Supplementary Material for Projected GANs Converge Faster

Axel Sauer[1,2] **Kashyap Chitta**[1,2] **Jens Müller**[3] **Andreas Geiger**[1,2]
[1]University of Tübingen [2]Max Planck Institute for Intelligent Systems, Tübingen
[3]Computer Vision and Learning Lab, University Heidelberg
[2]{firstname.lastname}@tue.mpg.de [3]{firstname.lastname}@iwr.uni-heidelberg.de

In this **supplementary document**, we first prove the theorem presented in the paper in Section 1. Section 2 provides additional evaluation metrics for StyleGAN-ADA [12], FastGAN [20], and Projected GAN, and FID of Projected GAN on nine more datasets. Section 3 presents uncurated samples for both baselines and our approach. Section 4 reports additional experiments. Lastly, we provide details on training configurations, hyperparameters, and compute in Section 5. The **supplementary videos** show interpolations between random samples of Projected GAN on all datasets. Code, models, and supplementary videos can be found on the project page https://sites.google.com/view/projected-gan.

1 Proofs

As described in the paper, Projected GAN training can be formulated as follows

$$\min_G \max_{\{D_l\}} \sum_{l \in \mathcal{L}} \Big(\mathbb{E}_{\mathbf{x}}[\log D_l(P_l(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}}[\log(1 - D_l(P_l(G(\mathbf{z}))))] \Big) \tag{1}$$

where $\{D_l\}$ is a set of independent discriminators operating on different feature projections. In the following, we first prove Theorem 1 for a deterministic projection. The second proof demonstrates the theorem's validity when training with stochastic differentiable augmentations.

Proof (deterministic). The following proof follows the consistency proofs in [23] and [7]. Let $\{P_l\}_{l \in \mathcal{L}}$ be a set of fixed feature projectors. Furthermore, let $\mathbb{P}_T$ be the density of the true data distribution and $\mathbb{P}_G$ the density of the distribution the generator G produces. As in Theorem 1, $P_l \circ T$ and $P_l \circ G$ are functional compositions of P_l and the true/generated data distribution. The minimax objective in (1) is then defined via

$$\min_G \max_{\{D_l\}} \sum_{l \in \mathcal{L}} V_l(D_l, G) \tag{2}$$

where

$$\begin{aligned} V_l\left(D_l, G\right) &= \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_T}\left[\log D_l\left(P_l(\mathbf{x})\right)\right] + \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_G}\left[\log\left(1 - D_l\left(P_l(\mathbf{x})\right)\right)\right] \\ &= \mathbb{E}_{\mathbf{y} \sim \mathbb{P}_{P_l \circ T}}\left[\log D_l(\mathbf{y})\right] + \mathbb{E}_{\mathbf{y} \sim \mathbb{P}_{P_l \circ G}}\left[\log\left(1 - D_l(\mathbf{y})\right)\right] \\ &= \int_{\mathbf{y}} \mathbb{P}_{P_l \circ T}(\mathbf{y}) \log(D_l(\mathbf{y})) + \mathbb{P}_{P_l \circ G}(\mathbf{y}) \log(1 - D_l(\mathbf{y})) d\mathbf{y} \end{aligned} \tag{3}$$

In the following we derive the optimal discriminator for a fixed G. For any $(a, b) \in \mathbb{R}^2 \setminus \{(0,0)\}$, the function $t \to a \log(t) + b \log(1-t)$ obtains its maximum in $[0,1]$ at $\frac{a}{a+b}$ [7]. Since the discriminators do not need to be defined outside $\operatorname{supp}(\mathbb{P}_{P_l \circ T}) \cup \operatorname{supp}(\mathbb{P}_{P_l \circ G})$, the maximum $\max_{\{D_l\}} V_l(D_l, G)$ is achieved for

$$D^*_{l,G}(\boldsymbol{y}) = \frac{\mathbb{P}_{P_l \circ T}(\mathbf{y})}{\mathbb{P}_{P_l \circ T}(\mathbf{y}) + \mathbb{P}_{P_l \circ G}(\mathbf{y})} \tag{4}$$

where G is fixed. Assuming a perfect discriminator, the minimax objective can be reformulated as

$$C(G) = \max_{\{D_l\}} \sum_l V_l(G, D_l) = \sum_l V_l(G, D^*_{l,G}) \tag{5}$$

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

Following the arguments of [7] and in [23], we obtain

$$C(G) = -2|\mathcal{L}|\log(2) + 2\sum_{l} JSD(\mathbb{P}_{P_l \circ T} || \mathbb{P}_{P_l \circ G}) \tag{6}$$

where JSD is the Jensen-Shannon divergence. Since the Jensen-Shannon divergence is non-negative and zero only in case of equality, the minimum is achieved iff $\mathbb{P}_{P_l \circ T} = \mathbb{P}_{P_l \circ G}$ for all l. This shows, that we achieve $\min_G C(G)$ iff $\mathbb{P}_{P_l \circ T} = \mathbb{P}_{P_l \circ G}$ for all l.

Proof (stochastic). We now show that the result above still holds when applying stochastic differentiable augmentations before the feature projections. Utilizing a stochastic augmentation $f_{\boldsymbol{\theta},l}$ before the projection through P_l, can be viewed as a functional composition, i.e. $P_{\boldsymbol{\theta},l} = P_l \circ f_{\boldsymbol{\theta},l}$. The parameter $\boldsymbol{\theta} \sim \mathbb{P}_\Theta$ encompasses both the probability of applying the augmentation and its parameters, e.g., translation direction and magnitude. As in the deterministic case, the minimax objective is defined as

$$\min_{G} \max_{\{D_l\}} \sum_{l \in \mathcal{L}} V_l(D_l, G) \tag{7}$$

where

$$\begin{aligned} V_l\left(D_l, G\right) &= \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_T}\left[\mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}[\log D_l\left(P_{\boldsymbol{\theta},l}(\mathbf{x})\right)]\right] + \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_G}\left[\mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}[\log\left(1 - D_l\left(P_{\boldsymbol{\theta},l}(\mathbf{x})\right)\right)]\right] \\ &= \mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}\left[\mathbb{E}_{\mathbf{x} \sim \mathbb{P}_T}\left[\log D_l\left(P_{\boldsymbol{\theta},l}(\mathbf{x})\right)\right] + \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_G}\left[\log\left(1 - D_l\left(P_{\boldsymbol{\theta},l}(\mathbf{x})\right)\right)\right]\right] \\ &= \mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}\left[\mathbb{E}_{\mathbf{y} \sim \mathbb{P}_{P_{\boldsymbol{\theta},l} \circ T}}\left[\log D_l(\mathbf{y})\right] + \mathbb{E}_{\mathbf{y} \sim \mathbb{P}_{P_{\boldsymbol{\theta},l} \circ G}}\left[\log\left(1 - D_l(\mathbf{y})\right)\right]\right] \\ &= \mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}\left[\int_{\mathbf{y}} \mathbb{P}_{P_{\boldsymbol{\theta},l} \circ T}(\mathbf{y}) \log(D_l(\mathbf{y})) + \mathbb{P}_{P_{\boldsymbol{\theta},l} \circ G}(\mathbf{y}) \log(1 - D_l(\mathbf{y})) d\mathbf{y}\right] \\ &= \int_{\mathbf{y}} \mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}[\mathbb{P}_{P_{\boldsymbol{\theta},l} \circ T}(\mathbf{y})] \log(D_l(\mathbf{y})) + \mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}[\mathbb{P}_{P_{\boldsymbol{\theta},l} \circ G}(\mathbf{y})] \log(1 - D_l(\mathbf{y})) d\mathbf{y} \end{aligned} \tag{8}$$

