Loppi, Niki; Kynkäänniemi, Tuomas

Multi-node Training for StyleGAN2

Published in:
Pattern Recognition. ICPR International Workshops and Challenges, 2021, Proceedings

DOI:
10.1007/978-3-030-68763-2_51

Published: 01/01/2021

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Loppi, N., & Kynkäänniemi, T. (2021). Multi-node Training for StyleGAN2. In A. Del Bimbo, R. Cucchiara, S. Sclaroff, G. M. Farinella, T. Mei, M. Bertini, H. J. Escalante, & R. Vezzani (Eds.), *Pattern Recognition. ICPR International Workshops and Challenges, 2021, Proceedings* (pp. 677-684). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 12661 LNCS). Springer. https://doi.org/10.1007/978-3-030-68763-2_51

Multi-node training for StyleGAN2

Niki A. Loppi[1] and Tuomas Kynkäänniemi[2]

[1] NVIDIA AI Technology Center
nloppi@nvidia.com
[2] Aalto University, Helsinki, Finland
tuomas.kynkaanniemi@aalto.fi

Abstract. StyleGAN2 is a Tensorflow-based Generative Adversarial Network (GAN) framework that represents the state-of-the-art in generative image modelling. The current release of StyleGAN2 implements multi-GPU training via Tensorflow's device contexts which limits data parallelism to a single node. In this work, a data-parallel multi-node training capability is implemented in StyleGAN2 via Horovod which enables harnessing the compute capability of larger cluster architectures. We demonstrate that the new Horovod-based communication outperforms the previous context approach on a single node. Furthermore, we demonstrate that the multi-node training does not compromise the accuracy of StyleGAN2 for a constant effective batch size. Finally, we report strong and weak scaling of the new implementation up to 64 NVIDIA Tesla A100 GPUs distributed across eight NVIDIA DGX A100 nodes, demonstrating the utility of the approach at scale.

Keywords: GAN · StyleGAN2 · GPU · Massively parallel architectures · Multi-node training

1 Introduction

The emergence of deep learning has directly led to tremendous improvements in generative modeling. In recent years, generative methods, such as generative adversarial networks (GAN) [1–5], variational-autoencoders (VAE) [6–8], autoregressive models [9, 10], and likelihood-based models [11, 12], have been applied to a wide range of tasks from image generation to drug discovery. GANs, in particular, have been at the forefront in the efforts to generate high-quality images that are indistinguishable from real images. However, the success of GANs is not only limited to generation of high-fidelity images – they have been successfully applied in image-to-image translation [13–16], super-resolution [17–19], and video synthesis [20].

Typical GAN architectures contain a large number of learnable parameters and they need to be trained with large data sets, which makes the process computationally very expensive and time-consuming. Moreover, despite recent advances in GAN research [21–24, 2, 4], almost every aspect of the GAN training, such as model architecture, loss function, optimizer, is sensitive to hyperparameter and

design choices [3]. One way to decrease training time and ease design experimentation with GANs is to scale up the process to harness more GPUs. In this paper, we present a multi-node training extension to the state-of-the-art generative image modelling framework StyleGAN2. This work has relevance to both production and research applications as it can significantly decrease turnaround time of GAN training.

The paper is structured as follows. Section 2 provides an overview of our multi-node StyleGAN2 implementation and details the modifications introduced to the original StyleGAN2 codebase. In Section 3, the new multi-node implementation is validated and its performance is measured against the original implementation. Section 4 presents strong and weak scaling studies to demonstrate the utility of the approach at scale. Finally, conclusions are drawn in Section 5.

2 Multi-node training via Horovod

GAN training consist of alternately optimizing a generator network, whose task is to generate images from a noise input, and a discriminator network, whose task is to recognize reals from generated images in order to guide the generator in producing more realistic images.

The original StyleGAN2 training program is launched as a single process where all available GPU devices are visible to the host. In the program, GPUs receive a copy of the networks and the optimizers to train the model in parallel. Parallel operations are designated to the individual GPUs using Tensorflow's device contexts. The common gradients for the global model update are averaged using Tensorflow's `nccl_ops.all_sum` method which outsources the reduction to the NVIDIA Collective Communications Library (NCCL) to leverage very fast NVLink interconnects. The majority of earlier work on StyleGAN2 have been undertaken using an NVIDIA DGX node with eight GPUs.

We implemented the multi-node training capability for StyleGAN2 using Horovod [25]. Horovod is a distributed deep learning training framework whose foundations are in Message Passing Interface (MPI). In contrast to the original StyleGAN2 context-based parallelism, the new Horovod-based implementation leverages parallel processes to distribute workloads. This makes the approach completely agnostic to the underlying system architecture. In this section, we detail the changes to the original StyleGAN2 to allow multi-node training via Horovod.

