A PROTOTYPE TOWARDS MODELING VISUAL DATA USING DECENTRALIZED GENERATIVE ADVERSARIAL NETWORKS

Dimitrios Kosmopoulos

University of Patras
dkosmo@upatras.gr

ABSTRACT

Decentralized computation is crucial for training on large data sets, which are stored in various locations. This paper proposes a method for collaboratively training generative adversarial networks (GANs) on several nodes to approach this problem. Each node has access to its local private dataset and model, but can influence and can be influenced by neighboring nodes. Such an approach avoids the need of a fusion center, offers better network load balance, higher robustness to network errors and improves data privacy. This work proposes an initial approach for decentralized GAN network optimization which is based on discriminator exchange among neighbors using an information gain criterion. We present our initial experiments to verify whether such a decentralized architecture can provide useful results.

Index Terms— decentralized learning, adversarial networks

1. INTRODUCTION

Distributed computation is crucial for training on large data sets, which are stored in various locations. Many state-of-the-art learning algorithms are rather simple, but base their accuracy on the quantity of the data they use to train and nowadays it is actually the amount of training data that makes a model more effective than the sophistication of the model itself [1]. Most often such data are acquired locally in a decentralized way by sensors, or personal devices.

Sharing local information with a central processor can be inefficient or even unfeasible, due to the large size of the network and data volume, time-varying network topology, bandwidth and energy constraints, or privacy issues. Performing the optimization in a centralized fashion may raise robustness concerns as well, since the central processor represents an single point of failure. This is very often the case for visual data (images and videos), which are produced in large volumes, stored locally and are difficult to transfer due to their size. The solution could be to have models that would be trained locally and then communicate with each other to learn better representations.

In this paper we propose a decentralized method for training generative networks. For the generative model for each node we use the generative adversarial network (GAN) [2]. In each node two models are trained simultaneously: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than the generative model. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to 0.5 everywhere.

The GANs have several benefits, such as, their ability to generate samples faster than fully visible belief nets (e.g. PixelRNN [3], or WaveNet [4]), the lack of any Monte Carlo approximations to train and generation using a single pass unlike the Boltzmann machines [5], the lack of deterministic bias and easier use of latent variables compared to variational autoencoders [6], the relaxation of the invertibility condition compared to nonlinear ICA (e.g. [7]).

The GANs have become popular and have been applied to several problems, such as, shape inpainting [8], image translation [9], object detection [10], scene generation [11] and many others.

Our contribution is a framework to employ adversarial networks in a decentralized setting, where each network is trained locally, but communicates with its neighbors to enhance itself. The following assumptions hold: i) the network is modeled as a symmetric and sparse graph; ii) agents know their local data only, which is adequate for training, iii) only inter-node communications between single-hop neighbors are possible, and iv) the node-specific GANs have the same structure. The goal here is to present a prototype and to verify experimentally whether the proposed method leads the overall network to stable and meaningful states. We verify it for the MNIST dataset. We are not aware of any other similar work in the literature doing decentralized learning using GANs.

The rest of the paper is structured as follows: in section 2 we presented the prior work related to this paper. Section 3 presents the proposed methodology, which is followed by the experimental results in section 4. Finally, section 5 concludes this paper.

2. RELATED WORK

The value of doing optimization in a decentralized fashion has been recognized for several decades (e.g., [12]). Decentralized optimization is very attractive in applications where data are collected by agents, and the communication to a fusion center is of high overhead or may compromise privacy. The recent research interest in big data processing also motivates the introduction of decentralized optimization to machine learning. Most works on this topic is devoted to optimization of a sum of convex functions, where the assumption on subgradient existence is essentially used.

