Alias-Free Generative Adversarial Networks

Tero Karras et al.

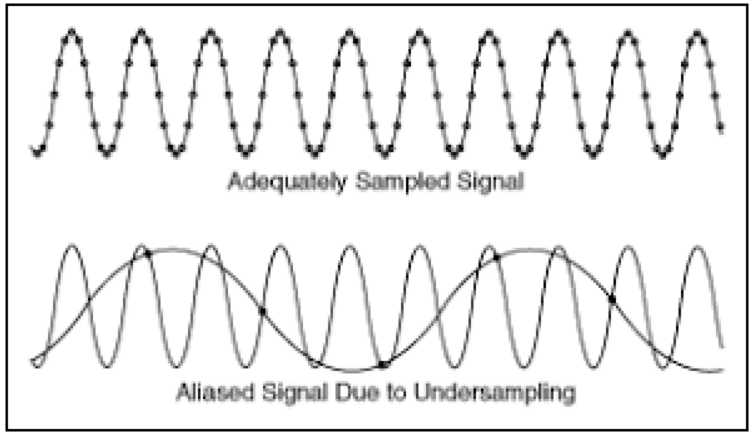
NVIDIA and Aalto University

NeurIPS 2021

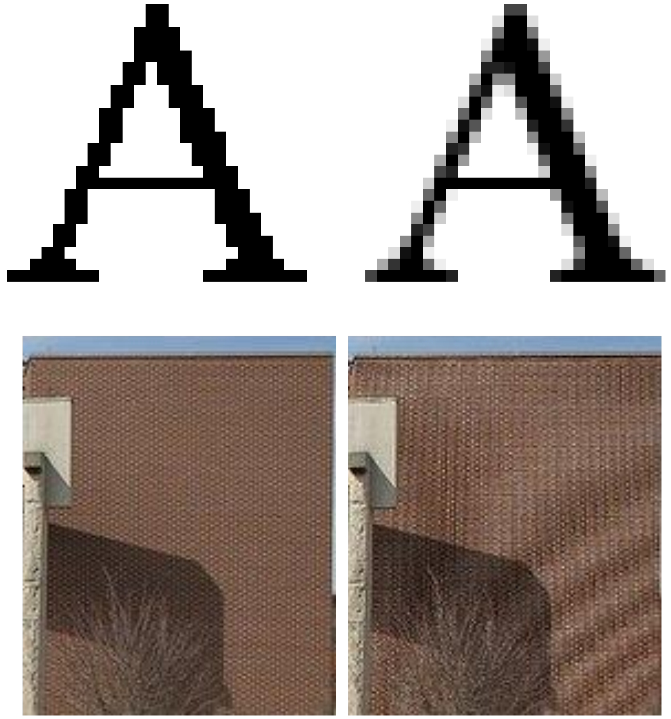
Presented by Minho Park

Motivation

- Aliasing is an effect that causes different signals to become indistinguishable (or aliases of one anot her) when sampled.


Adequately Sampled Signal
Aliased Signal Due to Undersampling

Definition of aliasing in signal processing



Various aliased examples

Motivation

- Current networks can bypass the ideal hierarchical construction by drawing on **unintentional positi onal reference** (e.g., image borders, per-pixel noise inputs, positional encodings, and aliasing).

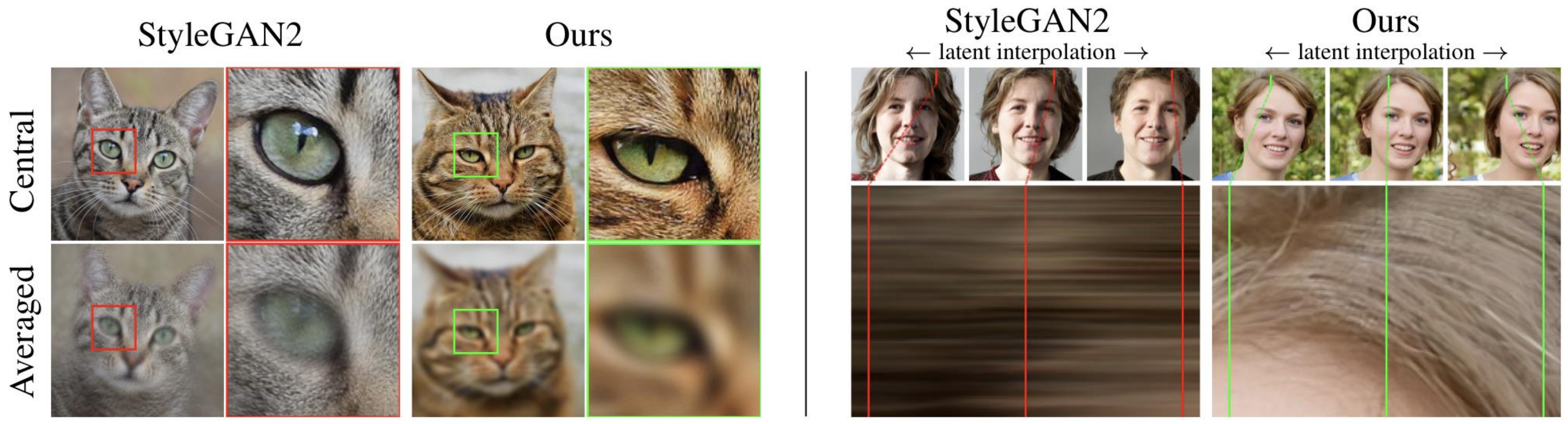


Figure 1: Examples of "texture sticking". **Left:** The average of images generated from a small neighborhood around a central latent (top row). The intended result is uniformly blurry because all details should move together. However, with StyleGAN2 many details (e.g., fur) stick to the same pixel coordinates, showing unwanted sharpness. **Right:** From a latent space interpolation (top row), we extract a short vertical segment of pixels from each generated image and stack them horizontally (bottom). The desired result is hairs moving in animation, creating a time-varying field. With StyleGAN2 the hairs mostly stick to the same coordinates, creating horizontal streaks instead.

More video examples: https://nvlabs.github.io/stylegan3/

What Causes Aliasing in GANs?

1. Faint after-images of the pixel grid resulting from non-ideal upsampling filters such as nearest, bilinear, or strided convolutions.
2. The point-wise application of non-linearities such as ReLU or swish.

- We will make alias-free generator!

