P2D: Plug and Play Discriminator for accelerating GAN frameworks

Min Jin Chong
ByteDance Inc.
minjin.chong@bytedance.com

Krishna Kumar Singh
Adobe Research
krishsin@adobe.com

Yijun Li
Adobe Research
yijli@adobe.com

Jingwan Lu
Adobe Research
jlu@adobe.com

David Forsyth
UIUC
daf@uiuc.edu

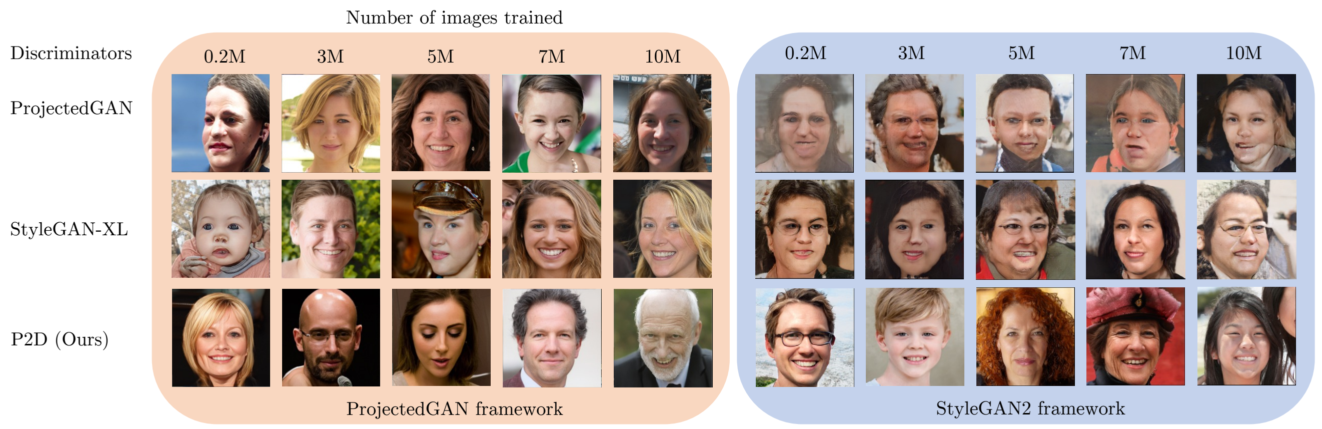

Figure 1. **P2D** is a pretrained plug-and-play discriminator that can be plugged into any adversarial framework. We compare different discriminators in the widely used ProjectedGAN and StyleGAN2 framework at different checkpoints of training. Results clearly show the advantage of P2D in terms of accelerating the GAN training across both frameworks.

Abstract

Most image classification tasks benefit from using pretrained feature stacks. In contrast, the discriminator for adversarial losses is trained at the same time as the model because using a pretrained feature stack yields a very poor model. Recent work has shown that an implicit regularization scheme allows using pretrained feature stacks to construct a discriminator, which improves both speed of training and quality of results. However, we observe that changes in hyperparameters can result in substantial changes in generator behavior.

We show that using a modified version of the R1 regularization scheme that regularizes in the feature space instead of the image space results in a plug-and-play discriminator – P2D. Our scheme results in a method that is highly stable across changes in architecture and framework; that significantly speeds up training; and that produces models that reliably beat SOTA in quality. The huge reduction in training resources required means that P2D could make training powerful generative models over specific datasets accessible to most researchers.

1. Introduction

One of the popular trends in recent computer vision research is to leverage pretrained features in various ways [2, 23, 30, 32], and train-from-scratch feature stacks are becoming less common. We describe a method, P2D, to apply a pretrained feature stack to GAN training that is *plug-and-play*. A general plug-and-play solution for accelerating GAN training should be:

1. **Generator-agnostic**: It should work for all generators in whatever codebase.
2. **Framework-stable**: Changes in framework should have minimal effect. Here *framework* refers to the

set of choices that apply to training a particular generator, including training procedures, hyperparameters and choice of losses (*cf*. cycle loss [36] vs straightforward adversarial loss).
3. **Transferable**: It should be easy to integrate into different codebases.
4. **Competitive**: It should accelerate training and achieve competitive results as measured by common metrics.

In Generative Adversarial Networks (GANs, originally [8]; major improvements in [12,14,15,22]), it is common to train discriminators from scratch. There are consequences: poor discriminators affect generator performance; training times are very long. An alternative choice would be using pretrained feature stacks in discriminators. But simply plugging a pretrained feature stack into GAN does not work, because the discriminator overfits immediately, resulting in unhelpful gradients. Recent work [17, 24, 25] demonstrates significant speed and quality improvement by using foundation models in discriminators by adopting a strong implicit regularizer that makes it hard for the discriminator to overfit. As we show in Section 3, the approaches of [24, 25] apply well to certain architectures and certain frameworks, but not to others due to insufficient regularization. They are thus not plug-and-play.

