DeCaf: Iterative Collaborative Processing over the Edge

Dhruv Kumar [0], Aravind Alagiri Ramkumar [0], Rohit Sindhu [0], Abhishek Chandra

University of Minnesota, Twin Cities

Abstract

The increase in privacy concerns among the users has led to edge based analytics applications such as federated learning which trains machine learning models in an iterative and collaborative fashion on the edge devices without sending the raw private data to the central cloud. In this paper, we propose a system for enabling iterative collaborative processing (ICP) in resource constrained edge environments. We first identify the unique systems challenges posed by ICP, which are not addressed by the existing distributed machine learning frameworks such as the parameter server. We then propose the system components necessary for ICP to work well in highly distributed edge environments. Based on this, we propose a system design for enabling such applications over the edge. We show the benefits of our proposed system components with a preliminary evaluation.

1 Introduction

Current Trends: In the past few years, there has been a proliferation of edge devices such as mobile phones, laptops, IoT sensors [9]. To perform any kind of analytics over the data generated on these edge devices, organizations typically need to bring in all the data into a central data center. Once the data is brought in one place, the organizations can perform a variety of analytics such as traditional SQL-based analytics, machine learning based model training and inference, and graph analytics. The large amount of data transfer from the edge devices to the central cloud leads to high data transfer cost and is often unacceptable [32]. This issue has been addressed in a number of geo-distributed analytics (GDA) systems [19, 21, 24]. However, since the actual data is transferred from the edge devices to the data center, the privacy of the user is violated in these GDA systems. The users may not always want to share sensitive data. Sending anonymized data from the edge device to the data center ensures the user privacy but this may not work in all cases. Hence, there is a need for approaches which enable analytics without transferring the raw data out of the edge devices. In this paper, we present a system for iterative collaborative processing (ICP) in a resource constrained edge environment with a focus on machine learning applications. The edge devices collaborate to complete the model training in multiple iterations. Example applications include distributed machine learning [27], federated learning [30] and federated multi-task learning [35]. The techniques proposed in this paper can be extended to other applications such as distributed graph analytics [22].

Challenges in Resource Constrained Edge Environments: Most of the existing approaches consider intra-DC environment where the resources are typically sufficient, homogeneous and the variation in resource availability is low [7, 27]. Recent approaches [19, 21, 24] have considered network and compute heterogeneity in geo-distributed inter-DC environments. But in edge environments, the resource availability has much more dynamism and constraints which introduces new challenges. The edge devices are constrained in terms of storage (disk space), compute (CPU, RAM), network bandwidth (3G, 4G, LTE) and battery power. Additionally, the devices are highly unreliable due to various reasons. For instance, the resource utilization may suddenly change due to user intervention or the device may go offline due to loss of network connectivity. Moreover, the user privacy needs to be ensured in edge environments. These challenges require us to specifically design systems for resource constrained edge environments.

Research Contributions: In this paper, we identify the unique challenges in performing ICP over the edge as compared to the data center environment. We propose the system components necessary for handling the dynamism and heterogeneity in resource availability. Specifically, these components adaptively tune the amount of computation by proactively monitoring the system resources and by balancing the computation load among the edge devices to run tasks efficiently in a fault tolerant manner even in the case of failures. We also present a prototype implementation and show the benefits of our proposed techniques with a preliminary evaluation.

2 Related Work and Challenges

Distributed Machine Learning Systems: There are a number of distributed machine learning systems for data center environments [7, 13, 14, 26, 27]. Parameter server framework [27] is the most popular approach for large scale distributed machine learning in data centers. It allows a large number of

[0] Equal contributions

workers to jointly build a global model by updating the shared parameters in the parameter server. It is mainly designed for homogeneous environments where there isn't much compute and network heterogeneity. It provides fault tolerance on the server side by replicating the parameters across multiple machines and on the worker side by replacing a dropped out worker or a straggler with another worker. The replication strategy is not feasible in edge environments where the resources are constrained and replication would waste critical resources which can be utilized elsewhere. Replacing the dropped out device with another device is also not trivial in edge environments where the raw data exists only on its source device as compared to the data center environment where the data is persisted outside the worker. Hence, parameter server framework cannot work out-of-the-box in edge environments. To mitigate the above mentioned dropout and straggler issues, we propose task migration and load balancing techniques based on proactive monitoring of system resources.

