Federated split GANs for collaborative training with heterogeneous devices[Formula presented]

Liang, Yilei

2022-11

Liang , Y , Kortoçi , P , Zhou , P , Lee , L H , Mehrabi , A , Hui , P , Tarkoma , S & Crowcroft , J 2022 , ' Federated split GANs for collaborative training with heterogeneous devices[Formula presented] ' , Software impacts , vol. 14 , 100436 . https://doi.org/10.1016/j.simpa.2022.100436

http://hdl.handle.net/10138/355746
https://doi.org/10.1016/j.simpa.2022.100436

Software Impacts 14 (2022) 100436

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

Federated split GANs for collaborative training with heterogeneous devices

Yilei Liang [a], Pranvera Kortoçi [b,*], Pengyuan Zhou [c], Lik-Hang Lee [d], Abbas Mehrabi [e], Pan Hui [f], Sasu Tarkoma [b], Jon Crowcroft [a]

[a] University of Cambridge, Cambridge, UK
[b] University of Helsinki, Helsinki, Finland
[c] USTC, Hefei, China
[d] KAIST, Daejeon, South Korea
[e] Northumbria University, Newcastle, UK
[f] HKUST, Hong Kong Special Administrative Region of China

ARTICLE INFO

Keywords:
Federated learning
Split learning
GAN
Hardware heterogeneous
Privacy preservation

ABSTRACT

Applications based on machine learning (ML) are greatly facilitated by mobile devices and their enormous volume and variety of data. To better safeguard the privacy of user data, traditional ML techniques have transitioned toward new paradigms like federated learning (FL) and split learning (SL). However, existing frameworks have overlooked device heterogeneity, greatly hindering their applicability in practice. In order to address such limitations, we developed a framework based on both FL and SL to share the training load of the discriminative part of a GAN to different client devices. We make our framework available as open-source software[1].

Code metadata

Current code version	v1
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2022-208
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/5643805/tree/v1
Legal Code License	*GNU General Public License v3.0*
Code versioning system used	git
Software code languages, tools, and services used	TensorFlow, Numpy
Compilation requirements, operating environments & dependencies	Jupyter Notebook
If available Link to developer documentation/manual	
Support email for questions	yl841@cst.cam.ac.uk

1. Introduction

GANs [1] are a subset of machine learning models that are utilized in semi-supervised and unsupervised learning. GANs are applied successfully in artificial intelligence (AI) applications such as medicine, computer vision, vocal AI, autonomous driving, and natural language processing [2–5]. Both the *generator* G and the *discriminator* D of a GAN are deep neural networks; specifically, D is a classifier designed to distinguish between genuine data stored locally (on the device) and bogus data generated by G. As a result, it is highly desired and, in certain circumstances, required to train D on the devices that generate and store the genuine (true) data. However, the varying processing capabilities of these devices may impede the on-site training of GAN

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: yl841@cst.cam.ac.uk (Y. Liang), pranvera.kortoci@helsinki.fi (P. Kortoçi), pyzhou@ustc.edu.cn (P. Zhou), likhang.lee@kaist.ac.kr (L.-H. Lee), abbas.mehrabidavoodabadi@northumbria.ac.uk (A. Mehrabi), panhui@cse.ust.hk (P. Hui), sasu.tarkoma@helsinki.fi (S. Tarkoma), jon.crowcroft@cl.cam.ac.uk (J. Crowcroft).

[1] https://github.com/YukariSonz/FSL-GAN

https://doi.org/10.1016/j.simpa.2022.100436
Received 27 September 2022; Received in revised form 16 October 2022; Accepted 25 October 2022

models. Enabling distributed learning of GAN models, particularly for XR applications, necessitates overcoming two significant obstacles:

(i) Limited resources (heterogeneity) on the devices where the model is trained.
(ii) Data privacy, e.g., medical applications are based on extremely sensitive patient data, therefore any machine learning model that operates on such data must maintain its confidentiality.

There are some recent works using FL and SL to ensure data and model privacy while taking client capabilities into account. FL enables data privacy by requiring each client to train a machine learning model privately, with no data sharing between participating clients. However, the limited resources of participating clients severely limit the applicability of FL [6], as clients may not be able to train a whole model. SL, on the other hand, solves the restrictions of FL by dividing the network architecture between participating clients and, perhaps, a coordinating entity such as a server. [7] presents a privacy-sensitive parallel SL approach in which each client trains a portion of a model with a preset size of mini batch that corresponds to the size of its (local) data collection. Similarly, [8] presents a strategy that combines FL and SL to train recurrent neural networks on distributed sequential data.

Our work makes improvements based on existing works by optimally splitting the neural network into smaller pieces to be trained by individual devices, in accordance to their on-board compute capabilities. As such, our method insures that a model is trained within a certain time budget by partitioning a model in accordance with the computational budget of single devices. Next, we describe the details of the software.

2. Description

Our software FSL-GAN [9] provided a simulation environment for federated GAN networks with split learning. In our software, we simulate the performance of device i through the $processing\_time\_factor_i$ variable, and model its memory consumption through the $capacity_i$ variable. We understand that this model is naïve and requires improvement, but users can easily plug in their selection algorithm for their experiment.

