Stable Parallel Training of Wasserstein Conditional Generative Adversarial Neural Networks

*Note: Sub-titles are not captured in Xplore and should not be used

1st Massimiliano Lupo Pasini
Computational Sciences and Engineering Division
Oak Ridge National Laboratory
Oak Ridge, USA
lupopasinim@ornl.gov

2nd Junqi Yin
National Center for Computational Sciences Division
Oak Ridge National Laboratory
Oak Ridge, USA
yinj@ornl.gov

Abstract—We use a stable parallel approach to train Wasserstein Conditional Generative Adversarial Neural Networks (W-CGANs). The parallel training reduces the risk of mode collapse and enhances scalability by using multiple generators that are concurrently trained, each one of them focusing on a single data label. The use of the Wasserstein metric reduces the risk of cycling by stabilizing the training of each generator. We apply the approach on the CIFAR10 and the CIFAR100 datasets, two standard benchmark datasets with images of the same resolution, but different number of classes. Performance is assessed using the inception score, the Fréchet inception distance, and image quality. An improvement in inception score and Fréchet inception distance is shown in comparison to previous results obtained by performing the parallel approach on deep convolutional conditional generative adversarial neural networks (DC-CGANs). Weak scaling is attained on both datasets using up to 100 NVIDIA V100 GPUs on the OLCF supercomputer Summit.

Index Terms—Artificial Intelligence, High Performance Computing, Multicore processing

I. Introduction

Generative adversarial neural networks (GANs) [1] [2] [3] [4] are deep learning (DL) models whereby a dataset is used by an agent, called the generator, to sample white noise from a latent space and simulate a data distribution to create new (fake) data that resemble the original data it has been trained on. Another agent, called the discriminator, has to correctly discern between the original data (provided by the external environment for training) and the fake data (produced by the generator). The generator prevails over the discriminator if the latter does not succeed in distinguishing anymore the original from the fake. The discriminator prevails over the generator if the fake data created by the generator is categorized as fake and the original data is still categorized as original. The interplay between generator and discriminator can be interpreted either as a collaborative game, or a competitive game, according to the specifics of the application. In national security applications, the generator and the discriminator are truly adversarial because the generator plays the role of a hacker that tries to breach through a security barrier and the discriminator aims at correctly distinguishing between legitimate operations of regular users against harmful illegal attacks operated by hackers. In other situations, the discriminator collaborates with the generator by helping it to improve its performance. An example is provided by data augmentation, where the generator aims at sampling from a given data distribution by extracting relevant features that can be used to produce new data samples. Another example occurs in healthcare, where GANs are used for the design of new drugs. In this case, the goal of the generator is to propose the composition of new drugs either to improve existing treatments or to propose treatments to types of diseases that are not curable yet, and the goal of the discriminator is to help the generator design more effective drugs by assessing their efficacy. GANs agents collaborate with each other also in animation applications, where the generator is in charge of creating virtual (but still realistic) representations of the reality, which are used in video games to formulate alternative scenarios, and the discriminator provides feedback to the generator to quantify how realistic the proposed scenario is. One final example of an application where the GANs training is set as a formally adversarial, but essentially a collaborative game is provided by photograph editing. In this context, GANs can be used for reconstructing images of faces to identify changes in features such as hair color, facial expressions, or gender, etc. This can help authorities identify criminals that might have undergone surgeries to modify their appearance.

In general, the training of GANs runs into two main numerical issues, namely cycling [5] and mode collapse [6]. Cycling happens when the generator alternates between differ-

ent regions of the data space without necessarily improving its performance. Mode collapse happens when the generator gets stuck in a small region of the data space and produces the same image over time. Avoiding cycling prevents the generator from wasting computational power by exploring the same region. Avoiding mode collapse allows the generator to escape local minima and more thoroughly explore the data space to ensure that the entire data distribution is equally represented in the new generated data. Cycling is due to large gradients. While large gradients are needed to escape local minima, thereby avoiding mode collapse, to prevent the cycling induced by large gradients, one needs stabilization. Existing approaches to train GANs address separately either cycling or mode collapse, but not both simultaneously. Performing a parallel independent [7] training of Conditional GANs (CGANs) [8]–[13] that assigns different classes to different processes has been recently showed to reduce the chances of mode collapse, but does not address cycling. Wasserstein GANs (WGANs) [14] address the problem of cycling, but do not address mode collapse.

To address simultaneously the problem of cycling and mode collapse in the parallel independent training of GANs, we use the Wasserstein metric to define the cost functions associated with the discriminator and the generator, and adapt WGANs to a conditional variant [7], where the data label is used as additional input to the generator in conjunction to the white noise. From now on, we refer to this variant as Wasserstein Conditional GANs (W-CGANs). In situations where the data is characterized by a large number of data classes, our approach fully takes advantage of high performance computing (HPC) resources because the number of processes engaged in the trainign of GANs scales with the number of classes. Numerical results performed on CIFAR10 [15] and CIFAR100 [16] show that W-CGANs stabilize the parallel training and lead to the production of better images with respect to past results obtained with the parallel training on deep convolutional GANs (DC-GANs) [7]. The better performance of W-CGANs with respect to DC-CGANs is validated both in quantitative terms using performance metrics as well as by visual inspection.