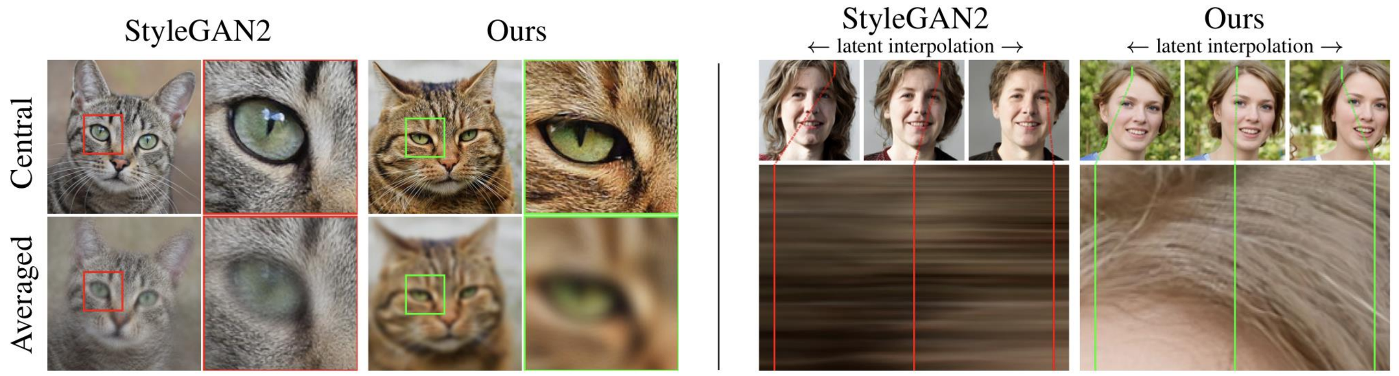


Figure 1: Examples of "texture sticking". **Left:** The average of images generated from a small neighborhood around a central latent (top row). The intended result is uniformly blurry because all details should move together. However, with StyleGAN2 many details (e.g., fur) stick to the same pixel coordinates, showing unwanted sharpness. **Right:** From a latent space interpolation (top row), we extract a short vertical segment of pixels from each generated image and stack them horizontally (bottom). The desired result is hairs moving in animation, creating a time-varying field. With StyleGAN2 the hairs mostly stick to the same coordinates, creating horizontal streaks instead.

More video examples: https://nvlabs.github.io/stylegan3/

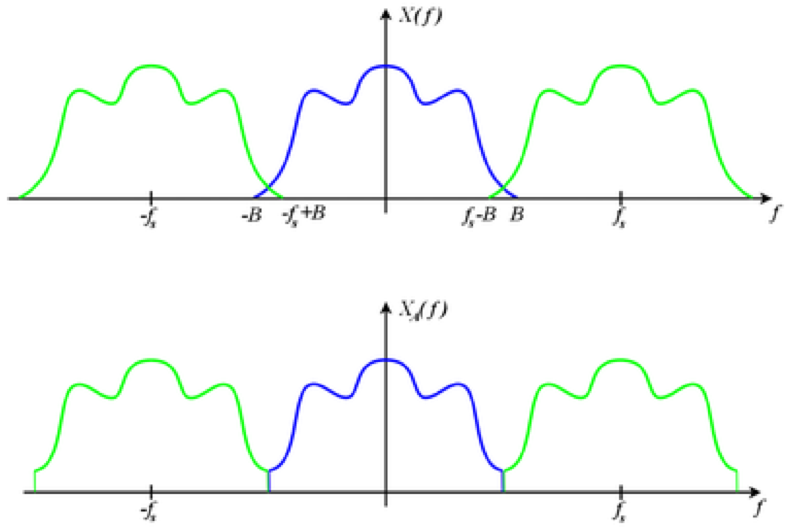
Why We Need Alias-free Generator?

1. **Equivariance generator**
    - Successful elimination of all sources of positional reference.
    - ⇒ Details can be generated equally well regardless of pixel coordinates.
    - ⇒ Equivalent to enforcing continuous equivariance to sub-pixel translation (and optionally rotation) in all layer.
2. **Consistent video generation.**

Equivariance Via Continuous Signal Interpretation

Nyquist-Shannon sampling theorem.

- Nyquist-Shannon sampling theorem.
  - A regularly sampled signal can represent any continuous signal containing frequencies between zero and half of the sampling rate.



Visualize Nyquist-Shannon sampling theorem.

https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem

Whittaker-Shannon interpolation formula

- Given $Z[x]$ and s.
- The corresponding continuous representation $z(x)$ is obtained by convolving the discretely sampled Dirac grid $Z[x]$ with an ideal interpolation filter ϕ_s, i.e., $z(x) = (\phi_s * Z)(x)$.
  - $*$: continuous convolution
  - $\phi_s(x) = \mathrm{sinc}(sx_0) \cdot \mathrm{sinc}(sx_1)$.

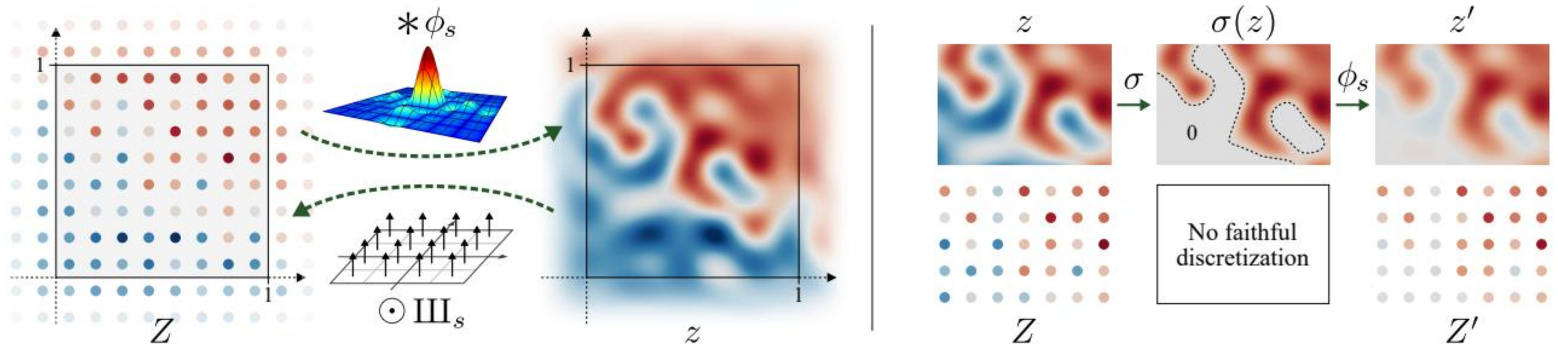


Figure 2: **Left:** Discrete representation Z and continuous representation z are related to each other via convolution with ideal interpolation filter ϕ_s and pointwise multiplication with Dirac comb $Ш_s$. **Right:** Nonlinearity σ, ReLU in this example, may produce arbitrarily high frequencies in the continuous-domain $\sigma(z)$. Low-pass filtering via ϕ_s is necessary to ensure that Z' captures the result.