Geo-Distributed Analytics Systems: Recently, there has been a growing interest in geo-distributed data analytics where the data remains at the geographically distributed data centers and the query is executed in a geo-distributed manner [21, 24, 32]. These systems consider compute and network heterogeneity but do not consider machine learning workloads and instead focus on map-reduce or SQL workloads. Gaia [19] is a recent geo-distributed machine learning system which addresses the network bandwidth scarcity and heterogeneity by aggregating insignificant updates of a parameter and communicating it to other data centers only when the aggregated updates are significant enough. It utilizes the parameter server framework which is not suitable for edge environments as discussed above. The solutions proposed for handling compute and network heterogeneity in these systems do not incorporate the drastic variation in resource availability over time as is the case in edge environments. In our work, we propose adaptive tuning of the amount of computations based on the current resource availability to handle the drastic variation in resource availability over time.

Federated Machine Learning: Federated learning [1] is an emerging machine learning paradigm which enables the edge devices to collaboratively learn a global shared prediction model while keeping all the training data on device, decoupling the ability to do machine learning from the need to store the data in the central data center. Example applications include next-word prediction and search query suggestions in smartphone keyboards [17, 37], human activity recognition using mobile phones [8] and predicting health events such as heart attack risk using wearable devices [31]. TensorFlow Federated [6] has been recently released for supporting such applications.

Federated multi-task learning [35] is another emerging paradigm which builds separate but related models for each of the edge devices instead of building a single global shared prediction model for all the devices. MOCHA [35] is a recently proposed federated multi-task learning framework which focuses on the algorithmic aspects of ensuring model convergence in a heterogeneous network by tolerating variable amount of computations on different devices. FedProx [33] is yet another federated learning framework for tackling the statistical heterogeneity in the data generated at the edge devices. It takes a simplistic approach of selecting a subset of available devices and uses the updates from this subset to build a global model. None of the above mentioned frameworks focus on when and how to enable the variable amount of computations on different devices. Additionally, they do not ensure model convergence in the cases when the device dropout rate is high. In our work, we propose system techniques for when and how to enable the variable amount of computations on edge devices and also address high drop out rate via task migration and load balancing strategies.

Google [3] has recently built a TensorFlow based system for Federated learning [11]. Their system makes many simplistic assumptions about the edge environment. Only those mobile devices which are idle, charging, and connected to WiFi, are selected for performing federated computations. The computations are aborted if any of these conditions change. Stragglers are simply ignored by the system. An iteration is considered complete if sufficient number of devices report back within the deadline. These assumptions fail to work in case of federated multi-task learning where each device has a separate model to train. In our work, we take an orthogonal approach to solve the dynamic resource availability, fault tolerance and straggler issues, and propose system techniques which also work for federated multi-task learning type of applications.

Machine Learning Inference at the Edge: Performing machine learning inference at the edge devices in collaboration with the edge and cloud servers is a very popular area of research [16, 20, 23, 25]. Machine learning inference is not iterative in general i.e. it does not require multiple rounds of communication between the server and the edge devices. Hence, the challenges in machine learning inference in resource constraint edge environments are different than those in iterative collaborative processing such as model training. As an example, the fault tolerance and device dropout issues discussed above are not relevant for inference. In our work, we focus on iterative collaborative processing over the edge.

3 Our Proposal

3.1 System Model

Our system considers a star topology similar to other systems mentioned in §2. As shown in Fig. 1, there is a central controller (can be an edge cloudlet or a cloud server) coordinating with all the user edge devices. These devices collaborate in two ways: via the central controller as well as directly in a peer-to-peer fashion. Note that the data is present at the user edge devices and it is not shared with the central controller to

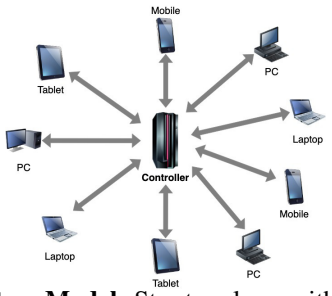

Figure 1: **System Model:** Star topology with a central controller coordinating with all the user edge devices.

ensure user privacy.