II. BACKGROUND ON GANS

In the context of GANs, original data and new generated data are described through two probability distributions. Given two probabilities p and q defined on a metric space $\mathcal{X}$, the Kullback–Leibler divergence (KL) [17]

$$KL(p, q) = \int_{\mathcal{X}} p(x) \log \left(\frac{p(x)}{q(x)} \right) dx, \qquad (1)$$

and its symmetrization, the Jensen–Shannon divergence (JS) [18]

$$JS(p, q) = \frac{1}{2} KL(p, q) + \frac{1}{2} KL(q, p) \qquad (2)$$

measure the distance between the probability distributions p and q by computing the pointwise discrepancy in the values

attained. The Wasserstein distance between p and q is defined as

$$W(p, q) = \sup_{Lip(f) \leq 1} \int_{\mathcal{X}} f(x) \big[p(x) - q(x) \big] dx, \qquad (3)$$

where $Lip(f) \leq 1$ represents the family of Lipschitz functions defined on $\mathcal{X}$ with Lipschitz constant less or equal to one. KL divergence and JS divergence attain indefinitely large values for any situations where the peaks of the two distributions do not overlap, and their value abruptly drops to zero only when the peaks of the distributions are located at the same point. In comparison, the Wasserstein metric represents a more informative estimate for measuring the distance between two probability distributions than KL divergence and JS divergence, since it measures the distance between the two expected values of the probabilities and suggests an update of the two probabilities to reduce that distance.

In order to contextualize the use of the Wasserstein metric for CGANs, we define the following input and output spaces, each with an associated probability distribution:

- Z is a noise space used to seed the generative model. $Z = \mathbb{R}^{d_Z}$, where d_Z is a hyperparameter. Values $\mathbf{z} \in Z$ are sampled from a noise distribution $p_{\mathbf{z}}(\mathbf{z})$. In our experiments $p_{\mathbf{z}}$ is a white noise distribution.
- Y is an embedding space used to condition the generative model on additional external information, drawn from the training data. $Y = \mathbb{R}^{d_Y}$, where d_Y is a hyperparameter. Using conditional information provided in the training data, we define a density model $p_{\mathbf{y}}(\mathbf{y})$.
- X is the data space which represents an image output from the generator or input to the discriminator. In our application the data are colored face images. Values are normalized pixel values: $X = [0, 1]^W \times C$, where W represents the number of pixels in the images, and C is the set of distinct color channels in the input images. Using the images in the training data and their associated conditional data, we can define a density model $p_{\text{data}}(\mathbf{x})$ of face images. This is exactly the density model we wish to replicate.

We now define two functions:

- $G : Z \times Y \to X$ is the conditional generative model (or generator), which accepts noise data $\mathbf{z} \in Z$ and produces an image $\mathbf{x} \in X$ conditional to the external information $\mathbf{y} \in Y$.
- $D : X \to [0, 1]$ is the discriminative model (or discriminator), which accepts an image $\mathbf{x}$ and condition $\mathbf{y}$ and predicts the probability under condition $\mathbf{y}$ that $\mathbf{x}$ came from the empirical data distribution rather than from the generative model.

The goal of the discriminator is to maximize the discrepancy between the data distribution and the probability distribution created by the generator G. For W-CGANs this means that the discriminator aims at maximizing the distance between the expected value of the probability distribution associated with the original data $\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} D(\mathbf{x})$ and the expected value of the probability distribution associated with fake data $\mathbb{E}_{\mathbf{z}} D(G(\mathbf{z}))$.

In order to use the Wasserstein metric to define the cost function of the discriminator, we choose $D(\mathbf{x})$ to play the role of the function f in Equation (3), and we substitute the two probabilities p and q with $p_{\text{data}} : \mathcal{X} \to [0,1]$ and $p_{\text{model}} : Z \to [0,1]$. Using the definition of expected value of D under the probability distribution p_{data} and p_{model} as follows

$$\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} D(\mathbf{x}) = \int_{\mathcal{X}} D(\mathbf{x}) p_{\text{data}}(\mathbf{x}) d\mathbf{x} \tag{4}$$

$$\mathbb{E}_{\mathbf{z} \sim p_{\text{model}}} D(G(\mathbf{z})) = \int_{\mathcal{Z}} D(G(\mathbf{z})) p_{\text{model}}(\mathbf{z}) d\mathbf{z}, \tag{5}$$

we define the cost function for the discriminator as

$$J_{Wasserstein}^{(D)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = \sup_{\boldsymbol{\theta}^{(D)}} \left[\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} D(\mathbf{x}) - \mathbb{E}_{\mathbf{z}} D(G(\mathbf{z})) \right]. \tag{6}$$

