Gossiping GANs

Position paper

Corentin Hardy
INRIA/Technicolor
corentin.hardy@technicolor.com

Erwan Le Merrer
Technicolor
elemerrer@acm.org

Bruno Sericola
INRIA
bruno.sericola@inria.fr

Abstract

A recently celebrated kind of deep neural networks is Generative Adversarial Networks. GANs are generators of samples from a distribution that has been learned; they are up to now *centrally* trained from local data on a single location.

We question the performance of training GANs using a spread dataset over a set of distributed machines, using a gossip approach shown to work on standard neural networks [1]. This performance is compared to the *federated learning* distributed method, that has the drawback of sending model data to a server. We also propose a gossip variant, where GAN components are gossiped independently. Experiments are conducted with Tensorflow with up to 100 emulated machines, on the canonical MNIST dataset.

The position of this paper is to provide a first evidence that gossip performances for GAN training are close to the ones of federated learning, while operating in a fully decentralized setup. Second, to highlight that for GANs, the distribution of data on machines is critical (*i.e.*, i.i.d. or not). Third, to illustrate that the gossip variant, despite proposing data diversity to the learning phase, brings only marginal improvements over the classic gossip approach.

1 Introduction

GANs are *generative* models, meaning that they are used to generate new realistic data from the distribution of an existing dataset. Those have been introduced by Goodfellow *et al* in seminal work [5]. Applications are for instance to generate pictures from text descriptions [12], or to get super-resolution from basic images [9]. A GAN is a machine learning model, and more specifically a certain type of deep neural network (noted DNNs hereafter). As for all other DNNs, GANs require a large training dataset in order to implement the target application. Nowadays, the norm is for service providers to collect large amounts of data into their datacenters; the learning phase is then taking place into those premises. There is thus an obvious interest and challenge to learn over spread datasets, possibly located out of the datacenter, in order to leverage as much data as possible.

Background on Generative Adversarial Networks The fundamental particularity of GANs is that their training phase is *unsupervised*, *i.e.*, no description labels are required to learn from the data. A classic GAN is composed of two elements : a *generator* $\mathcal{G}$ and a *discriminator* $\mathcal{D}$. Both are deep neural networks. The generator takes as input a noise signal (*e.g.*, random vectors of size k where each element follows a normal distribution $\mathcal{N}(0,1)$) and generates data with the same format as training dataset data (*e.g.*, a picture of 128x128 pixels and 3 color channels). The discriminator receives as input either some data generated by the generator, or data from training dataset. The goal of the discriminator is to guess from which source the data is coming from. At the beginning of the learning phase, the generator generates data from a probability distribution and the discriminator quickly learns how to differentiate that generated data from the training data. After some iterations, the generator learns to generate data which are closer to the dataset distribution. If eventually, the discriminator is not able to differentiate both, then the generator has learned the distribution of the data in the training dataset (and thus has fitted an unlabeled dataset in an unsupervised way).

Formally, let a given training dataset be included in the data space X, where $\mathbf{x}$ in that dataset follows a distribution probability P_{data}. Let a GAN, composed of generator $\mathcal{G}$ and discriminator $\mathcal{D}$, which tries to learn this distribution. As proposed in the original GAN paper [5], we model the generator by the function $\mathcal{G}_{\mathbf{w}} : \mathbb{R}^k \longrightarrow X$ where $\mathbf{w}$ contains the parameters of its DNN and k is fixed. Similarly, we model the discriminator by the function $\mathcal{D}_{\boldsymbol{\theta}} : X \longrightarrow [0,1]$ where $\mathcal{D}_{\boldsymbol{\theta}}(\mathbf{x})$ is the probability that $\mathbf{x}$ is a data from the training dataset, and $\boldsymbol{\theta}$ contains the parameters of the discriminator. The learning consists in finding the parameters $\mathbf{w}^*$ for the generator :

$$\arg\min_{\mathbf{w}} \max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}\sim P_{\text{data}}} [\log \mathcal{D}_{\boldsymbol{\theta}}(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim P_z} [\log (1 - \mathcal{D}_{\boldsymbol{\theta}}(\mathcal{G}_{\mathbf{w}}(\mathbf{z}))]$$

where $P_{\mathbf{z}}$ is a multivariate normal distribution of a k-dimensional random vector. In this equation, $\mathcal{D}$ tries to minimize the classification error on real data (the left side of the equation) and the error of classification on fake data (in the right side). $\mathcal{G}$ tries to maximize the right side $\mathbb{E}[\log(1 - \mathcal{D}(\mathcal{G}(\mathbf{z}))]$ (it does not have impact on the left side), that means it tries to maximize the classification error of $\mathcal{D}$ on generated data.