Our method, Plug and Play Discriminator (P2D), is a simple drop-in discriminator that is easy to integrate and significantly speeds up GAN training. Similar to Sauer *et al.*, P2D uses foundation models as feature extractors for the discriminator. However, to make P2D plug-and-play, we introduce a novel feature regularization loss that stabilizes the training process. We demonstrate that P2D is able to significantly improve training speed and quality across different generators, frameworks, and codebases. For example, training a StyleGAN2 on 256 $\times$ 256 FFHQ with a single NVIDIA A40 GPU takes only 34 hours to match the official reported FID [15], roughly a 23$\times$ increase from 781 hours. As another example, P2D applied to CycleGAN (a completely different framework) results in very strong FID improvements on standard datasets (including halving the FID of `horse2zebra`). We summarize our comparison with Sauer *et al.* in Table 1.

Our contributions are

1. We introduce a novel feature R1 regularization loss that stabilizes training P2D with foundation models.
2. We introduce P2D, a plug-and-play discriminator that is stable during training with any generator architectures, or frameworks within reasonable range of hyperparameters and significantly boosts training speed and image quality.

2. Related Work

Making GAN Training Faster is widely studied. Ngxande *et al.* [20] replace convolution layers with depthwise separable convolution layers to reduce trainable parameters, thereby reducing computation requirements. Zhong *et al.* [35] augment the training procedure with an additional adversarial loop for the discriminator, accelerating the training. Sinha *et al.* [27] subsample core-sets from large batches to improve batch coverage, leading to improved performance. No method results in sufficient speedup for it to be widely adopted. FastGAN [18] introduces an entirely new GAN architecture that aims to improve gradient signals, leading to significantly faster convergence with very competitive results. These methods apply to specific architectures or frameworks. In contrast, our method is plug-and-play – easily integrated into different frameworks and tasks while yielding a significant boost in training speed and image quality.

Pretrained Feature Stacks are now usual practice in computer vision. Such models such as CLIP [21] and EfficientNet [28] are trained on millions of images and have feature spaces that are very useful for downstream tasks. Learning a classifier on top of the features given by these models should be significantly easier than training a discriminator from scratch. But doing so disrupts the adversarial training, leading to poor results [24]. Kumari *et al.* [17] overcome this instability by first performing a standard GAN training and using a foundation model after the training has stabilized. While they show impressive quality gains, their process is not framework-agnostic. One must add new discriminators as the training progresses which requires significant changes to how GANs are trained.

ProjectedGAN [24] and StyleGAN-XL [25] by Sauer *et al.* hypothesize that foundation models cause unstable GAN training because their deep semantic features are easily overfitted by the discriminator. Sauer *et al.* propose to obstruct overfitting by applying a fixed random projection layer to features before passing them to the discriminator. The result is more stable training. In addition, DiffAug [34], a differentiable augmentation technique, is used to reduce the discriminator overfitting. ProjectedGAN and StyleGAN-XL share the same discriminator framework, with StyleGAN-XL having a ViT [7] in addition to an EfficientNet [28] for its feature extractor backbone. Unlike Kumari *et al.*, they do not require training a separate discriminator and training is the same start to finish. Section 3 demonstrates that these methods are significantly unstable with change of framework and codebase.

3. Is the Problem Solved Already?

We investigate whether ProjectedGAN [24] and StyleGAN-XL [25] are plug-and-play. We use Fréchet inception distance (FID) [10], a common metric for evaluating the quality and diversity of GANs. We evaluate using different generators and frameworks, but confine

	P2D	Sauer *et al.* [24, 25]
Accelerates training	✓	✓
Generator-agnostic	✓	✗
Framework-stable	✓	✗
Transferable	✓	✗
Robust to hyperparameters	✓	✗
Competitive results	✓	✓

Table 1. P2D is plug-and-play because it is able to achieve state-of-the-art results across different frameworks. ProjectedGAN and StyleGAN-XL by Sauer *et al.* do not provide this flexibility as verified by numbers in Table 2.

the evaluation to the FFHQ dataset [14]. We do so because problematic behavior on one dataset is enough to demonstrate plug-and-play problems, and because this is a popular face dataset with 70k images commonly used for benchmarking GANs.

3.1. Codebase and Generator

A plug-and-play discriminator should work for different codebases, generators, and frameworks. As a baseline, we test both ProjectedGAN and StyleGAN-XL discriminators in the native ProjectedGAN framework which uses a FastGAN generator. Note that StyleGAN-XL is built on top of ProjectedGAN and thus shares most of the codebase, with significant differences only in the generator. We refer to this codebase as Sauer *et al*. codebase.

We also investigate StyleGAN2 [14]. StyleGAN2's unique style-based architecture admits a variety of interesting image editing capabilities [4–6, 26, 30, 33]. Because Sauer *et al*. codebase is built on top of the official StyleGAN2 codebase, we choose to test the discriminators on Rosinality's reimplemented StyleGAN2 codebase [1] instead. This offers a different codebase on a different generator (StyleGAN2 vs FastGAN), and for them to be plug-and-play, ProjectedGAN and StyleGAN-XL should be expected to perform well.

3.2. Frameworks

A plug-and-play discriminator should be robust to change of framework. So, for example, various sensible choices of hyperparameter should all result in good behavior. There are many tweakable hyperparameters. We focus on varying a few commonly adjusted hyperparameters that have significant effects. For fair comparison, our baseline hyperparameter configuration for all training is a batch size of 8 (to fit in most GPUs) together with differentiable augmentation (as advised by [17, 24]).

Batch size: ProjectedGAN and StyleGAN-XL are trained on large batch sizes, at 64 and 256 respectively. Large batch sizes are typically unachievable without a sufficiently large number of GPUs or specific implementation of gradient accumulation within their codebase.