The generator G is parametrized with $\boldsymbol{\theta}^{(G)}$ and we define its cost function as

$$J_{Wasserstein}^{(G)}(\boldsymbol{\theta}^{(D)}, \boldsymbol{\theta}^{(G)}) = \sup_{\boldsymbol{\theta}^{(G)}} \left[\mathbb{E}_{\mathbf{z}} D(G(\mathbf{z})) \right]. \tag{7}$$

Contextualizing the difference between the Wasserstein metric and the KL divergence and JS divergence in the case of CGANs, we can explain why the Wasserstein metric reduces the occurrence of cycle during the training of CGANs with respect to situations where the training of CGANs is driven by the minimization of the KL divergence. Indeed, the KL divergence and JS divergence can be reduced by keeping the expected value of the probability distribution fixed, and just augmenting the variance. However, by doing so, the characteristics of the points sampled by the generator (new data) may not necessarily change significantly. On the other side, the minimization of the Wasserstein metric forces the probability distribution associated with the new generated data to shift in order to have the expected value $\mathbb{E}_{\mathbf{z}} D(G(\mathbf{z}))$ get closer to $\mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} D(\mathbf{x})$. Therefore, it is less likely that new data sampled at later iterations will resemble data generated at earlier iterations.

Our parallel approach to train W-CGANs relies on the equality

$$p_{\text{model}}(\mathbf{x}, \boldsymbol{\theta}) = \sum_{k=1}^{K} p_{\text{model}}(\mathbf{x}, \boldsymbol{\theta}|\mathbf{y}_k) p_{\mathbf{y}}(\mathbf{y}_k) \tag{8}$$

that allows to parallelize the computation of each term $p_{\text{model}}(\mathbf{x}, \mathbf{y}_k)$ by training K parallel W-CGANs, each one per class, and then we combine the results at the end of each training to yield $p_{\text{model}}(\mathbf{x})$. The numerical examples presented in this paper are characterized by a one-to-one mapping between $\mathbf{y}_k$ and the labels in the image dataset. The advantage of our approach consists in the fact that all the K parallel W-CGANs can be trained concurrently and independently of each other. If the complexity representation of the objects in each category is comparable, the training time for each parallel W-CGANs model is approximately the same, which in turn translates into promising performance in terms of weak
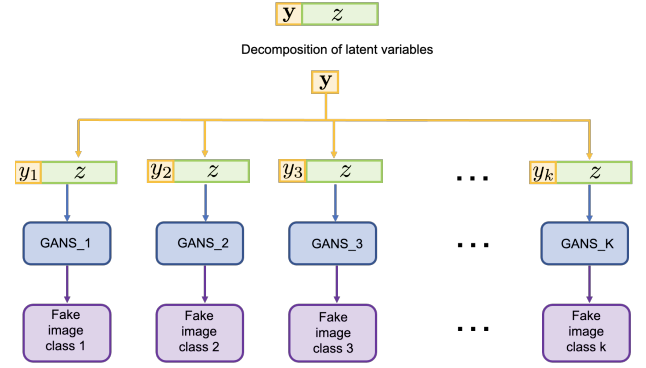

Fig. 1. Illustration of parallel W-CGANs.

scalability. An illustration that describe the distribution of W-CGANs is provided in Figure 1.

The parameters $\boldsymbol{\theta}$ for each replica of the W-CGAN model are updated independently using Adam. When the trained model is deployed, a random number generator provides the white noise and the label of the object whose image has to be generated. The randomly selected label determines which W-CGAN pair to call, and the white noise is passed to the selected W-CGAN pair to generate a new fake image for the specific object category associated with the label. Our approach differs from ensemble learning, where each model replica still has to span the entire dataset in a round-robin fashion. Our parallel approach confines each model replica to be trained on data associated with a single label. Ensemble learning can benefit the W-CGANs training in situations where interpolating across different classes is reasonable. However, image classification generally does not need interpolation (e.g., interpolating an image of a dog with an image of a car produces clearly unrealistic images). Therefore, parallel independent training is appropriate for this type of situations, and in turn it accelerates the training because the number of data batches processed by each GANs pair is significantly reduced. The partition of the data according to the classes facilitates our approach to scale, as confirmed by the weak scaling tests presented at the end of numerical section.

