PA-GAN: IMPROVING GAN TRAINING BY PROGRESSIVE AUGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite recent progress, Generative Adversarial Networks (GANs) still suffer from training instability, requiring careful consideration of architecture design choices and hyper-parameter tuning. The reason for this fragile training behaviour is partially due to the discriminator performing well very quickly; its loss converges to zero, providing no reliable backpropagation signal to the generator. In this work we introduce a new technique - *progressive augmentation of GANs (PA-GAN)* - that helps to overcome this fundamental limitation and improve the overall stability of GAN training. The key idea is to gradually increase the task difficulty of the discriminator by progressively augmenting its input space, thus enabling continuous learning of the generator. We show that the proposed progressive augmentation preserves the original GAN objective, does not bias the optimality of the discriminator and encourages the healthy competition between the generator and discriminator, leading to a better-performing generator. We experimentally demonstrate the effectiveness of the proposed approach on multiple benchmarks (MNIST, Fashion-MNIST, CIFAR10, CELEBA) for the image generation task.

1 INTRODUCTION

Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) are a recent development in the field of deep learning, that have attracted a lot of attention in the research community (Radford et al., 2016; Salimans et al., 2016; Arjovsky et al., 2017; Karras et al., 2018). GANs fall into the category of generative models, i.e. models that allow sampling of new data points from encoded high-dimensional data distributions, such as images. The GAN framework can be formulated as a competing game between the generator and the discriminator. Mathematically, training GANs requires solving a min-max problem. Since both the generator and the discriminator are typically parameterized as deep convolutional neural networks with millions of parameters, optimization is notoriously difficult in practice (Arjovsky et al., 2017; Gulrajani et al., 2017; Miyato et al., 2018).

The difficulty lies in maintaining the healthy competition between the generator and discriminator. A commonly occurring problem arises when the discriminator overshoots, leading to escalated gradients and oscillatory GAN behaviour (Mescheder et al., 2018). As a result the generator fails to learn the multimodal structure of the true distribution. Moreover, the supports of the data and model distributions typically lie on low dimensional manifolds and are often disjoint (Arjovsky & Bottou, 2017). Consequently, there exists a nearly trivial discriminator that can perfectly distinguish real data samples from synthetic ones. Once such a discriminator is produced, its loss quickly converges to zero and the gradients used for updating parameters of the generator become useless.

In this work we introduce a new technique to overcome this problem - *progressive augmentation of GANs (PA-GAN)* - that helps to control the behaviour of the discriminator and thus improve the overall training stability. The key idea is to progressively augment the input of the discriminator network with auxiliary random variables, enlarging the sample space dimensionality, in order to gradually increase the discrimination task difficulty (see Figure 1). In doing so, the discriminator can be prevented from becoming over-confident, enabling continuous learning of the generator. As opposed to standard data augmentation techniques (e.g. rotation, cropping, resizing), the proposed progressive augmentation does not directly modify the data samples, but rather is structurally appended to them. In particular, for the single level augmentation along with the data sample $\boldsymbol{x}$ the discriminator takes also as input the binary random variable $s \in \{0, 1\}$. The class of the augmented sample $(\boldsymbol{x}, s)$ is then set based on the combination $\boldsymbol{x}$ with s, resulting in real and synthetic samples contained in both true and fake classes. This presents a more challenging task for the discriminator, as it needs to tell the real and synthetic samples apart and additionally understand the association rule. We can further increase the task difficulty of the discriminator by progressively augmenting its input space and enlarging the dimensionality of s during the course of training.

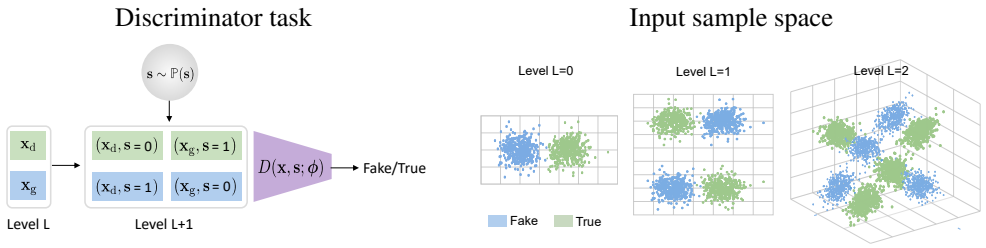

Figure 1: Visualization of progressive augmentation. With each extra augmentation level ($L \rightarrow L + 1$) the dimensionality of the discriminator input space is increased and the discrimination task gradually becomes harder. This strategy prevents the discriminator from easily finding a decision boundary between two classes and thus leads to meaningful gradients for the generator updates.

We show that the proposed progressive augmentation preserves the original GAN objective and is an outcome of a systematic derivation. In contrast to prior work (Salimans et al., 2016; Sønderby et al., 2017; Arjovsky & Bottou, 2017), it does not bias the optimality of the discriminator (see Sec. 4). Structurally augmenting the input sample space and mapping it to higher dimensions not only makes the discriminator task more challenging, but, in addition, encourages the generator to explore various paths towards the true data distribution, enabling it to do more than just intelligent memorization of the training set or interpolation between two data samples.

Our technique is orthogonal to existing work, it can be successfully employed with other regularizations strategies (Miyato et al., 2018; Gulrajani et al., 2017) and different network architectures (Chen et al., 2016; Radford et al., 2016), which we demonstrate in Sec. 5. We experimentally show the effectiveness of the progressive augmentation of GANs for image generation tasks on multiple benchmarks (MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017), CIFAR10 (Krizhevsky, 2009), CELEBA (Liu et al., 2015)) across different evaluation metrics (IS (Theis et al., 2016), FID (Huszár, 2015), KID (Bińkowski et al., 2018)).

2 Related Work

Many recent works have focused on improving the stability of GAN training and the overall visual quality of generated samples (Radford et al., 2016; Roth et al., 2017; Karras et al., 2018; Gulrajani et al., 2017; Miyato et al., 2018). As reported by Arjovsky & Bottou (2017), the reason for the unstable behaviour of GANs is partly due to a dimensional mismatch or non-overlapping support between the real data and the generative model distributions, resulting in an almost trivial task for the discriminator. Once the performance of the discriminator is maxed out, it provides a non-informative signal to train the generator. To avoid vanishing gradients, the original GAN paper (Goodfellow et al., 2014) proposed to modify the min-max based GAN objective (MM GAN) to a non-saturating loss (NS GAN). However, even with such a re-formulation the generator updates tend to get worse over the course of training and optimization becomes massively unstable (Arjovsky & Bottou, 2017).

Prior approaches tried to mitigate this issue by making the discriminator weaker using heuristics, such as decreasing its learning rate, adding label or input noise. Salimans et al. (2016) proposed a one-sided label smoothing technique to smoothen the classification boundary of the discriminator, thereby preventing it from being overly confident, but at the same time biasing its optimality. The works of Arjovsky & Bottou (2017) and Sønderby et al. (2017) made the job of the discriminator harder by adding Gaussian noise to both generated and real samples. Moving the manifolds of the data and model distributions closer to each other by adding the input noise ensures a meaningful overlap between their supports, which is desired in order for the generator to eventually approach the data distribution. However, adding high-dimensional noise introduces significant variance in the parameter estimation, which slows down the training and requires multiple samples for counteraction. Instead of adding noise to the input, Zhang et al. (2018b) created mixup samples by interpolating between synthetic and real ones, which leads to a more stable behaviour of GANs. Both techniques, i.e., additive noise and sample mixup, perform direct modifications on the data samples.

