MSG-GAN: Multi-Scale Gradients for Generative Adversarial Networks

Animesh Karnewar
TomTom
animesh.karnewar@tomtom.com

Oliver Wang
Adobe Research
owang@adobe.com

Figure 1: Results of our proposed MSG-GAN technique where the generator synthesizes images at all resolutions simultaneously and gradients flow directly to all levels from a single discriminator. The first column has a resolution of $4 \times 4$ which increases towards the right reaching the final output resolution of $1024 \times 1024$. Best viewed zoomed in on screen.

Abstract

While Generative Adversarial Networks (GANs) have seen huge successes in image synthesis tasks, they are notoriously difficult to adapt to different datasets, in part due to instability during training and sensitivity to hyperparameters. One commonly accepted reason for this instability is that gradients passing from the discriminator to the generator become uninformative when there isn’t enough overlap in the supports of the real and fake distributions. In this work, we propose the Multi-Scale Gradient Generative Adversarial Network (MSG-GAN), a simple but effective technique for addressing this by allowing the flow of gradients from the discriminator to the generator at multiple scales. This technique provides a stable approach for high resolution image synthesis, and serves as an alternative to the commonly used progressive growing technique. We show that MSG-GAN converges stably on a variety of image datasets of different sizes, resolutions and domains, as well as different types of loss functions and architectures, all with the same set of fixed hyperparameters. When compared to state-of-the-art GANs, our approach matches or exceeds the performance in most of the cases we tried.

1. Introduction

Since their introduction by Goodfellow et al. [10], Generative Adversarial Networks (GANs) have become the de facto standard for high quality image synthesis. The success of GANs comes from the fact that they do not require manually designed loss functions for optimization, and can therefore learn to generate complex data distributions without the need to be able to explicitly define them. While flow-based models such as [6, 7, 27, 18] and autoregressive models such as [32, 31, 29] allow training generative models directly using Maximum Likelihood Estimation (explicitly and implicitly respectively), the fidelity of the generated images has not yet been able to match that of the state-of-the-art GAN models [15, 16, 17, 3]. However, GAN training suffers from two prominent problems: (1) mode collapse and (2) training instability.

The problem of mode collapse occurs when the generator network is only able to capture a subset of the variance present in the data distribution. Although numerous works [28, 41, 15, 21] have been proposed to address this problem, it remains an open area of study. In this work, however, we address the problem of training instability. This is a fundamental issue with GANs, and has been widely reported by
Figure 2: Architecture of MSG-GAN, shown here on the base model proposed in ProGANs [15]. Our architecture includes connections from the intermediate layers of the generator to the intermediate layers of the discriminator. Multi-scale images sent to the discriminator are concatenated with the corresponding activation volumes obtained from the main path of convolutional layers followed by a combine function (shown in yellow).

Previous works [28, 22, 2, 11, 19, 33, 14, 15, 37, 25]. We propose a method to address training instability for the task of image generation by investigating how gradients at multiple scales can be used to generate high resolution images (typically more challenging due to the data dimensionality) without relying on previous greedy approaches, such as the progressive growing technique [15, 16]. MSG-GAN allows the discriminator to look at not only the final output (highest resolution) of the generator, but also at the outputs of the intermediate layers (Fig. 2). As a result, the discriminator becomes a function of multiple scale outputs of the generator and importantly, passes gradients to all the scales simultaneously (more details in section 1.1 and section 2).

Furthermore, our method is robust to different loss functions (we show results on WGAN-GP and Non-saturating GAN loss with 1-sided gradient penalty), datasets (we demonstrate results on a wide range of commonly used datasets and a newly created Indian Celebs dataset), and architectures (we integrate the MSG approach with both ProGANs and StyleGAN base architectures). Much like progressive growing [15], we note that multi-scale gradients account for a considerable improvement in FID score over the vanilla DCGAN architecture. However, our method achieves better performance with comparable training time to state-of-the-art methods on most existing datasets without requiring the extra hyperparameters that progressive growing introduces, such as training schedules and learning rates for different generation stages (resolutions). This robustness allows the MSG-GAN approach to be easily used “out-of-the-box” on new datasets. We also show the importance of the multi-scale connections on multiple generation stages (coarse, medium, and fine), through ablation experiments on the high resolution FFHQ dataset.