A particular formulation of a distributed optimization problem refers to the case where the optimization cost is expressed as a sum of functions and each function in the sum corresponds to an agent. In this formulation the agents interact subject to a communication network, usually modeled as a directed/undirected graph. A solution to this problem was proposed in [13]. However, GANs are far from being convex. More recent solutions which relax the convexity assumption have been proposed as well, but typically make additional assumptions, e.g., [14] assumes equality constraints, [15] assumes broadcast communication etc. However, it is not clear how these constraints can be applied to GANs. An additional problem is that for each local network different parts of the captured structure are reflected by different filters in a random fashion, and therefore it is hard to find the correspondences among the gradient coefficients of different nodes. Therefore the propagated gradients as implied by the above methods would only translate to noise for the receiving nodes.

Related work on GANs operating in a decentralized fashion practically does not exist. There are however a few works that do essentially parallel processing assuming synchronous communication among different nodes. The main idea is to exchange discriminators/generators among the nodes during training. Ghosh et al [16] use multiple agents each of them running a separate generative model. All generative models are trained in pair with the same single discriminative model. Furthermore, the first layers of all the generative models are tied. The objective function used, enforces diversity between different models. The approach is very interesting, however for decentralized settings these assumptions are too tight. In contrast, we don't assume any tied weights. The assumption of a single discriminative model is relaxed in our work, because it is against having a decentralized learning method.

In Liu and Tuzel [17] use a pair of generative adversarial nets, each responsible for generating images in a separate domain (e.g., color and depth images). The authors show that by enforcing a simple weight-sharing constraint, they learn to generate pairs of corresponding images without existence of any pairs of corresponding images in the two domains in the training set. The weights are shared between first layers of the generative and the last layers of the discriminative networks. Again, such a setting is too restrictive for decentralized learning and the problem of calculating the joint distribution of multidomain images is different from our problem which uses a single domain.

Im et al [18] propose a framework, which is the most similar to ours. They train many GANs or their variants simultaneously, exchanging their discriminators randomly. This extends the two-player game into a multiplayer game. They claim that this eliminates the tight coupling between a generator and discriminator, leading to improved convergence and improved coverage of modes. They also propose an improved variant of the recently proposed Generative Adversarial Metric and show how it can score individual GANs or their collections under the GAP model. The approach is interesting for parallel processing, however the requirement of having available all the discriminators along the whole network does not promote a decentralized approach and introduces a big communication overhead. Our work differs from theirs in the following points: (a) we do not assume a fully connected graph which is not applicable in decentralized settings and (b) we exchange discriminators randomly as well, but we propose an information gain criterion and avoid using the uniform random distribution for selecting the discriminator.

3. METHODOLOGY

3.1. Generative Adversarial Networks

In the typical Generative Adversarial Networks (GAN) setting, the goal is to learn the generators distribution p_g over data $\mathbf{x}$ [2]. To this end we define a prior on input noise variables $p_z(\mathbf{z})$, then represent a mapping to data space as $G(\mathbf{z}; \theta_g)$, where G is a differentiable nonlinear function with parameters θ_g. We also define a second differential nonlinear function $D(\mathbf{x}; \theta_d)$ that outputs a single scalar. $D(\mathbf{x})$ represents the probability that $\mathbf{x}$ came from the data rather than p_g. We train D to maximize the probability of assigning the correct label to both training examples and samples from G. We simultaneously train G to minimize $log(1 - D(G(\mathbf{z})))$. In other words, D and G play the following two-player minimax game with value function $V(G; D)$:

$$\min_{\theta_G} \max_{\theta_D} V(G; D) \tag{1}$$

$$V = E_{x \sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + E_{z \sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))] \tag{2}$$

The equilibrium is reached by updating the D ascending the stochastic gradient (Eq.(3)) and then updating the G by descending its stochastic graqdient (Eq.(4)).

$$\nabla\theta_d \frac{1}{m} \sum_{i=1}^{m} [\log D(\mathbf{x}^i) + \log(1 - D(G(\mathbf{z}^i)))] \tag{3}$$

$$\nabla\theta_g \frac{1}{m}\sum_{i=1}^{m}\log(1-D(G(\mathbf{z}^i))) \quad (4)$$

The proof of convergence is an open issue, however by following some architectural guidelines it is possible to reach meaningful solutions.