Iterative Collaborative Processing (ICP): ICP involves performing a set of computations in multiple iterations. Model training task in machine learning is one such example. The computations start with the central controller communicating the initial values of the model parameters to all the edge devices. Each device updates the model parameters using its local dataset and sends it to the central controller. The central controller then updates the model parameters using the individual model parameters from all the devices[1] and sends back the updated model parameters to the devices for next iteration. Further iterations happen until the model parameters has converged. Every iteration has a deadline within which all the devices must send back their updates. Any updates received after the deadline are simply ignored.

3.2 Proactive Monitoring and Prediction

As discussed in §2, the high variations in resource availability over time in edge environments, warrants the need for proactive monitoring of the system resources so that the system can quickly react to any changes in the resource availability. To this end, we propose resource monitors to be added as system components in edge-based systems for ICP. More specifically, we propose a compute monitor for monitoring the compute (CPU, GPU, FPGA) utilization, a battery monitor for monitoring the current battery level and battery drain rate, a network monitor for monitoring the uplink and downlink bandwidth and a storage monitor for monitoring the available persistent storage. We argue that the overhead of running all the monitors on the device is negligible since devices like laptops, mobile phones etc. already have them running as part of the background processes.

In addition to proactive monitoring, we also propose predicting the resource availability at any point in time by analyzing the resource utilization pattern of the edge devices. We argue that many of the edge devices will have resource utilization pattern based on the device's usage and device's location throughout the day. Prior work [36, 40] has shown that the application usage on mobile devices indeed exhibits temporal patterns. A possible approach is to find such patterns in the device's resource consumption by periodically logging the resource consumption statistics and then building a prediction model using the collected time series data with the help of existing learning methods [12, 18]. Additionally, there is some work in modeling and predicting the mobile devices application usage [28, 34, 38, 39]. These works consider various factors such as the user's application usage characteristics, the temporal context of application usage, point of location of device etc. for predicting the next application to be used by the user. Some of these techniques can be extended to incorporate the resource consumption statistics logs to predict the resource utilization in the near future. This will help the system react more intelligently to varying resource availability over time. We employ this proactive monitoring and prediction for adaptive tuning, task migration and load balancing as discussed next.

3.3 Adaptive Tuning

In order to make sure that every edge device finishes the model update within the deadline stipulated by the controller, we propose adaptively tuning the update computations based on the information received from the resource monitors. A single iteration[2] at the edge device for model update typically involves iterating through all the data points in the local dataset. Therefore, based on the resource availability, we can choose a subset of data points to iterate through, in the local dataset. The percentage of data points selected for performing the model update will affect the update quality[3]. Fig. 2 shows the impact of varying percentage of data points selected in every iteration on the update quality for an example application. Higher the percentage, higher the quality of the model. Therefore, the system must choose the maximum percentage of data points which can be processed within the stipulated deadline, considering the constraints on the available resources.

Resource-Computation Profile: Deciding the percentage of data points based on the resource availability requires building a resource-computation profile which can estimate the amount of computations that can be performed within the deadline based on the resource availability. This profile can be built by varying the system load by running various types of applications and identifying the amount of computations which the device was able to perform within a particular time period for various levels of system load. This only needs to be done once unless the system hardware, OS or training algorithm changes in which case the profile can be rebuilt.

As part of our initial experiments, we evaluated a simpler approach to schedule variable amounts of computations in order to meet the deadline. In this approach, during execution, our compute monitor measures the amount of time a device takes to process a small subset of the points and uses that to estimate the subset of points which can be processed within a deadline. This is done periodically to handle the scenarios where the system load changes during the execution. As shown in §5, the device is able to meet the deadline fairly

[1] Actual update rule depends on the specific algorithm or application.

[2] This is the iteration with respect to the controller.

[3] By quality, we refer to the rate of convergence of the model loss function.

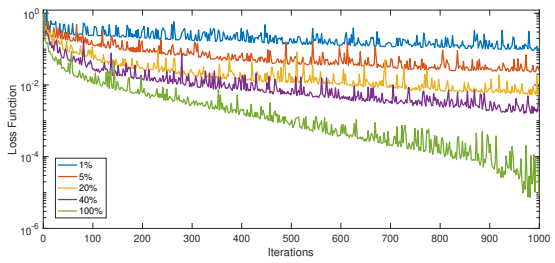

Figure 2: **Effect of the number of computations on the rate of convergence of the loss function:** Higher the number of points processed per iteration, faster the convergence.

well even with this simple approach.