Differentiable Augmentations: By default, DiffAug [34] is used by both Sauer *et al.* and Kumari *et al.* [17] to achieve SOTA results. This is uncommon as differentiable augmentations are typically used only when training with very small datasets and could even hurt the performance if the training data is sufficiently large [13]. FFHQ is sufficiently large and GANs are able to achieve SOTA results on it without differentiable augmentations [15].

3.3. Results and Discussion

As Table 2 shows, both ProjectedGAN and StyleGAN-XL are significantly affected by the change of codebase and the change of framework. On Rosinality's StyleGAN2 codebase, both discriminators cause unstable training and severe mode collapse, indicating that they are not agnostic to the change in the generator and codebase. As detailed in Section 5.3, we could not replicate the reported FIDs of ProjectedGAN on their official codebase. Both methods are very sensitive to batch sizes and DiffAug (a change in framework), with small batch sizes and removal of DiffAug having significantly worse FIDs. This is especially so for StyleGAN-XL, where using small batch sizes increased the FID from 3.22 to 27.1.

As Sauer *et al.* point out, it is challenging to use a pretrained feature extractor because the discriminator can overfit some features, providing bad gradient signals for the generator to learn. GAN training can be regularized by a variety of effects that are hard to account for, including generator structure and hyperparameter details. We speculate ProjectedGAN and StyleGAN-XL perform well in their own framework because of regularization effects in the generator. ProjectedGAN uses a FastGAN generator which is more "stable" than StyleGAN2 due to its Skip-Layer channel-wise Excitation module [18]. StyleGAN-XL's generator is trained in a progressively growing fashion, which strongly regularizes it. Changes in the generator, framework, and codebase appear to confirm Sauer *et al.*'s discriminators are not sufficiently regularized for stable training when the regularization effects in the generator itself are less pronounced. In turn, the way to obtain plug-and-play discriminators is to regularize the discriminator explicitly.

4. Methodology

We wish to regularize a set of discriminators, built as learned classifiers on top of frozen pretrained feature stacks (in our case, obtained from foundation models). The feature space produced by a foundation model is exceptionally rich and informative. In turn, the classifier might focus on one or a few features early in training and ignore the rest. We expect this behavior to produce poor gradient signals, because

	Sauer *et al.* Codebase			StyleGAN2 Codebase (Rosalinity)	
	Baseline	+Large batch size	+No DiffAug	Baseline	No DiffAug
ProjectedGAN	6.81	4.70	10.13	159	217
StyleGAN-XL	27.1	3.22	10.8	132	192
P2D (Ours)	**2.32**	**2.37**	**2.42**	**2.85**	**2.39**

Table 2. We compare FIDs of training discriminators with different hyperparameters across two different codebases and frameworks for FFHQ256. P2D is plug-and-play because it gives strong FID results for all cases while ProjectedGAN and StyleGAN-XL perform poorly outside of their intended settings. Refer to Figure 1 for qualitative comparisons.