III. NUMERICAL RESULTS

In this section we present numerical results using CIFAR10 and CIFAR100 benchmark image datasets to compare the performance of W-CGANs with DC-CGANs. The two benchmark datasets have the same total number of images, and images of both datasets have the same resolution, but the number of classes represented in the two datasets is different. Specifically, CIFAR10 has more images per class than CIFAR100. Since the GANs training is data intensive, the fact that CIFAR100 has fewer images per class results into a more challenging task for the model to produce good quality images. The specifics of the neural networks used to model generator and discriminator is provided in Tables I and II. Although the architectures of generator and discriminator can be tuned using hyperparameter optimization, this goes beyond the scope of this work since

Generator					
Layer	Input dim	Output dim	Kernel size	stride	padding
Input	100	8192	/	/	
LeakyReLU(slope = 0.2, inplace=True)					
Resizing					
Batch normalization(epsilon = 1e-5 , momentum = 0.1)					
Upsample(scale factor = 2)					
Conv.1	128	128	3	1	1
Batch normalization(epsilon = 0.8 , momentum = 0.1)					
LeakyReLU(slope = 0.2, inplace=True)					
Upsample(scale factor = 2)					
Conv.2	128	64	3	1	1
Batch normalization(epsilon = 0.8 , momentum = 0.1)					
leakyReLU(slope = 0.2, inplace=True)					
Conv.3	64	1 or 3	3	1	1
Tanh					

TABLE I

ARCHITECTURE OF THE GENERATOR.

Discriminator					
Layer	In. dim	Out. dim	Kernel size	stride	padding
Conv.1	1 or 3	16	3	2	1
leakyReLU(slope = 0.2, inplace=True)					
Dropout(0.25)					
Conv.2	16	32	3	2	1
LeakyReLU(slope = 0.2, inplace=True)					
Dropout(0.25)					
Batch normalization(epsilon = 0.8 , momentum = 0.1)					
Conv.3	32	64	3	2	1
leakyReLU(slope = 0.2, inplace=True)					
Dropout(0.25)					
Batch normalization(epsilon = 0.8 , momentum = 0.1)					
Conv.4	64	128	3	2	1
leakyReLU(slope = 0.2, inplace=True)					
Dropout(0.25)					
Batch normalization(epsilon = 0.8 , momentum = 0.1)					
output	2048	1	/	/	/
Sigmoid(for DC-CGANs), no activation function for W-CGANs					

TABLE II

ARCHITECTURE OF THE DISCRIMINATOR.

we aim at improving the performance of GANs for a fixed architecture.

Since ablation studies to assess how the parallel independent training improves the performance of GANs with respect to sequential training have already been performed in [7], in this work we focus on the benefit obtained from combining the Wasserstein metric with the parallel independent training. The training is performed using the optimizer Adam and a learning rate of $2e-4$, and a total number of 1,000 epochs. The comparison between parallel DC-CGANs and parallel W-CGANs is performed on a quantitative level by measuring the Inception Score (IS) [19] and the Fréchet Inception Distance (FID) [14]. The IS takes a list of images and returns a single floating point number, the score. The score is a measure of how realistic a GAN's output is. IS is an automatic alternative to having humans grade the quality of images. The score measures two things simultaneously: the image variety (e.g., each image is a different breed of dog), and whether each image distinctly looks like a real object. If both things are true, the score will be high. If either or both are false, the score will be low. The lowest score possible is zero. Mathematically the highest possible score is infinity, although in practice a finite ceiling is imposed. Unlike IS, which evaluates only the distribution of generated images, the FID compares the distribution of generated images with the distribution of real images that were used to train the generator. Lower values of FID correspond to the distribution of generated images approaching the distribution of real images, and this is interpreted as an improvement of the generator in creating more realistic images.

A. Hardware description

The numerical experiments are performed using Summit [20], a supercomputer at the Oak Ridge Leadership Computing Facility (OLCF) at Oak Ridge National Laboratory. Summit has a hybrid architecture, and each node contains two IBM POWER9 CPUs and six NVIDIA Volta GPUs all connected together with NVIDIA's high-speed NVLink. Each node has over half a terabyte of coherent memory (high bandwidth memory + DDR4) addressable by all CPUs and GPUs plus 1.6 TB of non-volatile memory (NVMe) storage that can be used as a burst buffer or as extended memory. To provide a high rate of communication and I/O throughput, the nodes are connected in a non-blocking fat-tree using a dual-rail Mellanox EDR InfiniBand interconnect.

B. Software description

The numerical experiments are performed using `Python3.7` with `PyTorch v1.3.1` package [21] for autodifferentiation to train the DL models with the use of GPUs, and the `mpi4py v3.0.2` tool is used for parallel computing.

C. CIFAR10

The training portion of the CIFAR10 dataset [15] consists of 50,000 32x32 color images in 10 classes, with 5,000 images per class. The classes represented in the dataset are airplanes, automobiles, birds, cats, deers, fogs, frogs, horses, ships, and trucks.

