Joint Task and Resource Allocation for Mobile Edge Learning

Amr Abutuleb*, Sameh Sorour†, and Hossam S. Hassanein†
*Department of Electrical and Computer Engineering
†School of Computing
Queen's University
Kingston, ON, Canada
{amr.abutuleb, sameh.sorour, hassanh}@queensu.ca

GLOBECOM 2020 - 2020 IEEE Global Communications Conference | 978-1-7281-8298-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/GLOBECOM42002.2020.9322399

***Abstract*—The exploding increase in the number of connected devices and growing sizes of their generated data gave more opportunities for distributed learning to dominate fast data analytics in mobile edge environments. In this work, we aim to jointly optimize the allocation of learning tasks and wireless resources in such environments with the aim of maximizing the number of local training cycles each device executes within a given time constraint, which was shown to achieve a faster convergence to the desired learning accuracy. This joint problem is formulated as a non-linear constrained integer-linear problem, which is proven to be NP-hard. The problem is then simplified into a simpler form by deducing the optimal solution for some parameters. We then employ numerical solvers to efficiently solve this simplified problem. Simulation results show gains up to 166% and 250% compared to the task allocation only and the resource allocation only techniques, respectively.**

***Index Terms*—Distributed Learning, Federated learning, Parallelized Learning, Wireless Resource Allocation**

I. Introduction

The Internet of Things (IoT) is increasingly becoming a conventional component in our everyday lives. With more than 22 billion IoT devices already connected as of 2020, each of which equipped with advanced sensing and storage capabilities, tremendous amounts of data are being generated every second with the aim to improve our daily lives and activities. Yet, these gigantic data sizes are becoming more and more expensive to transfer to central servers for processing/analytics given the limited communication resources of such typically wireless and mobile devices. In addition, a large portion of this data reflect information that is deemed by its owners as too private to share, thus exposing it to all sorts of privacy breaches. These factors made it more appealing for the mobile devices to keep this data locally and to the network to move the capabilities to process and learn from this data at or close to these devices, i.e., in the mobile edge domain.

Distributed learning (DL) arises as the paradigm that suits these mobile edge learning settings, as it enables the training of a global learning model from the local models that are concurrently trained at multiple mobile edge devices. DL branches into two major practical settings, namely the federated and parallelized learning. Both scenarios involves the presence of a governing node, known as the orchestrator, and a number of helping nodes known as learners. The ochestrator aims to train the global model by running local training cycles of the same model at the learners and aggregating their learning outcomes. The difference between the two scenarios lies in the location of the data. In federated learning (FL), different datasets are already stored at the different learners. This scenario is thus motivated by the high cost or privacy risks of moving data from the learners to the orchestrator. In parallelized learning (PL), the orchestrator initially possesses the entire dataset and decides to distributed fragments of it to learners for learning purposes. This scenario is motivated by the limited computational capabilities of edge IoT devices that need to learn from own collected data. This limitation drives them to parallelize their learning jobs on multiple near-by and usually trusted learners (e.g., a home local network).

The theoretical aspects and optimization requirements for DL in homogeneous computing and communication environments were extensively studied in the literature [1]–[3]. It was until very recently that the optimization of computation (i.e., CPU cycles) and communication (i.e., power, bandwidth) resources for FL were optimized in heterogeneous mobile edge environments [4], [5]. However, these works overlooked adapting task allocations to learners (i.e., performed local cycles and analyzed data samples by each learner per global aggregation cycle), especially in scenarios when such global aggregation cycles are time-constrained, a typical requirement imposed by many wireless and mobile settings. On the other

978-1-7281-8298-8/20/$31.00 ©2020 IEEE

hand, the task allocation was optimized in [7], [8] such highly heterogeneous edge environment, but without paying attention to optimizing the wireless resources concurrently to achieve the best performance.

In this paper, we consider the problem of joint task and resource allocation for both time-constrained PL and FL. The aim of these adaptive allocations is to maximize the number of local updates τ within the constrained duration of a global aggregation cycle, which was shown to achieve a faster convergence to the desired learning accuracy. The general problem is first formulated as a non-linear-constrained integer-linear problem, which is clearly NP-hard. The problem is then simplified to a more concise problem after identifying and setting the trivial optimal parameters as inputs to the problem. Being still very complex to analyze, we employ a numerical solver to obtain the globally optimal solution and compare its performance with those that optimize the tasks allocation only or the resources only in the very recent literature.

II. System Settings

This paper considers a system consisting of one orchestrator and K learners that perform one FL or PL job in a mobile edge environment. We will first start by describing the learning and data settings and parameters and then discuss the parameters and implication of executing this learning job in the considered mobile edge environment.