Another line of work resorts to cost function reformulation to improve the stability of GAN training, e.g. by using the Pearson χ^2 divergence for least square GANs (LS GANs) (Mao et al., 2016), kernel maximum mean discrepancy (MMD) for MMD-GANs (Li et al., 2017; Dziugaite et al., 2015), or f-divergence for f-GANs (Nowozin et al., 2016). Arjovsky et al. (2017) proposed the Wasserstein GAN (WGAN) with the training objective derived from the Wasserstein distance, aiming to mitigate the vanishing gradient problem. The drawback of this approach is the weight clipping of the

discriminator employed to enforce smoothness, which adversely reduces the capacity of the discriminator. Alternative to weight clipping, Gulrajani et al. (2017) added a soft penalty on the gradient norm which ensures a 1-Lipschitz discriminator. The gradient norm penalty can be seen as a weight regularization technique for the discriminator and was shown to improve the performance with other losses as well (Fedus et al., 2018). Similarly, Roth et al. (2017) proposed to add a penalty on the weighted gradient-norm of the discriminator in the context of f-divergences, showing its equivalence to adding input noise. On the downside, regularizing the discriminator with the gradient penalty depends on the model distribution, which changes during training and thus results in increased runtime. Miyato et al. (2018) proposed another way to stabilize the discriminator by normalizing its weights and limiting the spectral norm of each layer to constrain the Lipschitz constant. This normalization technique does not require intensive tuning of hyper-parameters and is computationally light. Most recently, Zhang et al. (2018a) showed that spectral normalization is also beneficial for the generator by preventing the escalation of parameter magnitudes and avoiding unusual gradients.

Several methods have proposed to modify the training methodology of GANs in order to further improve stability, e.g. by considering multiple discriminators with different roles (Durugkar et al., 2017) or growing both the generator and discriminator networks progressively (Karras et al., 2018).

In this work we introduce an orthogonal way to stabilize the GAN training by progressively increasing the discrimination task difficulty. To this end, a novel and structured way of augmenting the discriminator input space is proposed. In contrast to other techniques, our method does not bias the optimality of the discriminator or alter the training samples. Furthermore, the proposed augmentation is complementary to prior work. It can be employed with different GAN architectures, adapted to various divergence measures and combined with other regularization techniques (see Sec. 5).

3 Theoretical Background

For generative modeling, one common approach is to adopt divergence measures as loss functions for the generator. Our method belongs to this line of work. In contrast to prior work, our primary focus is not on explicitly minimizing the divergence between the data and model distributions defined on the sample space $\mathcal{X}$. Alternatively, we first structurally augment the training samples (both real and synthetic ones) and then minimize the divergence between distributions defined on the augmented sample space. For computing the divergence, we adopt the adversary process introduced by (Goodfellow et al., 2014) (Sec. 3.1), while the proposed augmentation is inspired by the information theory view of Jensen-Shannon (JS) divergence (Sec. 3.2). Both of them are briefly reviewed in this section to lay the theoretical groundwork for our method, which we then discuss in Sec. 4.

3.1 Adversary Process of GANs

Let $\mathcal{X}$ denote a compact metric space such as the image space $[0,1]^d$ of dimension d. The data distribution $\mathbb{P}_{\mathrm{d}}$ and the model distribution $\mathbb{P}_{\mathrm{g}}$ are both probability measures defined on $\mathcal{X}$. In the context of GANs, $\mathbb{P}_{\mathrm{g}}$ is commonly induced by a function G that maps a random noise vector $\boldsymbol{z}$, following a given prior distribution $\mathbb{P}_{\mathrm{z}}$, to a synthetic data sample, i.e. $\boldsymbol{x}_{\mathrm{g}} = G(\boldsymbol{z}) \in \mathcal{X}$.

The core idea behind GAN training is to set up a competing game between two players, commonly termed the discriminator D and generator G. Mathematically, their objective can be formulated as

$$\min_{G} \max_{D} L(D,G) \triangleq \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\mathrm{d}}} \left\{ \log \left[D(\boldsymbol{x}) \right] \right\} + \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_{\mathrm{g}}} \left\{ \log \left[1 - D(\boldsymbol{x}) \right] \right\}, \tag{1}$$

with $D : \mathcal{X} \mapsto [0,1]$. The optimal D^* respectively classifies $\boldsymbol{x} \sim \mathbb{P}_{\mathrm{d}}$ and $\boldsymbol{x} \sim \mathbb{P}_{\mathrm{g}}$ as TRUE and FAKE, i.e. binary classification. Its achieved maximum equals the JS divergence between $\mathbb{P}_{\mathrm{d}}$ and $\mathbb{P}_{\mathrm{g}}$, which is then used as the loss function by the generator to optimize G (Goodfellow et al., 2014).

3.2 Information Theory Viewpoint

Apart from measuring the similarity of a distribution pair, the JS divergence has an information theory interpretation that inspires our approach presented in Sec. 4. In accordance with the binary classification task of the discriminator, we introduce a binary random variable s with a uniform distribution $\mathbb{P}_{\mathrm{s}}$. Associating $s = 0$ and $s = 1$ respectively with $\boldsymbol{x} \sim \mathbb{P}_{\mathrm{d}}$ and $\boldsymbol{x} \sim \mathbb{P}_{\mathrm{g}}$, we obtain a joint distribution

$$\mathbb{P}(\boldsymbol{x}, s) = \mathbb{P}_{\mathrm{s}}(s)\mathbb{P}(\boldsymbol{x}|s) \quad \text{with} \quad \mathbb{P}(\boldsymbol{x}|s) \triangleq \begin{cases} \mathbb{P}_{\mathrm{d}}(\boldsymbol{x}) & \text{if } s = 0 \\ \mathbb{P}_{\mathrm{g}}(\boldsymbol{x}) & \text{if } s = 1 \end{cases}. \tag{2}$$

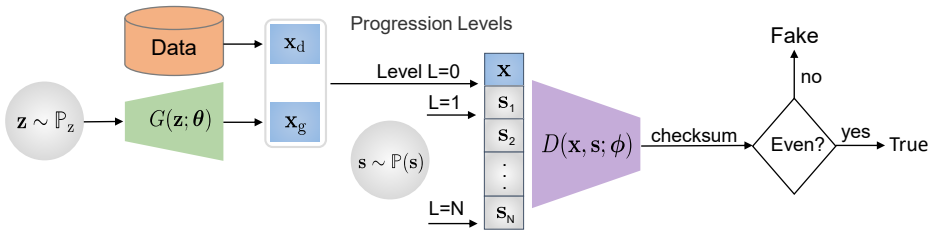

Figure 2: Overview of the proposed PA-GANs training. With each level of progressive augmentation L the dimensionality of $\boldsymbol{s}$ is enlarged from 1 to N, $\boldsymbol{s} = \{s_1, s_2, \ldots, s_N\}$. The task difficulty of the discriminator, checksum computation of $(\boldsymbol{x}, \boldsymbol{s})$, increases as the length of $\boldsymbol{s}$ grows.