2 Federated learning vs gossip

2.1 Experimental setup

We experiment on the Google Tensorflow platform.

DIDL, 2018, Rennes, France
2018. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

We emulate N machines, each containing an equal (i.i.d.) share of the training dataset, so that one GAN per machine is trained, and data shares never leave their original machine. We use the MNIST dataset, which is composed of 60,000 images of handwritten digits of size 28×28 pixels. The dataset is composed of 10 classes (we do not leverage the class labels in experiments as GANs are unsupervised learning methods).

The GAN model used comes from the open source models of the Tensorflow plaftform: $\mathcal{G}$ is composed of two fully-connected layers of sizes 1024 and 6272, followed by 2 transposed convolutional layers with respectively 64 and 32 filters, and a final layer convolutional layer using one filter. $\mathcal{D}$ is composed of 2 convolutional layers with respectively 64 and 128 filters, followed by a fully-connected layer of 1024 neurons and a last one of 1 neuron. We use the same training configuration as for the standard neural network case. $\mathcal{G}$ takes as input batches of random vectors of size 128 where each element is generated from a normal distribution $\mathcal{N}(0,1)$. We set the total number of iterations to $I = 20,000$ for $\mathcal{G}$ in all experiments.

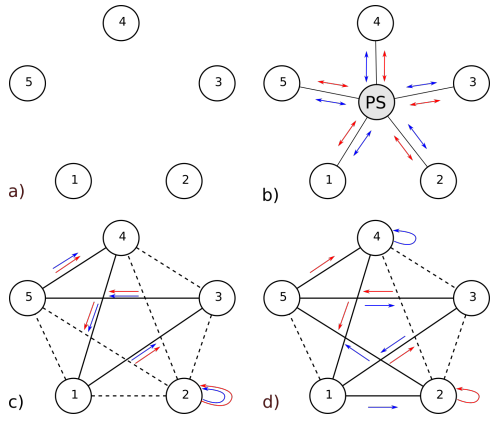

Figure 1. The different communication setups considered: *a)* Stand-alone (*i.e.*, no collaboration between machines), *b)* Federated learning, *c)* Gossip DDL, and *c)* Gossip_ind DDL. Red and blue arrows represent the movement of $\mathcal{G}$ and $\mathcal{D}$.

Competing approaches *(i)* A stand-alone version (Figure 1*a*), intended to illustrate the baseline of a distributed dataset, but non-collaborative computation. In this setup, one GAN is trained per machine without any communication or collaboration with other participants. Each GAN uses only local data on its machine.
(ii) The federated learning [11] approach (Figure 1*b*). We apply it to GANs, by considering that each machine i trains its GAN and then sends its parameters (both $\boldsymbol{w}_i$ and $\boldsymbol{\theta}_i$) every K iterations to a parameter server (PS). This server averages all received $\boldsymbol{w}_i$ and $\boldsymbol{\theta}_i$, and sends them to all machines at the beginning of the new iteration. Please note that federated learning was not proposed for the learning of GANs; we are using this server-centric approach as an expected upper bound on the performance of the trained of GANs.
(iii) The state of the art of distributed deep learning using gossip communication [1], that was not previously experimented on GANs. Every K local iterations, each machine i sends its parameters (both $\boldsymbol{w}_i$ and $\boldsymbol{\theta}_i$) to a neighbour j (Figure 1*c*). We consider the neighbour j as randomly selected among all machines at each communication step. The machine j averages received $\boldsymbol{w}_i$ and $\boldsymbol{\theta}_i$ with their local parameters $\boldsymbol{w}_j$ and $\boldsymbol{\theta}_j$ before running a new learning step of K iterations. We name this approach Gossip DDL.
(iv) A variant of the Gossip DDL where $\mathcal{G}$ parameters ($\boldsymbol{w}$) and $\mathcal{D}$ parameters ($\boldsymbol{\theta}$) are not sent to the same machine during each communication step (independent destination selection, as on Figure 1*d*). So a machine i averages its parameters $\boldsymbol{w}_i$ and $\boldsymbol{\theta}_i$ with $\boldsymbol{w}_j$ and $\boldsymbol{\theta}_k$, where j and k are two different machines. The idea is to mix $\mathcal{G}$ and $\mathcal{D}$ couples during the learning process, in the hope for better performance facing data variety, as illustrated in [8] in a central setup. We name this variant Gossip_ind DDL.