Using the same arguments as above, we obtain that the maximum $\max_{\{D_l\}} V_l(D_l, G)$ is achieved for

$$D^*_{l,G}(\boldsymbol{y}) = \frac{\mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}[\mathbb{P}_{P_{\boldsymbol{\theta},l} \circ T}(\mathbf{y})]}{\mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}[\mathbb{P}_{P_{\boldsymbol{\theta},l} \circ T}(\mathbf{y})] + \mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}[\mathbb{P}_{P_{\boldsymbol{\theta},l} \circ G}(\mathbf{y})]} \tag{9}$$

where G is fixed. Note that $\mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}[\mathbb{P}_{P_{\boldsymbol{\theta},l} \circ T}]$ and $\mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}[\mathbb{P}_{P_{\boldsymbol{\theta},l} \circ G}]$ are densities. Similar to above, we obtain $\min_G C(G)$ iff $\mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_{\Theta,l}}[\mathbb{P}_{P_l \circ T}] = \mathbb{E}_{\boldsymbol{\theta} \sim \mathbb{P}_\Theta}[\mathbb{P}_{P_{\boldsymbol{\theta},l} \circ G}]$ for all l.

2 Additional Metrics and Datasets

Metrics. In addition to FID and *Imgs* reported in the paper, we compute the following metrics:

- **Kernel Inception Distance (KID) [2].** KID is an unbiased alternative to FID, hence, especially suitable for small datasets.
- **Precision and Recall [28].** Precision measures the quality of samples, and recall measures image diversity. Generally, GANs produce high-quality samples while being prone to mode collapse (high precision, low recall), compared to VAEs [16], which generate lower quality but more diverse samples (low precision, high recall). This observation is evidenced empirically in [28]. For our evalution, we use the improved formulation by [18].
- **SwAV-FID [22].** Instead of utilizing an image classifier feature space, SwAV-FID uses self-supervised representations. More specifically, SwAV-FID computes the Fréchet distance in the penultimate layer of a ResNet-50 trained with the contrastive SwAV objective [3]. Morozov et al. [22] show that in some cases, SwAV-FID is more consistent with human judgment of visual quality than FID.
- **CLIP-FID & Virtex-FID.** FID uses an Inception network trained on ImageNet. Our feature network F has also been trained on ImageNet. To rule out training data as a confounding factor between F and the evaluation metric, we propose to use FID with non-ImageNet features. We evaluate CLIP-FID (using a ResNet50 trained with the CLIP objective on the dataset collected by [25]) and VirTex-FID (using a ResNet50 trained on COCO Captions with the VirTex objective [5]).
- **Sliced Wasserstein Distance (SWD).** SWD is a non-neural metric that computes the Wasserstein distance between local image patches drawn from a Laplacian pyramid. We follow the evaluation protocol proposed by [11].

In addition to the metrics above, we conduct a **human preference study** with 28 participants unfamiliar with our method. The structure of the study is as follows:

- For each dataset, we first show samples of the real dataset for context.
- We then present two sample sheets and ask the participants to rank the sheets relative to each other based on sample fidelity and diversity. We define these as follows: (i) Fidelity is the degree to which the generated samples resemble the real ones. (ii) Diversity measures whether the generated samples cover the full variability of the real samples.
- Each sheet contains nine random samples; we present three sample sheet pairs per dataset.
- Samples and comparison pairs are randomized per participant. We make sure that all possible pairings are equally represented.
- Evaluated Models: StyleGAN2-ADA, FastGAN, Projected GAN, and real data (for control)
- Evaluated datasets: all 256x256 datasets
- We count how often a given model wins a comparison and report the relative amount of wins.

Results. On all datasets, KID, SwaV-FID, CLIP-FID, VirTex-FID, and SWD mirror the ranking obtained via FID. The low SwAV-FIDs indicate that Projected GAN's low FIDs are not due to correlations between the feature network used for projection and the inception network used in FID.

On the large datasets, the baselines only outperform Projected GAN in precision on FFHQ, Cityscapes, and LSUN Church (Table 1). However, when comparing recall in these cases, it is apparent that the baselines suffer from mode collapse. The high recall generally obtained by Projected GAN hints at the reason for its superior performance on FID, KID, and SwAV-FID: the generated images are very diverse. Hence, we conclude that Projected GANs alleviate mode collapse. The sample diversity is also evident in the qualitative comparisons in Section 3. On small datasets (Table 2), Projected GAN is outperformed in precision on Art Painting by FastGAN; however, FastGANs very low recall of 0.044 hints at mode collapse. Only on flowers, Projected GAN appears to cover fewer modes than the baselines as indicated by lower recall.

At high resolutions (Table 3), Projected GAN performs slightly worse in precision. It appears that Projected GAN incurs small losses in image quality while obtaining a better mode coverage, which can be observed in the quantitative comparisons, e.g., on AFHQ-Cat, some samples exhibit artifacts. These artifacts indicate that projected GAN training at higher resolutions warrants closer inspection.

On small datasets, overfitting is a problem that is not detected well by FID and other metrics [27]. Therefore, it is instructive to inspect latent interpolations for which we refer the reader to the supplementary videos. Projected GAN generates smooth interpolations between random samples on all datasets suggesting that it generalizes rather than memorizing training samples.

The results of the human preference study are shown in Table 4. The study results largely agree with the results obtained via FID. On FFHQ, the study demonstrates our reported failure case for projected GANs. Interestingly, on AnimalFace projected GAN outperforms real data. We hypothesize that this is because for AnimalFace there is a significant portion of low-quality images (blurry, compression artifacts) in the dataset, and possibly projected GAN generates fewer of those. Of course, human studies are not optimal, as it is not straightforward to evaluate sample diversity - which is a strong suit of projected GANs - given only a few samples.

Table 5 reports the FID achieved by Projected GAN for nine more datasets, all at a resolution of 256^2. We compare on LSUN cat and horse [31], ADE indoor (a subset of ADE [34] proposed in [1]), the full Oxford flowers dataset with 8k images [24], KITTI-fisheye (a subset of KITTI-360 [19], consisting of fisheye images), STL-10 [4], CUB200 [30], Stanford Dogs [14], and Stanford Cars [17]. We do not change the hyperparameters of Projected GAN. On each dataset, we report the lowest FID achieved in previous literature. We train FastGAN as a baseline for ADE indoor and KITTI-fisheye.