2.1 Process parallelism

A setup of a single GPU per process is adopted. After importing Horovod, parallel processes can by launched using Horovod's initializer, as shown in Figure 1. Processes can be bound to unique GPU devices by setting the `CUDA_VISIBLE_DEVICES` environment variable as the local rank in the `run_training.py` module. Horovod's helper functions `hvd.rank` and

```
import horovod.tensorflow as hvd

hvd.init()
os.environ['CUDA_VISIBLE_DEVICES'] = str(hvd.local_rank())
run(**vars(args))
```

Fig. 1: Initialising Horovod processes and binding them to unique GPU devices.

```
dset = tf.data.TFRecordDataset(tfr_file, compression_type='',
                               buffer_size=buffer_mb<<20)
if self._sharding:
    dset = dset.shard(num_shards=hvd.size(), index=hvd.rank())
```

Fig. 2: TFRecord dataset sharding.

`hvd.local_rank` return the ID of a process globally and within a node, respectively.

Since GANs heavily build upon random number generators to draw samples from the latent space, it is important to initialise all processes with a unique random seed. In StyleGAN2, this can be done by perturbing the default seed with the global rank IDs in the `run_training.py` module as `tf_config ={'rnd.np_random_seed': 1000 + hvd.rank()}`. Note that different random seeds also affect random initialisers, causing differences in the initial states across ranks. Before the start of the training process in the `training_loop.py` module, all global variables must be broadcasted from the root rank (0) using `hvd.broadcast_global_variables(0)` to ensure that training starts from the same initial state.

2.2 Data sharding

Sharding is a term used to describe the process where the input dataset is split into smaller partitions to ensure that all parallel copies train with unique data. By default, StyleGAN2 uses the TFRecord binary format to load the data from disk and the data loader can automatically split the data for multiple GPUs within a single process. However, as we now deal with multiple processes, and hence also with multiple copies of the data loader, we need to explicitly tell each process to use its own shard. Figure 2 illustrates the sharding implementation in the `dataset.py` module, where the `dset.shard(num_shards=hvd.size(), index=hvd.rank())` takes every nth sample into the process-specific shard, with n being the global rank.

2.3 Gradient averaging

A common practice with Horovod is to use a distributed optimizer class which abstracts all communication. StyleGAN2 uses a custom optimizer class and therefore the communication was implemented using Horovod's reduction primitive wrappers. Specifically, gradients can be summed using `hvd.allreduce(grad,`

`average=False)` in the `apply_updates` method in the `optimizer.py` module, which replaces the `nccl_ops.all_sum` method. In contrast to the original `nccl_ops.all_sum` method which takes all device arrays as an input argument, `hvd.allreduce` takes only the unique tensor available to the process. This makes communication completely agnostic to the system configuration as Horovod formulates the reduction pattern using the local and global rank hierarchy. The default allreduce operator is based on the ring-allreduce approach that is bandwidth optimal [25]. After the reduction operation has been executed, the model updates occur efficiently per device.

To ensure that Horovod is performing the reduction operations efficiently using NCCL, using a clean containerised environment is recommended. In this work, we run the `nvcr.io/nvidia/tensorflow:20.08-tf1-py3` container cloned from the NVIDIA NGC registry which contains Horovod.

2.4 Multi-node metrics

StyleGAN2 uses Fréchet Inception Distance (FID) [23] metric to assess the quality of generated images. Multi-node acceleration for the computation of this metric in the `frechet_inception_distance.py` module was enabled as follows.

First, the Inception network [26] activations for the real images need to be computed only once and therefore it is unnecessary to distribute this one-off task. These activations are stored as a pickled Python object onto disk during the first metric evaluation step. It is important to consider that the activations are computed using the entire dataset, not the process-specific data shards. Second, all processes load the pickled object as well as the Inception network to compute the activations for synthetic images. Third, all processes take part in generation of synthetic images using rank-specific random seeds. Subsequently, these are run through the Inception network and the activation results from all processes are gathered using `hvd.allgather`. Finally, the FID metric between the activations of real and synthetic images is computed using NumPy on the CPU.

3 Validation

Three training runs were performed to validate that:

- New implementation does not deteriorate single-node performance.
- New implementation does not compromise the quality of the generated images.