A comparison in quantitative terms between parallel DC-CGANs and the parallel W-CGANs is shown in Table III where the performance of the models is measured in terms of IS and FID, and W-CGANs outperform DC-CGANs with respect to both indices. A visual comparison between the images produced by DC-CGANs in Figure 2 and the images produced by W-CGANs in Figure 3 shows that W-CGANs succeeds produces objects with more refined contours, and the objects are much easier to recognize with respect to the object class they attempt at sampling. Weak scaling plot for DC-CGANs and W-CGANs trained on the CIFAR10 dataset is presented on the left side of Figure 6 by reporting the runtime of the slowest process. The independence of the generator-discriminator pairs allows the code to scale with the number of processors that take care of separate data classes. The average GPU utilization is 87.5% with a standard deviation of 2.2%, and the memory utilization is 7,839 mebibytes (MiB).

	IS	FID
Parallel DC-CGANs	6.43	9.41
Parallel W-CGANs	7.43	8.53

TABLE III
INCEPTION SCORE (IS) AND FRÉCHET INCEPTION DISTANCE (FID) FOR THE TRAINING OF PARALLEL DC-CGANS AND PARALLEL W-CGANS ON CIFAR10.
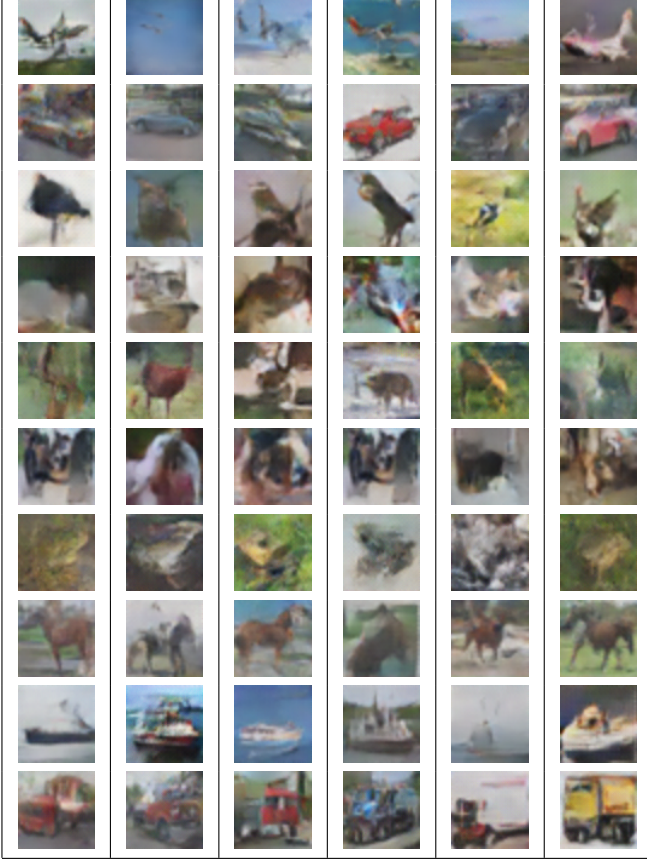

Fig. 2. Fake images generated by parallel DC-CGANs trained on the CIFAR10 dataset.

Fig. 3. Fake images generated by parallel W-CGANs trained on the CIFAR10 dataset.

	IS	FID
Parallel DC-CGANs	6.61	9.23
Parallel W-CGANs	6.93	8.92

TABLE IV
INCEPTION SCORE (IS) AND FRÉCHET INCEPTION DISTANCE (FID) FOR THE TRAINING OF PARALLEL DC-CGANS AND PARALLEL W-CGANS ON THE CIFAR100 DATASET.

D. CIFAR100

The training portion of the CIFAR100 dataset [16] consists of 50,000 32x32 color images in 100 classes, with 500 images per class. We refer the reader to [16] for details about the objects represented in each class.

The fact that CIFAR100 has fewer image data per class than CIFAR10 makes the GANs training more difficult, but the use of the Wasserstein metric stabilizes the parallel training of W-CGANs, which results into a higher IS score and lower FID score as shown in Table IV. A visual comparison between the images produced by DC-CGANs in Figure 4 and the images produced by W-CGANs in Figure 5 shows that W-CGANs outperforms DC-CGANs on objects that are particularly complex to represent, such as houses surrounded by a garden (second row, third picture from the left), butterflies (second row, fifth picture from the left), and camels (second row, sixth picture from the left).