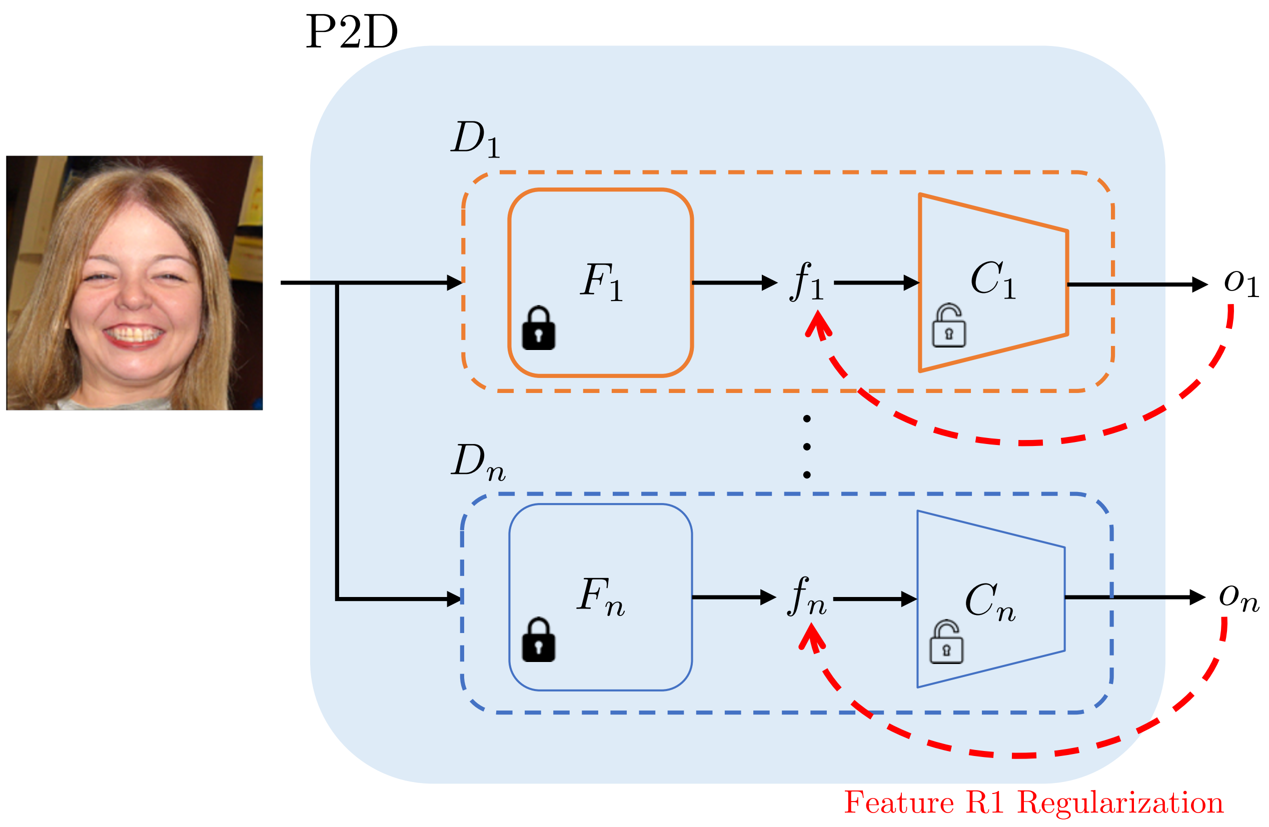

Figure 2. P2D uses n foundation models F_i as frozen pretrained feature stacks and regularizes the classifiers C_i with Feature R1 Regularization. P2D works well with different generators and frameworks, see Section 5.2.

Method	FID
Baseline	4.98
P2D (Ours)	**2.39**
(A) No relativistic loss	2.87
(B) Random init G	2.84
(C) Full pretrained G	2.97

Table 3. We look at ablations on StyleGAN2 trained on FFHQ for 10M images.

we expect each feature direction in a foundation model to have some useful information about an image. This suggests using a form of regularization that encourages the discriminator to use the features relatively evenly.

The R1 regularizer [19] is commonly used as a form of gradient penalty that penalizes the discriminator for deviating from the Nash equilibrium. Given x as the input, D as the discriminator, and γ as the weighting term, the R1 regularizer takes the form

$$\mathcal{L}_{\text{R1}} = \frac{\gamma}{2}\mathbb{E}_{x\sim p_D}\left[||\nabla_x D(x)||_2^2\right] \tag{1}$$

and encourages the gradient with respect to each input to be close to zero, preventing the model from overfitting to a particular input. We adopt a variant of this regularizer.

4.1. Feature R1 regularization

Understanding our regularizer requires a review of how our discriminators are integrated into the model. Figure 2 summarizes our approach. We have multiple feature extractors (as in [24, 25]); write $F_{1,\dots,k}$ for feature extractors. Each is taken from a pretrained foundation model, and frozen. On top of each we place a trainable classifier; write $C_{1,\dots,k}$ for the classifiers. Thus, given an image x, the output logits for discriminator D_i is

$$o_i = D_i(x) = C_i(F_i(x)) \tag{2}$$

Because we keep F_i fixed and only update C_i during training, any instability can be caused only by C_i. To prevent this, we apply R1 regularization to C_i *with respect to its input features*, so

$$\mathcal{L}_{\text{FR1}} = \frac{\gamma}{2}\mathbb{E}\Big[||\nabla_{f_i} C_i(f_i)||_2^2\Big] \tag{3}$$

Unlike normal R1 regularization, we are computing the loss with respect to the features extracted from foundation models ($F_i(x)$) instead of image pixels (x). We call this procedure *Feature R1 Regularization*. Computing gradients over only C_i instead of D_i is cheaper as it avoids computing gradients through multiple layers of large foundation models. Furthermore, our experiments show that using R1 regularization directly leads to poor results.

Heuristics for R1 weighting: A plug-and-play discriminator should not have too many hyperparameters to be simple for users. In P2D, because we are using multiple feature extractors, their outputs f_i might have varying sizes. As a result, we could need to tune γ separately for each feature extractor in Equation 3. Simple experiments show that using the same γ for each results in poor overall training stability.

A simple procedure tunes γ_i automatically and successfully. We observed that features with larger L2 norms tend to require a larger gradient penalty (larger γ), we thus scale γ_i by the feature's average L2 norm across the batch. Thus, we set γ_i using

$$\gamma_i = \frac{\lambda}{N}\sum_{j=1}^{N} ||f_{ij}||_2 \tag{4}$$

where f_{ij} represents the jth feature in the batch of features extracted by F_i and λ is a constant weighting term that is used for all discriminators. Now, instead of adjusting γ_i for each feature extractor, we only need to tune a single λ for all discriminators. Empirically, we notice that the same λ can be used across different datasets and generators, so that P2D thus does not require any form of hyperparameter tuning.

4.2. Loss function

There is no consensus on the best loss function for GANs, with different frameworks using different versions of the loss function. For example, StyleGAN uses the non-saturating loss [8] while ProjectedGAN uses the hinge loss. In our framework, we show that using relativistic loss [11] along with the nonsaturating loss results in better stability and provides a substantial improvement in results (see Table 3 A).

Relativistic loss changes the discriminator's output to give the probability of the real data being *more realistic* than the sampled fake data, on average. It is of the form

$$\hat{D}(x) = \begin{cases} D(x) - \mathbb{E}_{x_f} D(x_f) & \text{if x is real} \\ D(x) - \mathbb{E}_{x_r} D(x_r) & \text{if x is fake} \end{cases} \tag{5}$$

Because we have several feature extractors, we have multiple discriminator outputs. To compute the final loss, we average the losses over all discriminator outputs.