	Large Datasets (256^2)				
	CLEVR	**FFHQ**	**Cityscapes**	**Bedroom**	**Church**
	$FID \downarrow$				
StyleGAN2-ADA [12]	10.17	7.32	8.35	11.53	5.85
FastGAN [20]	3.24	12.69	8.78	8.24	8.43
Projected GAN	**0.89**	**3.08**	**3.41**	**1.52**	**1.59**
	$KID \times 10^3 \downarrow$				
StyleGAN2-ADA [12]	8.15	1.49	3.34	7.42	4.70
FastGAN [20]	2.64	5.34	5.45	5.90	4.61
Projected GAN	**0.51**	**0.44**	**0.91**	**0.36**	**0.50**
	$Precision \uparrow$				
StyleGAN2-ADA [12]	0.373	0.669	**0.649**	0.429	0.565
FastGAN [20]	0.600	**0.716**	0.557	0.602	**0.645**
Projected GAN	**0.640**	0.654	0.619	**0.614**	0.612
	$Recall \uparrow$				
StyleGAN2-ADA [12]	0.569	0.445	0.146	0.202	0.416
FastGAN [20]	0.650	0.184	0.227	0.189	0.207
Projected GAN	**0.735**	**0.464**	**0.361**	**0.346**	**0.438**
	$SwAV - FID \downarrow$				
StyleGAN2-ADA [12]	3.50	1.24	1.35	8.47	2.51
FastGAN [20]	1.46	2.55	1.29	5.38	3.64
Projected GAN	**0.56**	**0.85**	**0.60**	**1.44**	**1.01**
	$CLIP - FID \downarrow$				
StyleGAN2-ADA [12]	4.70	10.3	5.88	42.12	15.85
FastGAN [20]	4.24	19.23	6.46	31.10	35.47
Projected GAN	**0.80**	**7.55**	**2.96**	**11.97**	**13.71**
	$VirTex - FID \downarrow$				
StyleGAN2-ADA [12]	0.78	1.20	1.15	2.20	1.10
FastGAN [20]	0.64	2.47	1.48	2.66	3.61
Projected GAN	**0.35**	**0.64**	**0.49**	**0.81**	**0.82**
	$SWD \times 10^{-3} \downarrow$				
StyleGAN2-ADA [12]	17.50	7.42	10.71	12.53	14.62
FastGAN [20]	28.51	10.19	9.45	14.68	14.42
Projected GAN	**12.90**	**6.41**	**7.27**	**6.83**	**8.37**

Table 1: **Metrics on Large Datasets** (256^2)**.** Projected GAN compares favorably on most metrics. Exceptions are precision on FFHQ, Cityscapes, and LSUN Church. As argued by [13], shifting from precision to recall is generally desirable, since recall can be traded into precision via truncation.

	Small Datasets (256^2)				
	Art Painting	**Landscape**	**AnimalFace**	**Flowers**	**Pokemon**
	$FID \downarrow$				
StyleGAN2-ADA [12]	43.07	15.99	60.90	21.66	40.38
FastGAN [20]	44.02	16.44	62.11	26.23	81.86
Projected GAN	**27.96**	**6.92**	**17.88**	**13.86**	**26.36**
	$KID \times 10^3 \downarrow$				
StyleGAN2-ADA [12]	10.23	4.39	22.52	3.56	13.49
FastGAN [20]	13.00	3.40	22.11	6.61	80.30
Projected GAN	**1.25**	**1.30**	**0.03**	**0.38**	**1.32**
	$Precision \uparrow$				
StyleGAN2-ADA [12]	0.691	0.709	0.841	0.731	0.735
FastGAN [20]	**0.858**	0.768	0.849	0.611	0.731
Projected GAN	0.762	**0.774**	**0.998**	**0.816**	**0.809**
	$Recall \uparrow$				
StyleGAN2-ADA [12]	0.218	0.213	0.036	0.095	0.197
FastGAN [20]	0.044	0.160	0.015	**0.100**	0.004
Projected GAN	**0.239**	**0.258**	**0.095**	0.058	**0.259**
	$SwAV - FID \downarrow$				
StyleGAN2-ADA [12]	3.32	2.98	16.26	5.02	6.71
FastGAN [20]	3.29	2.42	15.07	7.45	9.25
Projected GAN	**2.25**	**1.42**	**4.22**	**2.70**	**2.04**
	$CLIP - FID \downarrow$				
StyleGAN2-ADA [12]	44.13	24.89	46.18	26.30	13.96
FastGAN [20]	40.47	19.84	54.69	40.12	87.65
Projected GAN	**22.91**	**13.71**	**16.89**	**15.83**	**9.93**
	$VirTex - FID \downarrow$				
StyleGAN2-ADA [12]	4.15	2.78	8.83	3.25	3.69
FastGAN [20]	5.72	3.86	9.41	4.08	17.49
Projected GAN	**3.53**	**1.98**	**3.79**	**2.19**	**2.55**
	$SWD \times 10^{-3} \downarrow$				
StyleGAN2-ADA [12]	25.55	19.06	22.31	14.04	14.73
FastGAN [20]	21.94	29.87	29.23	17.39	46.81
Projected GAN	**11.44**	**15.38**	**14.34**	**9.61**	**11.65**

Table 2: **Metrics on Small Datasets (256^2).** Projected GAN performs best on most metrics.

	1024^2		512^2		
	Art Painting	**Pokemon**	**AHFQ-Cat**	**AFHQ-Dog**	**AFHQ-Wild**
			$FID \downarrow$		
StyleGAN2-ADA [12]	41.69	56.76	3.55	7.40	3.05
FastGAN [20]	46.71	56.46	4.69	13.09	3.14
Projected GAN	**32.07**	**33.96**	**2.16**	**4.52**	**2.17**
			$KID \times 10^3 \downarrow$		
StyleGAN2-ADA [12]	26.59	15.31	0.63	1.21	0.47
FastGAN [20]	12.70	29.40	1.72	5.51	0.74
Projected GAN	**1.70**	**7.76**	**0.16**	**0.80**	**0.12**
			$Precision \uparrow$		
StyleGAN2-ADA [12]	0.619	**0.791**	0.767	**0.753**	**0.765**
FastGAN [20]	**0.776**	0.777	**0.784**	0.746	0.761
Projected GAN	0.706	0.780	0.693	0.718	0.705
			$Recall \uparrow$		
StyleGAN2-ADA [12]	0.168	0.053	0.411	0.470	0.137
FastGAN [20]	0.033	0.080	0.305	0.380	0.201
Projected GAN	**0.235**	**0.215**	**0.565**	**0.643**	**0.292**
			$SwAV - FID \downarrow$		
StyleGAN2-ADA [12]	3.68	5.03	1.23	1.98	1.89
FastGAN [20]	3.41	5.14	1.73	3.07	1.77
Projected GAN	**2.28**	**3.83**	**0.68**	**1.12**	**1.08**

Table 3: **Metrics on Small Datasets (512^2 and 1024^2).** The results are in line with the findings at a resolution of 256^2. Only with respect to precision, the baselines slightly outperform Projected GAN.