Discrete and continuous representation of network layers

- Operating F on a discrete feature map $Z' = F(Z)$.
- Operating f on a discrete feature map $z' = f(z)$.
- $f(z) = \phi_{s'} * F(\text{III}_s \odot z)$
- $F(Z) = \text{III}_{s'} \odot f(\phi_s * Z)$

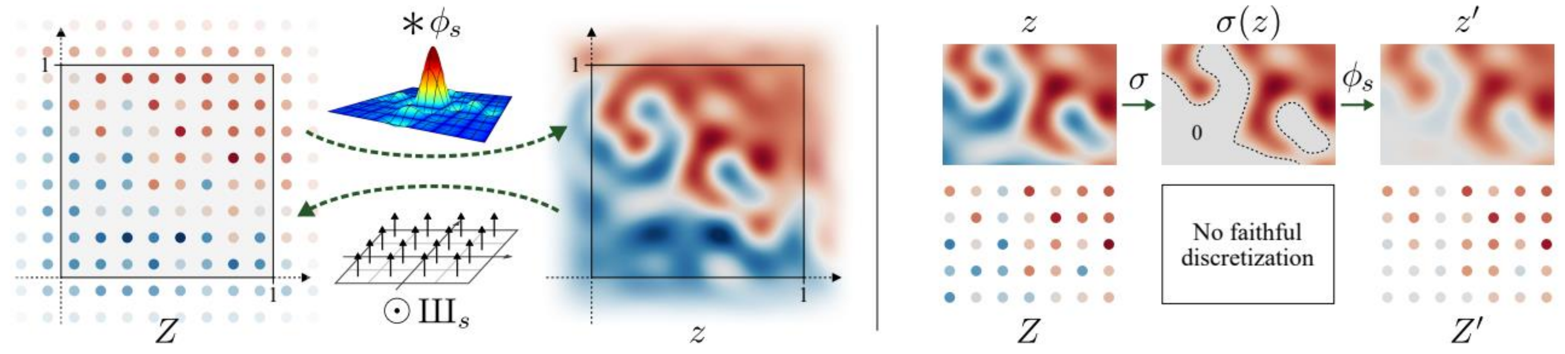


Figure 2: **Left:** Discrete representation Z and continuous representation z are related to each other via convolution with ideal interpolation filter ϕ_s and pointwise multiplication with Dirac comb III_s. **Right:** Nonlinearity σ, ReLU in this example, may produce arbitrarily high frequencies in the continuous-domain $\sigma(z)$. Low-pass filtering via ϕ_s is necessary to ensure that Z' captures the result.

Equivariant network layers

Equivariant Network Layers

- Definition:
  - Operation f is equivariant with respect to a spatial transformation t of the 2D plane if it commutes with it in the continuous domain: $t \circ f = f \circ t$.
  - If inputs are not bandlimited to $s/2$, an equivariant operation must not generate frequency content above the output bandlimit of $s'/2$, as otherwise no faithful discrete output representation exists.
- We focus on translation and rotation equivariance.
- Measurement:

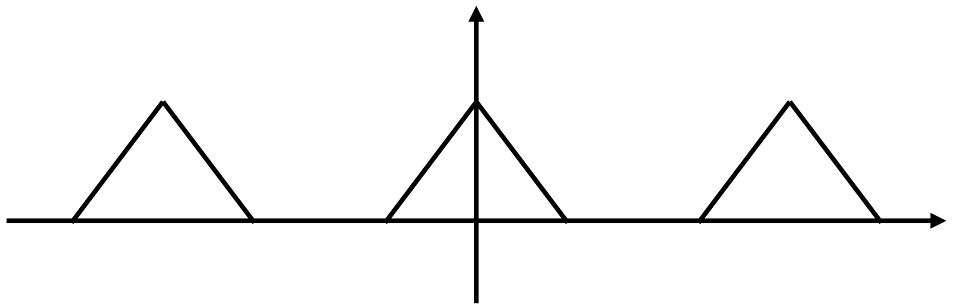
$$\text{EQ-T} = 10 \cdot \log_{10}\left(I_{max}^2 / \mathbb{E}_{\mathbf{w}\sim\mathcal{W},x\sim\mathcal{X}^2,p\sim\mathcal{V},c\sim\mathcal{C}}\left[\left(\mathbf{g}(\mathbf{t}_x[z_0];\mathbf{w})_c(p) - \mathbf{t}_x[\mathbf{g}(z_0;\mathbf{w})]_c(p)\right)^2\right]\right)$$

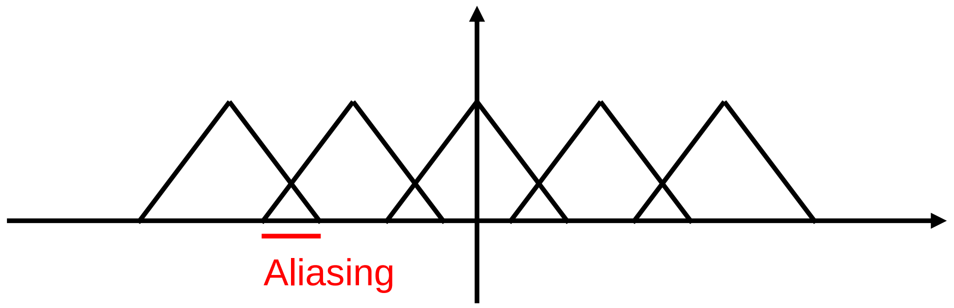
Convolution

- $F_{conv}(Z) = K * Z$.
- $f_{conv}(z) = \phi_s * \left(K * (\mathrm{III}_s \odot z)\right) = K * \phi_s * \mathrm{III}_s \odot z = K * z$.
- **This convolution introduces no new frequencies.**
- Convolution also commutes with translation in the continuous domain**.**
- For rotation equivariance, the discrete kernel K needs to be radially symmetric (e.g., 1×1).