4.3. Pretrained Generator

Specifically for StyleGAN2 [15], Grigoryev *et al.* [9] showed using an ImageNet-pretrained generator speeds up the training and gives superior results compared to training from scratch. In our experiments, we notice that initializing with an ImageNet-pretrained generator generally speeds up early training but gives poorer final results, see Table 3 B and C.

We hypothesize that the drop in performance is due to the mapping network of StyleGAN2. The mapping network is a MLP that produces the style code used to modulate the convolution layers of the generator. For stability reasons, during training, the learning rate of the mapping network is set to 100 times lower than that of the rest of the generator. Due to this low learning rate, it is difficult for the mapping network to change drastically if our new dataset differs from the pretrained dataset significantly. This results in poorer performance as the mapping network is not able to properly disentangle the representation.

Instead of initializing the entire generator with the pretrained weights, we leave out the mapping network and choose to randomly initialize them. We refer to this as partial initialization. Doing so gives us a boost in training speed and final results, see P2D in Table 3.

5. Experiments

In this section, we first talk about the implementation details in Section 5.1. Next, in Section 5.2, we show P2D is generator-agnostic and framework stable. After that, in section 5.3, we show P2D generates better results than the competing methods. Finally, in section 5.4, we compare the wallclock time of P2D with the baseline approaches.

Discriminators (Frameworks)	FID	Imgs	FID	Imgs	FID	Imgs	FID	Imgs
	FFHQ		Churches		AFHQ		Art Painting	
StyleGAN2 (StyleGAN2)*	3.62	25 M	3.39	88 M	-	-	43.07	3.2 M
ProjectedGAN (ProjectedGAN)*	3.39	7.1 M	1.59	9.2 M	-	-	28.0	0.8 M
StyleGAN2 (StyleGAN2)	4.98	9.4 M	4.00	9.4 M	6.36	3.4 M	46.7	2.0 M
ProjectedGAN (ProjectedGAN)	4.70	7.0 M	2.04	9.7 M	2.16	9.0 M	28.7	1.6 M
StyleGAN-XL (ProjectedGAN)	3.22	9.6 M	1.78	8.6 M	2.42	9.6 M	27.2	7.7 M
P2D (ProjectedGAN)	**2.32**	9.4 M	2.08	4.8 M	**1.65**	10 M	**25.5**	9.6 M
P2D (StyleGAN2)	2.39	9.8 M	**1.55**	9.4 M	2.10	8.0 M	26.2	8.2 M

Table 4. P2D generally gets better FID for a range of datasets compared to ProjectedGAN and StyleGAN-XL without being sensitive to hyperparameters and frameworks, see Table 2. Brackets indicate the framework we apply the discriminators to. $*$ represents results reported in [24]; the rest are reruns by us.

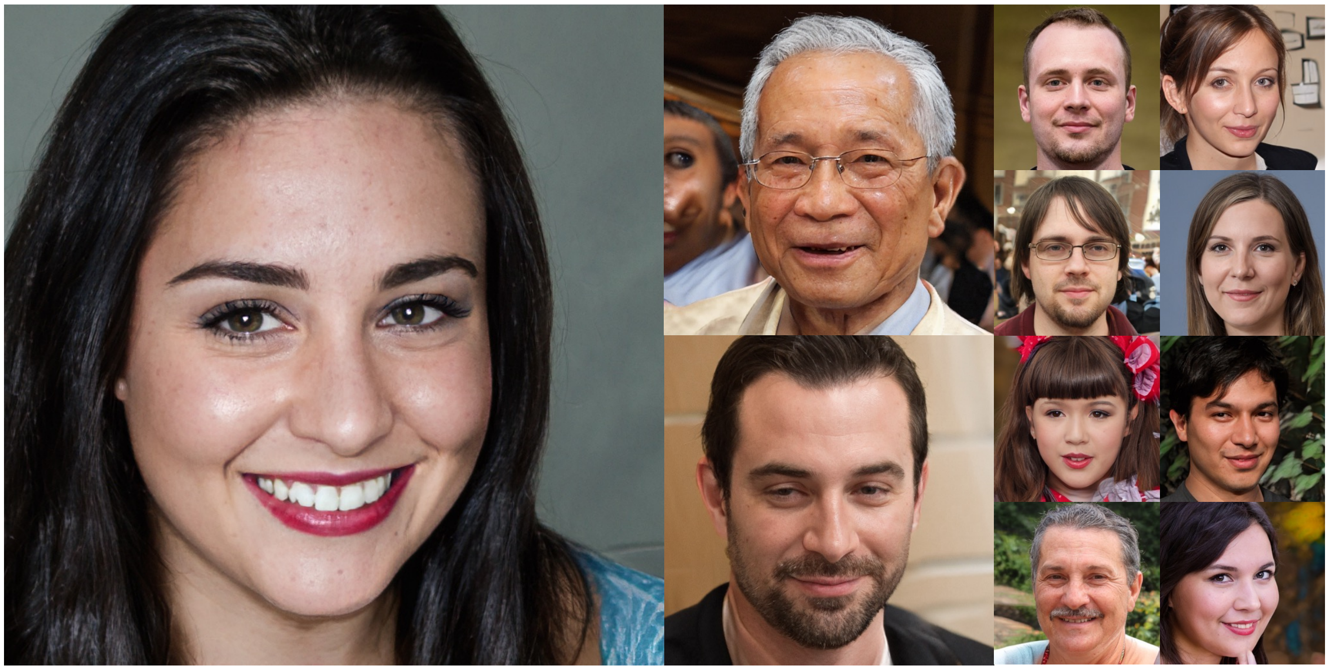
Figure 3. Random samples from P2D. P2D achieves an FID of 2.19 on FFHQ1024, beating the baseline StyleGAN2 of FID 2.84.