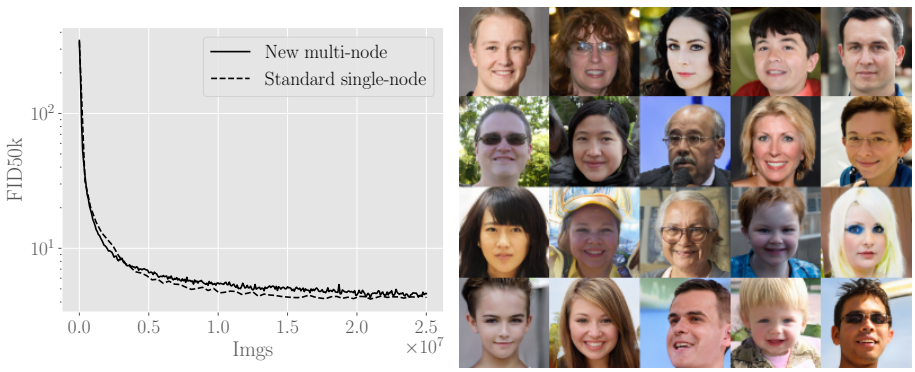
We used StyleGAN2 configuration-`f` and Flickr-Faces-HQ (FFHQ) dataset, in 256x256 resolution, to train our models. FFHQ is a dataset of human faces, containing 70,000 unique images. StyleGAN2 constructs the network and sets the hyperparameters automatically based on the given configuration identifier. For StyleGAN2 configuration details, see [5]. Table 1 describes the run configurations and reports their performance in wall-time seconds per thousand images. An

Table 1: Run configurations and their performance in wall-time seconds per thousand images.

Case	N_{GPU}	Performance (s/kimgs)
Standard single-node	4	30.8
New single-node	4	28.5
New multi-node	16	9.3

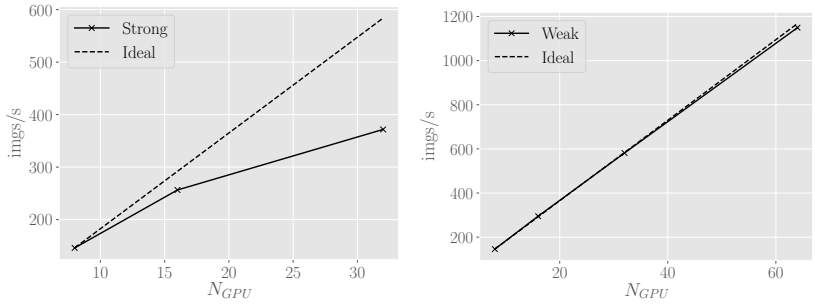
effective batch size (batchsize $\times$ number of nodes) of 32 was used throughout the study. All cases were run on CSC's Puhti-AI cluster, each node containing four NVIDIA Tesla V100 GPUs connected via NVLink. The nodes are connected with Mellanox HDR100 InfiniBand links. From the performance numbers, it can be seen that Horovod-based training outperforms the standard device context-based implementation by a factor 1.08x on a single node. Using four nodes the speed-up factor relative to the standard single-node implementation goes as high as 3.3x. Figure 3 shows the FID metric, computed using 50,000 real and generated images, for the 4 GPU single-node and 16 GPU multi-node setting. The models were trained until 25×10^6 real images had been processed. The multi-node implementation does not have any negative effect on the training dynamics and the model convergences to a level comparable to that of standard StyleGAN2 [5].

Figure 3b shows synthetic images generated using the multi-node-trained model. These images do not show any signs of quality issues relative to the standard single-node implementation. On the whole, the evolution of the FID50k metric and visual quality of the images demonstrate that multi-node training with the new Horovod-based implementation does not compromise the accuracy of StyleGAN2.



(a) (b)

Fig. 3: (a) Evolution of FID for the new multi-node StyleGAN2 case, together with reference data obtained with single-node StyleGAN2. (b) Images generated using the model that was trained with 16 GPUs across four nodes.



(a) (b)

Fig. 4: (a) Strong scaling up 32 GPUs on NVIDIA DGX A100 nodes, using an effective batch size of 64. (b) Weak scaling up 64 GPUs on NVIDIA DGX A100 nodes, using an effective batch size of 64 × the number of nodes.

4 Scaling tests

In addition to the validation tests, strong and weak scaling of the multi-node StyleGAN2 implementation was studied using state-of-the-art hardware. All results presented in this section were run on NVIDIA's Selene supercomputer with DGX A100 nodes, each containing eight NVIDIA Tesla A100 GPUs. We used StyleGAN2 configuration-`f` and FFHQ dataset, in 256x256 resolution, to train the models for strong and weak scaling tests.