Metrics Evaluating generative models such as GANs is a difficult task. Ideally, it requires human judgment to assess the quality of the generated data. Fortunately, in the domain of GANs, interesting methods are proposed to simulate this human judgment. The main one is named the *Inception Score* (we denote it by IS), and has been proposed by Salimans *et al.* [13], and shown to be correlated to human judgment. The IS consists in applying a pre-trained Inception classifier over the generated data. The Inception Score evaluates the confidence on generated data classification (*i.e.*, generated data are, or not, well recognized by the Inception network), and the diversity of output (*i.e.*, generated data are not all the same). To evaluate the competing approaches on MNIST, we use the MNIST score (we denote it by MS), similar to the Inception score but using a classifier adapted to MNIST data instead of Inception net. Heusel *et al.* propose a second metric named the Fréchet Inception Distance (FID) in [7]. The FID measures a distance between the distribution of generated data $P_{\mathcal{G}}$ and real data P_{data}. It applies the Inception network on a sample of generated data and another sample of real data and suppose their outputs are Gaussian distributions. The FID computes the Fréchet Distance between the Gaussian distribution obtained using the generated data and the Gaussian distribution obtained using real data. As for the Inception distance, we use a classifier more adapted to compute the FID on the MNIST dataset. We use the implementation of the MS and FID available in Tensorflow[1].

2.2 Scalability vs performance

We consider that each machine hosts $1/N$ of the MNIST dataset, randomly i.i.d. split. The overall number of samples remains constant (60,000 pictures). The goal is to assess

[1]We use code from https://github.com/tensorflow/models/blob/master/research/gan/mnist/util.py.

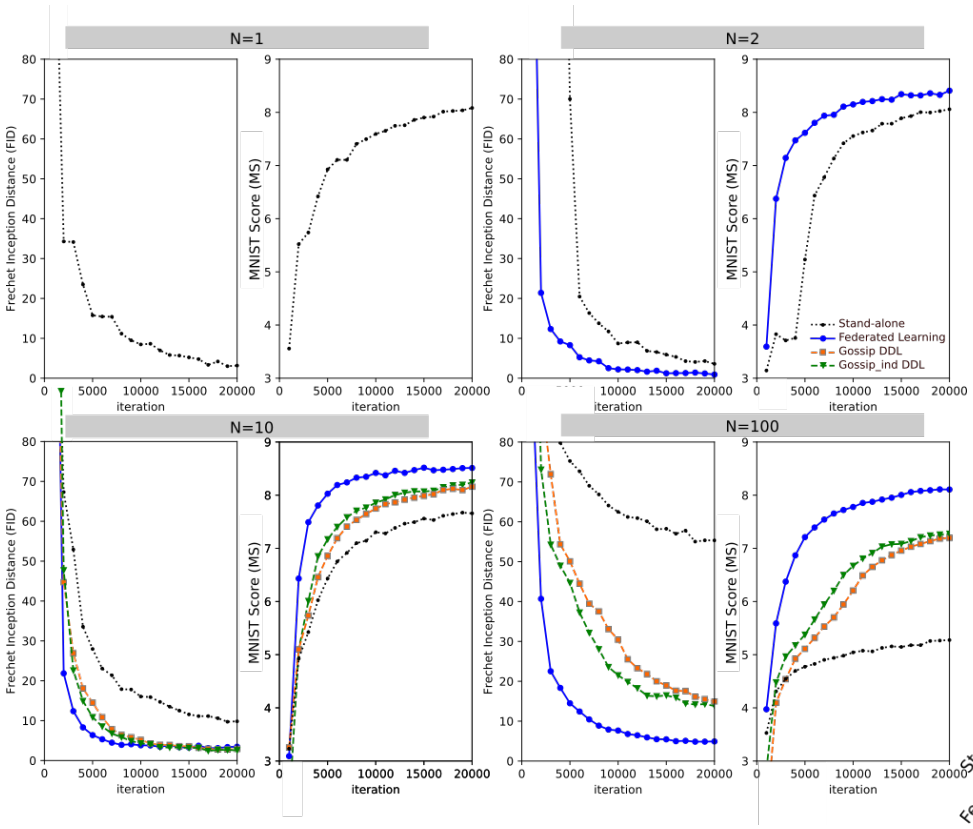

Figure 2. Inception score for MNIST (MS) and the Fréchet Inception Distance (FID) for the four competitors with $N \in \{1, 2, 10, 100\}$. Higher MS and lower FID are better.