	CLEVR	**FFHQ**	**Cityscapes**	**Bedroom**	**Church**	**ArtPainting**	**Landscape**	**AnimalFace**	**Flowers**	**Pokemon**	**All Datasets**
						Fidelity					
Stylegan2-ADA [12]	14 %	32 %	17 %	5 %	16 %	16 %	21 %	4 %	17 %	10 %	15 %
FastGAN [20]	7 %	2 %	15 %	3 %	9 %	0 %	8 %	7 %	4 %	0 %	6 %
Projected GAN	42 %	5 %	15 %	16 %	9 %	28 %	17 %	55 %	25 %	38 %	25 %
Data	37 %	61 %	53 %	76 %	66 %	56 %	54 %	34 %	54 %	52 %	54 %
						Diversity					
Stylegan2-ADA [12]	13 %	24 %	9 %	10 %	19 %	11 %	25 %	9 %	17 %	6 %	14 %
FastGAN [20]	13 %	4 %	11 %	2 %	0 %	4 %	13 %	9 %	17 %	0 %	7 %
Projected GAN	31 %	12 %	21 %	21 %	14 %	27 %	16 %	49 %	29 %	40 %	27 %
Data	43 %	60 %	59 %	67 %	67 %	58 %	46 %	33 %	37 %	54 %	52 %

Table 4: **Human Preference Study.** The obtained results largely agree with the rankings of other metrics.

	LSUN cat	LSUN horse	ADE indoor	Flowers Full	KITTI fisheye
Prev. SotA (Approach)	5.57 (ADM [6])	2.57 (ADM [6])	30.33 (FastGAN [20])	19.60 (MSG-StyleGAN [10])	6.64 (FastGAN [20])
Projected GAN	**3.89**	**2.17**	**6.70**	**3.86**	**2.72**

	STL-10	CUB200	Stanford Dogs	Stanford Cars
Prev. SotA (Approach)	25.32 (TransGAN [9])	11.25 (FineGAN [29])	25.66 (FineGAN [29])	16.03 (FineGAN [29])
Projected GAN	**13.68**	**2.79**	**11.75**	**2.09**

Table 5: **FID on Additional Datasets (**256^2**)** Without any hyperparameter changes, Projected GAN outperforms the previous state-of-the art on all evaluated datasets.

3 Qualitative Comparisons

We show generated images for CLEVR (Fig. 1), FFHQ (Fig. 2), Cityscapes (Fig. 3), Bedroom (Fig. 4), Church (Fig. 5), Art Painting (Fig. 6, 11), Landscape (Fig. 7), AnimalFace Dog (Fig. 8), Flowers (Fig. 9), Pokemon (Fig. 10, 12), AFHQ-Cat (Fig. 13), AFHQ-Dog (Fig. 14), and AFHQ-Wild (Fig. 15). Following [12], we select a global seed per dataset. We do not perform truncation on any of the models. Projected GAN produces convincing results on all datasets. The sample diversity in particular is apparent in comparison to the baselines, e.g., on AFHQ-Cat or AFHQ-Wild, all baselines generate high-quality samples, but Projected GAN captures more variability of the training data.

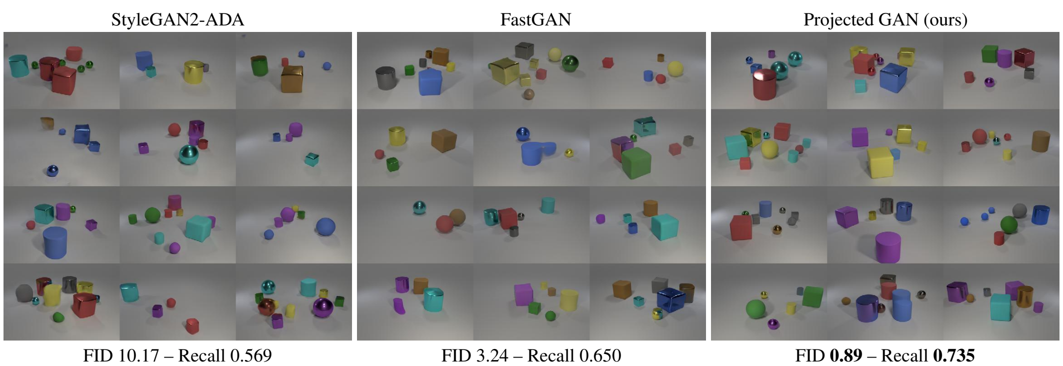

Figure 1: **Uncurated Results for CLEVR (256^2).** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

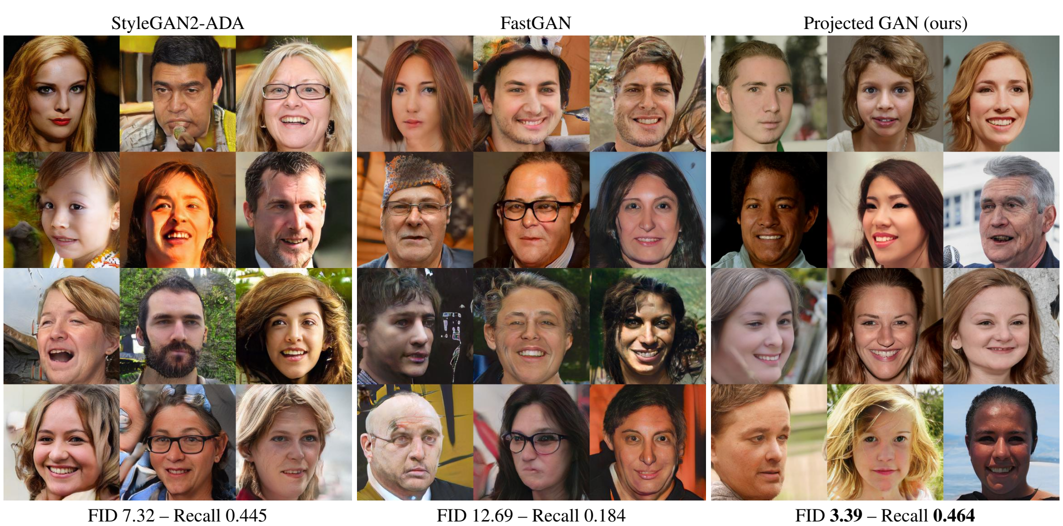

Figure 2: **Uncurated Results for FFHQ (256^2).** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

StyleGAN2-ADA | FastGAN | Projected GAN (ours)

FID 8.35 – Recall 0.146 | FID 8.78 - Recall 0.227 | FID **3.41** - Recall **0.361**

Figure 3: **Uncurated Results for Cityscapes** (256^2)**.** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

StyleGAN2-ADA | FastGAN | Projected GAN (ours)

FID 11.53 – Recall 0.202 | FID 8.24 - Recall 0.189 | FID **1.52** - Recall **0.346**

Figure 4: **Uncurated Results for LSUN bedroom** (256^2)**.** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

StyleGAN2-ADA | FastGAN | Projected GAN (ours)

FID 5.85 – Recall 0.416 | FID 8.43 - Recall 0.207 | FID **1.59** - Recall **0.438**

Figure 5: **Uncurated Results for LSUN church** (256^2)**.** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