3.2. Decentralized GAN

We propose a framework to employ adversarial networks in a decentralized setting, where each network is trained locally, but communicates with its neighbors to enhance itself. We assume a sparse connectivity, each node is connected with a small percentage of neighbors.

In our scenario we seek to minimize the function:

$$\min_{\theta_g,\theta_d}\sum_{k=1}^{K} V_i(\theta_g^k,\theta_d^k) \quad (5)$$

where K is the total number of nodes. It is the sum of the functions defined locally in the typical GAN problem (see Eq.(2)), one for each node.

The overall process for a single node is given by Algorithm 1. Most of the optimization refers to the standard GAN learning algorithm (lines 3-5, 7-8). We used an application-specific threshold for the loss function of D, which gave better results and which is typically reduced with the number of epochs. Then the decision to consider one of the neighbors or not, is taken by sampling a binomial pdf (line 9). In case that we decide to consider a neighbor, the exact one is chosen by sampling a discrete pdf. To this end we define the Kullback-Leibler divergence between the discriminator D_k of the current node and the discriminator D_j of the neighbor j given the noisy samples $\mathbf{z}^i$,...,$\mathbf{z}^m$:

$$D_{KL}(D_j||D_k)=\sum_{i=1}^{m} D_j(G_k(\mathbf{z}^i))\log\frac{D_j(G_k(\mathbf{z}^i))}{D_k(G_k(\mathbf{z}^i))} \quad (6)$$

Then the discriminator to train the generator against is selected by sampling a discrete probability distribution $p(D)$, which is given for each discriminator D_j by:

$$p(D_j)=\frac{D_{KL}(D_j||D_k)}{\sum_{j'\in N(k)} D_{KL}(D_{j'}||D_k)} \quad (7)$$

This selection policy is equivalent to stochastically selecting to be influenced the node that gives the maximum information gain. In other words the generator is stochastically chosen to be trained against the neighbor j whose discriminator D_j gives the maximum deviation from the discriminator D_k of the current node. This is expected to promote uniform training throughout the network.

Algorithm 1 Training algorithm for one network node

1: **for** $i = 1$ to $TrainingCycles$ **do**
2: **while** $Loss > LossThres(i)$ **do**
3: Sample batch of m noise samples $\mathbf{z}^1,...,\mathbf{z}^m$ from noise prior $p_g(\mathbf{z})$
4: Sample batch of m examples $\mathbf{x}^1,...,\mathbf{x}^m$ from noise prior $p_{data}(\mathbf{x})$
5: Update the discriminator by ascending its stochastic gradient according to Eq.(3)
6: **end while**
7: Sample minibatch of m noise samples $\mathbf{z}^1,...,\mathbf{z}^m$ from noise prior $p_g(\mathbf{z})$
8: Update θ_g by descending its stochastic gradient according to Eq.(4)
9: **if** $Sample(ConsiderNeighbors)$ == true **then**
10: Read θ_d^j from neighboring nodes
11: Select the neighbor j_0 to use, by sampling the discrete pdf defined by Eq.(7)
12: Update θ_g by descending its stochastic gradient according to Eq.(4) using $D_{j_0}(G(\mathbf{z}^i))$
13: **end if**
14: **end for**

4. EXPERIMENTS

In this section we report the experiments we did so far to verify the validity of the proposed method. To this end we have used for our experiments the well-known dataset MNIST [19] which is extensively used for prototype evaluation. The MNIST database of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

We have set up a network of five collaborating nodes as in Fig. 1. In each of them a GAN operates using an approach similar to that of [20], however the GANs do not have to follow the same architecture in each node. Each node can see only 10,000 samples and these samples are unique for each node. The images have been normalized to [-1,1] and tanh has been used for the output layer of the generator. The noise sampling was based on the Gaussian distribution with zero mean and σ=1. We used batches of only real or only noise data and we didn't mix them. We added noise of zero mean and of standard deviation of 0.2 for the labels of real and noisy data, which is expected to give better results [21].