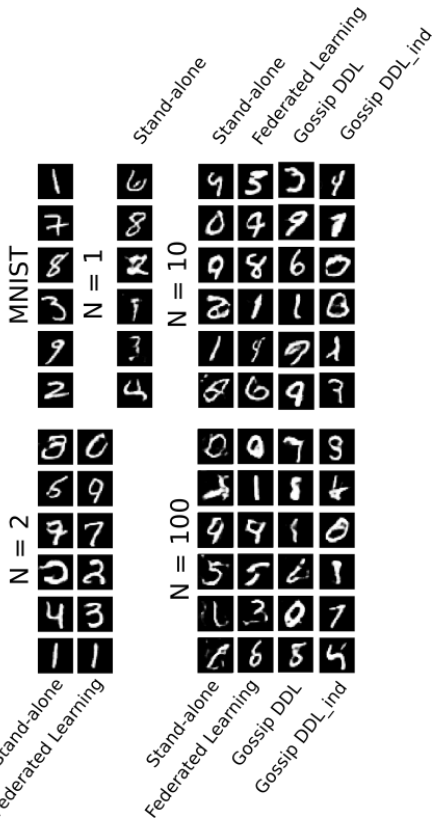

Figure 3. Samples generated by the best GAN of each competing method.

the scalability of competitor performances. We train all approaches with a number $K = 200$ of local iterations between two communication steps (not applicable for the stand-alone method) and a total number of $I = 20,000$ iterations. We run the training for $N \in \{1, 2, 10, 100\}$. Note that $N = 1$ corresponds to the central case, and that for $N = 2$, federated learning and the two gossip methods are equivalents, *i.e.*, the parameters of both nodes are averaged at each communication step (solely the federated learning curve is presented).

The MS (Inception score) and the FID are measured after each communication step. The mean score of all GANs on all machines, and for each competitor, is reported in Figure 2. To better understand the performance of final GANs in practice, we plot few samples generated by those at the end of each run (i.e., after 20,000 iterations), on Figure 3. Finally, Figure 4 presents the final GANs best scores at the end of runs, as well as standard deviations on individual machines.

Experiment Results We first observe that the Stand-alone baseline shows larger score deviations due to non collaboration (Figure 4). Very clearly, Stand-alone learning suffers from the lack of local data in the case where $N = 100$, and thus cannot provide satisfying learning results (Figure 3 for $N = 100$). From this moment onward, the need for a collaborative approach is clearly required for performance reasons.

As expected, federated learning obtains the best results on MS and FID with $N = 10$ and $N = 100$, which are similar or greater than the result of centralized run ($N = 1$). This comes at the cost of sending all $\mathcal{G}$ and $\mathcal{D}$ parameters of each machine to a server to average them at each communication step, as well as requires a large bandwidth on that server for gathering all model updates from machines [6].

Third observation is that the gossip learning methods for GANs can obtain good results too. Gossip DLL methods obtain close results for $N = 10$ and $N = 100$; second we observe that the variant Gossip_ind only marginally improve results. For $N = 10$, Gossip DDL obtains similar results to federated learning for the FID metric, and close enough for the MS metric. For $N = 100$, Gossip DDL results decrease compared to federated learning but are yet much better than the ones of the stand-alone method. Score deviations for gossip methods are larger than for federated learning; this may justify a final aggregation step at the end of runs, where all machines distributedly converge to keeping the best $\mathcal{G}$ and $\mathcal{D}$ couple only. The figure 3 shows that samples generated by gossip methods also look realistic for most of them.

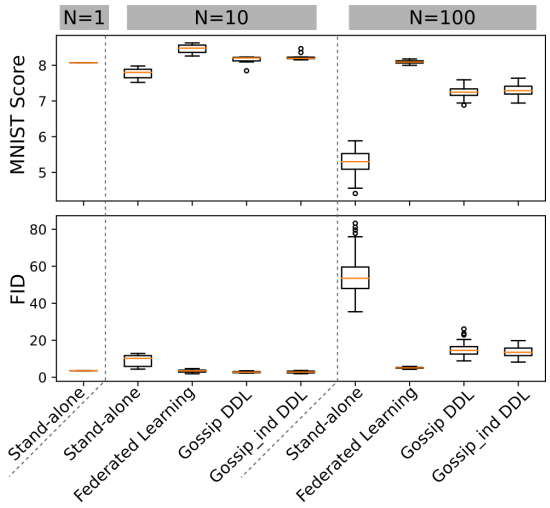

Figure 4. Distribution of final GAN scores (FID and MS) for each competitor, with $N \in \{1, 10, 100\}$.