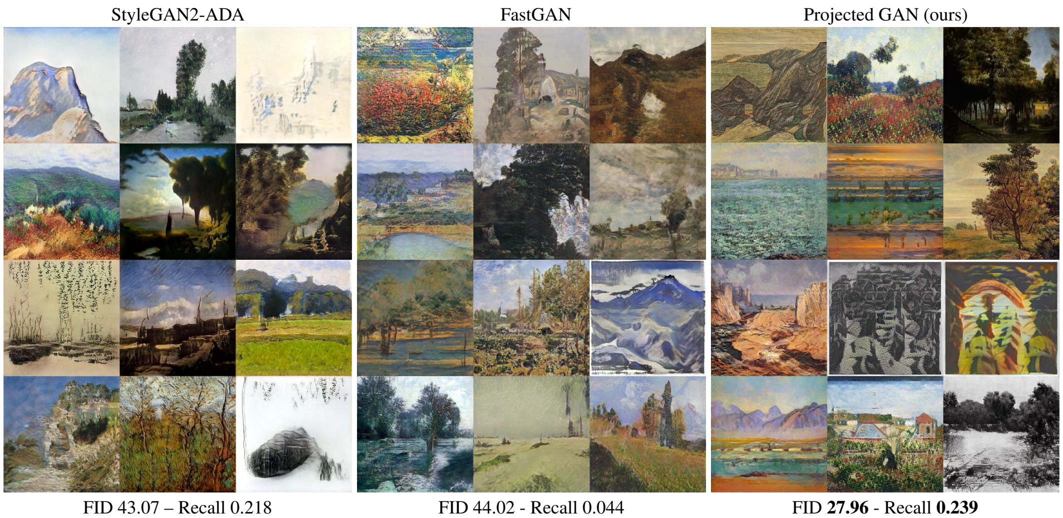

Figure 6: **Uncurated Results for Art Painting (256^2).** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

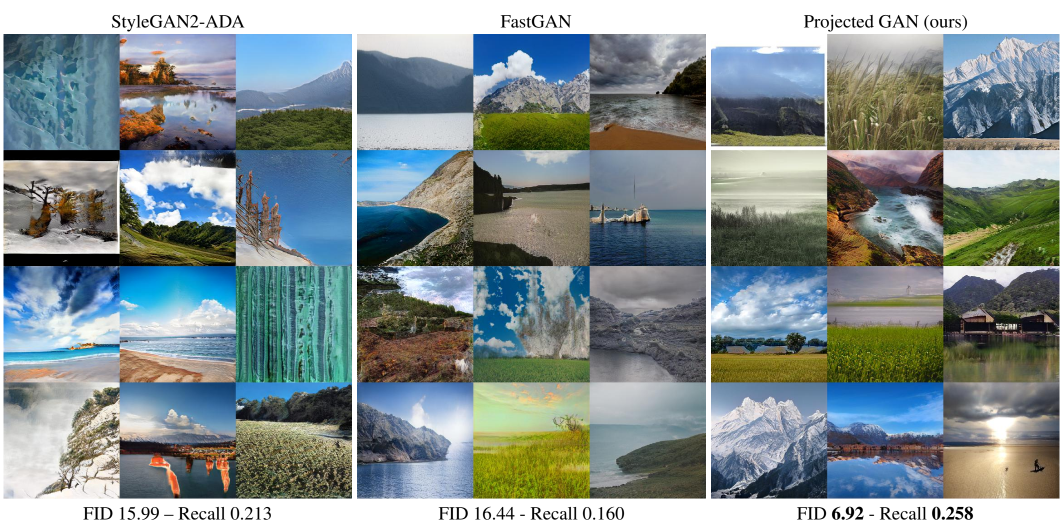

Figure 7: **Uncurated Results for Landscape (256^2).** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

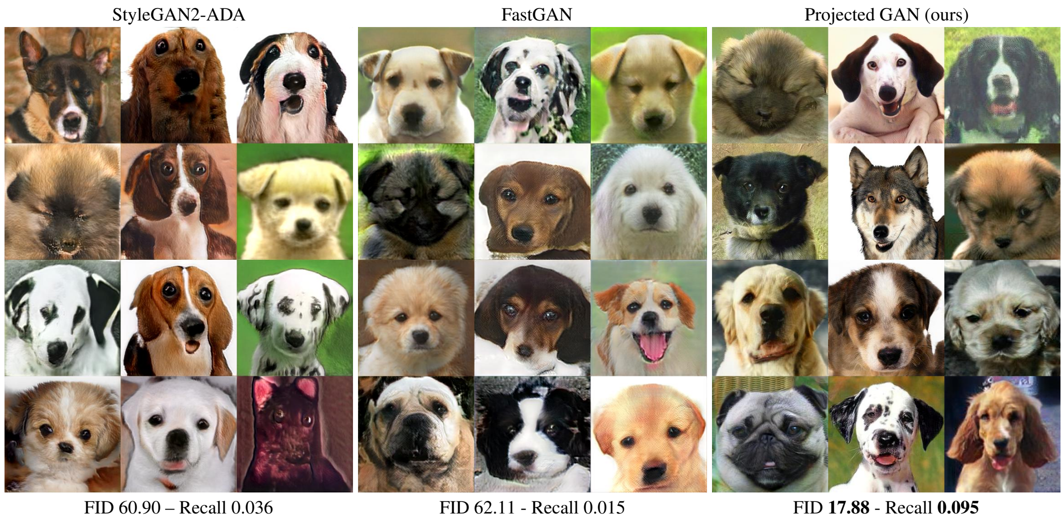

Figure 8: **Uncurated Results for AnimalFace-Dog (256^2).** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

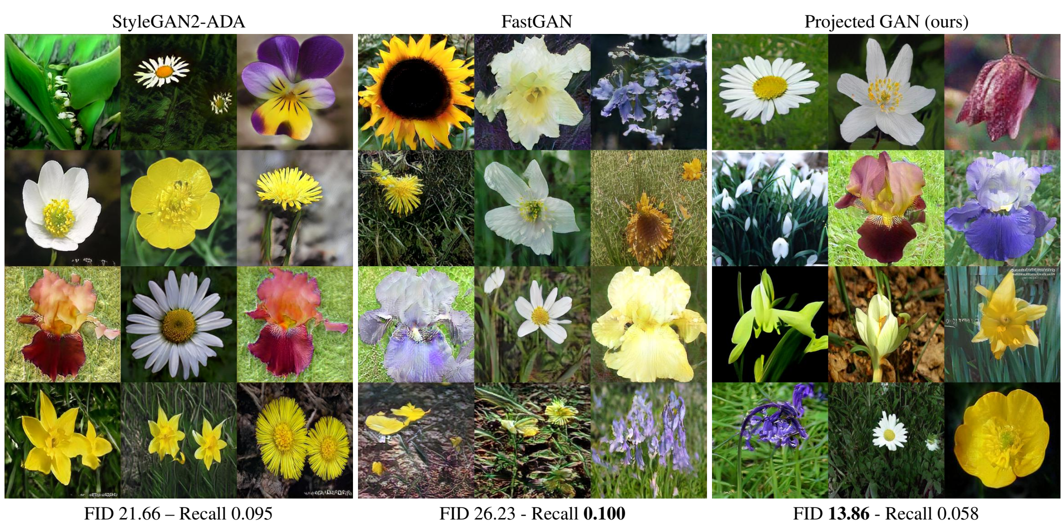

Figure 9: **Uncurated Results for Flowers (256^2).** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

Figure 10: **Uncurated Results for Pokemon** (256^2)**.** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