In summary, we present the following contributions. First, we introduce a multiscale gradient technique for image synthesis that improves the stability of training as defined in prior work. Second, we show that we can robustly generate high quality samples on a number of commonly used datasets, including CIFAR10, Oxford102 flowers, CelebA-HQ, LSUN Churches, Flickr Faces HQ and our new Indian Celebs all with the same fixed hyperparameters. This makes our method easy to use in practice.

1.1. Motivation

Arjovsky and Bottou [1] pointed out that one of the reasons for the training instability of GANs is due to the passage of random (uninformative) gradients from the discriminator to the generator when there is insubstantial overlap between the supports of the real and fake distributions. Since the inception of GANs, numerous solutions have been proposed to this problem. One early example proposes adding instance noise to the real and the fake images so that the supports minimally overlap [1, 30]. More recently, Peng et al. [25] proposed a mutual information bottleneck

![Figure 2: Architecture of MSG-GAN, shown here on the base model proposed in ProGANs [15]. Our architecture includes connections from the intermediate layers of the generator to the intermediate layers of the discriminator. Multi-scale images sent to the discriminator are concatenated with the corresponding activation volumes obtained from the main path of convolutional layers followed by a combine function (shown in yellow).](image-url)
between input images and the discriminator’s deepest representation of those input images called the variational discriminator bottleneck (VDB) [25], and Karras et al. [15] proposed a progressive growing technique to add continually increasing resolution layers. The VDB solution forces the discriminator to focus only on the most discriminating features of the images for classification, which can be viewed as an adaptive variant of instance noise. Our work is orthogonal to the VDB technique, and we leave an investigation into a combination of MSG-GAN and VDB to future work.

The progressive growing technique tackles the instability problem by training the GAN layer-by-layer by gradually doubling the operating resolution of the generated images. Whenever a new layer is added to the training it is slowly faded in such that the learning of the previous layers are retained. Intuitively, this technique helps with the support overlap problem because it first achieves a good distribution match on lower resolutions, where the data dimensionality is lower, and then partially-initializes (with substantial support overlap between real and fake distributions) higher resolution training with these previously trained weights, focusing on learning finer details.

While this approach is able to generate state-of-the-art results, it can be hard to train, due to the addition of hyper-parameters to be tuned per resolution, including different iteration counts, learning rates (which can be different for the Generator and Discriminator [12]) and the fade-in iterations. In addition, a concurrent submission [17] discovered that it leads to phase artifacts where certain generated features are attached to specific spatial locations. Hence our main motivation lies in addressing these problems by providing a simpler alternative that leads to high quality results and stable training.

Although the current state-of-the-art in class conditional image generation on the Imagenet dataset, i.e. BigGAN [4], doesn’t employ multi-scale image generation, note that the highest resolution they operate on is 512x512. All high resolution state-of-the-art methods [15, 16, 17, 34, 40] use some or the other form of multi-scale image synthesis. Multi-scale image generation is a well established technique, with methods existing well before deep networks became popular for this task [20, 35]. More recently, a number of GAN-based methods break the process of high resolution image synthesis into smaller sub-tasks [36, 39, 38, 8, 9, 15, 34, 40]. For example, LR-GAN [36] uses separate generators for synthesizing the background, foreground and compositor masks for the final image. Works such as GMAN and StackGAN employ a single generator and multiple discriminators for variation in teaching and multi-scale generation respectively [8, 39, 38]. MAD-GAN [9], instead uses multiple generators to address mode-collapse by training a multi-agent setup in such a way that different generators capture different modalities in the training dataset. LapGAN [5] models the difference between the generated multi-scale components of a Laplacian pyramid of the images using a single generator and multiple discriminators for different scales. Pix2PixHD [34] uses three architecturally similar discriminators acting upon three different resolutions of the images obtained by downsampling the real and the generated images.

Our proposed method draws architectural inspiration from all these works and builds upon their teachings and ideologies, but has some key differences. In MSG-GAN, we use a single discriminator and a single generator with multi-scale connections, which allows for the gradients to flow at multiple resolutions simultaneously. There are several advantages (driven largely by the simplicity) of the proposed approach. If multiple discriminators are used at each resolution [39, 38, 5, 40, 34], the total parameters grow exponentially across scales, as repeated downsampling layers are needed, whereas in MSG-GAN the relationship is linear. In addition, multiple discriminators with different effective fields [34, 40] are not able to share information across scales, which could make the task easier. Besides having fewer parameters and design choices required, our approach also avoids the need for an explicit color consistency regularization term across images generated at multiple scales, which was necessary, e.g. in StackGAN [38].