Upsampling and Downsampling

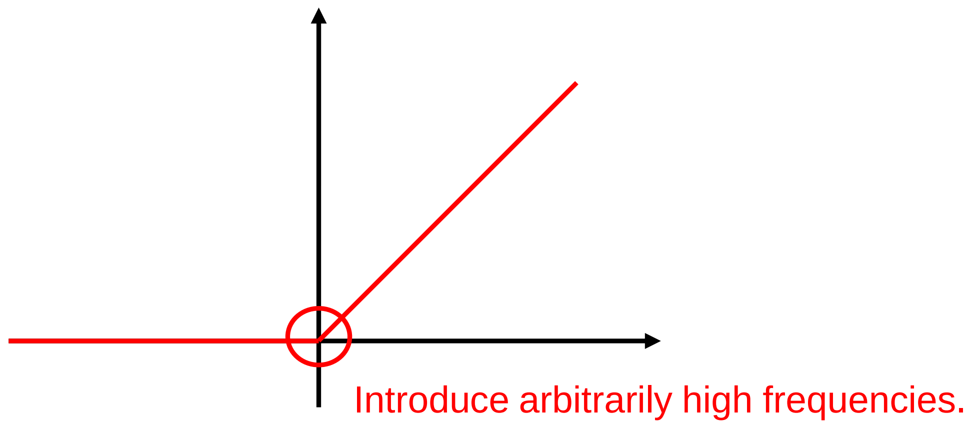
- Ideal upsampling does not modify the continuous representation.
  - $f_{up}(z) = z$.
  - $F_{up}(Z) = \mathrm{III}_{s'} \odot (\phi_s * Z)$.
- In downsampling, we must low-pass filter *z* to remove frequencies above the output bandlimit.
  - We have to use ideal low-pass filter $\psi_s \coloneqq s^2 \cdot \phi_s$.
  - $F_{down}(Z) = \mathrm{III}_{s'} \odot \left(\psi_{s'} * (\phi_s * Z)\right) = \frac{s'^2}{s^2} \cdot \mathrm{III}_{s'} \odot (\phi_{s'} * Z)$. $(\because \psi_{s'} * \psi_s = \psi_{\min(s,s')})$.
  - For rotation equivariance, we use $\phi_s{}^{\circ}(x) = \mathrm{jinc}(s\|x\|)$.



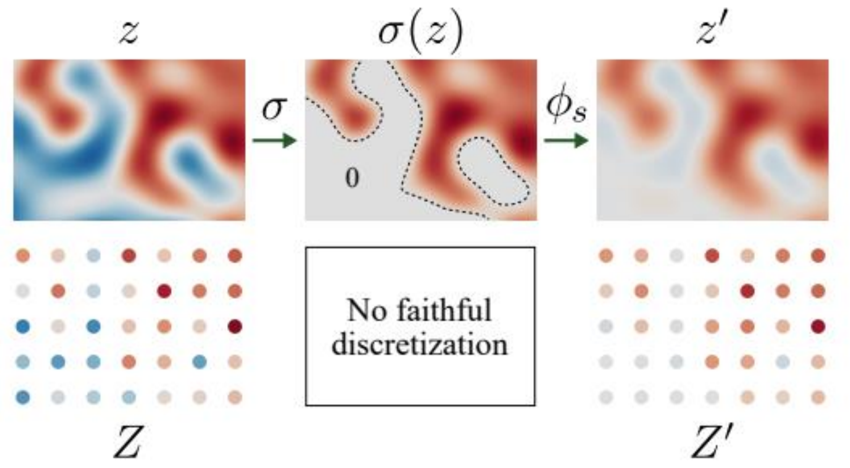


Nonlinearity

- A pointwise nonlinearity σ in the discrete domain does not commute with fractional translation or rotation.
  - E.g., ReLU in the continuous domain may introduce arbitrarily high frequencies that cannot be represented in the output.
- Again, use ideal low-pass filter ψ_s.



ReLU function

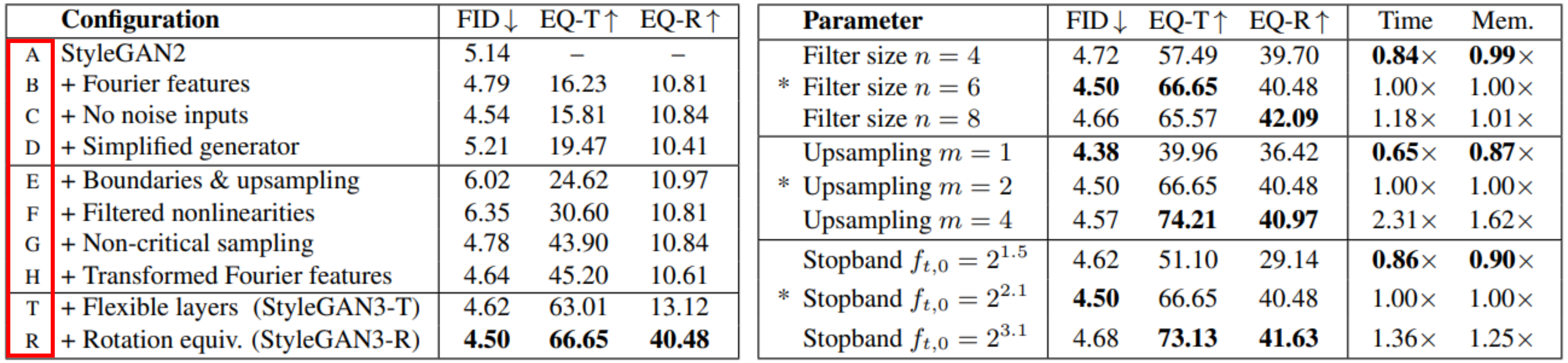


Practical Application to Generator Network

Practical Application to Generator Network

- Converting the well-established StyleGAN2 generator to be fully equivariant to translation and rotation.
- The discriminator remains unchanged in our experiments.

step-by-step



	Configuration	FID↓	EQ-T↑	EQ-R↑
A	StyleGAN2	5.14	–	–
B	+ Fourier features	4.79	16.23	10.81
C	+ No noise inputs	4.54	15.81	10.84
D	+ Simplified generator	5.21	19.47	10.41
E	+ Boundaries & upsampling	6.02	24.62	10.97
F	+ Filtered nonlinearities	6.35	30.60	10.81
G	+ Non-critical sampling	4.78	43.90	10.84
H	+ Transformed Fourier features	4.64	45.20	10.61
T	+ Flexible layers (StyleGAN3-T)	4.62	63.01	13.12
R	+ Rotation equiv. (StyleGAN3-R)	**4.50**	**66.65**	**40.48**