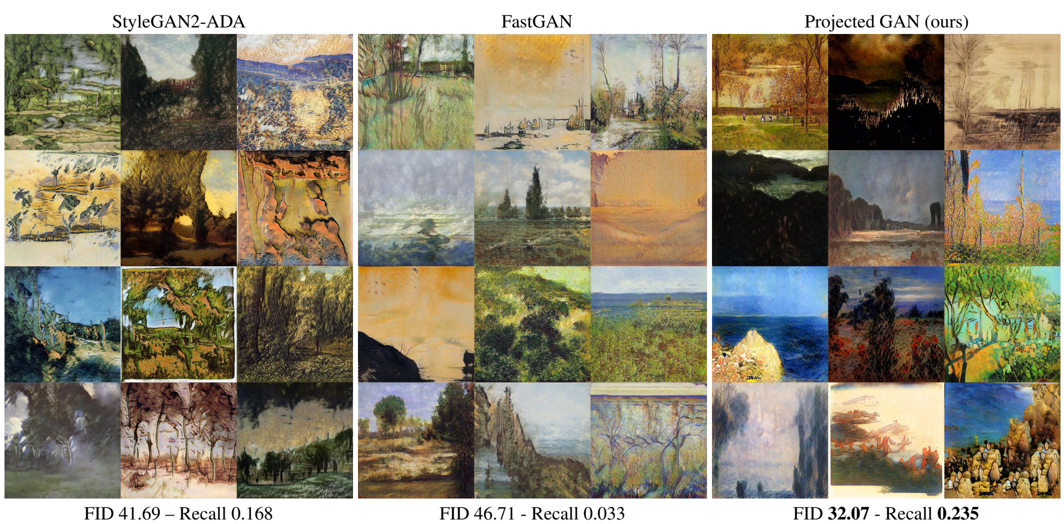

Figure 11: **Uncurated Results for Art Painting** (1024^2)**.** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

Figure 12: **Uncurated Results for Pokemon** (1024^2)**.** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

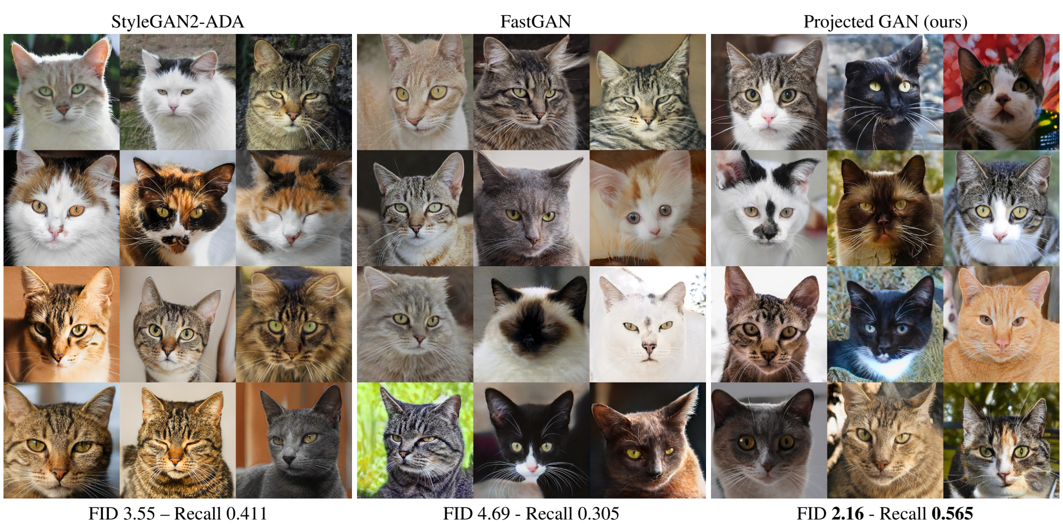

Figure 13: **Uncurated Results for AFHQ-Cat** (512^2)**.** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

Figure 14: **Uncurated Results for AFHQ-Dog (512^2).** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

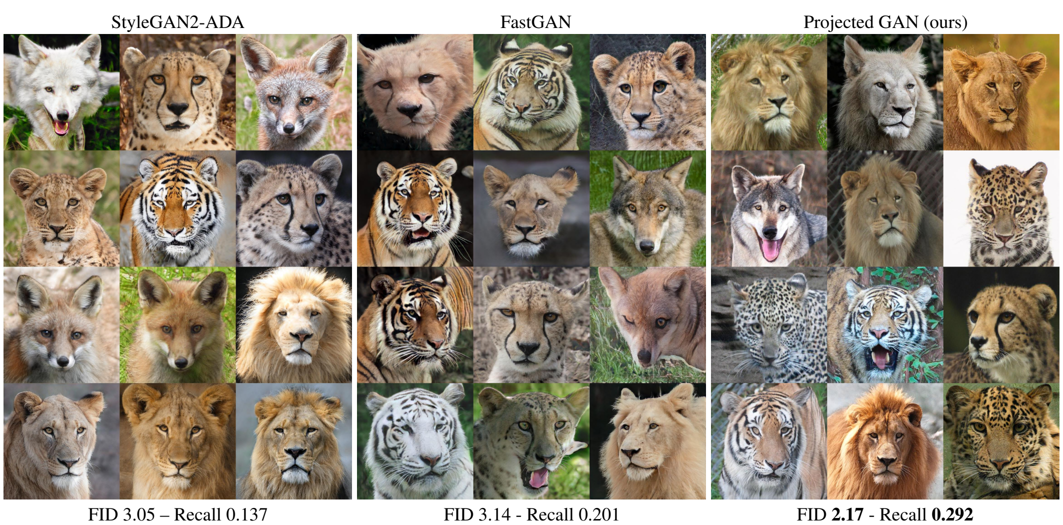

Figure 15: **Uncurated Results for AFHQ-Wild (512^2).** The images are selected randomly given one global random seed. We recommend zooming in for comparison.

4 Additional Experiments

This section presents additional experiments referenced in the paper. The experiments explore alternative setups to our final configuration entailing a pretrained, fixed feature network F, fixed 1x1 convolutions for cross-channel mixing (CCM), and fixed convolutions for cross-scale mixing (CSM). We follow the setup of Section 4 of the paper: training on LSUN church at a resolution of 256^2, with a batch size of 64 for 1 million *Imgs*, four discriminators, and a EfficientNet-Lite1 as feature network F. Again, we report FID normalized by the FID obtained by a model with a standard single RGB image discriminator. Values > 1 indicate worse performance than the RGB baseline. The results are summarized in Table 6.

No Projection. In all experiments, we utilize pretrained representations. As a sanity check, it is instructive to test if the architectural bias of F alone is helpful. The results in Table 6 demonstrate that randomly initializing F results in much higher FID.

CCM. We explore three different options for CCM. The first option, *Feature Norm*, does not mix features; rather, it normalizes the features to exhibit zero mean and a standard deviation of 1. This option investigates the importance of input scaling. The two other options utilize random convolution with different initializations. *CCM-rotation* is initialized with a random rotation matrix, *CCM-Kaiming* utilizes Kaiming initialization. As shown in Table 6, *CCM-Kaiming* (0.77) improves over the baselines (*Feature Norm* (1.27), *CCM-rotation* (1.27)), over the RGB baseline (1.0), and pretrained F without projection (1.15). This supports our hypothesis, that we need a sufficient amount of channel mixing.