2. Multi-Scale Gradient GAN

We conduct experiments with the MSG-GAN framework applied to two base architectures, ProGANs [15] and StyleGAN [16]. We call these two methods MSG-ProGAN and MSG-StyleGAN respectively. Despite the name, there is no progressive growing used in any of the MSG variants, and we note that ProGANs without progressive growing is essentially the DCGAN [26] architecture. Figure 2 shows an overview of our MSG-ProGAN architecture, which we define in more detail in this section, and include the MSG-StyleGAN model details in the supplemental material.

Let the initial block of the generator function \( g_{gen} \) be defined as \( g_{gen} : Z \mapsto A_{begin} \), such that sets \( Z \) and \( A_{begin} \) are respectively defined as \( Z = \mathbb{R}^{512} \), where \( z \sim N(0, I) \) such that \( z \in Z \) and \( A_{begin} = \mathbb{R}^{4 \times 4 \times 512} \) contains \( 4 \times 4 \times 512 \) dimensional activations. Let \( g_i \) be a generic function which acts as the basic generator block, which in our implementation consists of an upsampling operation followed by two conv layers.

\[
g_i : A_{i-1} \mapsto A_i
\]

where, \( A_i = \mathbb{R}^{2^{i+2} \times 2^{i+2} \times c_i} \) (1)

and, \( i \in \mathbb{N}; A_0 = A_{begin} \) (2)

where \( c_i \) is the number of channels in the \( i^{th} \) intermediate activations of the generator. We provide the sizes of \( c_i \) in all layers in the supplemental material. The full generator
where, \( r' \) is yet another (1x1) convolution operation similar to \( r \) and \([;]\) is a simple channelwise concatenation operation. We compare these different combine functions in Sec 4.

The final discriminator function is then defined as:

\[
\begin{align*}
\text{DIS}(y', o_0, o_1, \ldots, o_{k-1}) &= \\
&= d_{\text{critic}} \circ d^k(\cdot, o_0) \circ d^{k-1}(\cdot, o_1) \circ \ldots d^1(\cdot, o_i) \circ d^0(y')
\end{align*}
\]  

We experimented with two different loss functions for the \( d_{\text{critic}} \) function namely, WGAN-GP [11] which was used by ProGAN [15] and Non-saturating GAN loss with 1-sided GP [10, 23] which was used by StyleGAN [16]. Please note that since the discriminator is now a function of multiple input images generated by the generator, we modified the gradient penalty to be the average of the penalties over each input.

3. Experiments

While evaluating the quality of GAN generated images is not a trivial task, the most commonly used metrics today are the Inception Score (IS, higher is better) [28] and Fréchet Inception Distance (FID, lower is better) [12]. In order to compare our results with the previous works, we use the IS for the CIFAR10 experiments and the FID for the rest of the experiments, and report the “number of real images shown” as done in prior work [15, 16].

New Indian Celebs Dataset In addition to existing datasets, we also collect a new dataset consisting of Indian celebrities. To this end, we collected the images using a process similar to CelebA-HQ. First, we downloaded images for Indian celebrities by scraping the web for related search queries. Then, we detected faces using an off the shelf face-detector and cropped and resized all the images to 256x256. Finally, we manually cleaned the images by filtering out low-quality, erroneous, and low-light images. In the end, the dataset contained only 3K samples, an order of magnitude less than CelebA-HQ.

3.1. Implementation Details

We evaluate our method on a variety of datasets of different resolutions and sizes (number of images); CIFAR10 (60K images at 32x32 resolution); Oxford flowers (8K images at 256x256), LSUN churches (126K images at 256x256), Indian Celebs (3K images at 256x256 resolution), CelebA-HQ (30K images at 1024x1024) and FFHQ (70K images at 1024x1024 resolution).

For each dataset, we use the same initial latent dimensionality of 512, drawn from a standard normal distribution \( N(0, I) \) followed by hypersphere normalization [15]. For all experiments, we use the same hyperparameter settings for MSG-ProGAN and MSG-StyleGAN (\( \text{lr}=0.003 \), with
the only differences being the number of upsampling layers (fewer for lower resolution datasets).