	horse2zebra	facades	ukiyoe2photo
CycleGAN	63.6	108	109
P2D (Ours)	**31.9**	**81.6**	**87.1**

Table 5. P2D even works for CycleGAN, a vastly different framework from standard GANs, improving FIDs significantly from the baseline.

5.1. Implementation

Similar to StyleGAN-XL, P2D uses two foundation models as feature extractors, one CNN-based (EfficientNet [28]), another ViT-based [7]. However, instead of using DeiT [29], we follow the suggestion of Kumari *et al.* [17] and use a CLIP [21] encoder instead. Similarly, we extract several intermediate features from the foundation models and train a shallow classifier on top of each feature. Our classifiers consist of several ResBlocks along with a minibatch standard deviation layer from ProGAN [12]. The major difference between P2D and StyleGAN-XL is we remove the random projection layer and explicitly regularize P2D with feature R1 regularization from Section 4.1. All experiments with P2D implement relativistic loss from Section 4.2. Additionally, for StyleGAN2 experiments, P2D uses the partial initialization we discussed in Section 4.3.

We use the default hyperparameter settings for ProjectedGAN and StyleGAN-XL. For P2D, we use Adam optimizer [16] learning rate of 0.002 and betas of 0 and 0.99. We use the R1 weighting heuristic from Section 4.1 with a fixed $\lambda = 0.2$ for all experiments. More implementation details are in the Supplementary Material.

	Time (hours) taken to reach target FID			
	FFHQ (4.70)	Churches (2.08)	AFHQ (2.42)	Art Painting (28.7)
P2D (ProjectedGAN)	**21**	96	**43**	23
ProjectedGAN	76	106	82	**18**
StyleGAN-XL	47	**85**	151	19
	FFHQ (4.98)	Churches (4.00)	AFHQ (6.36)	Art Painting (46.7)
P2D (StyleGAN2)	**25**	**8**	**9**	**1**
StyleGAN2	240	251	92	50

Table 6. For the **top** table, we compare P2D trained in ProjectedGAN framework with ProjectedGAN and StyleGAN-XL. P2D reaches target FID (in brackets) significantly faster on FFHQ and AFHQ while matching the speed for the other datasets. For the **bottom** table, we compared P2D trained in StyleGAN2 framework with baseline StyleGAN2. P2D trains significantly faster across all datasets.

5.2. Is P2D generator-agnostic and framework-stable?

Following Section 3.1, we test P2D on the same experiments and show the results in Table 2. Unlike ProjectedGAN and StyleGAN-XL, P2D is stable for both codebases and different hyperparameters, producing strong and consistent FIDs across all setups. Refer to Figure 1 for qualitative results.

P2D on I2I: We have validated that P2D works across two GAN frameworks. However, we can take it one step further. P2D is not just useful in a normal GAN generation task and we demonstrate its success in the task of image-to-image translation (I2I). We integrate P2D into CycleGAN [36], a popular I2I framework, while keeping all hyperparameters and settings the same. The only modification we made to P2D is to change all convolution strides to 1 to convert it to a patch-based discriminator which is more suitable for I2I. As shown in Table 5, P2D significantly outperforms the CycleGAN baseline. In particular, P2D reduces the FID of horse2zebra by half, from 63.6 to 31.9. This lends more evidence that P2D is generator-agnostic and framework-stable.

Integrating P2D into ProjectedGAN, StyleGAN2, and CycleGAN frameworks was simple and required no hyperparameter tweaks. We thus claim that P2D is transferable.

5.3. Does P2D produce competitive results?

A plug-and-play discriminator must produce competitive results across different datasets. We again compare P2D with ProjectedGAN and StyleGAN-XL under their best settings (*e.g.* FastGAN generator, large batch size, *etc.*) over a number of datasets. For P2D, we train in ProjectedGAN and StyleGAN2 framework with a small batch size of 8 and without DiffAug. For datasets, we use FFHQ, LSUN-Churches [31], AFHQ [3], and Art painting [24]. For Art painting dataset, we train P2D with DiffAug because it contains only 1000 images.

Table 4 reports our results. In our own experiments, we were not able to reproduce the FIDs as reported in ProjectedGAN even when using the official codebase and default hyperparameters. Thus in the table, we report their reported FIDs denoting with a $*$, and also report our own reruns. The brackets represent the framework we train the discriminator on.

Even under unfavorable comparisons with the reported FIDs that we cannot reproduce, P2D still achieves the best FIDs across all the datasets for the two frameworks. For FFHQ and LSUN-Churches, we achieve an FID of 2.39 and 1.55 which to the best of our knowledge, is the current state-of-the-art for a StyleGAN2 generator.