Configuration	FID
No Projection	
Random F	11.03
Pretrained F	1.15
CCM	
Feature Norm	1.27
CCM-rotation	1.27
CCM-Kaiming	0.77
CCM + CSM	
Denoising AE (t_0)	0.24
Denoising AE (t_1)	0.37
Denoising AE (t_2)	0.44
Denoising AE (t_3)	0.59
Train F, Train P_{rand}	3.09
Fixed F, Train P_{rand}	0.97
Fixed F, Fixed P_{rand}	0.24

Table 6: **Ablations.**

CSM. We investigate if training the random projections P_{rand} in CCM and CSM is advantageous for FID. We consider two options: training P_{rand} before or during GAN training. First, we train P_{rand} before GAN training, the feature network F remains fixed. For this purpose, we add a head on the last CSM layer to map back to full resolution and three output channels, forming an autoencoder architecture. This model trains with a denoising autoencoder loss on ImageNet: the input image is augmented with gaussian blur, JPEG compression, coarse and fine region dropout, and conversion to grayscale. All augmentations are applied with a probability of 0.5. The target for reconstruction is the non-augmented image. We keep four models, at the beginning of training (t_0), at convergence (t_3), and two in between with equal spacing in terms of reconstruction loss. After autoencoder training, we use the model (without the additional head) for projected GAN training, denoted as *Denoising AE* (t_i). Interestingly, the longer the AE trained, the higher the FID, see Table 6. For the second ablation, different parts of the projection stay fixed during GAN training: (i) training both F and P_{rand}, (ii) fixed F, training only P_{rand}, (iii) both F and P_{rand} stay fixed. Again, the results in Table 6 suggest that training any part of the projection results in worse performance. We conclude that training the random projection is not advantageous, neither before nor during GAN training.

The signed real logits of the discriminator $sign(D(\mathbf{x}))$ are the portion of the training set that gets positive discriminator outputs. Karras et al. [12] find this a helpful heuristic for quantifying discriminator overfitting. The signed logits should remain constant, which they achieve via adapting the augmentation probability during training. Fig. 16 shows that the logits of the RGB baseline steadily increase throughout training, whereas the logits remain mostly constant for Projected GAN training. This observation coincides with the finding that adaptive augmentation is unnecessary for Projected GANs as the logits are already stable.

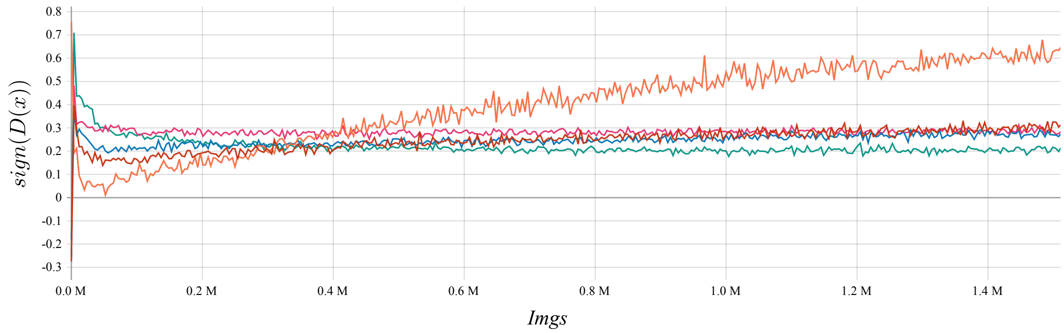

Figure 16: **Signed Discriminator Logits.** For this experiment, we project through F and train with up to four discriminators; we leave the augmentation probability constant. ($|D_i| = 1$: red, $|D_i| = 2$: blue, $|D_i| = 3$: pink, $|D_i| = 4$: green, RGB baseline: orange). For projected GAN training, the logits remain stable throughout training.

5 Implementation Details

This section highlights the codebases we used and details hyperparameters and training configurations.

5.1 Code and Compute

For dataset preparation, training, and evaluation, we build on top of the Stylegan2-ADA codebase[1]. The evaluation employs the official pretrained Inception network to compute FID and KID. For SwAV-FID, we integrate the model of the SwaV-FID codebase[2] using weights by [3]. SAGAN [32] and Gansformers [8] are trained in the Gansformers codebase[3], we use their default hyperparameters. For the feature networks' implementation and weights, we use timm[4], except for R50-CLIP[5]. Lastly, we utilize official implementations for differentiable augmentation[6] and FastGAN[7].

We conduct our experiments on an internal cluster with several nodes, each with up to 8 Quadro RTX 6000 or NVIDIA V100 using PyTorch 1.7.1 and CUDA 11.0.

5.2 Wall Clock-Time

On images at resolution 256^2, the wall-clock training times measured in sec/kimg using 8 Quadro RTX 6000 are shown in Table 7. StyleGAN2 is the fastest overall, which is expected as we enable mixed-precision and use the custom CUDA kernels provided by the authors. These are not available for the FastGAN generator; hence, Projected GAN can only be compared to FastGAN in a fair manner. FastGANs wall-clock times are higher because it uses a reconstruction loss on the discriminator features. This reconstruction loss adds computational overhead. In contrast, projected GANs exhibit lower wall-clock times as we do not need any regularization other than spectral normalization.

Model	sec/kimg
StyleGAN2-ADA	5.6
FastGAN	7.2
Projected GAN	10.1

Table 7: **Training Speed.**

5.3 Hyperparameters

Below, we describe the hyperparameter search for each method. For optimization, we always use Adam [15] with $\beta_1 = 0, \beta_2 = 0.99$, and $\epsilon = 10^{-8}$. All models employ exponential moving average for the generator weights [11].

StyleGAN2-ADA. The official codebase supplies standard configurations of architectures and hyperparameters for different resolutions. Furthermore, an automatic configuration option is available,

[1] https://github.com/NVlabs/stylegan2-ada-pytorch
[2] https://github.com/stanis-morozov/self-supervised-gan-eval/
[3] https://github.com/dorarad/gansformer
[4] https://github.com/rwightman/pytorch-image-models
[5] https://github.com/openai/CLIP
[6] https://github.com/mit-han-lab/data-efficient-gans
[7] https://github.com/odegeasslbc/FastGAN-pytorch

entailing several heuristics for different hyperparameters. We find the automatic option very robust for both architecture and most hyperparameters. The exception is the R1 gradient penalty [21], which is highly dependent on the dataset. For all large datasets, the Gansformer codebase suggests suitable values for StyleGAN2. For the small datasets at 256^2 and 1024^2, we perform a grid search over $\gamma \in \{1, 10, 20, 50\}$. We first train for 1 M *Imgs*, then continue training only the best one. For all AFHQ datasets, we use the same setting as [12]. All experiments employ adaptive discriminator augmentation with the default target value of 0.6.