Additionally, there are two time horizons which we need to consider while tuning: the amount of computations which can be performed in the next iteration and the amount of computations which can be performed in the entire training period. Both are highly correlated. Choosing a higher amount of computations for the next iteration may lead to high resource consumption (such as battery power). Hence, the amount of computations needs to be tuned such that the device's overall resource consumption is not too high until the end of the training period.

3.4 Fault Tolerance and Load Balancing

As discussed in §3.3, the amount of computations performed in one iteration affects the update quality. Therefore, the system should not reduce the amount of computations below a certain threshold. If the resource availability is too low, then the system will not be able to perform the minimum amount of computations. In such a case, we propose migrating the task to another edge device. But since the data also needs to be moved to another device, we introduce the concept of trusted device set so as to ensure the user privacy which is one of the primary reasons for edge-based ICP.

Trusted Device Set: Every device will have a set of devices whom it trusts and to whom it can migrate the task to, if required. We argue that this is a fair assumption since in today's time, each person owns multiple digital devices (mobile phone, laptop, tablet, desktop etc.) [5] and it is possible to have devices belonging to people in the same organization, family, social circle etc in the same trusted device set.

Whenever any device encounters that it is not possible for it to perform the minimum amount of computations, it will send a migration request to all the devices in its trusted device set. The prospective devices will reply back if they are willing to accept another task along with their most recent resource availability information. Then the constrained device can choose a device from the candidate devices and migrate the task with its most recent model parameters and its local dataset. The constrained device can choose a device with the minimum migration time and the minimum increase in load

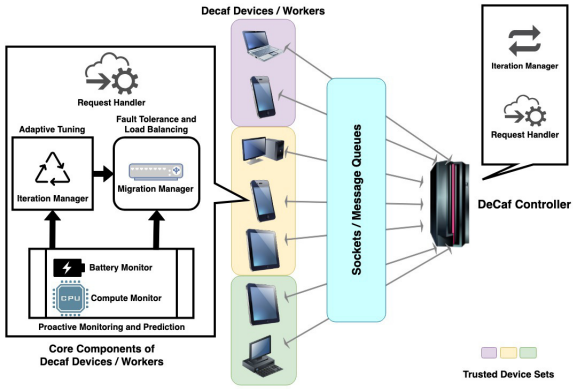

Figure 3: **DeCaf Architecture**

on the other device[4]. Losing few iterations during migration is acceptable as long as the number of missed iterations is not high. Also, the local dataset is going to be small in general and if the trusted device set is in the same local network, then the total migration time is going to be small.

In addition to task migration, we also propose load balancing which involves migrating a task from a device with high resource utilization to another device in the trusted device set with low resource utilization. This will ensure that no device is over-utilized and the overall quality of the trained model(s) is improved.

4 Implementation

We built a prototype called DeCaf which incorporates the proposed techniques in §3. Fig. 3 shows its various components. DeCaf has two main modules:

- **DeCaf Controller:** The controller module runs on the central server and coordinates with all the workers (running on edge devices). It has a request handler for handling all the requests from the worker and passing on the received parameters from the workers to the iteration manager. The iteration manager updates the model parameters at the end of every iteration. The updated parameters are sent to the workers via the request handler.
- **DeCaf Worker:** The worker module runs on the edge device. The compute monitor monitors the compute resources[5]. The iteration manager in the worker module fetches the updated parameters from the controller. It also tunes the amount of computations to the current available resources information fetched from the resource monitors. Migration manager implements the fault tolerance and load balancing strategies.

All the communication between the worker and the controller is initiated by the worker and happens over sockets. The architecture can be easily extended to use message queues like RabbitMQ [4] or Kafka [2] for the communication mechanism.

[4]This is just one policy. Other policies can also be considered.
[5]Implementing other monitors is a work-in-progress.

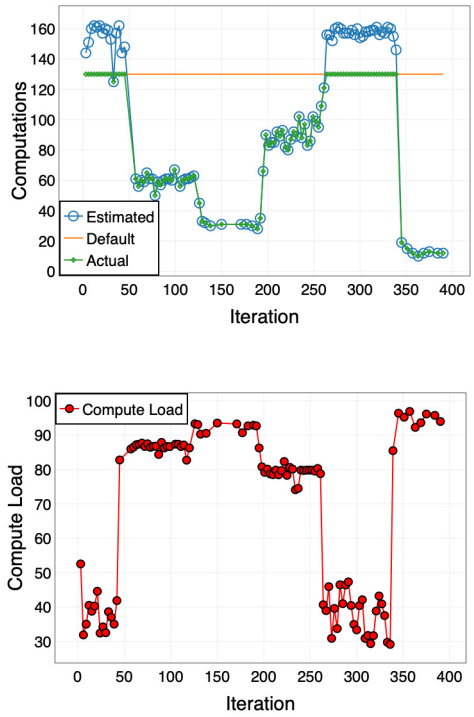

Figure 4: **Adaptive Tuning:** The system tunes the amount of computations performed per iteration based on the resource availability to meet the deadline.