	Parameter	FID↓	EQ-T↑	EQ-R↑	Time	Mem.
	Filter size $n = 4$	4.72	57.49	39.70	**0.84**×	**0.99**×
*	Filter size $n = 6$	**4.50**	**66.65**	40.48	1.00×	1.00×
	Filter size $n = 8$	4.66	65.57	**42.09**	1.18×	1.01×
	Upsampling $m = 1$	**4.38**	39.96	36.42	**0.65**×	**0.87**×
*	Upsampling $m = 2$	4.50	66.65	40.48	1.00×	1.00×
	Upsampling $m = 4$	4.57	**74.21**	**40.97**	2.31×	1.62×
	Stopband $f_{t,0} = 2^{1.5}$	4.62	51.10	29.14	**0.86**×	**0.90**×
*	Stopband $f_{t,0} = 2^{2.1}$	**4.50**	66.65	40.48	1.00×	1.00×
	Stopband $f_{t,0} = 2^{3.1}$	4.68	**73.13**	**41.63**	1.36×	1.25×

Figure 3: Results for FFHQ-U (unaligned FFHQ) at 256^2. **Left:** Training configurations. FID is computed between 50k generated images and all training images [26, 32]; lower is better. EQ-T and EQ-R are our equivariance metrics in decibels (dB); higher is better. **Right:** Parameter ablations using our final configuration (R) for the filter's support, magnification around nonlinearities, and the minimum stopband frequency at the first layer. * indicates our default choices.

Fourier Features and Baseline Simplifications

- **B:** Sample the frequencies uniformly within the circular frequency band $f_c = 2$, matching the original 4×4 input resolution.
  - Allows us to compute the equivariance metrics without having to approximate the operator t by changing phase.
- **C:** Remove the per-pixel noise inputs.
  - The exact sub-pixel position of each feature should be exclusively inherited from the underlying coarse features.
- **D:** Decrease the mapping network depth as recommended by ADA and disable mixing regularization and path length regularization.

Fourier Features and Baseline Simplifications

- Finally, we also **eliminate the output skip connections.**
- We hypothesize that their benefit is mostly related to **gradient magnitude dynamics** during training.
- We track the **exponential moving average** $\sigma^2 = \mathbb{E}[x^2]$ over all pixels and feature maps during training, and divide the feature maps by $\sqrt{\sigma^2}$.

Boundaries and Upsampling (config E)

- This explicit extension is necessary as border padding is known to leak absolute image coordinates into the internal representations: **10-pixel margin to be enough.**
- Replace the bilinear 2× upsampling filter with a better approximation of the ideal low-pass filter.
  - We use a **windowed sinc filter** with a relatively large Kaiser window of size $n = 6$.

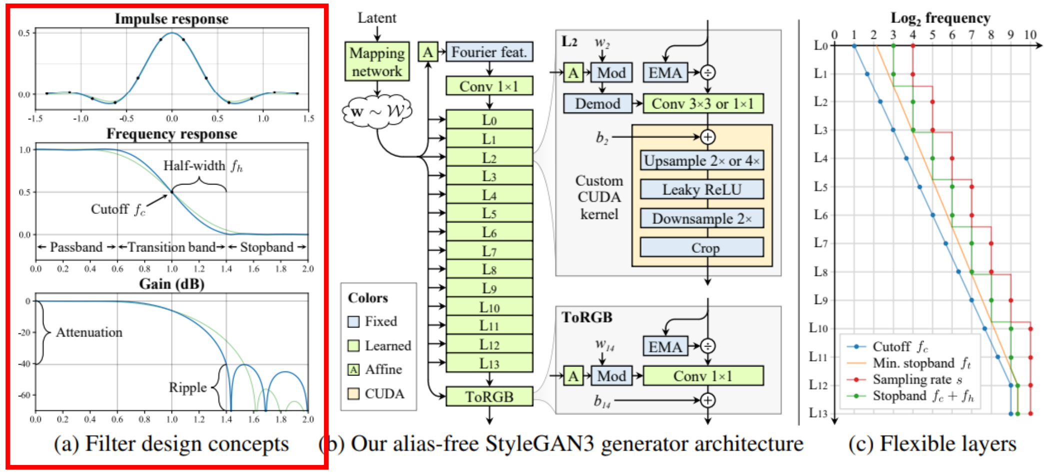
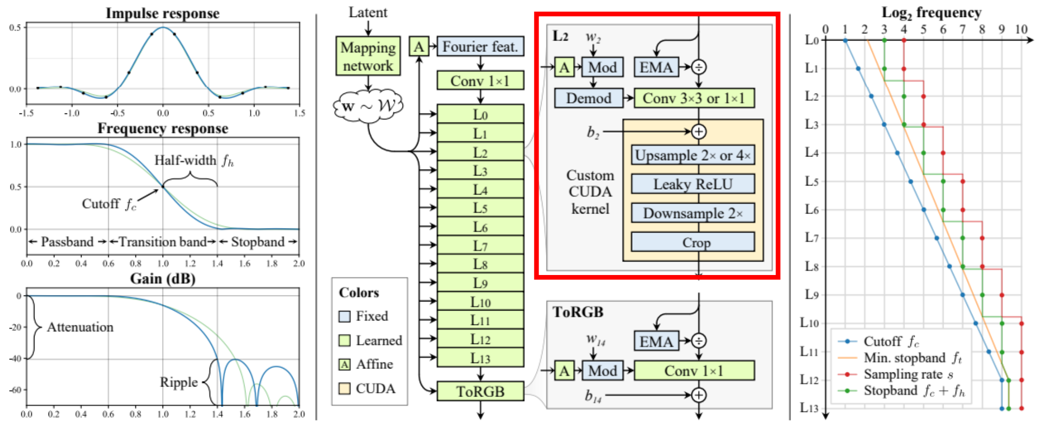


Figure 4: **(a)** 1D example of a 2× upsampling filter with $n = 6$, $s = 2$, $f_c = 1$, and $f_h = 0.4$ (blue). Setting $f_h = 0.6$ makes the transition band wider (green), which reduces the unwanted stopband ripple and thus leads to stronger attenuation. **(b)** Our alias-free generator, corresponding to configs T and R in Figure 3. The main datapath consists of Fourier features and normalization (Section 3.1), modulated convolutions [34], and filtered nonlinearities (Section 3.2). **(c)** Flexible layer specifications (config T) with $N = 14$ and $s_N = 1024$. Cutoff f_c (blue) and minimum acceptable stopband frequency f_t (orange) obey geometric progression over the layers; sampling rate s (red) and actual stopband $f_c + f_h$ (green) are computed according to our design constraints.