We deploy the DCGAN [10] model in our software; however, the user can deploy any model they like. To customize the model on FSL-GAN, the user needs to define each split-able part of the model and the model complexity (to simulate the client selection algorithm). The user also needs to notice that in our software one client means one discriminator, and hence each discriminator can also access a portion of the data (a simulation of real federated learning use case).

Listing 1: Split DCGAN

```
# Num_of_params = 1664
def make_discriminator_part_A():
  model = tf.keras.Sequential()
  model.add(layers.Conv2D(64, (5, 5),
                strides=(2, 2), padding='same',
                input_shape=[28, 28, 1]))
  model.add(layers.LeakyReLU())
  model.add(layers.Dropout(0.3))
  return model

# Num_of_params = 204,928
def make_discriminator_part_B():
  model = tf.keras.Sequential()
  model.add(layers.Conv2D(128, (5, 5),
                strides=(2, 2), padding='same'))
  model.add(layers.LeakyReLU())
  model.add(layers.Dropout(0.3))
  return model
# Num_of_params = 823,553
def make_discriminator_part_C():
  model = tf.keras.Sequential()
  model.add(layers.Conv2D(256, (5, 5),
                strides=(2, 2), padding='same'))
  model.add(layers.LeakyReLU())
  model.add(layers.Dropout(0.3))
  model.add(layers.Flatten())
  model.add(layers.Dense(1))
  return model
```

3. Software impact

Our FSL-GAN methodology consists of cooperatively training the discriminator model of a GAN across multiple devices of a single client, fully leveraging their heterogeneous resources. Its heuristic device selection strategy takes into account their capabilities to achieve better system performance. Increasing the number of discriminators in the system can further reduce generator loss and training time.

Our software provides a solution for running GANs on mobile devices, enabling such devices to jointly train a complex GAN for use cases such as GAN-based Render [11] and Scene Synthesis [12]. Enterprises could serve the pre-trained model to the user for these cases. These models can be trained on devices with low computational capabilities, such as mobile phones, while preserving data privacy on the client side.

4. Limitations and future development

We plan to investigate the following aspects:

1. minimizing the communication overhead between devices,
2. considering the trade-off between memory size and computational performance (e.g., the bandwidth and the number of cores on GPU/CPU),
3. removing the slowest discriminator in the system, and
4. investigating the effect of data heterogeneity on model convergence.

CRediT authorship contribution statement

Yilei Liang: Software implementation, Data curation, Draft preparation. **Pranvera Kortoçi:** Supervision, Split model, Draft preparation, Editing. **Pengyuan Zhou:** Supervision, Draft preparation. **Lik-Hang Lee:** Supervision. **Abbas Mehrabi:** Split model, Supervision. **Pan Hui:** Supervision. **Sasu Tarkoma:** Supervision. **Jon Crowcroft:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial networks, Commun. ACM 63 (11) (2020) 139–144.
[2] K. Lin, D. Li, X. He, Z. Zhang, M.-T. Sun, Adversarial ranking for language generation, Adv. Neural Inf. Process. Syst. 30 (2017).
[3] C.-C. Hsu, H.-T. Hwang, Y.-C. Wu, Y. Tsao, H.-M. Wang, Voice conversion from unaligned corpora using variational autoencoding wasserstein generative adversarial networks, 2017, arXiv preprint arXiv:1704.00849.
[4] O. Mogren, C-RNN-GAN: Continuous recurrent neural networks with adversarial training, 2016, arXiv preprint arXiv:1611.09904.
[5] A. Jabbar, X. Li, B. Omar, A survey on generative adversarial networks: Variants, applications, and training, ACM Comput. Surv. 54 (8) (2021) 1–49.

[6] P. Zhou, H. Xu, L. Lee, P. Fang, P. Hui, Are You Left Out? An Efficient and Fair Federated Learning for Personalized Profiles on Wearable Devices of Inferior Networking Conditions, Proc. ACM Interact. Mobile Wearable Ubiquitous Technol. 6 (3) (2022).

[7] J. Jeon, J. Kim, Privacy-sensitive parallel split learning, in: 2020 International Conference on Information Networking, ICOIN, IEEE, 2020, pp. 7–9.

[8] A. Abedi, S.S. Khan, FedSL: Federated Split Learning on Distributed Sequential Data in Recurrent Neural Networks, 2020, arXiv preprint arXiv:2011.03180.

[9] P. Kortoçi, Y. Liang, P. Zhou, L.-H. Lee, A. Mehrabi, P. Hui, S. Tarkoma, J. Crowcroft, Federated Split GANs, 2022, arXiv preprint arXiv:2207.01750.

[10] A. Radford, L. Metz, S. Chintala, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, in: Y. Bengio, Y. LeCun (Eds.), 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings, 2016.

[11] Y. Zhang, W. Chen, H. Ling, J. Gao, Y. Zhang, A. Torralba, S. Fidler, Image GANs meet Differentiable Rendering for Inverse Graphics and Interpretable 3D Neural Rendering, 2020, CoRR, abs/2010.09125.

[12] E. Chan, M. Monteiro, P. Kellnhofer, J. Wu, G. Wetzstein, pi-GAN: Periodic implicit generative adversarial networks for 3D-aware image synthesis, in: ArXiv, 2020.