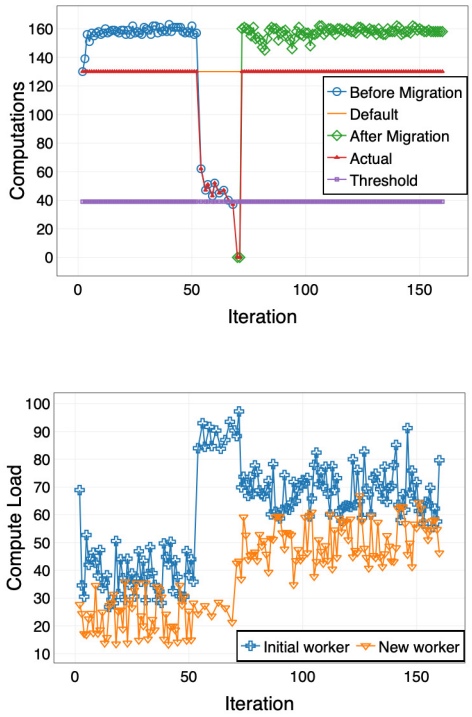

Figure 5: **Fault Tolerance:** If the compute resources becomes too constrained, the system migrates the task to another worker which has low compute load.

5 Evaluation

We show a preliminary evaluation of the proposed techniques in this section. We run DeCaf in an edge environment consisting of 2.9 GHz Intel Core i5 MacBook Pro as the worker nodes. The workers communicate with a remote central controller running on a Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz machine. We use MOCHA [35] for implementing Support Vector Machines as our test application.

Adaptive Tuning: We show how the iteration manager tunes the number of computations performed per iteration based on the current resource availability so as to meet the deadline set by the central controller. We disable the migration manager to independently evaluate the performance of adaptive tuning. We vary the compute load on the worker node and show the number of computations performed per iteration. In Fig. 4, we can see that the number of computations performed per iteration is high when the compute load is low. Similarly, the number of computations is low when the compute load is high. In all cases, the iteration manager chooses the minimum of the default number of computations and the estimated number of computations possible within the deadline. An important point to note here is that at higher compute load, the iteration manager has to tune the number of computations by a relatively larger amount as compared to the lower compute load scenario. This is because at higher compute load, the worker has relatively more number of processes competing for the limited compute resources.

Fault Tolerance: We show how the migration manager identifies the need to migrate the task from the highly resource constrained worker to a less utilized worker so as to ensure the minimum number of computations performed per iteration. In Fig. 5, we can see that when the number of computations per iteration goes below a user-defined threshold (30% of the default in this case), the migration manager identifies the need to migrate the task to another trusted worker. Using the mechanism explained in §3.4, the migration manager chooses a trusted worker and migrates the task to this trusted worker. After migration, the task resumes performing computations (in this case, equal to the default number of computations) on the trusted worker. Note that the task missed a couple of iterations during migration and hence, the number of computations for those iterations are shown as zero in the figure.

6 Conclusion

User privacy has been an important aspect while designing any software system in recent times. In this paper, we explored the challenges associated with systems for iterative collaborative processing (ICP) in a dynamic edge environment and presented system components to tackle them. The proposed components adaptively tune the amount of computations by proactively monitoring the system resources and by balancing the computation load among the edge devices to run tasks efficiently in a fault tolerant manner. We focused on machine learning applications but the proposed techniques can be extended to handle any ICP based application. We believe that a substantial amount of additional research work is required for implementing ICP in edge-based systems.

7 Discussion

Our work on DeCaf is a preliminary attempt on tackling the challenges in ICP in heterogeneous edge environment. There are several open and possibly undiscovered challenges. We discuss some of these in this section.