On 1024×1024 resolution: While the foundation models are trained on 224×224 resolution images, we test its capabilities to generalize to FFHQ1024. We again use the StyleGAN2 framework written by Rosinality for this experiment. Because of the computationally expensive attention layers in CLIP, we resize the images to 224×224 before passing them to CLIP. This is less of a problem for EfficientNet because it is a convolutional network and thus, to ensure that we do not lose the high frequency details, we directly pass the high resolution images to EfficientNet. P2D is able to generate high quality results, achieving an FID of 2.19, beating out the reported StyleGAN2 baseline of 2.84. See Figure 3 for qualitative results.

5.4. Wallclock Time

We compare the wallclock training time between P2D, ProjectedGAN, StyleGAN-XL, and StyleGAN2. Because ProjectedGAN and StyleGAN-XL do not work in StyleGAN2 framework (see Section 3.1), we compare P2D and StyleGAN2 separately.

We first compare P2D with ProjectedGAN and StyleGAN-XL. For fair comparisons, we implement everything in ProjectedGAN's codebase and framework, with hyperparameters from Section 3.2 that are favorable for them. In the first table of Table 6, we show the training time needed to reach a target FID over several datasets on a single NVIDIA A40. We choose the target FID as the largest converged FID among the 3 methods so that all methods

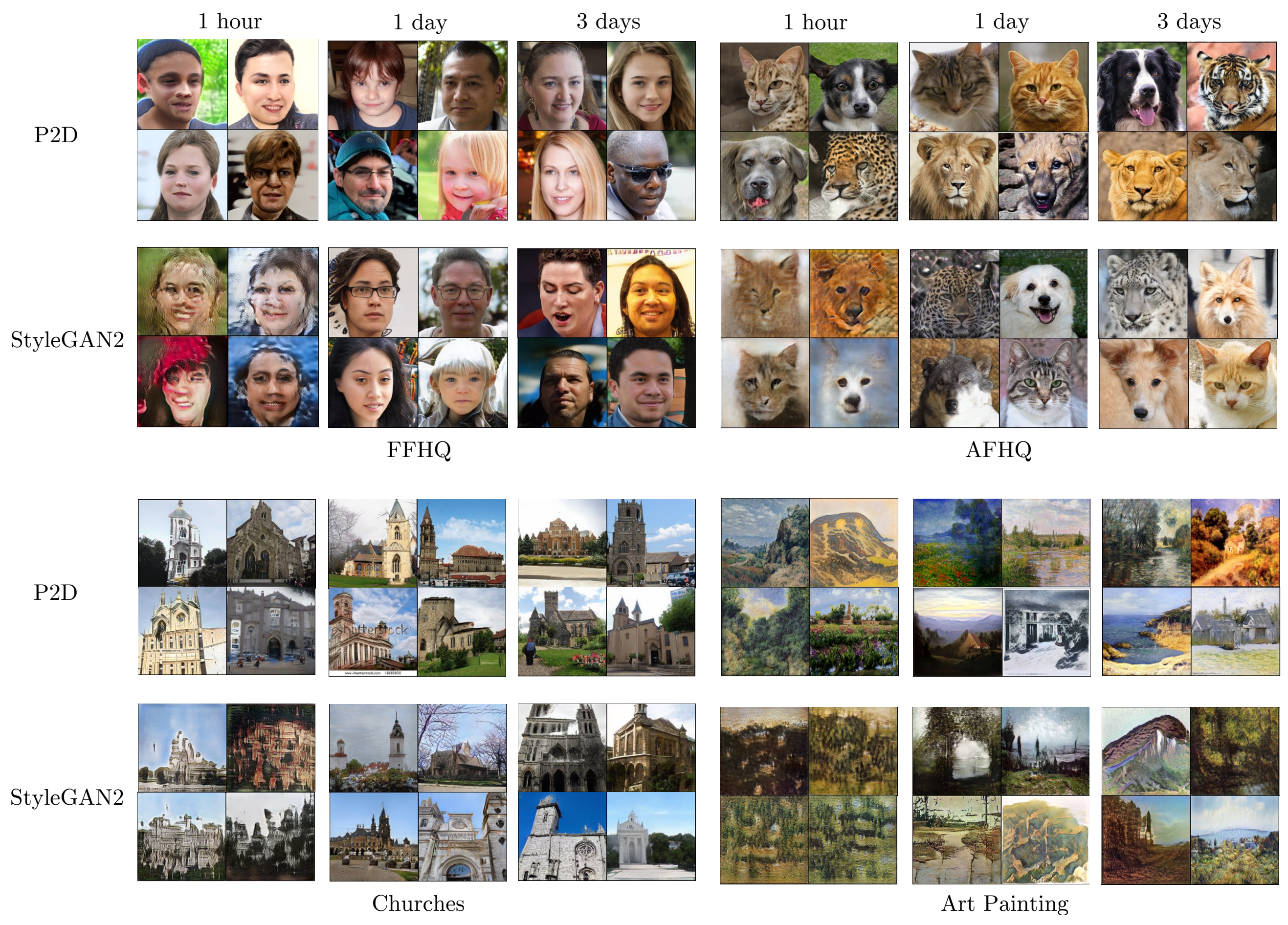

Figure 4. We compare P2D with baseline StyleGAN2 at different wallclock time. Overall P2D produces better quality and more diverse images at the same timestep. In just 1 hour, P2D produces cohesive images with good textures while StyleGAN2 struggle to learn proper textures. In 3 days, P2D produces very good photorealistic images while StyleGAN2 still struggles with textures.

will achieve that FID at some point in their training. P2D trains significantly faster on FFHQ and AFHQ, requiring only about half the time to reach the target FID compared to ProjectedGAN and StyleGAN-XL. On Churches and Art Painting, P2D's training speed is comparable with the other methods. While we cannot claim that P2D always trains faster than ProjectedGAN or StyleGAN-XL, P2D is significantly more stable across generators, frameworks, and codebases.