The marginal distribution with respect to $\boldsymbol{x}$ (a.k.a. the mixture distribution) is equal to

$$\mathbb{P}_{\mathrm{m}} \triangleq \frac{\mathbb{P}_{\mathrm{d}} + \mathbb{P}_{\mathrm{g}}}{2}. \tag{3}$$

Computing the mutual information of the two random variables s and $\boldsymbol{x}$ based on $\mathbb{P}(\boldsymbol{x}, s)$, we have

$$\begin{aligned} I(\boldsymbol{x}; s) = H(\boldsymbol{x}) - H(\boldsymbol{x}|s) &\overset{(a)}{=} 0.5 \int p_{\mathrm{d}}(\boldsymbol{x}) \log p_{\mathrm{d}}(\boldsymbol{x}) \mathrm{d}\mathbb{P}_{\mathrm{m}}(\boldsymbol{x}) + 0.5 \int p_{\mathrm{g}}(\boldsymbol{x}) \log p_{\mathrm{g}}(\boldsymbol{x}) \mathrm{d}\mathbb{P}_{\mathrm{m}}(\boldsymbol{x}) \\ &= D_{\mathrm{JS}}\left(\mathbb{P}_{\mathrm{d}} \| \mathbb{P}_{\mathrm{g}}\right), \end{aligned} \tag{4}$$

where the equality (a) is the outcome of computing the two entropy terms based on the reference measure $\mathbb{P}_{\mathrm{m}}$. The minimum of the JS divergence $D_{\mathrm{JS}}\left(\mathbb{P}_{\mathrm{d}} \| \mathbb{P}_{\mathrm{g}}\right)$ equal to zero is attainable iff $\mathbb{P}_{\mathrm{d}} = \mathbb{P}_{\mathrm{g}}$. This condition makes the joint distribution function $\mathbb{P}(\boldsymbol{x}, s)$ factorizable, indicating the independence between $\boldsymbol{x}$ and s, and thereby yielding zero mutual information.

4 Progressively Augmented GAN Training

Relying on the information theory view of the JS divergence given in the previous section, we can cast the optimization objective of the generator as mutual information minimization

$$\min_G D_{\mathrm{JS}}\left(\mathbb{P}_{\mathrm{d}} \| \mathbb{P}_{\mathrm{g}}\right) \equiv \min_G I(\boldsymbol{x}; s). \tag{5}$$

Based on (5), in Sec. 4.1 we will first derive a series of equivalent problems, particularly showing how the auxiliary random bit s leads us to a novel and structured way to augment the sample $\boldsymbol{x} \in \mathcal{X}$ for training the discriminator. By further identifying a common principle behind the series of equivalent problems, in Sec. 4.2 we extend the single level augmentation based on one bit s to progressive multi-level augmentation with an arbitrarily long random bit sequence. Progressively increasing the number of augmentation levels equips us with a new mechanism to balance the two-player competing game. In Sec. 4.3 we describe the integration of the proposed augmentation into neural networks and present how to schedule the augmentation progression during training.

4.1 Single Level Augmentation

Starting from the case of single level augmentation with one bit s, we first prove the equivalence of the following problems:

$$\min_G D_{\mathrm{JS}}\left(\mathbb{P}_{\mathrm{d}} \| \mathbb{P}_{\mathrm{g}}\right) \overset{(a)}{\equiv} \min_G I(\boldsymbol{x}; s) \overset{(b)}{\equiv} \min_G D_{\mathrm{KL}}\left(\mathbb{P}(\boldsymbol{x}, s) \| \mathbb{P}(\boldsymbol{x})\mathbb{P}_{\mathrm{s}}(s)\right) \tag{6}$$

$$\overset{(c)}{\equiv} \min_G D_{\mathrm{JS}}\left(\mathbb{P}(\boldsymbol{x}, s) \| \mathbb{P}(\boldsymbol{x})\mathbb{P}_{\mathrm{s}}(s)\right) \overset{(d)}{\equiv} \min_G D_{\mathrm{JS}}\left(\mathbb{P}(\boldsymbol{x}, s) \| \mathbb{Q}(\boldsymbol{x}, s)\right). \tag{7}$$

The first $\equiv$, i.e. (a) in (6), is a convenient copy of (5) followed by (b) due to the relation of mutual information with the Kullback-Leibler (KL) divergence. Namely, the mutual information $I(\boldsymbol{x}; s)$ can also be expressed as the KL divergence of the product of the marginal distributions, $\mathbb{P}(\boldsymbol{x})\mathbb{P}_{\mathrm{s}}(s)$, of the two random variables $\boldsymbol{x}$ and s from their joint distribution $\mathbb{P}(\boldsymbol{x}, s)$. Since the KL divergence as a loss function aims to reduce the difference between two distributions, it can be replaced by the JS divergence as they are minimized under the same condition. This yields the equivalence (c) in (7). Proceeding to (d) in (7), we note that the product of the two marginal distributions equals

$$\mathbb{P}(\boldsymbol{x})\mathbb{P}_{\mathrm{s}}(s) = 0.5\mathbb{P}(\boldsymbol{x}, s) + 0.5\mathbb{Q}(\boldsymbol{x}, s), \tag{8}$$

where $\mathbb{P}(\boldsymbol{x}, s)$ has been defined in (2) and $\mathbb{Q}(\boldsymbol{x}, s)$ is given as follows

$$\mathbb{Q}(\boldsymbol{x}, s) = \mathbb{P}_{\mathrm{s}}(s)\mathbb{Q}(\boldsymbol{x}|s) \quad \text{with} \quad \mathbb{Q}(\boldsymbol{x}|s) \triangleq \begin{cases} \mathbb{P}_{\mathrm{d}}(\boldsymbol{x}) & \text{if } s = 1 \\ \mathbb{P}_{\mathrm{g}}(\boldsymbol{x}) & \text{if } s = 0 \end{cases}. \tag{9}$$

Comparing (9) with (2), $\mathbb{Q}(\boldsymbol{x}, s)$ is simply an outcome of flipping the association rule of $s \in \{0, 1\}$ with $\boldsymbol{x}_{\mathrm{d}}$ and $\boldsymbol{x}_{\mathrm{g}}$. With the relation given in (8), minimizing the JS divergence between $\mathbb{P}(\boldsymbol{x})\mathbb{P}_{\mathrm{s}}(s)$ and $\mathbb{P}(\boldsymbol{x}, s)$ requires $\mathbb{P}(\boldsymbol{x}, s)$ and $\mathbb{Q}(\boldsymbol{x}, s)$ being mutually identical, i.e. $D_{\mathrm{JS}}\left(\mathbb{P}(\boldsymbol{x}, s) \| \mathbb{Q}(\boldsymbol{x}, s)\right) = 0$. With that, we close the proof of the equivalences given in (6)-(7).