Network Heterogeneity: A thorough evaluation is necessary to analyze the impact of network variability as it involves various factors like latency, bandwidth and network availability especially in WAN environment. Additionally, we have to take into consideration the model size along with the network heterogeneity. For instance, the size of the parameters being exchanged can range from KBs to GBs and as a result, the time taken to transfer over the variable bandwidth link will impact the amount of computation that can be performed within the deadline.

System Topology: In addition to the star topology considered in this work, some of the recent works [10] also consider a fully decentralized peer-to-peer topology for federated learning where each node exchanges the parameters from its neighbors and transmits the updated weight for next iteration. Here, each node has a dual responsibility of being an edge node as well as a controller. As a result, each node has to potentially cater to different deadlines set by different neighboring nodes. Satisfying these different requirements concurrently is a challenge which needs to be further explored.

Other Applications for ICP: In this work, we focused primarily on federated machine learning. Distributed graph analytics is another application domain which fits in ICP. The existing work in distributed graph analytics is mostly aimed at single data center environment [15, 29] or geo-distributed data center environments [22]. The techniques proposed in this paper are general and not specific to machine learning but further research is required to identify the unique challenges posed by graph analytics in an edge environment.

Resource Prediction: The prior techniques [38, 39] on predicting the application usage in edge devices requires aggregating logs from all the edge devices to a central location and then building a prediction model using the collected data. This also raises privacy concerns which is the motivation for ICP in edge environment. Hence, these techniques need to be modified to ensure privacy before they can be used to predict the resource availability in edge devices.

Fault Tolerance: The central controller approach is not scalable in a geo-distributed environment since it might become a point of contention. Hence, enabling fault tolerance at the controller level is a separate challenge. One possibility is to consider an intermediate layer of secondary controllers with each secondary controller aggregating updates from a group of edge devices and then sending the aggregated update to the central controller. Similar design is discussed in Google's federated learning system [11].

For the edge devices, the proposed fault tolerance mechanism aims to identify scenarios where a device is not able to perform the minimum amount of computations required for maintaining model quality. It is also helpful in scenarios where the system is able to predict drop out events of a device such as very low battery, loss of network connectivity or surge in resource consumption. We note that this mechanism will not be able to handle scenarios where the device drops out unexpectedly due to reasons such as unexpected shutdown of devices or manual interference of a user etc. Further research is required to handle such scenarios.

Acknowledgements

We thank the anonymous reviewers and our shepherd Bernard Wong, for many constructive comments and suggestions that greatly improved the quality of this paper. This work was sponsored in part by NSF under Grants CNS-1717834, CNS-1619254 and III-1422802.

References

[1] https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[2] https://kafka.apache.org.

[3] https://www.google.com.

[4] https://www.rabbitmq.com.

[5] https://www.statista.com/statistics/678739/forecast-on-connected-devices-per-person/.

[6] https://www.tensorflow.org/federated.

[7] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[8] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *21st European Symposium on Artificial Neural Networks, ESANN 2013, Bruges, Belgium, April 24-26, 2013*, 2013.

[9] Suman Banerjee and Dapeng Oliver Wu. Final report from the NSF workshop on future directions in wireless networking. 2013.

[10] Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. Personalized and private peer-to-peer machine learning. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, pages 473–481, 2018.

[11] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. *CoRR*, abs/1902.01046, 2019.

[12] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Inc., San Francisco, CA, USA, 1990.

[13] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274, 2015.

[14] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1223–1231, USA, 2012. Curran Associates Inc.

[15] Joseph E. Gonzalez, Reynold S. Xin, Ankur Dave, Daniel Crankshaw, Michael J. Franklin, and Ion Stoica. Graphx: Graph processing in a distributed dataflow framework. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, pages 599–613, Berkeley, CA, USA, 2014. USENIX Association.

[16] Seungyeop Han, Haichen Shen, Matthai Philipose, Sharad Agarwal, Alec Wolman, and Arvind Krishnamurthy. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '16, pages 123–136, New York, NY, USA, 2016. ACM.

[17] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604, 2018.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[19] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R. Ganger, Phillip B. Gibbons, and Onur Mutlu. Gaia: Geo-distributed machine learning approaching lan speeds. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, NSDI'17, pages 629–647, Berkeley, CA, USA, 2017. USENIX Association.