All models were trained with RMSprop (lr=0.003) for generator and discriminator. We initialize parameters according to the standard normal $N(0, I)$ distribution. To match previously published work, StyleGAN and MSG-StyleGAN models were trained with Non-saturating GAN loss with 1-sided GP while ProGANs and MSG-ProGAN models were trained with the WGAN-GP loss function.

We also extend the MinBatchStdDev technique [15, 16], where the average standard deviation of a batch of activations is fed to the discriminator to improve sample diversity, to our multiscale setup. To do this, we add a separate MinBatchStdDev layer at the beginning of each block in the discriminator. This way, the discriminator obtains batch-statistics of the generated samples along with the straight-path activations at each scale, and can detect some degree of mode collapse by the generator.

When we trained the models ourselves, we report training time and GPUs used. We use the same machines for corresponding set of experiments so that direct training time comparisons can be made. Please note that the variation in numbers of real images shown and training time is because, as is common practice, we report the best FID score obtained in a fixed number of iterations, and the time that it took achieve that score. All the code and the trained models required for reproducing our work are made available for research purposes at https://github.com/akanimax/msg-stylegan-tf.

### 3.2. Results

#### Quality

Table 1 shows quantitative results of our method on various mid-level resolutions datasets. Both our MSG-ProGAN and MSG-StyleGAN models achieve better FID
scores than the respective baselines of ProGANs and StyleGAN on the (256x256) resolution datasets of Oxford Flowers, LSUN Churches and Indian Celebs. While each iteration of MSG-GAN is slower than the initial lower resolution iterations of progressive growing, due to all layers being trained together, MSG-GAN tends to converge in fewer iterations, requiring fewer total hours of GPU training time to achieve these scores. Figure 3 shows random samples generated on these datasets for qualitative evaluation.

For high-resolution experiments (Table 2), the MSG-ProGAN model trains in comparable amount of time and gets similar scores on the CelebA-HQ and the FFHQ datasets (8.02 vs 7.79) and (8.36 vs 8.04) respectively. We note a small difference in the author reported scores and what we were able to achieve with the author provided code. This could be due to subtle hardware differences or variance between runs. Our MSG-StyleGAN model was unable to beat the FID score of StyleGAN on the CelebA-HQ dataset (6.37 vs 5.17) and the FFHQ dataset (5.8 vs 4.40). We discuss some hypotheses for why this might be in Sec 4, but note that our method does have other advantages, namely that it seems to be easier to generalize to different datasets as shown in our other experiments. Also, our generated images do not show any traces of the phase artifacts [17] which are prominently visible in progressively grown GANs.

Stability during training To compare the stability of MSG-ProGAN with ProGANs during training, we measure the changes in the generated samples for the same fixed latent points as iterations progress (on CelebA-HQ dataset). This method was introduced by [37] as a way to measure stability during training, which we quantify by calculat-
Figure 5: During training, all the layers in the MSG-GAN synchronize across the generated resolutions fairly early in the training and subsequently improve the quality of the generated images at all scales simultaneously. Throughout the training the generator makes only minimal incremental improvements to the images generated from fixed latent points.

Figure 6: Image stability during training. These plots show the MSE between images generated from the same latent code at the beginning of sequential epochs (averaged over 36 latent samples) on the CelebA-HQ dataset. MSG-ProGAN converges stably over time while ProGANs [15] continues to vary significantly across epochs.

Robustness to learning rate
It has been observed by prior work [28, 14, 24, 23] and also our experience, that convergence of GANs during training is very heavily dependent on the choice of hyperparameters, in particular, learning rate. To validate the robustness of MSG-ProGAN,
Table 3: Robustness to learning rate on CIFAR-10. We see that our approach converges to similar IS scores over a range of learning rates.