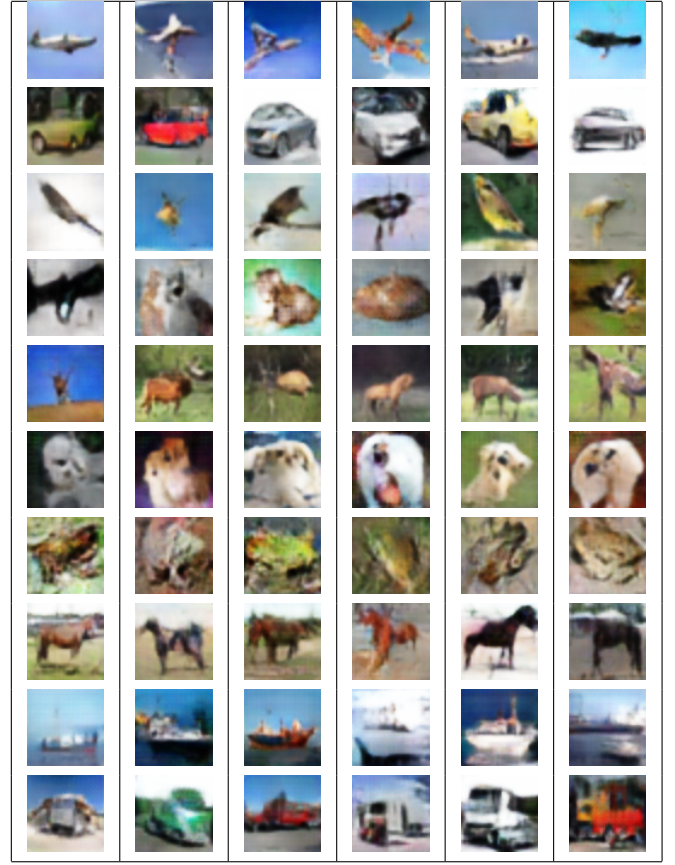Weak scaling plot for DC-CGANs and W-CGANs trained on the CIFAR100 dataset is presented on the right side of Figure 6, by reporting the runtime of the slowest process. Also in this case, the independence of the generator-discriminator pairs allows the code to scale with the number of processors that take care of separate data classes. The average GPU utilization is 93.8% with a standard deviation of 0.7%, and the memory utilization is 7,839 MiB.