[20] Chien-Chun Hung, Ganesh Ananthanarayanan, Peter Bodík, Leana Golubchik, Minlan Yu, Paramvir Bahl, and Matthai Philipose. Videoedge: Processing camera streams using hierarchical clusters. In *2018 IEEE/ACM Symposium on Edge Computing, SEC 2018, Seattle, WA, USA, October 25-27, 2018*, pages 115–131, 2018.

[21] Chien-Chun Hung, Ganesh Ananthanarayanan, Leana Golubchik, Minlan Yu, and Mingyang Zhang. Wide-area analytics with multiple resources. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys '18, pages 12:1–12:16, New York, NY, USA, 2018. ACM.

[22] Anand Padmanabha Iyer, Aurojit Panda, Mosharaf Chowdhury, Aditya Akella, Scott Shenker, and Ion Stoica. Monarch: Gaining command on geo-distributed graph analytics. In *Proceedings of the 10th USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'18, pages 5–5, Berkeley, CA, USA, 2018. USENIX Association.

[23] Hyuk-Jin Jeong, Hyeon-Jae Lee, Chang Hyun Shin, and Soo-Mook Moon. Ionn: Incremental offloading of neural network computations from mobile devices to edge servers. In *Proceedings of the ACM Symposium on Cloud Computing*, SoCC '18, pages 401–411, New York, NY, USA, 2018. ACM.

[24] Albert Jonathan, Abhishek Chandra, and Jon Weissman. Multi-query optimization in wide-area streaming analytics. In *Proceedings of the ACM Symposium on Cloud Computing*, SoCC '18, pages 412–425, New York, NY, USA, 2018. ACM.

[25] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '17, pages 615–629, New York, NY, USA, 2017. ACM.

[26] Tim Kraska, Ameet S. Talwalkar, John C. Duchi, Rean Griffith, Michael J. Franklin, and Michael I. Jordan. Mlbase: A distributed machine-learning system. In *CIDR*, 2013.

[27] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, pages 583–598, Berkeley, CA, USA, 2014. USENIX Association.

[28] Keun-Woo Lim, Stefano Secci, Lionel Tabourier, and Badis Tebbani. Characterizing and predicting mobile application usage. *Computer Communications*, 95:82–94, December 2016.

[29] Grzegorz Malewicz, Matthew H. Austern, Aart J.C Bik, James C. Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: A system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146, New York, NY, USA, 2010. ACM.

[30] H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. Federated learning of deep networks using model averaging. *CoRR*, abs/1602.05629, 2016.

[31] A. Pantelopoulos and N. G. Bourbakis. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 40(1):1–12, Jan 2010.

[32] Qifan Pu, Ganesh Ananthanarayanan, Peter Bodik, Srikanth Kandula, Aditya Akella, Paramvir Bahl, and Ion Stoica. Low latency geo-distributed data analytics. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15*, pages 421–434. ACM Press.

[33] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks. *CoRR*, abs/1812.06127, 2018.

[34] Choonsung Shin, Jin-Hyuk Hong, and Anind K. Dey. Understanding and prediction of mobile application usage for smart phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 173–182, New York, NY, USA, 2012. ACM.

[35] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. Federated multi-task learning. page 19, 2017.

[36] Qiang Xu, Jeffrey Erman, Alexandre Gerber, Zhuoqing Mao, Jeffrey Pang, and Shobha Venkataraman. Identifying diverse usage behaviors of smartphone apps. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 329–344, New York, NY, USA, 2011. ACM.

[37] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *CoRR*, abs/1812.02903, 2018.

[38] Donghan Yu, Yong Li, Fengli Xu, Pengyu Zhang, and Vassilis Kostakos. Smartphone app usage prediction using points of interest. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(4):174:1–174:21, January 2018.

[39] Sha Zhao, Zhiling Luo, Ziwen Jiang, Haiyan Wang, Feng Xu, Shijian Li, Jianwei Yin, and Gang Pan. Appusage2vec: Modeling smartphone app usage for prediction. In *International Conference on Data Engineering, ICDE 2019, 8-11 April 2019, Macau SAR, China*, 04 2019.

[40] Sha Zhao, Julian Ramos, Jianrong Tao, Ziwen Jiang, Shijian Li, Zhaohui Wu, Gang Pan, and Anind K. Dey. Discovering different kinds of smartphone users through their application usage behaviors. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '16, pages 498–509, New York, NY, USA, 2016. ACM.