. *CoRR*, abs/1606.03657, 2016. 6

[9] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a Laplacian pyramid of adversarial networks. *CoRR*, abs/1506.05751, 2015. 3

[10] G. Desjardins, A. Courville, and Y. Bengio. Disentangling factors of variation via generative entangling. *CoRR*, abs/1210.5474, 2012. 6

[11] T. Doan, J. Monteiro, I. Albuquerque, B. Mazoure, A. Durand, J. Pineau, and R. D. Hjelm. Online adaptative curriculum learning for GANs. *CoRR*, abs/1808.00020, 2018. 3

[12] J. Donahue, P. Krähenbühl, and T. Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016. 6

[13] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. *CoRR*, abs/1411.5928, 2014. 1

[14] H. Drucker and Y. L. Cun. Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, 3(6):991–997, 1992. 3

[15] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville. Adversarially learned inference. In *Proc. ICLR*, 2017. 6

[16] V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. d. Vries, A. Courville, and Y. Bengio. Feature-wise transformations. *Distill*, 2018. https://distill.pub/2018/feature-wise-transformations. 2

[17] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. *CoRR*, abs/1610.07629, 2016. 2

[18] I. P. Durugkar, I. Gemp, and S. Mahadevan. Generative multi-adversarial networks. *CoRR*, abs/1611.01673, 2016. 3

[19] C. Eastwood and C. K. I. Williams. A framework for the quantitative evaluation of disentangled representations. In *Proc. ICLR*, 2018. 6

[20] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proc. CVPR*, 2016. 6

[21] G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *CoRR*, abs/1705.06830, 2017. 2

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. In *NIPS*, 2014. 1, 3, 9

[23] W.-S. Z. Guang-Yuan Hao, Hong-Xing Yu. MIXGAN: learning concepts from different domains for mixture generation. *CoRR*, abs/1807.01659, 2018. 2

[24] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of Wasserstein GANs. *CoRR*, abs/1704.00028, 2017. 1, 2

[25] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Proc. NIPS*, pages 6626–6637, 2017. 2

[26] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *Proc. ICLR*, 2017. 6

[27] X. Huang and S. J. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. *CoRR*, abs/1703.06868, 2017. 1, 2

[28] X. Huang, M. Liu, S. J. Belongie, and J. Kautz. Multimodal unsupervised image-to-image translation. *CoRR*, abs/1804.04732, 2018. 2

[29] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. 9

[30] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017. 1, 2, 7, 8, 9

[31] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Proc. CVPR*, 2014. 8

[32] H. Kim and A. Mnih. Disentangling by factorising. In *Proc. ICML*, 2018. 6

[33] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 9

[34] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *CoRR*, abs/1807.03039, 2018. 3, 8

[35] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 6

[36] K. Kurach, M. Lucic, X. Zhai, M. Michalski, and S. Gelly. The gan landscape: Losses, architectures, regularization, and normalization. *CoRR*, abs/1807.04720, 2018. 1

[37] S. Laine. Feature-based metrics for exploring the latent space of generative models. *ICLR workshop poster*, 2018. 1, 6

[38] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang. Universal style transfer via feature transforms. In *Proc. NIPS*, 2017. 2

[39] Y. Li, N. Wang, J. Liu, and X. Hou. Demystifying neural style transfer. *CoRR*, abs/1701.01036, 2017. 6

[40] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet. Are GANs created equal? a large-scale study. *CoRR*, abs/1711.10337, 2017. 1

[41] A. L. Maas, A. Y. Hannun, and A. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. International Conference on Machine Learning (ICML)*, volume 30, 2013. 9

[42] M. Marchesi. Megapixel size image creation using generative adversarial networks. *CoRR*, abs/1706.00082, 2017. 3, 8

[43] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017. 6

[44] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for GANs do actually converge? *CoRR*, abs/1801.04406, 2018. 1, 3, 9

[45] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018. 1, 9

[46] T. Miyato and M. Koyama. cGANs with projection discriminator. *CoRR*, abs/1802.05637, 2018. 3

[47] G. Mordido, H. Yang, and C. Meinel. Dropout-gan: Learning from a dynamic ensemble of discriminators. *CoRR*, abs/1807.11346, 2018. 3

[48] S. Mukherjee, H. Asnani, E. Lin, and S. Kannan. ClusterGAN : Latent space clustering in generative adversarial networks. *CoRR*, abs/1809.03627, 2018. 3

[49] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. ICML*, 2014. 6

[50] K. Ridgeway. A survey of inductive biases for factorial representation-learning. *CoRR*, abs/1612.05299, 2016. 6

[51] A. S. Ross and F. Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. *CoRR*, abs/1711.09404, 2017. 3

[52] T. Sainburg, M. Thielk, B. Theilman, B. Migliori, and T. Gentner. Generative adversarial interpolative autoencoding: adversarial training on latent space interpolations encourage convex latent distributions. *CoRR*, abs/1807.06650, 2018. 1, 3

[53] T. Salimans, I. J. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NIPS*, 2016. 7

[54] J. Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*, 4(6):863–879, 1992. 6

[55] R. Sharma, S. Barratt, S. Ermon, and V. Pande. Improved training with curriculum gans. *CoRR*, abs/1807.09295, 2018. 3

[56] K. Shoemake. Animating rotation with quaternion curves. In *Proc. SIGGRAPH '85*, 1985. 7

[57] A. Siarohin, E. Sangineto, and N. Sebe. Whitening and coloring transform for GANs. *CoRR*, abs/1806.00420, 2018. 2