Optimizing G based on (7) instead of (5), the objective of the discriminator D subsequently converges to estimation of the JS divergence between $\mathbb{P}(\boldsymbol{x}, s)$ and $\mathbb{Q}(\boldsymbol{x}, s)$

$$\max_{D} \mathbb{E}_{(\boldsymbol{x}, s) \sim \mathbb{P}} \left\{\log\left[D(\boldsymbol{x}, s)\right]\right\} + \mathbb{E}_{(\boldsymbol{x}, s) \sim \mathbb{Q}} \left\{\log\left[1 - D(\boldsymbol{x}, s)\right]\right\}. \tag{10}$$

Comparing with the original discrimination task in GANs, i.e. (1), two differences are worth noting. First, the above discriminator takes s in addition to the sample $\boldsymbol{x} \in \mathcal{X}$ as the input. We, therefore, view s as a single level of augmentation to the sample $\boldsymbol{x}$. Second, the distributions that form the two classes (i.e. TRUE vs. FAKE) become $\mathbb{P}(\boldsymbol{x}, s)$ and $\mathbb{Q}(\boldsymbol{x}, s)$, instead of the original data and model distributions. Based on the definitions in (2) and (9), we identify $(\boldsymbol{x}_{\mathrm{d}}, s = 0)$ and $(\boldsymbol{x}_{\mathrm{g}}, s = 1)$ as belonging to the TRUE class, whereas $(\boldsymbol{x}_{\mathrm{d}}, s = 1)$ and $(\boldsymbol{x}_{\mathrm{g}}, s = 0)$ to the FAKE class. Thus, the real samples are no longer always in the TRUE class, and the synthetic samples are no longer always in the FAKE class. TRUE and FAKE now depend on the combination of $\boldsymbol{x}$ with s (see Figure 1).

Here, we introduce a simple trick to easily detect the class of a given pair. Namely, let the data and synthetic samples respectively convey one bit of information, with $\boldsymbol{x}_{\mathrm{d}}$ encoding bit zero and $\boldsymbol{x}_{\mathrm{g}}$ encoding bit one from now on. Then, the checksum of the pair $(\boldsymbol{x}, s)$ determines the respective class, i.e. checksum zero for TRUE and one for FAKE. [1] The checksum computation poses a more challenging task for the discriminator, as it needs to tell the real and synthetic samples apart and additionally understand the checksum rule. Therefore, such augmentation is usable for preventing early maxing-out of the discriminator. More importantly, it does not compromise the core role of the discriminator in GAN training: informing the generator about the difference between the data and generative model distribution. This statement is confirmed by the equivalence at (7).

4.2 Progressive Multi-Level Augmentation

Next, we extend the single level augmentation with one bit s to multi-level augmentation with an arbitrarily long random bit sequence $\boldsymbol{s}$. Note that the first minimization problem in (6) and the last one in (7) share the same structure except for the different distribution pair. Let us replace the data and model distributions, i.e. $\mathbb{P}_{\mathrm{d}}$ and $\mathbb{P}_{\mathrm{g}}$, respectively with $\mathbb{P}(\boldsymbol{x}, s)$ and $\mathbb{Q}(\boldsymbol{x}, s)$. This implies that the last minimization problem takes the starting position in our previous derivation. Following the same line of argumentation, we can systematically add a new bit. Repeating this procedure L times will give us a bit sequence $\boldsymbol{s}$ with length L plus a series of equivalent problems with the same structure

$$\begin{aligned} \min_{G} D_{\mathrm{JS}}\left(\mathbb{P}_{\mathrm{d}} \| \mathbb{P}_{\mathrm{g}}\right) \equiv \min_{G} D_{\mathrm{JS}}\left(\mathbb{P}(\boldsymbol{x}, s_1) \| \mathbb{Q}(\boldsymbol{x}, s_1)\right) \equiv \min_{G} D_{\mathrm{JS}}\left(\mathbb{P}(\boldsymbol{x}, s_1, s_2) \| \mathbb{Q}(\boldsymbol{x}, s_1, s_2)\right) \\ \cdots \equiv \min_{G} D_{\mathrm{JS}}\left(\mathbb{P}(\boldsymbol{x}, \boldsymbol{s}) \| \mathbb{Q}(\boldsymbol{x}, \boldsymbol{s})\right). \end{aligned} \tag{11}$$

To estimate the JS divergence in (11), the augmented discriminator as defined in (10) now takes a bit sequence $\boldsymbol{s}$ instead of the single bit s, in addition to $\boldsymbol{x}$. The combination of the sample $\boldsymbol{x}$ and $\boldsymbol{s}$ yields a multi-level augmentation. Following the analysis of the single bit case, it is not difficult to notice that the checksum mechanism remains. Namely, the discriminator needs to retrieve the one bit information carried by $\boldsymbol{x}$ and then perform a checksum together with the bit sequence $\boldsymbol{s}$. The task difficulty increases as the length of $\boldsymbol{s}$ grows (see Figure 1). Therefore it makes sense to increase the augmentation level by adding more bits, whenever the discriminator becomes too powerful. More importantly, the consistency of the checksum mechanism across different augmentation levels permits progressive augmentation. The same discriminator can be trained from a lower augmentation level and gradually take more bits into consideration (see Figure 2).

4.3 Implementation

Network architecture. In this work, we aim to maximally reuse existing neural network architectures tailored for GANs, such as DCGAN with spectral normalization (SN DCGAN) (Miyato et al., 2018) and InfoGAN (Chen et al., 2016). According to the above-introduced approach (and Figure 2), the generator architecture can remain unchanged, while the discriminator network requires an alteration to incorporate the augmentation $\boldsymbol{s}$. To this end, we only modify the input layer of the discriminator network, yielding minimal changes.

[1] By checksum, we mean to conduct the XOR operation over a bit sequence.

First, the bit sequence $\boldsymbol{s}$ is preprocessed into a form compatible with $\boldsymbol{x}$. Consider an image sample $\boldsymbol{x}$ with three coordinates, i.e. height, width and color channel (RGB). Each entry of $\boldsymbol{s}$ creates one extra augmentation channel, whereas the bit value is replicated to match the height and width of $\boldsymbol{x}$. It is worth noting that we let each bit take on values $\{0, 1\}$ as input to the network. This choice of values is mainly due to the progressive augmentation during the course of training. When increasing the augmentation level, the additional bit 0 does not change the checksum and thus the output of D. On the contrary, the additional bit 1 flips the even(odd) checksum to odd(even). An effective change at the discriminator output is necessary to match the discrimination goal, thereby requiring a non-zero input in the first place. Using $\{0, 1\}$ rather than other pairs of values, e.g. $\{-1, 1\}$, helps the discriminator to timely catch the change when progression takes place. Second, we keep the input layer in the network to process $\boldsymbol{x}$ and copy its configuration for processing the reformed $\boldsymbol{s}$. Its kernel size, stride and padding type remain, but the input channel size is changed to L to process each entry of $\boldsymbol{s}$. When a new augmentation level is reached, one extra input channel is instantiated to process the bit $L + 1$. All the following layers of the discriminator remain unchanged.

Minibatch discrimination. The gradients for updating D are computed from the loss function given in (10), where the augmentation bit s is replaced by the bit sequence $\boldsymbol{s}$ of length L depending on the current augmentation level. The two expectations are approximated by using minibatches. Each minibatch is constructed with the same number of real data samples, synthetic samples and bit sequences. Each bit sequence is randomly sampled and associated with one real and one synthetic sample. By computing the checksums of the formed pairs, we can decide the correct class of each pair $(\boldsymbol{x}, \boldsymbol{s})$ in the minibatch and feed it into the discriminator to compute the cross-entropy loss. This way of generating $(\boldsymbol{x}, \boldsymbol{s})$ guarantees a balanced number of TRUE/FAKE samples.