Next, we compare P2D and StyleGAN2 in the second table of Table 6. For all datasets, P2D is significantly faster than StyleGAN2 which trains its discriminator from scratch. We visualize this in Figure 4 where we plot the images P2D and StyleGAN2 produce at different timesteps. Within 1 hours, P2D is getting good textures and cohesive images while StyleGAN2 struggles with textures. After training for only 1, P2D generates good quality and diverse images, while obvious artifacts can be seen from StyleGAN2. By day 3, P2D is already producing photo-realistic images that beat the officially reported FID from StyleGAN2.

6. Conclusion

In this work, we introduce P2D, a plug-and-play discriminator that can be easily integrated into most GAN frameworks to accelerate and improve their training. By stabilizing the training with the novel feature R1 regularization, P2D is able to achieve competitive FIDs across a range of generators, frameworks, and different hyperparameter choices. We hope that P2D can make training state-of-the-art GANs achievable for more people. In future work, we will explore more variety of pretrained feature stacks for the discriminator. We will also explore P2D's applicability to more tasks.

References

[1] stylegan2-pytorch. https://github.com/rosinality/stylegan2-pytorch, 2019. 3

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022. 1

[3] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 7

[4] Min Jin Chong, Wen-Sheng Chu, Abhishek Kumar, and David Forsyth. Retrieve in style: Unsupervised facial feature transfer and retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3887–3896, October 2021. 3

[5] Min Jin Chong and David Forsyth. Jojogan: One shot face stylization. *arXiv preprint arXiv:2112.11641*, 2021. 3

[6] Min Jin Chong, Hsin-Ying Lee, and David Forsyth. Stylegan of all trades: Image manipulation with only pretrained stylegan. *arXiv preprint arXiv:2111.01619*, 2021. 3

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 6

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. 2, 5

[9] Timofey Grigoryev, Andrey Voynov, and Artem Babenko. When, why, and which pretrained gans are useful? *arXiv preprint arXiv:2202.08937*, 2022. 5

[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 2

[11] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018. 5

[12] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 2, 6

[13] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *Advances in Neural Information Processing Systems*, 33:12104–12114, 2020. 3

[14] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 2, 3

[15] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *Proc. CVPR*, 2020. 2, 3, 5

[16] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[17] Nupur Kumari, Richard Zhang, Eli Shechtman, and Jun-Yan Zhu. Ensembling off-the-shelf models for gan training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10651–10662, 2022. 2, 3, 6

[18] Bingchen Liu, Yizhe Zhu, Kunpeng Song, and Ahmed Elgammal. Towards faster and stabilized gan training for high-fidelity few-shot image synthesis. In *International Conference on Learning Representations*, 2020. 2, 3

[19] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Which training methods for gans do actually converge? In *International Conference on Machine Learning (ICML)*, 2018. 4

[20] Mkhuseli Ngxande, Jules-Raymond Tapamo, and Michael Burke. Depthwisegans: Fast training generative adversarial networks for realistic image synthesis. In *2019 Southern African Universities Power Engineering Conference/Robotics and Mechatronics/Pattern Recognition Association of South Africa (SAUPEC/RobMech/PRASA)*, pages 111–116. IEEE, 2019. 2

[21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 2, 6

[22] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 2

[23] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. 2022. 1

[24] Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 2, 3, 5, 6, 7

[25] Axel Sauer, Katja Schwarz, and Andreas Geiger. Stylegan-xl: Scaling stylegan to large diverse datasets. volume abs/2201.00273, 2022. 2, 3, 5

[26] Yujun Shen and Bolei Zhou. Closed-form factorization of latent semantics in gans. In *CVPR*, 2021. 3

[27] Samarth Sinha, Han Zhang, Anirudh Goyal, Yoshua Bengio, Hugo Larochelle, and Augustus Odena. Small-gan: Speeding up gan training using core-sets. In *International Conference on Machine Learning*, pages 9005–9015. PMLR, 2020. 2

[28] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International*

conference on machine learning, pages 6105–6114. PMLR, 2019. 2, 6

[29] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 6

[30] Xintao Wang, Yu Li, Honglun Zhang, and Ying Shan. Towards real-world blind face restoration with generative facial prior. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 3

[31] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 7

[32] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 1

[33] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In *CVPR*, 2021. 3

[34] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 2, 3

[35] Jiachen Zhong, Xuanqing Liu, and Cho-Jui Hsieh. Improving the speed and quality of gan by adversarial training. *arXiv preprint arXiv:2008.03364*, 2020. 2

[36] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. 2, 7