FastGAN. We replicate the generator and discriminator architecture of the official FastGAN codebase. FastGAN is robust to most hyperparameters; we always use a learning rate of 0.0002 and train with a hinge loss. The only sensitive hyperparameter with direct impact on performance is the batch size. Interestingly, FastGAN profits of smaller batch sizes. The default suggested by [20] is a batch size of 8. We conduct a search over 8, 16, 32, and 64, a batch size of 16 further improves the results, while larger batch sizes decrease performance and even result in divergence in some cases. We employ differentiable augmentation [33] of color, translation, and cutout.

Projected GAN. We use the same architecture, learning rate (0.0002), batch size (64), and hinge loss for all experiments at all resolutions. Compared to FastGAN, we see a slight improvement when increasing model capacity; we double the channel count in each dimension, from a base value of 64 to 128. The multipliers for the base value are as follows (resolution: multiplier): $4^2 : 16, 8^2 : 8, 16^2 : 4, 32^2 : 2, 64^2 : 2, 128^2 : 1, 256^2 : 0.5, 512^2 : 0.25, 1024^2 : 0.125$. We did not observe similar improvements for FastGAN when increasing capacity. We employ differentiable augmentation [33] of color, translation, and cutout. Lastly, to extract feature maps of intermediate layers of the feature networks, both CNNs and visual transformers, we follow the protocol presented in the MiDAS [26] codebase[8]. For all EfficientNets and ResNets, we use features at spatial resolutions $r = \{64^2, 32^2, 16^2, 8^2\}$, for DeiT and ViT, we use layers $l = \{3, 6, 9, 12\}$. The CSM blocks follow a residual design typically used in architectures for dense prediction [26]. The lower-resolution feature is passed through a residual 3x3 convolution block; the higher-resolution feature is added, followed by a second residual block and bilinear upsampling, followed by a 1x1 convolution.

The discriminator architectures are shown in Table 8 where n_i are the channels of the different feature network stages, c_{in} and c_{out} are the input and output channels of the DownBlock DB. A DownBlock consists of a convolution with k size $k = 4$ and stride $s = 2$, BatchNorm, and LeakyReLU with a slope of 0.2. We apply spectral normalization on all convolution layers $Conv$.

Discriminator L_1	Discriminator L_2	Discriminator L_3	Discriminator L_4
$DB(c_in = c_1, c_out = 64)$	$DB(c_{in} = c_2, c_{out} = 128)$	$DB(c_{in} = c_3, c_{out} = 256)$	$DB(c_{in} = c_4, c_{out} = 512)$
$DB(c_{in} = 64, c_{out} = 128)$	$DB(c_{in} = 128, c_{out} = 256)$	$DB(c_{in} = 256, c_{out} = 512)$	$Conv(c_{in} = 512, c_{out} = 1, k = 4)$
$DB(c_{in} = 128, c_{out} = 256)$	$DB(c_{in} = 256, c_{out} = 512)$	$Conv(c_{in} = 512, c_{out} = 1, k = 4)$	
$DB(c_{in} = 256, c_{out} = 512)$	$Conv(c_{in} = 512, c_{out} = 1, k = 4)$		
$Conv(c_{in} = 512, c_{out} = 1, k = 4)$			

Table 8: **Discriminator Architectures.**

Acknowledgments and Disclosure of Funding

We acknowledge the financial support by the BMWi in the project KI Delta Learning (project number 19A19013O). Andreas Geiger was supported by the ERC Starting Grant LEGO-3D (850533). Kashyap Chitta was supported by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039B and the International Max Planck Research School for Intelligent Systems (IMPRS-IS). Jens Müller received funding by the Heidelberg Collaboratory for Image Processing (HCI). We thank the Center for Information Services and High Performance Computing (ZIH) at Dresden University of Technology for generous allocations of computation time. Lastly, we would like to thank Vanessa Sauer for her general support.

[8]https://github.com/intel-isl/MiDaS

References

[1] S. Azadi, M. Tschannen, E. Tzeng, S. Gelly, T. Darrell, and M. Lucic. Semantic bottleneck scene generation. *arXiv.org*, 2019. 3

[2] M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton. Demystifying MMD gans. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018. 2

[3] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv.org*, 2006.09882, 2020. 2, 16

[4] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proc. of the Conf. on Artificial Intelligence (AAAI)*, 2011. 3

[5] K. Desai and J. Johnson. Virtex: Learning visual representations from textual annotations. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[6] P. Dhariwal and A. Nichol. Diffusion models beat gans on image synthesis. *arXiv.org*, 2021. 7

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 1, 2

[8] D. A. Hudson and C. L. Zitnick. Generative adversarial transformers. *arXiv.org*, 2103.01209, 2021. 16

[9] Y. Jiang, S. Chang, and Z. Wang. Transgan: Two transformers can make one strong gan. *arXiv.org*, 2021. 7

[10] A. Karnewar and O. Wang. Msg-gan: Multi-scale gradients for generative adversarial networks. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 7

[11] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2018. 2, 16

[12] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. Training generative adversarial networks with limited data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 1, 4, 5, 6, 8, 15, 17

[13] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila. Analyzing and improving the image quality of StyleGAN. 2020. 4

[14] A. Khosla, N. Jayadevaprakash, B. Yao, and F.-F. Li. Novel dataset for fine-grained image categorization: Stanford dogs. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2011. 3

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2015. 16

[16] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv.org*, abs/1312.6114, 2013. 2

[17] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3D object representations for fine-grained categorization. In *Proc. of the IEEE International Conf. on Computer Vision (ICCV) Workshops*, 2013. 3

[18] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved precision and recall metric for assessing generative models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2

[19] Y. Liao, J. Xie, and A. Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *arXiv.org*, 2109.13410, 2021. 3

[20] B. Liu, Y. Zhu, K. Song, and A. Elgammal. Towards faster and stabilized gan training for high-fidelity few-shot image synthesis. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021. 1, 4, 5, 6, 7, 17

[21] L. Mescheder, A. Geiger, and S. Nowozin. Which training methods for gans do actually converge? In *Proc. of the International Conf. on Machine learning (ICML)*, 2018. 17

[22] S. Morozov, A. Voynov, and A. Babenko. On self-supervised image representations for GAN evaluation. In *Proc. of the International Conf. on Learning Representations (ICLR)*, 2021. 2

[23] B. Neyshabur, S. Bhojanapalli, and A. Chakrabarti. Stabilizing gan training with multiple random projections. *arXiv.org*, 1705.0783, 2017. 1, 2

[24] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proc. Indian Conf. on Computer Vision, Graphics & Image Processing*, 2008. 3

[25] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. *arXiv.org*, 2103.00020, 2021. 2

[26] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 2020. 17

[27] E. Robb, W.-S. Chu, A. Kumar, and J.-B. Huang. Few-shot adaptation of generative adversarial networks. *arXiv.org*, 2010.11943, 2020. 3

[28] M. S. Sajjadi, O. Bachem, M. Lucic, O. Bousquet, and S. Gelly. Assessing generative models via precision and recall. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2

[29] K. K. Singh, U. Ojha, and Y. J. Lee. Finegan: Unsupervised hierarchical disentanglement for fine-grained object generation and discovery. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 7

[30] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 3

[31] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv.org*, 1506.03365, 2015. 3

[32] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *Proc. of the International Conf. on Machine learning (ICML)*, 2019. 16

[33] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han. Differentiable augmentation for data-efficient gan training. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 17

[34] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3