Non-saturating loss (NS). When employing non-saturating loss for G in the experiments, we follow the reformulation introduced by (Goodfellow et al., 2014). Since the two expectation terms in (10) depend on G, this reformulation is applied for both of them, namely

$$\min_G -\mathbb{E}_{(\boldsymbol{x},\, \boldsymbol{s})\, \sim\, \mathbb{P}} \left\{\log\left[1 - D(\boldsymbol{x}, \boldsymbol{s})\right]\right\} - \mathbb{E}_{(\boldsymbol{x},\, \boldsymbol{s})\, \sim\, \mathbb{Q}} \left\{\log\left[D(\boldsymbol{x}, \boldsymbol{s})\right]\right\}. \tag{12}$$

Progression scheduling. Bińkowski et al. (2018) introduced the kernel inception distance (KID) to quantify the quality of the synthetic samples and proposed to reduce the learning rate by tracking the reduction of KID over iterations. Here we use KID to decide if the performance of G at the current augmentation level saturates or even starts degrading (typically happens when D starts overfitting or becomes too powerful). Specifically, after t discriminator iterations [2] , we compute the KID between synthetic samples and data samples drawn from the training set. If the current KID score is less than 5% of the average of the two previous ones attained at the same augmentation level, the augmentation is leveled up, i.e. L is increased by one.

Once reaching a new augmentation level, we introduce the following mechanisms to assist the discriminator in quickly picking up the change in the input space. First, the new augmentation bit is generated from a non-uniform distribution, i.e. $\mathbb{P}(s = 1) = p$ and $\mathbb{P}(s = 0) = 1 - p$ with $p < 0.5$. As mentioned before, critical changes on the discriminator side are required for bit 1. For it to gradually comprehend the new bit, we on purpose create more 0s than 1s and gradually increase p up to 0.5 (the uniform case) after a certain number of iterations. A simple linear model is adopted

$$p = \min\{0.5 * (t - t_{\mathrm{st}})/t_{\mathrm{r}}, 0.5\} \tag{13}$$

where t and t_{st} are the current iteration index and the iteration index when the current augmentation level is started, respectively; and t_{r} controls the rate of increase. It is important to note that $p \neq 0.5$ does not cause unbalanced TRUE and FAKE classes in the constructed minibatches. It only introduces some bias in the generation of the new augmentation bit.

Finally, it is advisable to slow down the learning rate of G when a new augmentation level is reached. When D is not properly adjusted to the new level, its feedback to G can be noisy. For instance, when using the Adam optimizer (Kingma & Ba, 2015), we reset the time step recorded by the G optimizer.

5 EXPERIMENTS

In this section, we empirically evaluate the effectiveness of PA-GAN and compare its performance with other GAN-type algorithms across multiple datasets and different network architectures.

[2] Each update of D parameters counts as one discriminator iteration. We assume that the update frequency for D can be as or more frequent than that of G, but not the opposite.

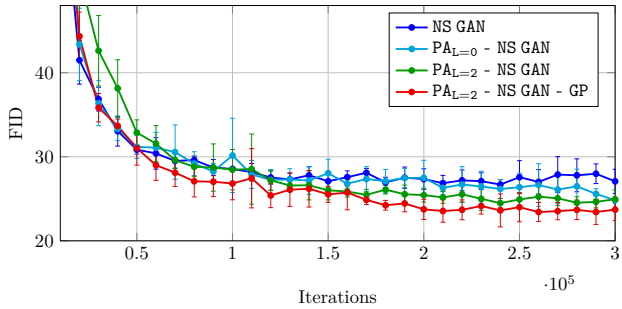

Figure 3: Mean FID values attained over iterations with different NS GAN variants across five independent runs on CIFAR10.

	200k	300k
NS GAN (Kurach et al., 2018)	26.7	–
NS GAN (ours)	26.3	25.7
NS GAN - GP (Kurach et al., 2018)	26.2	–
PA$_{L=0}$ - NS GAN	25.7	24.6
PA$_{L=2}$ - NS GAN	24.5	23.8
PA$_{L=2}$ - NS GAN - GP	**23.2**	**22.5**

Table 1: Median FID values attained with different NS GAN variants on CIFAR10. Applying PA and GP on top of NS GAN reduces FID by $\sim 12.5\%$.

Datasets. In our experiments we consider MNIST (LeCun et al., 1998), Fashion-MNIST (Xiao et al., 2017), CIFAR10 (Krizhevsky, 2009) and CELEBA (Liu et al., 2015) datasets, with the training set sizes equal to 60k, 60k, 50k and 162k respectively.

Network architectures. We employ two well established deep convolutional GAN architectures, SN-DCGAN (Miyato et al., 2018) and InfoGAN (Chen et al., 2016) (see Appendix A.1 for detailed configurations). As they respectively employ spectral normalization (SN) and batch normalization (BN), our aim is to explore the compatibility of PA-GAN with these normalization techniques.

Evaluation metrics. We use Fréchet inception distance (FID) (Huszár, 2015) as the primary performance evaluation metric.[3] Additionally, we also report inception score (IS) (Theis et al., 2016) and kernel inception distance (KID) (Bińkowski et al., 2018) in Appendix A.2. For quantifying the quality of synthetic samples, all measures are computed based on 10k test data and 10k synthetic samples, following the evaluation framework of Lučić et al. (2018) and Kurach et al. (2018).[4]

Training details. We use the minibatch size of 64 and the Adam optimizer (Kingma & Ba, 2015) with the default setting: $\beta_1 = 0.5$, $\beta_2 = 0.999$ and learning rate 2×10^{-4} for both the generator and discriminator, which have an equal update rate. The dimension of $\boldsymbol{z}$ is set to 64 and 128 respectively for InfoGAN and SN-DCGAN. The prior distribution $\mathbb{P}_{\mathrm{z}}$ is uniform. For scheduling progressive augmentation KID is evaluated every 10^4 discriminator iterations, using 10k generated samples and 10k samples randomly drawn from the training set, and t_{r} in (13) is set to 5×10^3.

5.1 CIFAR10 WITH SN-DCGAN

In this experiment, we evaluate the progressive augmentation (PA) with NS GAN (GAN with the non-saturating loss) using the SN-DCGAN architecture on CIFAR10. We analyze the benefits of applying PA on top of NS GAN, experiment with starting PA from different augmentation levels and investigate the complementarity of using both PA and the gradient penalty regularization (GP) (Gulrajani et al., 2017). For fair comparison, we follow the experimental setup of (Kurach et al., 2018).

NS GAN with PA. Figure 3 and Table 1 compare NS GAN results with and without applying PA. We are able to closely reproduce the NS GAN results reported in (Kurach et al., 2018, Table 6), after 200k iterations we obtain the median FID value of 26.3 vs. original 26.7. By applying PA on top of NS GAN and starting from the augmentation level 0 (PA$_{L=0}$ - NS GAN), we achieve superior performance, with the median FID of 25.7 vs. 26.3 of NS GAN. Training for extra 100k iterations boosts the performance of PA$_{L=0}$ - NS GAN (25.7 vs. 24.6). Starting PA from the level 2 (PA$_{L=2}$ - NS GAN) further improves the median FID (23.8 vs. 25.7); as CIFAR10 contains diverse images a start from a higher augmentation level is recommended. It is worth noting that at early iterations (up to 50k) PA$_{L=2}$ - NS GAN has worse performance than NS GAN. Starting at the augmentation level 2 imposes a more challenging task for the discriminator, thereby showing slower improvement at initial iterations but being beneficial in the longer term. The FID value of PA$_{L=2}$ - NS GAN saturates at a slower pace than NS GAN leading to better overall results.