Filtered Nonlinearities (config F)

- Wrapping each leaky ReLU (or any other commonly used non-linearity) between $m\times$ upsampling and $m\times$ downsampling, for some magnification factor m ($m = 2$ is sufficient).
- They implement a custom CUDA kernel leading to $10\times$ faster training.

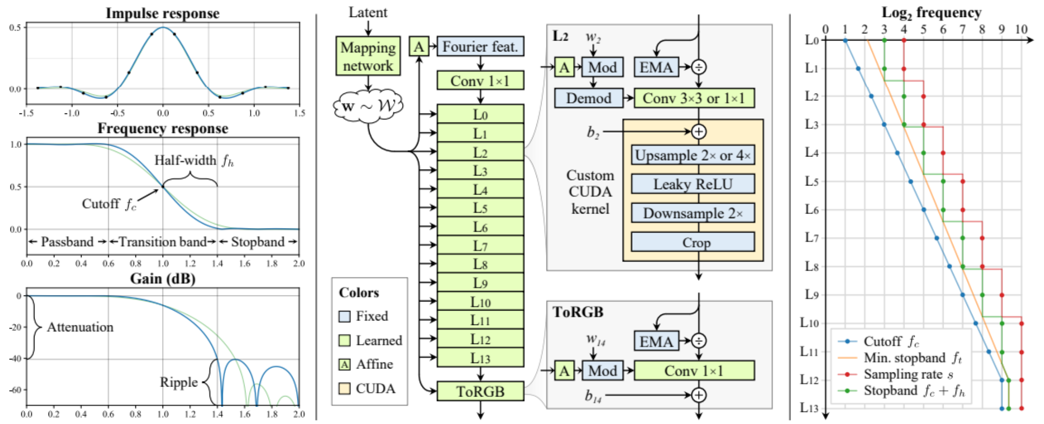


(a) Filter design concepts (b) Our alias-free StyleGAN3 generator architecture (c) Flexible layers

Figure 4: **(a)** 1D example of a 2× upsampling filter with $n = 6$, $s = 2$, $f_c = 1$, and $f_h = 0.4$ (blue). Setting $f_h = 0.6$ makes the transition band wider (green), which reduces the unwanted stopband ripple and thus leads to stronger attenuation. **(b)** Our alias-free generator, corresponding to configs T and R in Figure 3. The main datapath consists of Fourier features and normalization (Section 3.1), modulated convolutions [34], and filtered nonlinearities (Section 3.2). **(c)** Flexible layer specifications (config T) with $N = 14$ and $s_N = 1024$. Cutoff f_c (blue) and minimum acceptable stopband frequency f_t (orange) obey geometric progression over the layers; sampling rate s (red) and actual stopband $f_c + f_h$ (green) are computed according to our design constraints.

Non-critical sampling (config G)

- To suppress aliasing, we can simply lower the cutoff frequency to $f_c = \frac{s}{2} - f_h$.
- As the signals now contain less spatial information, they tune the number of feature maps to be inversely proportional to f_c instead of the sampling rate s.

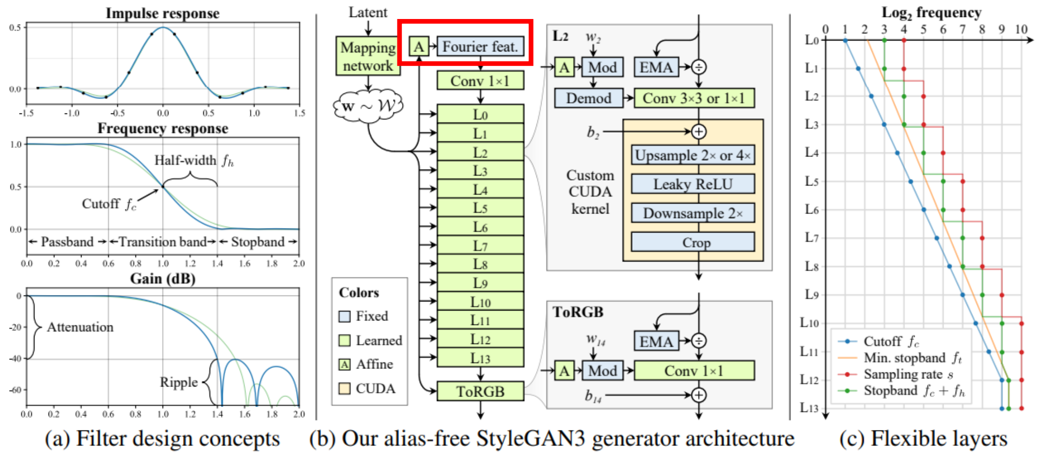


(a) Filter design concepts (b) Our alias-free StyleGAN3 generator architecture (c) Flexible layers

Figure 4: **(a)** 1D example of a 2× upsampling filter with $n = 6$, $s = 2$, $f_c = 1$, and $f_h = 0.4$ (blue). Setting $f_h = 0.6$ makes the transition band wider (green), which reduces the unwanted stopband ripple and thus leads to stronger attenuation. **(b)** Our alias-free generator, corresponding to configs T and R in Figure 3. The main datapath consists of Fourier features and normalization (Section 3.1), modulated convolutions [34], and filtered nonlinearities (Section 3.2). **(c)** Flexible layer specifications (config T) with $N = 14$ and $s_N = 1024$. Cutoff f_c (blue) and minimum acceptable stopband frequency f_t (orange) obey geometric progression over the layers; sampling rate s (red) and actual stopband $f_c + f_h$ (green) are computed according to our design constraints.

Transformed Fourier features (config H)

- The input features z_0 play a crucial role in defining the global orientation of z_N.
  - The generator should have the ability to transform z_0 based on w.
- **A learned affine layer** that outputs global translation and rotation parameters for the **input Fourier features.**

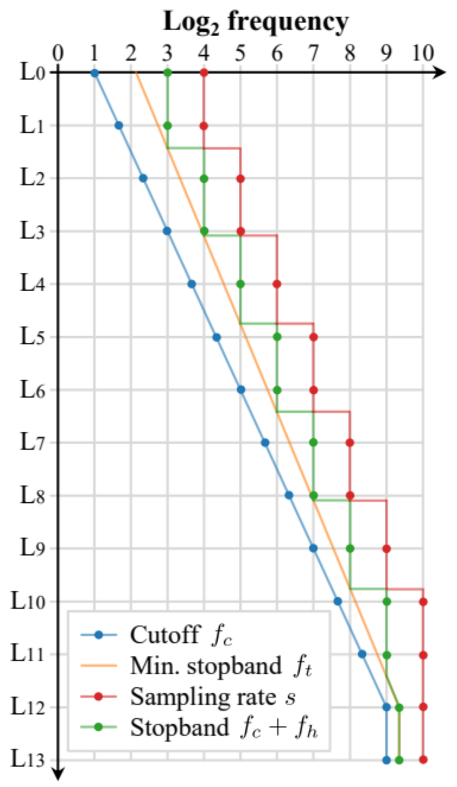


(a) Filter design concepts (b) Our alias-free StyleGAN3 generator architecture (c) Flexible layers