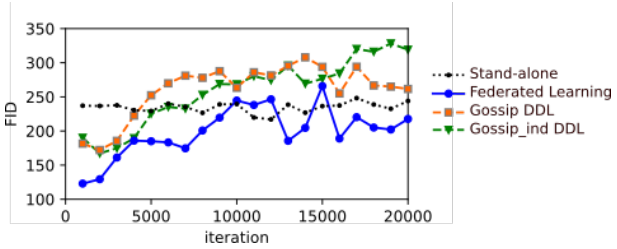

Figure 5. FID for competitors: non i.i.d. training dataset. $N = 10$: each machine hosts the data of a single digit.

We finally run a scenario where data on machines is not i.i.d.: each of the 10 machines hosts images from a single digit. We observe on Figure 5 that none of the competitors manage to reach a decent FID score, underlining the non converge of the learning approaches.

3 Related Work

Distributing the learning of classic deep neural networks over multiple machines is generally performed with the Parameter Server model proposed by Dean et al. in [4]. It has been followed by speed up proposals [2, 3, 10]. Distributed learning on edge-devices (i.e., out of the datacenter) has also recently been proposed [6, 14].

We reviewed the GAN concept [5] in the introduction. The specifics of distributing GANs efficiently for distributed datasets is still an open topic for research, since the tight coupling of generators and discriminators are to be taken into account. This paper compares a gossip approach adapted from [1], as well as a variant with independent gossiping of generators and discriminators; both are in turn compared to a baseline and to federated learning [11].

4 Conclusion

The conclusions of this position paper are that:

(i) the gossiping process of GANs gets the learning performances close enough to the one of federated learning, with the benefit of an absence of a central server. This empirical fact was not reported in [1] even for the gossip of a classic neural model. Futureworks should confirm this observation drawn from the MNIST dataset on more datasets and GAN architectures. *(ii)* for GANs, the initial distribution of data (*i.e.*, i.i.d. or not) is crucial for convergence. *(iii)* the intuition (led by reference [8]) that the competition of shuffled $\mathcal{G}$ and $\mathcal{D}$, in order to bring more diversity, only brings marginally better scores as compared to the basic gossip method.

This calls for improved distributed learning techniques in the specific context of GANs, including more advanced gossiping ones, in order to bridge the gap between federated learning (centralized) and a fully distributed learning step.

References

[1] M. Blot, D. Picard, M. Cord, and N. Thome. 2016. Gossip training for deep learning. *ArXiv e-prints* (Nov. 2016). arXiv:cs.CV/1611.09726

[2] Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. 2016. Revisiting Distributed Synchronous SGD. In *ICLR, Workshop Track*.

[3] Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. 2014. Project Adam: Building an Efficient and Scalable Deep Learning Training System. In *OSDI*.

[4] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, Quoc V. Le, and Andrew Y. Ng. 2012. Large Scale Distributed Deep Networks. In *NIPS*.

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. Generative Adversarial Networks. *ArXiv e-prints* (June 2014). arXiv:stat.ML/1406.2661

[6] C. Hardy, E. Le Merrer, and B. Sericola. 2017. Distributed deep learning on edge-devices: Feasibility via adaptive compression. In *NCA*.

[7] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. (2017). arXiv:cs.LG/1706.08500

[8] Daniel Jiwoong Im, He Ma, Chris Dongjoo Kim, and Graham W. Taylor. 2016. Generative Adversarial Parallelization. *CoRR* abs/1612.04021 (2016). arXiv:1612.04021

[9] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. 2016. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. (2016). arXiv:cs.CV/1609.04802

[10] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. 2014. Scaling Distributed Machine Learning with the Parameter Server. In *OSDI*.

[11] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated Learning of Deep Networks using Model Averaging. *CoRR* abs/1602.05629 (2016). arXiv:1602.05629

[12] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee. 2016. Generative Adversarial Text to Image Synthesis. (2016). arXiv:1605.05396

[13] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. 2016. Improved Techniques for Training GANs. (2016). arXiv:cs.LG/1606.03498

[14] S. Teerapittayanon, B. McDanel, and H. T. Kung. 2017. Distributed Deep Neural Networks Over the Cloud, the Edge and End Devices. In *ICDCS*.