NS GAN with PA and GP. For GP the interpolates are created analogously to (Gulrajani et al., 2017), i.e. $[\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{s}}] = \alpha[\boldsymbol{x}, \boldsymbol{s}]_{\mathrm{TRUE}} + (1-\alpha)[\boldsymbol{x}, \boldsymbol{s}]_{\mathrm{FAKE}}$ with $\alpha \sim \mathrm{U}(0,1)$. Note that interpolation takes place in the augmented space $[\boldsymbol{x}, \boldsymbol{s}]$, yielding $\tilde{\boldsymbol{s}} \in [0,1]^L$.

[3] How to precisely evaluate the performance of GANs is an open question in itself. From the comparison conducted in (Borji, 2018), FID is considered to be the most informative of the measures.

[4] https://github.com/google/compare_gan

	MNIST	Fashion-MNIST	CIFAR10	CELEBA
MM GAN (Goodfellow et al., 2014)	9.8 ± 0.9	29.6 ± 1.6	72.7 ± 3.6	65.6 ± 4.2
LS GAN (Mao et al., 2016)	7.8 ± 0.6	30.7 ± 2.2	87.1 ± 0.9	53.9 ± 2.8
WGAN (Arjovsky et al., 2017)	6.7 ± 0.4	21.5 ± 1.6	55.2 ± 2.3	41.3 ± 2.0
DRAGAN (Kodali et al., 2017)	7.6 ± 0.4	27.7 ± 1.2	69.8 ± 2.0	42.3 ± 3.0
NS GAN (Goodfellow et al., 2014)	6.8 ± 0.5	26.5 ± 1.6	58.5 ± 1.9	55.0 ± 3.3
PA - NS GAN	8.8 ± 1.1	$\mathit{18.4} \pm 1.5$	$\mathit{44.6} \pm 1.9$	$\mathit{46.9} \pm 3.3$
PA - NS GAN (*)	$\mathbf{6.6} \pm 0.8$	$\mathbf{15.8} \pm 1.1$	$\mathbf{43.1} \pm 1.6$	$\mathit{46.8} \pm 3.2$
WGAN - GP (Gulrajani et al., 2017)	20.3 ± 5.0	24.5 ± 2.1	55.8 ± 0.9	30.0 ± 1.0
PA - WGAN - GP	$\mathit{13.9} \pm 1.5$	26.4 ± 2.8	-	$\mathit{29.2} \pm 1.7$
PA - WGAN - GP (*)	$\mathit{8.6} \pm 1.1$	$\mathit{20.7} \pm 2.1$	-	$\mathbf{29.1} \pm 1.7$

Table 2: FID values achieved by the listed algorithms with InfoGAN. The numbers except for PA are taken from (Lučić et al., 2018). For the four datasets (from left to right), the results are attained after 20, 20, 100 and 40 epochs, respectively, except for the PA results marked with (*). For (*) the training time is not constrained by the previously specified number of epochs. See A.3 for details.

In (Kurach et al., 2018), when GP is applied on top of NS GAN a marginal improvement is observed (26.2 vs. 26.7). However, employing PA results in a more noticeable gain for GP (22.5 vs. 23.8). Furthermore, GP helps to accelerate the learning speed of $\mathtt{PA}_{\mathtt{L=2}}$ - NS GAN at the initial iterations. As indicated in Figure 1, in the augmented space (e.g., $L \in \{1, 2\}$), more paths are created for the generative model to approach the data distribution. GP can help to smoothen the decision boundaries along these paths. As a result, GP and PA jointly improve the performance of NS GAN, being mutually beneficial (22.5 vs. 26.3).

5.2 Comparisons among Datasets and GAN-type Algorithms

Lučić et al. (2018) compared various GAN-type algorithms under the InfoGAN architecture and reported their FID scores after a wide range of hyper-parameter searching. Following their experimental setup, we select hyper-parameters within the candidate set considered by Lučić et al. (2018) (see Appendix A.3 for details) and evaluate PA with NS GAN as well as Wasserstein GAN with GP (WGAN - GP). The maximal number of augmentation levels achieved by PA corresponds to seven. The results are reported in Table 2. Overall, we observe significant gains of employing PA with NS GAN and WGAN - GP across different datasets.

As shown in Table 2, the NS GAN performance is quite stable across the datasets. By applying PA with NS GAN, we achieve a pronounced improvement, particularly when the dataset (and hence the image generation task) becomes more complicated. Note that the PA - NS GAN performance on MNIST is worse than NS GAN. This is mainly because NS GAN already performs very well on such a simple dataset and PA requires additional iterations to reach similar results or further improve them as in PA - NS GAN (*). In contrast to NS GAN, WGAN - GP is sensitive to the dataset as shown in Table 2 and reported by Mescheder et al. (2018). However, applying PA helps to stabilize the WGAN - GP performance across different benchmarks. Similarly to NS GAN, longer training leads to better results for PA with WGAN - GP. The results in Table 2 indicate that PA generalizes well across different distance measures and is not limited to the JS divergence.

6 Conclusion

In this work we have proposed a novel method - progressive augmentation (PA) - to improve the stability of GAN training, and showed a way to integrate it into existing GAN architectures with minimal changes. Different to standard data augmentation our approach does not modify the training samples, instead it progressively increases the dimension of the discriminator input space by augmenting it with auxiliary random variables. Higher sample space dimensionality helps to entangle the discriminator and thus to avoid its early performance saturation. Moreover, in the augmented space the generator can explore more paths to approach the data distribution, improving variation of the generated samples. We experimentally have shown pronounced performance improvements of employing the proposed PA with state-of-the-art GAN methods across multiple benchmarks. We demonstrated that PA generalizes well across different network architectures and loss functions and is complementary to other regularization techniques. For future work, we find a joint optimization of PA with neural architectures an interesting direction, for instance, combining it with progressive growing of GANs (Karras et al., 2018). Apart from generative modeling, our approach can also be exploited for semi-supervised learning, generative latent modeling and transfer learning.

REFERENCES

Michael Arbel, Dougal J. Sutherland, Mikołaj Bińkowski, and Arthur Gretton. On gradient regularizers for MMD GANs. *arXiv:1805.11565*, 2018.

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Mikołaj Bińkowski, Dougal J. Sutherland, Michael N. Arbel, and Athur Gretton. Demystifying MMD GANs. In *International Conference on Learning Representations (ICLR)*, 2018.

Ali Borji. Pros and cons of GAN evaluation measures. *arXiv:1802.03446*, 2018.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Ishan P. Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Gintare Karolina Dziugaite, Daniel M. Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. In *UAI*, 2015.

William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M. Dai, Shakir Mohamed, and Ian J. Goodfellow. Many paths to equilibrium: GANs do not need to decrease a divergence at every step. In *International Conference on Learning Representations (ICLR)*, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv: 1511.05101*, 2015.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations (ICLR)*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Naveen Kodali, James Hays, Jacob Abernethy, and Zsolt Kira. On convergence and stability of GANs. *arXiv:1705.07215*, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Karol Kurach, Mario Lučić, Xiaohua Zhai, Marcin Michalski, and Sylvain Gelly. The GAN landscape: Losses, architectures, regularization, and normalization. *arXiv: 1807.04720*, 2018.

Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The MNIST database of handwritten digits. Technical report, 1998.

Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. MMD GAN: Towards deeper understanding of moment matching network. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision (ICCV)*, 2015.

Mario Lučić, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. Are GANs created equal? A large-scale study. In *Advances in Neural Information Processing Systems (NIPS)*, 2018.

Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class generative adversarial networks with the L2 loss function. *arXiv:1611.04076*, 2016.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In *International Conference on Machine learning (ICML)*, 2018.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations (ICLR)*, 2016.

Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training GANs. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Casper Kaae Sønderby, Jose Caballero, Lucas Theis, Wenzhe Shi, and Ferenc Huszár. Amortised map inference for image super-resolution. In *International Conference on Learning Representations (ICLR)*, 2017.

Lucas Theis, Aaron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations (ICLR)*, 2016.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv: 1708.07747*, 2017.

Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv: 1805.08318*, 2018a.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018b.

A APPENDIX

A.1 NEURAL NETWORK ARCHITECTURES

Following Kurach et al. (2018) for SN-DCGAN we employed the same architecture as in (Miyato et al., 2018), which we present in Table A1. For the InfoGAN architecture we followed Lučić et al. (2018) and used the network structure of (Chen et al., 2016), which is described in Table A2. For both experiments with SN-DCGAN and InfoGAN we exploited the implementation provided in `https://github.com/google/compare_gan`.

Table A1: SN-DCGAN architecture.

(a) Discriminator

Layer	Kernel	Output shape
Conv, lReLU	$[3, 3, 1]$	$h \times w \times 64$
Conv, lReLU	$[4, 4, 2]$	$\frac{h}{2} \times \frac{w}{2} \times 128$
Conv, lReLU	$[3, 3, 1]$	$\frac{h}{2} \times \frac{w}{2} \times 128$
Conv, lReLU	$[4, 4, 2]$	$\frac{h}{4} \times \frac{w}{4} \times 256$
Conv, lReLU	$[3, 3, 1]$	$\frac{h}{4} \times \frac{w}{4} \times 256$
Conv, lReLU	$[4, 4, 2]$	$\frac{h}{8} \times \frac{w}{8} \times 512$
Conv, lReLU	$[3, 3, 1]$	$\frac{h}{8} \times \frac{w}{8} \times 512$
Linear	$-$	1

(b) Generator

Layer	Kernel	Output shape
$\mathbf{z}$	$-$	128
Linear, BN, ReLU	$-$	$\frac{h}{8} \times \frac{w}{8} \times 512$
DeConv, BN, ReLU	$[4, 4, 2]$	$\frac{h}{4} \times \frac{w}{4} \times 256$
DeConv, BN, ReLU	$[4, 4, 2]$	$\frac{h}{2} \times \frac{w}{2} \times 128$
DeConv, BN, ReLU	$[4, 4, 2]$	$h \times w \times 64$
DeConv, Tanh	$[3, 3, 1]$	$h \times w \times 3$

Table A2: InfoGAN architecture.

(a) Discriminator

Configuration per Layer
4×4 conv. 64 lReLU, stride 2
4×4 conv. 128 lReLU, stride 2, BN
Linear, 1024 lReLU, BN
Linear, 1 output

(b) Generator

Configuration per Layer
Linear, 1024 ReLU, BN
Linear, $7 \times 7 \times 128$ ReLU, BN
4×4 DeConv, 64 ReLu, stride 2, BN
4×4 Deconv, 1 or 3 channels

A.2 CIFAR10 WITH SN-DCGAN

Figure A1 and A2 together with Table A3 and Table A4 report the inception scores and KID values that are attained following the same experimental setup as in Sec. 5.1 on CIFAR10. The results on both measures are consistent with our observation drawn from the FID values. In the recent work (Arbel et al., 2018), the reported best KID value for CIFAR10 with SN-DCGAN is 0.015, whereas $\texttt{PA}_{\texttt{L}=2}$ - `NS GAN` - `GP` reduces it to 0.013.

Next we present some further remarks on the implementation side. First, to explicitly investigate the combination of GP and PA, we add the GP as an extra regularizer to the discriminator loss function only when the augmentation takes place, and GP is computed with respect to the augmented sample after interpolation, i.e. $[\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{s}}]$. Figures A4 - A6 together with Tables A5 - A7 report results evaluated with the FID, IS and KID metrics. With the use of GP, the starting level of the augmentation has negligible influence on the performance after a sufficient number of iterations.

Second, the chosen hyper-parameters are not optimized for PA. In fact, they are dedicatedly chosen by (Kurach et al., 2018) for `NS GAN`, i.e. $\beta_1 = 0.5$, $\beta_2 = 0.999$, $\lambda = 1$ (weighting factor for GP) and the learning rate 2×10^{-4}. We adopt them for PA for the purpose of fair comparison. Therefore, further optimization on the hyper-parameters for PA may potentially yield better results than the reported ones.

Third, in our implementation all of progressively added augmentation channels take on the values $\{0, 1\}$ to ease the progression, as described in Sec. 4.3. For those augmentation channels that are

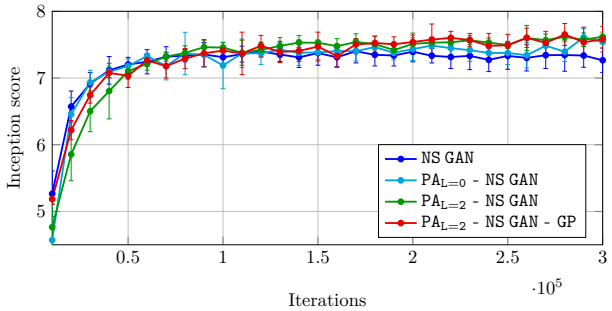

Figure A1: Inception scores (IS) attained over iterations with different NS GAN variants averaged across five independent runs on CIFAR10.

	200k	300k
NS GAN (ours)	7.54	7.58
$\mathrm{PA}_{\mathrm{L}=0}$ - NS GAN	7.63	7.64
$\mathrm{PA}_{\mathrm{L}=2}$ - NS GAN	7.64	7.72
$\mathrm{PA}_{\mathrm{L}=2}$ - NS GAN - GP	7.59	7.76

Table A3: Median inception scores (IS) attained with different NS GAN variants on CIFAR10.

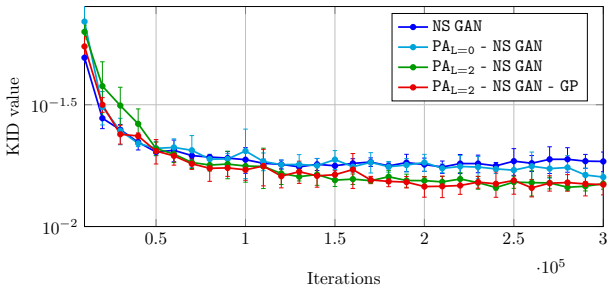

Figure A2: KID values attained over iterations with different NS GAN variants averaged across five independent runs on CIFAR10.