For training the D we used stochastic gradient descent, while for training the G we used the Adam optimizer [22]. The threshold in the while-loop in Algorithm 1 was set to decrease according to $1/n$ from an initial value, where n is the epoch number. The probability of considering a neighbors' D was set to 0.5. A series of three fractionally-strided convolutions convert the 100-dimensional $\mathbf{z}$ input into a 20x20 pixel image by G. We followed the architectural guidelines of [20]

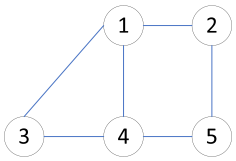

Fig. 1. Experimental network graph demonstrating the neighborhood relations among node-specific GANs.

without fully-connected or pooling layers and with ReLU activation in generator for all layers except for the output, which used tanh. For D we used LeakyReLU activation in the discriminator for all layers and batchnorm in both the generator and the discriminator. Fig. 2 gives the GAN architecture. In Fig. 3 we present qualitative results of the generator for each of the five nodes.

Furthermore, for quantitative evaluation, we have used the trained discriminators to train a multiclass SVM, which we then used to classify the test images. We introduced each image as input to the discriminators and used the flattened output in each layer as (8049-dimensional) features vectors. The same process was followed for each of the network nodes. Exactly the same process was used for the second experiment with the difference that the discriminator was selected based on the information gain criterion. The results are presented in Table 1. The exchange of discriminators leads to better solutions than standalone nodes. The use of the information gain criterion leads to more homogeneous results among nodes and to even better representation. The results are behind the state of the art of 0.23% error rate [23]. The relatively high error rate may be justified by the fact (a) that we did not do exhaustive cross validation to find the GAN structure or the SVM that give the best results, (b) the rather low number of epochs we run the GAN - less than 80, and (c) by the relatively low amount of training data for each node. However, in the first place our goal was not to compete against the state of the art classification methods for MNIST, but (a) to present a method that reaches an equilibrium that provides reasonable results and (b) to demonstrate the value of selecting D based on information gain compared to uniform sampling. Both of these goals have been demonstrated. In both settings (with and without information gain selection) all the parameters were identical, so that the methods were comparable.

node number	stand alone	uniform selection	inform.-gain selection
1	3.15	2.71	2.22
2	3.35	3.12	2.41
3	3.45	3.14	2.73
4	3.22	2.72	2.54
5	3.55	3.12	2.60

Table 1. Error rates (%) for each node for the MNIST dataset when using the features generated by the discriminator of each separate node for feeding an SVM. Three separate cases are displayed: nodes working alone, nodes exchanging discriminators using uniform sampling, and the proposed selection based on the information gain.

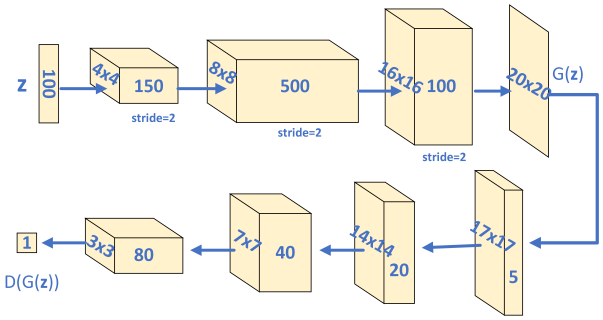

Fig. 2. Architecture of a GAN node in our MNIST experiments.