Figure 4: **(a)** 1D example of a 2× upsampling filter with $n = 6$, $s = 2$, $f_c = 1$, and $f_h = 0.4$ (blue). Setting $f_h = 0.6$ makes the transition band wider (green), which reduces the unwanted stopband ripple and thus leads to stronger attenuation. **(b)** Our alias-free generator, corresponding to configs T and R in Figure 3. The main datapath consists of Fourier features and normalization (Section 3.1), modulated convolutions [34], and filtered nonlinearities (Section 3.2). **(c)** Flexible layer specifications (config T) with $N = 14$ and $s_N = 1024$. Cutoff f_c (blue) and minimum acceptable stopband frequency f_t (orange) obey geometric progression over the layers; sampling rate s (red) and actual stopband $f_c + f_h$ (green) are computed according to our design constraints.

Flexible layer specifications (config T)

- We would like f_h to be high in the lowest-resolution layers to maximize attenuation in the stopband, but low in the highest-resolution layers to allow matching high-frequency details of the training data.



(c) Flexible layers

No more hyper-parameters

Rotation equivariance (config R)

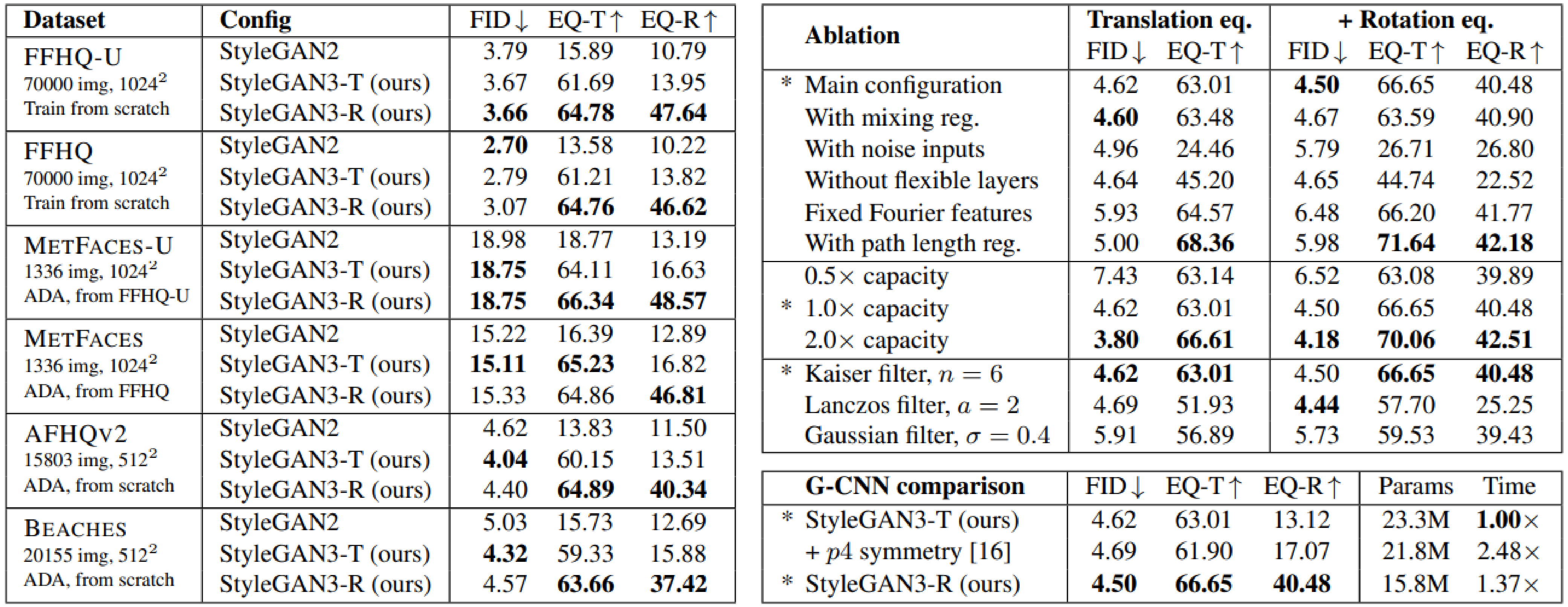
- First, we replace the 3×3 convolutions with 1×1 on all layers and compensate for the reduced capacity by doubling the number of feature maps.
- Second, we replace the sinc-based downsampling filter with a radially symmetric jinc-based one that we construct using the same Kaiser scheme (Appendix C).

Additional Stabilization Trick

- Early on in the training, we blur all images the discriminator sees using a Gaussian filter.
- We start with $\sigma = 10$ pixels, which we ramp to zero over the first 200k images.
- This prevents the discriminator from focusing too heavily on high frequencies early on.

Results

Quantitative Results



Dataset	Config	FID↓	EQ-T↑	EQ-R↑
FFHQ-U 70000 img, 1024^2 Train from scratch	StyleGAN2	3.79	15.89	10.79
	StyleGAN3-T (ours)	3.67	61.69	13.95
	StyleGAN3-R (ours)	**3.66**	**64.78**	**47.64**
FFHQ 70000 img, 1024^2 Train from scratch	StyleGAN2	**2.70**	13.58	10.22
	StyleGAN3-T (ours)	2.79	61.21	13.82
	StyleGAN3-R (ours)	3.07	**64.76**	**46.62**
METFACES-U 1336 img, 1024^2 ADA, from FFHQ-U	StyleGAN2	18.98	18.77	13.19
	StyleGAN3-T (ours)	**18.75**	64.11	16.63
	StyleGAN3-R (ours)	**18.75**	**66.34**	**48.57**
METFACES 1336 img, 1024^2 ADA, from FFHQ	StyleGAN2	15.22	16.39	12.89
	StyleGAN3-T (ours)	**15.11**	**65.23**	16.82
	StyleGAN3-R (ours)	15.33	64.86	**46.81**
AFHQV2 15803 img, 512^2 ADA, from scratch	StyleGAN2	4.62	13.83	11.50
	StyleGAN3-T (ours)	**4.04**	60.15	13.51
	StyleGAN3-R (ours)	4.40	**64.89**	**40.34**
BEACHES 20155 img, 512^2 ADA, from scratch	StyleGAN2	5.03	15.73	12.69
	StyleGAN3-T (ours)	**4.32**	59.33	15.88
	StyleGAN3-R (ours)	4.57	**63.66**	**37.42**