	200k	300k
NS GAN (ours)	0.0168	0.0164
$\mathrm{PA}_{\mathrm{L}=0}$ - NS GAN	0.0165	0.0153
$\mathrm{PA}_{\mathrm{L}=2}$ - NS GAN	0.0149	0.0139
$\mathrm{PA}_{\mathrm{L}=2}$ - NS GAN - GP	0.0140	0.0133

Table A4: Median KID values attained with different NS GAN variants on CIFAR10.

present from the start of training and if the pixel values of the image are normalized to $[-1, 1]$, we accordingly experimented with the values $\{\pm 1\}$. As a result, in the current experiment (i.e., CIFAR10 with SN-DCGAN), we used $\{\pm 1\}$ and switch to $\{0, 1\}$ for subsequent InfoGAN experiment. In general, it is not a critical choice to the performance from our observation.

At last, a set of synthetic images generated by $\mathrm{PA}_{\mathrm{L}=2}$ - NS GAN - GP with the FID value of 22.5 on CIFAR10 is shown in Figure A3.

A.3 Comparisons among Datasets and GAN-type Algorithms

In Sec. 5.2, we apply PA with NS GAN and WGAN - GP and report the achieved FID values in Table 2.

For PA - NS GAN and PA - NS GAN (*) across all datasets we start augmentation with level two, i.e. $L = 2$. The other adopted hyper-parameters for PA - NS GAN are listed in Table A8. It is important to note that they are selected considering the limit on the training epochs specified by Lučić et al. (2018). Allowing more iterations, the performance can be further improved. For instance, PA - NS GAN (*) in Table 2 shows the achieved performance if we run 40k iterations for MNIST and Fashion-MNIST, and run 140k iterations for CIFAR10 and CELEBA. The gains can be further enlarged if the hyper-parameters are adjusted towards the longer training time. For instance, by reducing the learning rate from 10^{-3} to 2×10^{-4} and performing 140k iterations for MNIST, we can achieve the FID of 4.5 ± 0.35.

	MNIST	Fashion-MNIST	CIFAR10	CELEBA
β_1	0.5	0	0.5	0.5
β_2	0.999	0.999	0.999	0.999
Learning rate	10^{-3}	4×10^{-4}	4×10^{-4}	2×10^{-4}

Table A8: Hyper-parameters for generating our numbers associated to PA - NS GAN in Table 2.

Figure A3: Synthetic images generated by $\texttt{PA}_{\texttt{L=2}}$ - NS GAN - GP with the FID 22.5 on CIFAR10.

	MNIST	Fashion-MNIST	CIFAR10	CELEBA
β_1	0	0	-	0
β_2	0.999	0.999	-	0.999
λ	0.1	0.1	-	0.1
Learning rate	10^{-3}	10^{-3}	-	4×10^{-4}

Table A9: Hyper-parameters for generating our numbers associated to PA - WGAN - GP in Table 2.

Proceeding to the case PA - WGAN - GP, the hyper-parameters are listed in Table A9, the batch normalization for the discriminator is disabled. For both MNIST and Fashion-MNIST, the augmentation level starts from one to ensure that it is in place within the 20 epoch training time. For CELEBA, we start from the augmentation level zero and will reach the level one augmentation within the 40 training epochs. Analogous to the previous case, better performance is achievable by PA with the number of iterations increased beyond the original limit, i.e. 40k for MNIST and Fashion-MNIST and 140k for CELEBA.

Finally, we note that to stabilize progression in WGAN - GP the weighting factor λ for GP requires a careful adjustment. In this experiment, we relax λ following the idea behind (13), namely by linearly increasing λ from zero to its original value within t_{r} iterations (in this case $t_{\mathrm{r}} = 5 \times 10^3$). Interestingly, we did not find such adaptation necessary when using GP with NS GAN. One possible reason is that GP is critical to WGAN due to the Lipschitz constraint, but optional to NS GAN. Augmentation changes the input space of the discriminator, and the number of terms involved in the GP also increases with the number of augmentation channels. Further investigation on the weighting factor adjustment to fully exploit the benefit of combining PA with WGAN - GP is a part of our future work. The results reported in this work have confirmed that PA is not limited to the JS divergence.

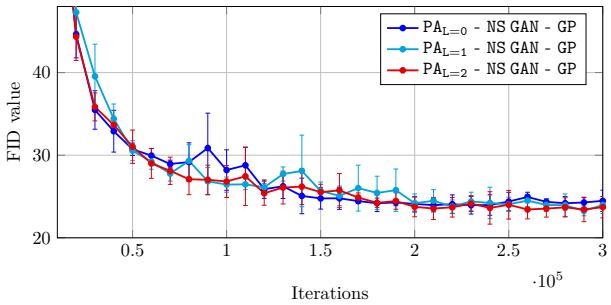

Figure A4: FID attained over iterations with different progression starting levels averaged across five independent runs on CIFAR10.

	200k	300k
NS GAN (ours)	26.3	25.7
PA$_{L=0}$ - NS GAN - GP	23.2	22.8
PA$_{L=1}$ - NS GAN - GP	23.9	23.0
PA$_{L=2}$ - NS GAN - GP	23.2	22.5

Table A5: FID values attained with different progression starting levels on CIFAR10.

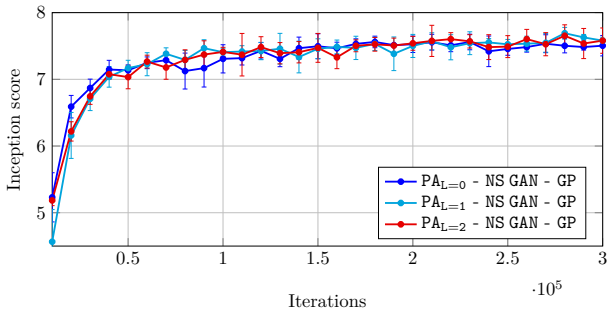

Figure A5: IS attained over iterations with different progression starting levels averaged across five independent runs on CIFAR10.

	200k	300k
NS GAN (ours)	7.54	7.58
PA$_{L=0}$ - NS GAN - GP	7.63	7.67
PA$_{L=1}$ - NS GAN - GP	7.64	7.77
PA$_{L=2}$ - NS GAN - GP	7.59	7.76

Table A6: Median IS attained with different progression starting levels on CIFAR10.

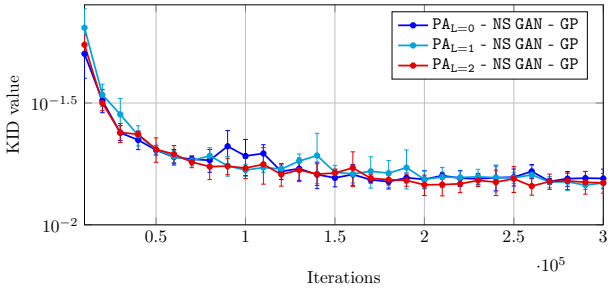

Figure A6: KID values attained over iterations with different progression starting levels averaged across five independent runs on CIFAR10.

	200k	300k
NS GAN (ours)	0.0168	0.0164
PA$_{L=0}$ - NS GAN - GP	0.0141	0.0141
PA$_{L=1}$ - NS GAN - GP	0.0144	0.0141
PA$_{L=2}$ - NS GAN - GP	0.0140	0.0133

Table A7: Median KID values attained with different progression starting levels on CIFAR10.