[58] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 6

[59] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 9

[60] T. Wang, M. Liu, J. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. *CoRR*, abs/1711.11585, 2017. 3

[61] T. White. Sampling generative networks: Notes on a few effective techniques. *CoRR*, abs/1609.04468, 2016. 7

[62] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao. LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop. *CoRR*, abs/1506.03365, 2015. 9

[63] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. *CoRR*, abs/1805.08318, 2018. 3, 9

[64] R. Zhang. Making convolutional networks shift-invariant again, 2019. 2, 9

[65] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. CVPR*, 2018. 6, 7

StyleGAN Summary

Prepared By: Sushant Gautam
Part of 30 Day GAN Paper Reading
https://github.com/sushant097/30-Days-GANs-Paper-Reading

Problem: Lack of control over synthesized Images. Eg: ProGAN (2017)

Solution: Control Style Using New Generator Model. This is the main motivation of StyleGAN.

StyleGAN is an extension of the progressive growing GAN architecture to give control over the disentangled style properties of generated images.

ProGAN + Style Control => StyleGAN

StyleGAN mainly improves the existing architecture of generator but Discriminator network and loss functions remain same.

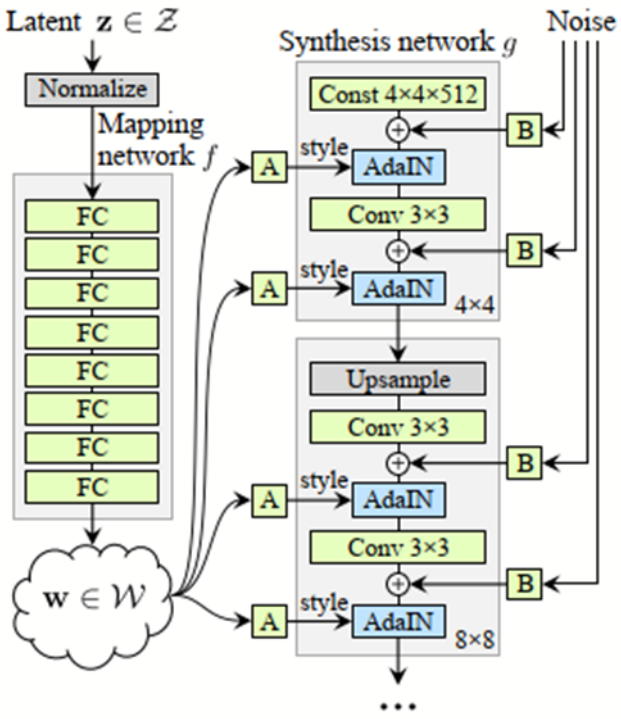
NO Latent space Input to Generator

Input from two new sources of randomness used to generate a synthetic image: a mapping network (8 layers of FCN) and noise input.

The purpose of mapping network convert latent input to the style vectors through non-linear transformation and styles that is integrated at each point in the generator model via a new layer called adaptive instance normalization(AdaIN). The use of this style vector gives control over the style of the generated image.

Noise supplied at each stage in the generator model introduces stochastic variance. The noise is introduced to whole feature maps, allowing the model to comprehend the style at the pixel level.

Style GAN Model Architecture



Latent input (z) is normalized as send to Mapping network(f) - 8 layers of dense layer, which perform non-linear transformation and convert to w.

As, separately learned affine operation A to transform latent vector z in each layer. This operation A generates a separate style y[ys, yb] (these both are scalars) from w which is applied to each feature map when performing the AdaIN.

In the AdaIN operation, each feature map is normalized first and then scale(ys) + bias(yb) is applied to place the respective style information to feature maps after Conv layers.

Scaling — Normalization — Bias Add

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

At a given degree of detail, the noise (B) noise is utilized to introduce style-level variance.

Each Conv layer in the synthesis network is made up of a block of activation maps, each of which has Gaussian noise added to it before the AdaIN process.

For each block, a separate sample of noise is generated and interpreted using per-layer scaling factors.

Important changes in StyleGAN:

Prepared By: Sushant Gautam
Part of 30 Day GAN Paper Reading
https://github.com/sushant097/30-Days-GANs-Paper-Reading

1. Addition of tuning and bilinear upsampling:

   The progressive GAN uses nearest interpolation for upsampling and Average pooling for downsampling. In StyleGAN, bilinear sampling is used for up/downsampling in both networks.

2. Mapping Network and AdaIN:

   8 layers fully connected layers as a standalone mapping network which converts randomly sampled point from the latent space as input and generates a style vector.

3. Addition of Noise:

   The random noise supplied to the input vector adds a slight feature variation that helps the fakes appear more authentic.
   Before the AdaIN procedures, each of the activation maps is given a Gaussian noise (represented by B). For each block, a separate sample of noise is created and interpreted using the layer's scaling factors.

4. Mixing Regularization:

   Using the intermediate vector at each level of the synthesis network may lead the network to learn correlation between levels. This association must be erased, and this model chooses two input vectors (z1 and z2) at random and creates the intermediate vectors (w1 and w2) for them. It then trains part of the levels using the first and switches to the other (at a random split point) to train the remaining levels. This switch in random split points guarantees that the network does not learn significant correlation and delivers a variety of outputs.

5. Remove of latent point Input -> Constant Input:

   Generator network no longer takes a point from the latent space as input but relies on a learned constant 4x4x512 value input to start the image synthesis process.