IV. CONCLUSIONS AND FUTURE DEVELOPMENTS

We used the Wasserstein metric to stabilize the parallel training of CGANs, thereby maintaining scalability while avoiding both cycling and mode collapse. Numerical results have been presented using CIFAR10 and CIFAR100, two benchmark image dataset with images of the same resolution but different number of classes and different number of image data per class. The fact that CIFAR100 has fewer image data per class makes the GANs training more difficult than for CIFAR10, but the use of the Wasserstein metric stabilizes the parallel training of W-CGANs, which leads to better results as
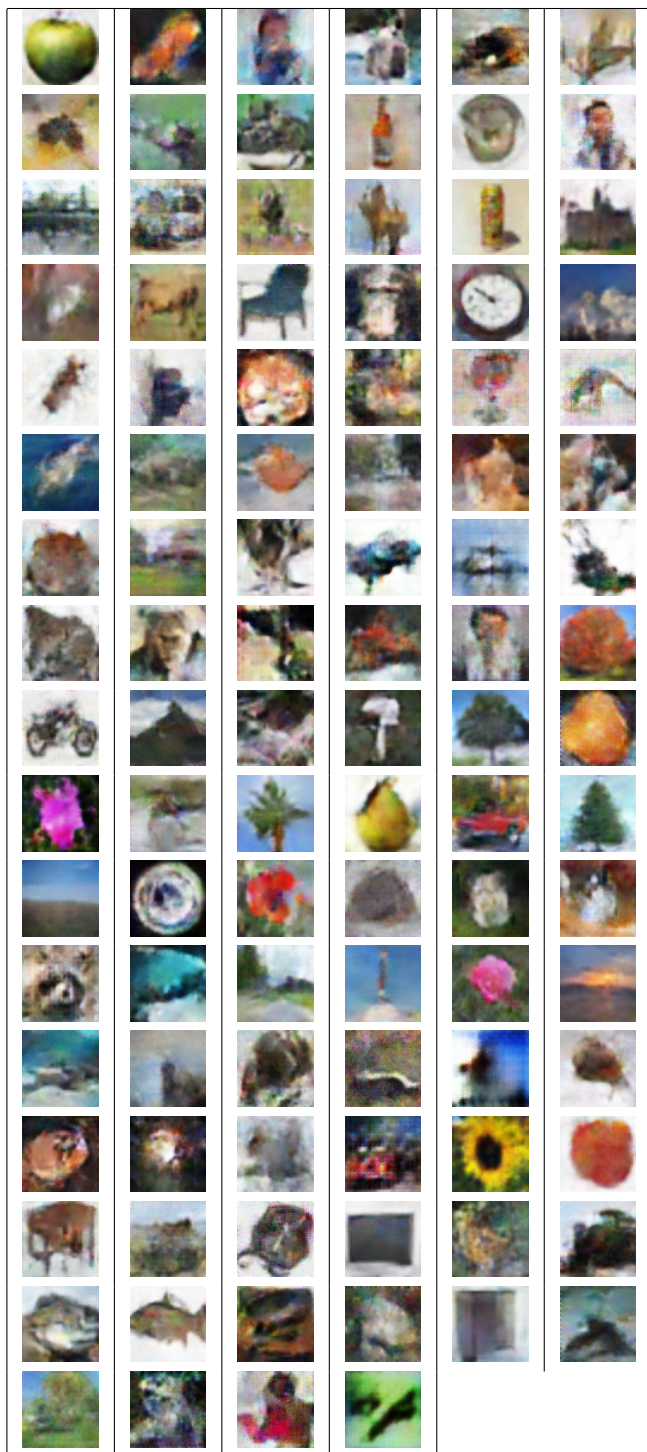
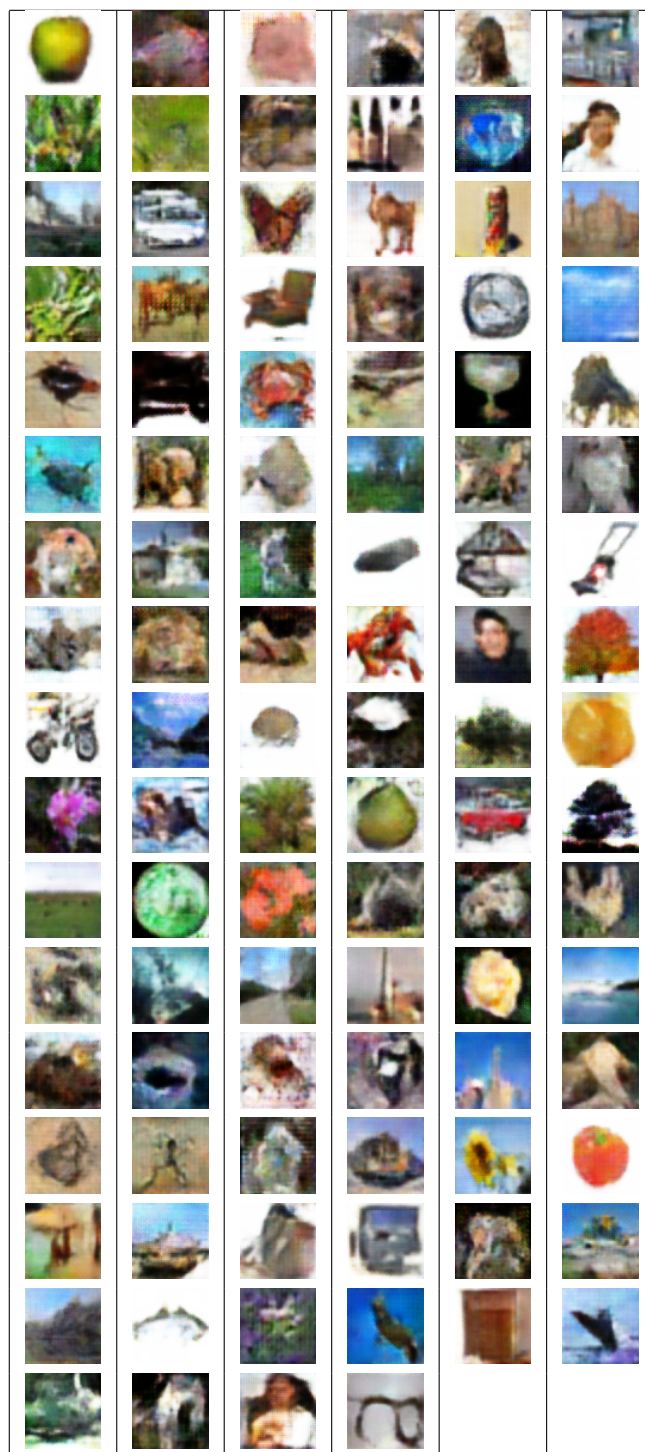Fig. 4. Fake images generated by parallel DC-CGANs trained on the CIFAR100 dataset.

Fig. 5. Fake images generated by parallel W-CGANs trained on CIFAR100.
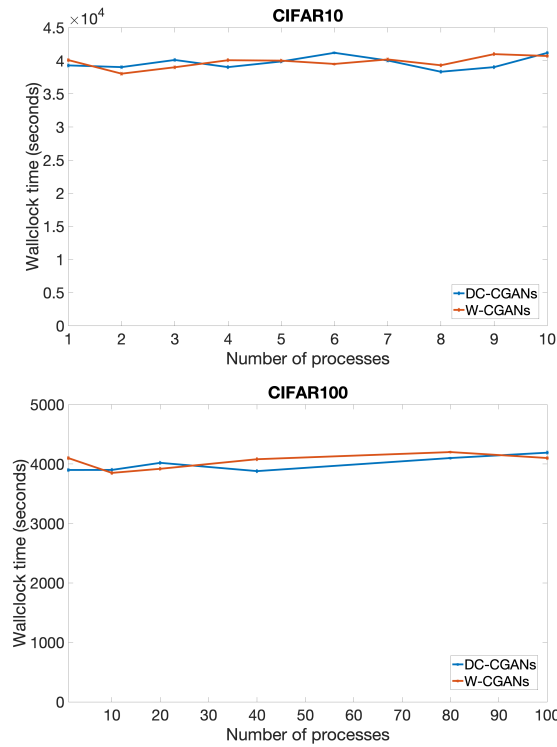
Fig. 6. Results of weak scaling tests for CIFAR10, and CIFAR100. Wall-clock time measured in seconds is reported for the slowest processor.

it is validated by higher IS score, lower FID score, and better quality images than what previously obtained with a parallel training of DC-CGANs [7]. Weak scaling plots reporting the wall-clock time of the slowest process show that the stabilization obtained with W-CGANs does not compromise the scalability of the approach.