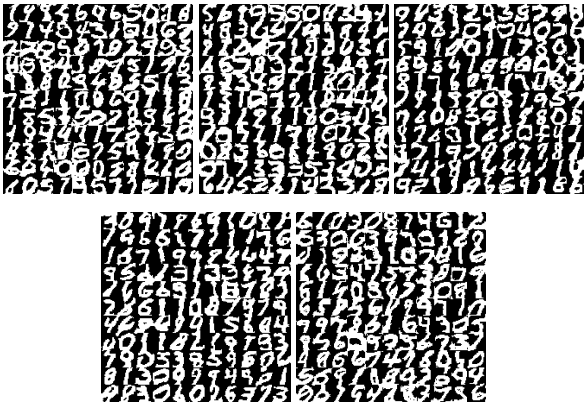

Fig. 3. Sample results of the generators on the five nodes after training

5. CONCLUSIONS

We have presented our first steps towards learning generative models using a decentralized GAN network. We have described the method and presented some initial results on the MNIST dataset. This is the first such work that we are aware of. The benefits of implementing such architectures are obvious for the image processing community. The validity of the method has been demonstrated experimentally and the results are promising. At this stage the size of the network is limited by the rather small size of the dataset, as well as the effects of the communication delays. We plan to extend evaluation on much larger networks using much bigger datasets and we will analyze convergence for network nodes based on connectivity and private dataset size. Furthermore, the proposed network is vulnerable to malfunctioning or malicious nodes and therefore further investigation is required on robustness issues.

6. REFERENCES

[1] A. Halevy, P. Norvig, and F. Pereira, "The unreasonable effectiveness of data," *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, March 2009.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014.

[3] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu, "Pixel recurrent neural networks," *CoRR*, vol. abs/1601.06759, 2016.

[4] Aron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio.," in *SSW*. 2016, p. 125, ISCA.

[5] Geoffrey E. Hinton, *A Practical Guide to Training Restricted Boltzmann Machines*, pp. 599–619, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[6] Carl Doersch, "Tutorial on variational autoencoders," 2016, cite arxiv:1606.05908.

[7] Laurent Dinh, David Krueger, and Yoshua Bengio, "NICE: non-linear independent components estimation," *CoRR*, vol. abs/1410.8516, 2014.

[8] Weiyue Wang, Qiangui Huang, Suya You, Chao Yang, and Ulrich Neumann, "Shape inpainting using 3d generative adversarial network and recurrent convolutional networks," *ICCV*, pp. 2317–2325, 2017.

[9] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros, "Image-to-image translation with conditional adversarial networks," in *CVPR*, July 2017.

[10] Jianan Li, Xiaodan Liang, Yunchao Wei, Tingfa Xu, Jiashi Feng, and Shuicheng Yan, "Perceptual generative adversarial networks for small object detection," in *CVPR*, July 2017.

[11] VSR Veeravasarapu, Constantin Rothkopf, and Ramesh Visvanathan, "Adversarially tuned scene generation," in *CVPR*, July 2017.

[12] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *Automatic Control, IEEE Transactions on*, vol. 31, no. 9, pp. 803–812, Sept. 1986.

[13] A. Nedic and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Transactions on Automatic Control*, vol. 54, no. 1, pp. 48–61, Jan 2009.

[14] Ion Matei and John Baras, "A non-heuristic distributed algorithm for nonconvex constrained optimization," Technical report, The Institute for systems research, University of Maryland, 2013.

[15] Ying Sun, Gesualdo Scutari, and Daniel P. Palomar, "Distributed nonconvex multiagent optimization over time-varying networks," *CoRR*, vol. abs/1607.00249, 2016.

[16] Arnab Ghosh, Viveka Kulharia, Vinay P. Namboodiri, Philip H. S. Torr, and Puneet Kumar Dokania, "Multi-agent diverse generative adversarial networks," *CoRR*, vol. abs/1704.02906, 2017.

[17] Ming-Yu Liu and Oncel Tuzel, "Coupled generative adversarial networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds., pp. 469–477. Curran Associates, Inc., 2016.

[18] Daniel Jiwoong Im, He Ma, Chris Dongjoo Kim, and Graham W. Taylor, "Generative adversarial parallelization," *CoRR*, vol. abs/1612.04021, 2016.

[19] Yann LeCun and Corinna Cortes, "MNIST handwritten digit database," 2010.

[20] Alec Radford, Luke Metz, and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *CoRR*, vol. abs/1511.06434, 2015.

[21] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, "Improved techniques for training gans," *CoRR*, vol. abs/1606.03498, 2016.

[22] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[23] Dan Cirean, Ueli Meier, and Jurgen Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, USA, 2012, CVPR '12, pp. 3642–3649, IEEE Computer Society.