Poster Abstract: a ChainSGD-reduce Approach to Mobile Deep Learning for Personal Mobile Sensing

Yu Zhang
RMIT University
zac.lhjzyzzoo@gmail.com

Tao Gu
RMIT University
tao.gu@rmit.edu.au

Xi Zhang
RMIT University
zaibuer@gmail.com

ABSTRACT

MDLdroid is a novel decentralized mobile deep learning framework, which enables resource-aware on-device collaborative learning for personal mobile sensing applications. To address resource limitation, MDLdroid uses a *chain-directed* Synchronous Stochastic Gradient Descent (ChainSGD-reduce) approach to effectively reduce overhead among multiple devices. In addition, MDLdroid includes an agent-based *multi-goal* reinforcement learning mechanism to balance resources in a fair and efficient manner. Real-world experiments demonstrate that our model training on off-the-shelf mobile devices achieves 2x to 3.5x faster than single-device training, and 1.5x faster than the master-slave approach.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Distributed computing methodologies**; *Reinforcement learning*; *Mobile agents*; • **Networks** → **Network resources allocation**; *Network protocol design.*

KEYWORDS

Mobile deep learning, Neural networks, Distribute computing, Resource allocation, Reinforcement learning

1 INTRODUCTION

Personal mobile sensing is fast permeating our daily lives to enable activity monitoring, healthcare and rehabilitation. Combined with *Deep Learning* (DL), these applications have achieved significant success in recent years.

Personal sensing data are significantly privacy-sensitive as the data contain a variety of human motion, biological contexts and identification information. Different from conventional cloud-based approaches, running deep learning on devices can be an ideal solution to effectively preserve sensing data privacy without being transmitted over the public network [6]. Besides, personal mobile sensing applications are mostly user-specific and highly affected by environment [4]. Continually training a local model with new data is a fundamental requirement. In practice, continually transmitting sensing data to the server and downloading model updates for training can incur fast battery drain and considerable latency for mobile devices especially when the network connection is unstable or broken. By contrast, continuous on-device training can enable quick local model inference and update response without exposing data [5]. Since data collection is costly in reality, Google's Federated Learning [2] offers not only complete data privacy but also better model robustness based on multiple user data. However, continuous local changes may seriously affect the performance of a global model generated by Federated Learning. In addition, deploying Federated Learning on a local server, e.g., edge server, may quickly reach the bottleneck due to resource constraint [3] and serious failure by attacks [1].

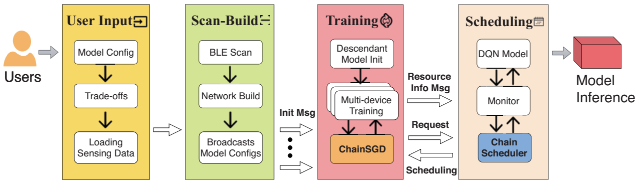


Figure 1: MDLdroid Architecture consists of three-stage processes: 1) user model configuration input; 2) network scan & build; 3) model training & task scheduling.

This paper proposes MDLdroid, a novel decentralized **M**obile **D**eep **L**earning framework to enable resource-aware on-device collaborative learning for personal mobile sensing applications. The key idea is to decentralize the Synchronous Stochastic Gradient Descent algorithm running on a single device to multiple devices with dynamic *chain-directed* model aggregation. MDLdroid targets to fully operate on multiple off-the-shelf An**droid** smartphones connected in a mesh network, and achieve high training accuracy and reliable execution of the state-of-the-art DL models.

MDLdroid defines two implication models. The *individual* model is used for an individual who has sufficient personal data and multiple mobile devices to offer shared resources. The data can only be safely distributed to the given mobile devices verified by the same identity (e.g., Google account). The *non-individual* model is applied for a group of people to explore specific local sensing features. The data will be strictly kept on device to preserve data privacy, and only the model gradient parameters of each individual will be exchanged to improve model robustness. Moreover, MDLdroid can be potentially used in many multi-user sensing scenarios, such as specific family behaviour recognition in a smart home.

2 SYSTEM OVERVIEW

Figure 1 demonstrates the architecture of MDLdroid. Since MDLdroid is designed to operate full-scale DL on Android based on a mesh network, we employ both Bluetooth Low Energy (BLE) and Bluetooth Socket (BS) to build the mesh network due to accessibility and low energy consumption. In principle, any on-device mesh-based protocol can be applied. In the **first** stage, users input different model configurations based on their demand to train models. MDLdroid uses two combined network topology in the **second** stage. The BS-based mesh topology is applied to perform the decentralized model aggregation between devices in the training stage, while the BLE-based tree topology is used for the centralized

DOI 10.1109/IPSN48710.2020.00-24

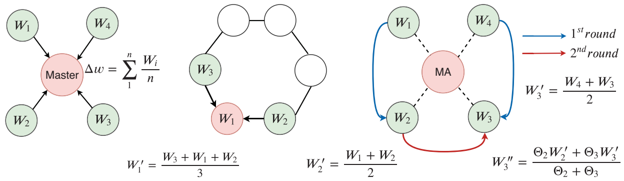

W_1
W_4
Master
$\Delta w = \sum_{1}^{n} \frac{W_i}{n}$
W_2
W_3
$W_1' = \frac{W_3 + W_1 + W_2}{3}$
W_3
W_1
W_2
$W_2' = \frac{W_1 + W_2}{2}$
W_1
W_4
MA
W_2
W_3
1^{st} round
2^{nd} round
$W_3' = \frac{W_4 + W_3}{2}$
$W_3'' = \frac{\Theta_2 W_2' + \Theta_3 W_3'}{\Theta_2 + \Theta_3}$

Figure 2: Model aggregation structure comparison: CentralSGD vs. NeighborSGD vs. ChainSGD