Future work will extend this study to more complex but also more accurate neural network architectures, such as auxiliary classifier GANs (AC-GANs) [22], residual neural networks (ResNet) [23], and self-attention generative adversarial neural networks (SAGANs) [24], We will also introduce regular communications between the model replicas to allow distributed training in situations where interpolating between data from different classes is useful.

REFERENCES

[1] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Networks," *arXiv:1406.2661 [cs, stat]*, June 2014. arXiv: 1406.2661.

[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," *arXiv:1511.06434 [cs]*, Jan. 2016. arXiv: 1511.06434.

[3] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training GANs," *arXiv:1606.03498 [cs]*, June 2016. arXiv: 1606.03498.

[4] D. Bertsekas, "Multiagent Rollout Algorithms and Reinforcement Learning," *arXiv:1910.00120 [cs]*, Sept. 2019. arXiv: 1910.00120 version: 1.

[5] P. Mertikopoulos, C. Papadimitriou, and G. Piliouras, "Cycles in Adversarial Regularized Learning," *Proceedings of the 2018 Annual ACM-SIAM Symposium on Discrete Algorithms*, 2017.

[6] E. Hazan, K. Singh, and C. Zhang, "Learning Linear Dynamical Systems via Spectral Filtering," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, pp. 6702–6712, Curran Associates, Inc., 2017.

[7] M. Lupo Pasini, V. Gabbi, J. Yin, S. Perotto, and N. Laanait, "Scalable Balanced Training of Conditional Generative Adversarial Neural Networks on Image Data," *arXiv:2102.10485 [cs.CV, cs.AI]*, Feb. 2021. arXiv:2102.10485.

[8] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," *arXiv:1411.1784 [cs, stat]*, Nov. 2014. arXiv: 1411.1784.

[9] D. Yang, S. Hong, Y. Jang, T. Zhao, and H. Lee, "Diversity-Sensitive Conditional Generative Adversarial Networks," in *International Conference on Learning Representations*, 2019.

[10] P. Zhou, L. Xie, X. Zhang, B. Ni, and Q. Tian, "Searching towards Class-Aware Generators for Conditional Generative Adversarial Networks," *arXiv:2006.14208 [cs.CV, cs.LG]*, June 2020. arXiv:2006.14208.

[11] H. Zhang, V. Sindagi, and V. M. Pate, "Image De-Raining Using a Conditional Generative Adversarial Network," *IEEE Transactions on Circuits and Systems for Video Technology*, Nov. 2020. 10.1109/TCSVT.2019.2920407.

[12] T. Miyato and M. Koyama, "CGANs with Projection Discriminator," in *International Conference on Learning Representations*, 2018.

[13] I. Kavalerov, W. Czaja, and R. Chellappa, "cGANs with Multi-Hinge Loss," *CoRR*, vol. abs/1912.04216, 2019.

[14] 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," *arXiv:1706.08500 [cs.CV, cs.LG]*, June 2017. arXiv:1706.08500.

[15] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)," 2009.

[16] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-100 (canadian institute for advanced research)," 2009.

[17] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[18] J. Lin, "Divergence Measures Based on the Shannon Entropy," *IEEE Transactions on Information Theory*, vol. 37, no. 1, p. 145, 1991.

[19] S. Barratt and R. Sharma, "A Note on the Inception Score," *arXiv:1801.01973 [cs, stat]*, June 2018. arXiv: 1801.01973.

[20] Oak Ridge Leadership Facility, "Summit - oak ridge national laboratory's 200 petaflop supercomputer." https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/, 2018. Accessed: 2021-08-21.

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32* (H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, and R. Garnett, eds.), pp. 8026–8037, Curran Associates, Inc., 2019.

[22] A. Odena, C. Olah, and J. Shlens, "Conditional Image Synthesis with Auxiliary Classifier GANs," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, (International Convention Centre, Sydney, Australia), pp. 2642–2651, PMLR, 06–11 Aug 2017.

[23] M. Wang, H. Li, and F. Li, "Generative Adversarial Network based on Resnet for Conditional Image Restoration," *ArXiv*, vol. abs/1707.04881, 2017.

[24] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention Generative Adversarial Networks," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 7354–7363, PMLR, 09–15 Jun 2019.