<table>
<thead>
<tr>
<th>Method</th>
<th># Real Images</th>
<th>Learning rate</th>
<th>IS (↑)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Images</td>
<td>-</td>
<td>-</td>
<td>11.34</td>
</tr>
<tr>
<td>MSG-ProGAN</td>
<td>12M</td>
<td>0.003</td>
<td>8.63</td>
</tr>
<tr>
<td>MSG-ProGAN</td>
<td>12M</td>
<td>0.001</td>
<td>8.24</td>
</tr>
<tr>
<td>MSG-ProGAN</td>
<td>12M</td>
<td>0.005</td>
<td>8.33</td>
</tr>
<tr>
<td>MSG-ProGAN</td>
<td>12M</td>
<td>0.01</td>
<td>7.92</td>
</tr>
</tbody>
</table>

Table 4: Ablation experiments for varying degrees of multiscale gradient connections on the high resolution (1024x1024) FFHQ dataset. Coarse contains connections at (4x4) and (8x8), middle at (16x16) and (32x32); and fine at (64x64) till (1024x1024).

<table>
<thead>
<tr>
<th>Level of Multi-scale connections FID (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No connections (DC-GAN) 14.20</td>
</tr>
<tr>
<td>Coarse Only                   10.84</td>
</tr>
<tr>
<td>Middle Only                   9.17</td>
</tr>
<tr>
<td>Fine Only                     9.74</td>
</tr>
<tr>
<td>All (MSG-ProGAN)             8.36</td>
</tr>
<tr>
<td>ProGAN∗                       9.49</td>
</tr>
</tbody>
</table>

Table 5: Experiments with different combine functions on the high resolution (1024x1024) FFHQ dataset.

<table>
<thead>
<tr>
<th>Method</th>
<th>Combine function</th>
<th>FID (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG-ProGAN</td>
<td>φ_{lin,cat}</td>
<td>11.88</td>
</tr>
<tr>
<td></td>
<td>φ_{cat,lin}</td>
<td>9.63</td>
</tr>
<tr>
<td></td>
<td>φ_{simple}</td>
<td>8.36</td>
</tr>
<tr>
<td>MSG-StyleGAN</td>
<td>φ_{simple}</td>
<td>6.46</td>
</tr>
<tr>
<td></td>
<td>φ_{lin,cat}</td>
<td>6.12</td>
</tr>
<tr>
<td></td>
<td>φ_{cat,lin}</td>
<td>5.80</td>
</tr>
</tbody>
</table>

4. Discussion

Ablation Studies We performed two types of ablations on the MSG-ProGAN architecture. Table 4 summarizes our experiments on applying ablated versions of the Multi-Scale Gradients, where we only add subsets of the connections from the generator to the discriminator at different scales. We can see that adding multi-scale gradients at any level to the ProGANs/DCGAN architecture improves the FID score. Interestingly, adding only mid-level connections performs slightly better than adding only coarse or fine-level connections, however the overall best performance is achieved with the connections at all levels.

Table 5 presents our experiments with the different variants of the combine function φ on the MSG-ProGAN and the MSG-StyleGAN architectures. φ_{simple} (Eq 11) performed best on the MSG-ProGAN architecture while the φ_{cat,lin} (Eq 13) has the best FID score on the MSG-StyleGAN architecture. All results shown in this work employ these respective combine functions. We can see through these experiments that the combine function also plays an important role in the generative performance of the model, and it is possible that a more advanced combine function such as multi-layer densenet or AdaIN [13] could improve the results even further.

Limitations and Future Work Our method is not without limitations. We note that using progressive training, the first set of iterations at lower resolutions take place much faster, whereas each iteration of MSG-GAN takes the same amount of time. However, we observe that MSG-GAN requires fewer total iterations to reach the same FID, and often does so after a similar length of total training time.

In addition, because of our multi-scale modification in MSG-StyleGAN, our approach cannot take advantage of the mixing regularization trick [16], where multiple latent vectors are mixed and the resulting image is forced to be realistic by the discriminator. This is done to allow the mixing of different styles at different levels at test time, but also improves overall quality. Interestingly, even though we do not explicitly enforce mixing regularization, our method is still able to generate plausible mixing results (see supplementary material).

Conclusion Although huge strides have been made towards photo-realistic high resolution image synthesis [3, 16, 17], true photo-realism has yet to be achieved, especially with regards to domains with substantial variance in appearance. In this work, we presented the MSG-GAN technique which contributes to these efforts with a simple approach to enable high resolution multi-scale image generation with GANs.

5. Acknowledgements

We would like to thank Alexia Jolicoeur-Martineau (Ph.D. student at MILA) for her guidance over Relativism in GANs and for proofreading the paper. Finally we extend special thanks to Michael Hoffman (Sr. Mgr. Software Engineering, TomTom) for his support and motivation.
References