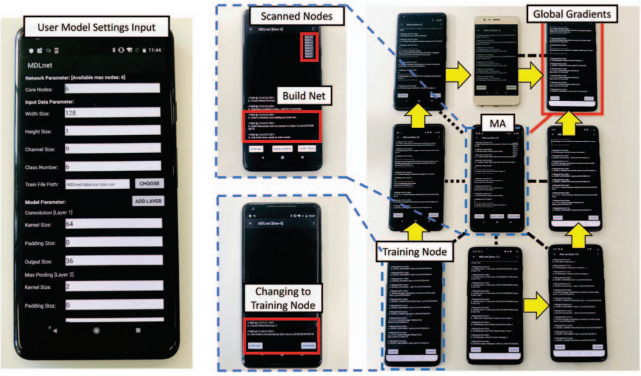
resource condition monitoring in the task scheduling stage. In the **third** stage, each device is required to continually report resource condition to a mobile agent (MA). Once all training tasks request the model aggregation for each iteration, the Chain-scheduler on the MA can manage the resource-efficiency scheduling paths as a *chain-directed* graph. When model aggregation of each iteration are completed, a copy of the aggregated model parameters will be resource-aware broadcasted to all devices for the next iteration. **Finally**, once training is completed, a global model will be distributed to all devices via broadcasting for model inference Especially, MDLdroid can also reload a pre-trained model to continually train with new local sensing data.

To reduce memory overhead and latency, we propose a ChainSGD-reduce approach with a mesh-based decentralized topology. In this approach, the number of neighbors is constantly managed as one for every model aggregation to achieve a *minimal-peak* in memory and communication overhead. Our approach also includes an agent-based reinforcement learning Chain-scheduler to schedule the neighbor aggregation task as a dynamic *chain-directed* graph in a resource efficiency way. Compared with centralized graph (CentralSGD) and decentralized neighbor graph (NeighborSGD), Figure 2 demonstrates that the major differences of ChainSGD-reduce are twofold: 1) the model aggregation is managed only with one of neighbors at a time; 2) the order of the aggregation tasks is dynamically scheduled depending on the real-time resource condition of device.

3 EVALUATION AND IMPLEMENTATION

We fully implement MDLdroid on off-the-shelf smartphones based on modified DL4J libraries. In particular, we essentially modify DL4J to enable the proposed ChainSGD-reduce approach on device. With minor model configurations, MDLdroid is fully compatible with a range of DL models without scaling down the model. Figure 3a plots a screenshot in which user customizes model configuration such as the parameters for certain datasets, customized hidden layer structures, and the required number of training devices. Figure 3b plots a screenshot during an execution of training on 9 smartphones using MDLdroid. The MA device scans all nearby devices, and build a BLE mesh network. The black dash lines represent the BLE connections between MA and training devices for resource condition monitoring. The yellow lines indicates a particular *chain-directed* model aggregation process via BS.

To evaluate MDLdroid, we use standard Convolutional Neural Network (CNN) models since CNN can be used to effectively process multi-channel sensing data. We evaluate the training performance using 6 public datasets, containing diverse personal mobile sensing data. Results show that MDLdroid achieves high training accuracy which is comparable to the state-of-the-art accuracy, speeds up training by 2x to 3.5x compared to the single-device baseline and 1.5x compared to Federated Learning. In addition, MDLdroid reduces latency overhead due to busy condition by 23% and 53% compared to *Tree-scheduler* in the Tree-Allreduce approach and *Ring-scheduler* in the Ring-Allreduce approach, respectively. Moreover, MDLdroid reduces the variance in battery consumption among devices by 40% compared to *Tree-scheduler*.



11:44

MDLnet

Network Parameter: [Available max nodes: 6]

Core Nodes: 6

Input Data Parameter:

Width Size: 128

Height Size: 1

Channel Size: 9

Class Number: 6

Train File Path: /MDLnet/data/nor_train.csv CHOOSE

Model Parameter: ADD LAYER

Convolution [Layer 1]

Kernel Size: 64

Padding Size: 0

Output Size: 36

Max Pooling [Layer 2]

Kernel Size: 2

Padding Size: 0

(a)

(b)

Figure 3: Experiment screenshot. (a) User model input; (b) Training execution screenshot

4 CONCLUSION AND FUTURE WORK

In this work, we present MDLdroid, a novel decentralized mobile DL framework to enable resource-aware on-device collaborative learning for personal mobile sensing applications without central server support, which achieves a reliable state-of-the-art model training accuracy, low resource overhead, low latency for model inference and update. we plan to embed the MDLdroid into mobile OS to offer automatic background training, and develop a wider variety of applications to fully explore the capability of MDLdroid in our future work.

ACKNOWLEDGMENTS

This work is supported by Australian Research Council (ARC) Discovery Project grants DP180103932 and DP190101888.

REFERENCES

[1] Arjun Nitin Bhagoji, Supriyo Chakraborty, Prateek Mittal, and Seraphin Calo. 2019. Analyzing Federated Learning through an Adversarial Lens. In *ICML'19*.

[2] Jakub Konecný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *CoRR* (2016).

[3] Xiangru L., Ce Z., Huan Z., Cho-Jui H., Wei Z., and Ji L. 2017. Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In *NIPS'17*.

[4] Francisco Laport-López, Emilio Serrano, Javier Bajo, and Andrew T. Campbell. 2019. A review of mobile sensing systems, applications, and opportunities. *KAIS'19* (2019).

[5] J. Wang, B. Cao, P. Yu, L. Sun, W. Bao, and X. Zhu. 2018. Deep Learning towards Mobile Applications. In *ICDCS'18*.

[6] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated Machine Learning: Concept and Applications. *TIST'19* (2019).