4.1 Strong scaling

The strong scaling study was undertaken with a constant effective batch size of 64. Keeping the effective batch size constant ensures that increasing the GPU and node count does not have adverse effects on training dynamics. Figure 4a shows the strong scaling up 32 GPUs distributed across four nodes. We can see that parallel efficiency decreases with the GPU count since the batch size per GPU decreases. In this case, 63 % parallel efficiency is retained at 32 GPUs which is still substantial, considering that the batch size per GPU is as low as two. By quadrupling the GPU count one can speed-up the training process by more than 2.5x.

4.2 Weak scaling

The weak scaling study was undertaken with an effective batch size of 64 × the number of nodes. Scaling the batch size with the number of nodes allows significantly more images to be processed in parallel. Figure 4b shows the weak scaling up to 64 GPUs across eight nodes. In this case, the scaling is nearly linear and 98 % parallel efficiency is retained at 64 GPUs. However, increasing the effective batch size can change the underlying training dynamics and cause instabilities. In [3], it was shown that GANs can benefit from large batch sizes when stabilization techniques such as spectral normalisation [24] and R_1 gradient

penalty [27] for the discriminator are being are used. Training dynamics in the context of very large batch sizes has been identified as an avenue for future work.

5 Conclusion

Typical GAN architectures contain a large number of learnable parameters and they need to be trained with large data sets, which makes the process computationally very expensive and time-consuming. We implemented a data-parallel multi-node training capability to StyleGAN2 via Horovod to allow effective utilisation of larger system architectures. We demonstrated that the new Horovod-based multi-node implementation can outperform the previous device context-based multi-GPU implementation approach on a single node. Furthermore, we demonstrated that multi-node training does not compromise the accuracy of StyleGAN2 for a constant effective batch size. Finally, we reported strong and weak scaling of the new implementation on NVIDIA Tesla A100 GPUs. In a weak scaling sense i.e. keeping the effective batch size constant, the implementation achieves 63% parallel efficiency up to 32 GPUs. In a strong scaling sense i.e. scaling the batch size with the number of nodes, the implementation achieves more than 98% parallel efficiency up to 64 GPUs. On the whole, these results demonstrate the utility of the new multi-node training approach at scale.

Acknowledgment

The authors would like to thank Associate Professor Jaakko Lehtinen and Tero Karras for help with the StyleGAN2 codebase, and CSC – IT Center for Science for the GPU resources on Puhti-AI via project ID 2002415.

References

1. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf
2. T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *CoRR*, vol. abs/1710.10196, 2017. [Online]. Available: http://arxiv.org/abs/1710.10196
3. A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *Proc. ICLR*, 2019.
4. T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
5. T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

6. D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," *CoRR*, vol. abs/1406.5298, 2014. [Online]. Available: http://arxiv.org/abs/1406.5298
7. A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proc. NIPS*, 2017.
8. A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," in *Proc. NeurIPS*, 2019.
9. A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel recurrent neural networks," in *ICML*, 2016, pp. 1747–1756.
10. A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional image generation with PixelCNN decoders," *CoRR*, vol. abs/1606.05328, 2016.
11. L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using Real NVP." *CoRR*, vol. abs/1605.08803, 2016.
12. D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," *CoRR*, vol. abs/1807.03039, 2018.
13. J. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *CoRR*, vol. abs/1703.10593, 2017.
14. Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," *CoRR*, vol. abs/1711.09020, 2018.
15. Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha, "Stargan v2: Diverse image synthesis for multiple domains," *CoRR*, vol. abs/1912.01865, 2019.
16. J. Kim, M. Kim, H. Kang, and K. Lee, "U-gat-it: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation," *CoRR*, vol. abs/1907.10830, 2019.
17. C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *CoRR*, vol. abs/1609.04802, 2016.
18. T. R. Shaham, T. Dekel, and T. Michaeli, "Singan: Learning a generative model from a single natural image," in *Proc. ICCV*, 2019.
19. S. Bell-Kligler, A. Shocher, and M. Irani, "Blind super-resolution kernel estimation using an internal-gan," in *Proc. NeurIPS*, 2019.
20. K. S. Aidan Clark, Jeff Donahue, "Adversarial video generation on complex datasets," *CoRR*, vol. abs/1907.06571, 2019.
21. M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," *CoRR*, vol. abs/1701.07875, 2017.
22. I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," *CoRR*, vol. abs/1704.00028, 2017.
23. M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local Nash equilibrium," in *NIPS*, 2017, pp. 6626–6637.
24. T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *CoRR*, vol. abs/1802.05957, 2018.
25. A. Sergeev and M. D. Balso, "Horovod: fast and easy distributed deep learning in TensorFlow," *arXiv preprint arXiv:1802.05799*, 2018.
26. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. CVPR*, 2016.
27. L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for GANs do actually converge?" *CoRR*, vol. abs/1801.04406, 2018.