Ablation	Translation eq.		+ Rotation eq.		
	FID↓	EQ-T↑	FID↓	EQ-T↑	EQ-R↑
* Main configuration	4.62	63.01	**4.50**	66.65	40.48
With mixing reg.	**4.60**	63.48	4.67	63.59	40.90
With noise inputs	4.96	24.46	5.79	26.71	26.80
Without flexible layers	4.64	45.20	4.65	44.74	22.52
Fixed Fourier features	5.93	64.57	6.48	66.20	41.77
With path length reg.	5.00	**68.36**	5.98	**71.64**	**42.18**
0.5× capacity	7.43	63.14	6.52	63.08	39.89
* 1.0× capacity	4.62	63.01	4.50	66.65	40.48
2.0× capacity	**3.80**	**66.61**	**4.18**	**70.06**	**42.51**
* Kaiser filter, $n = 6$	**4.62**	**63.01**	4.50	**66.65**	**40.48**
Lanczos filter, $a = 2$	4.69	51.93	**4.44**	57.70	25.25
Gaussian filter, $\sigma = 0.4$	5.91	56.89	5.73	59.53	39.43

G-CNN comparison	FID↓	EQ-T↑	EQ-R↑	Params	Time
* StyleGAN3-T (ours)	4.62	63.01	13.12	23.3M	**1.00**×
+ *p4* symmetry [16]	4.69	61.90	17.07	21.8M	2.48×
* StyleGAN3-R (ours)	**4.50**	**66.65**	**40.48**	15.8M	1.37×

Figure 5: **Left:** Results for six datasets. We use adaptive discriminator augmentation (ADA) [32] for the smaller datasets. "StyleGAN2" corresponds to our baseline config B with Fourier features. **Right:** Ablations and comparisons for FFHQ-U (unaligned FFHQ) at 256^2. * indicates our default choices.

Qualitative Results

- https://nvlabs.github.io/stylegan3/

Internal Representations

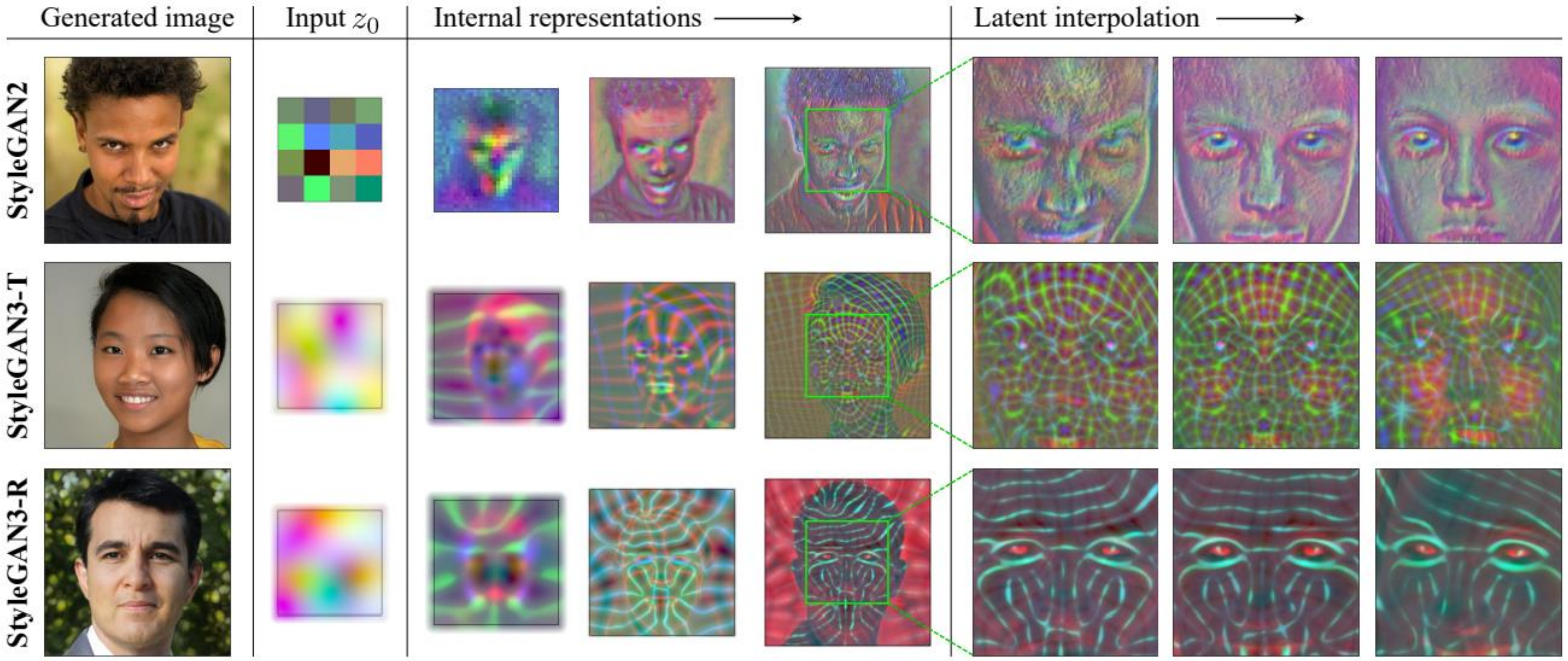


Figure 6: Example internal representations (3 feature maps as RGB) in StyleGAN2 and our generators.

Limitations

- Further benefits would be available by making the discriminator equivariant as well.
    - E.g., in our FFHQ results the teeth do not move correctly when the head turns, and we suspect that this is caused by the discriminator accidentally preferring to see the front teeth at certain pixel locations.
- Our alias-free generator architecture contains implicit assumptions about the nature of the training data, and violating these may cause training difficulties.
    - E.g., Edges are jagged, low-quality JPEGs, and retro pixel graphics.
    - Our generator is not partially equivariant.
- Re-introduce noise inputs (stochastic variation) and path length regularization.
- Recent attention-based GANs that start with a tokenizing transformer (e.g., VQGAN) may be at odds with equivariance.