A. Learning and Data Settings

The learning setting considered in this paper consists of any arbitrary DL model (e.g., linear regression, support vector machine, K-means, deep neural network) that can employ a Stochastic Gradient Descent (SGD) approach to train the K local models deployed by the orchestrator on the network's K learners. Unlike Gradient Descent (GD) approaches that requires exhaustic iteration on all stored data samples to do one single update of the learners' parameter (a.k.a. weight) vectors, SGD enables cycles of training using randomly selected samples from the training set to update these vectors in each of these cycles [9]. SGD is clearly more suitable to both FL and PL as it allows the adaptation of allocated tasks to the learners according to their capabilities and resources. It also fits the PL concept as the orchestrator can distribute randomly selected sets of data samples to each of the learners in each cycle. Luckily, it has been proven that SGD often converges much faster in comparison to GD, though the error function may not be as well minimized as in the case of GD. Yet, the close approximation obtained using SGD for the parameter values is usually enough in most cases, and particularly in mobile edge environments where speed is typically more critical than absolute accuracy [10].

The learning process for each global aggregation cycle occurs in three steps:

Step 1: The orchestrator conveys the global learning model parameters to the learners. For PL, this step also involves sending a set of d_k data samples to each learner $k \in \{1, \ldots, K\}$. Defining F as the number of features in the data set, and P_d as the precision of each data feature (i.e., the number of bits representing each feature), the total number of sent data bits in PL can be expressed as:

$$B_k^{data} = d_k F P_d \tag{1}$$

note that in FL, these bits are not sent by the orchestrator, but rather each learner k selects a random set of d_k data samples from its own stored data set to use them in the training process explained in Step 2. Each data sample i assigned/selected by learner k is defined by the input-output pair $\{x_i^k, y_i^k\}_{i=1}^{d_k}$, $k \in \{1, \ldots, K\}$. In addition, defining S_d and S_m as the data-size dependent and data-size independent model parameters, and P_m as the precision of the learning model (i.e. number of bits representing each parameter/weight), the total number of sent bits to convey the employed FL or PL model is:

$$B_k^{model} = P_m(d_k S_d + S_m) \tag{2}$$

Step 2: Each learner starts training the received model with the d_k received (in PL) or selected (in FL) data samples for τ local cycles. The typical goal of the training is to minimize the global loss function of the model expressed as:

$$L_{global} = \sum_{k=1}^{K} \sum_{i=1}^{d_k} f\left(\underline{w}_k, x_i^k, y_i^k\right) \tag{3}$$

where $\underline{w}_k \in \mathbb{R}^{B_k^{model}}$ is the local model parameter vector and $f(.)$ is a loss function in building the relationship between an x_i^k and y_i^k through $\underline{w}_k$. To achieve this target, each learner needs to C_m flops to execute the training calculations per data sample in each local cycle, resulting in a total of:

$$X_k = d_k C_m \tag{4}$$

flop computations per local cycle. It is important to note here that, in a typical SGD algorithm, it has been shown that L_{global} is a decreasing function of τ, i.e., minimizing the loss function can be efficiently achieved by maximizing the number of learning iterations [12], which translates in our setting to maximizing the number of local cycles.

Step 3: At the end of each global cycle of duration T, the orchestrator collects the local model parameter vectors from all learners and aggregates them to build the global model

parameter vector. One popular method for such aggregation is the weighted averaging approach expressed as:

$$\underline{w} = \frac{\sum_{k=1}^{K} d_k \underline{w}_k}{D} \tag{5}$$

where $D = \sum_{k=1}^{K} d_k$ defines the total number of samples that needs to be analyzed in each global cycle, usually imposed by the orchestrator given the considered learning job.

Once these three steps are done, the orchestrator chooses to either stop the process, typically if it converged to the desired level of accuracy, or start another cycle. Interested readers in the local/global loss function minimization and local parameter aggregations are referred to [6] and [11] for more details.

B. Mobile Edge Settings

From the mobile edge environment viewpoint, the aforementioned learning steps and settings must be performed by wireless/mobile edge devices. This physically translates into three different time epochs to complete the above three steps of each global update cycle. The first epoch represents the time needed to send the model and data (in PL) to each of the learners given their channel characteristics. If learner k is assigned forward bandwidth B_k^F and transmit power P_k^F, this epoch will take[1]:

$$t_k^S = \frac{d_k F P_d + P_m(d_k S_d + S_m)}{B_k^F log_2(1 + \frac{P_k^F h_k}{N_o})} \tag{6}$$

for learner k, where h_k is the power gain of the channel between the orchestrator and learner k. The second epoch represents the duration taken by each learner k to finish all its assigned computations to generate $\underline{w}_k$. If learner k has a CPU flop speed of f_k, this duration is equal to:

$$t_k^C = \frac{\tau X_k}{f_k} = \frac{\tau d_k C_m}{f_k} \tag{7}$$

the third and final epoch represents the time needed for each learner to send back its $\underline{w}_k$ to the orchestrator. If the assigned reverse bandwidth and power to learner k are B_k^R and P_k^R, the duration of this epoch is:

$$t_k^R = \frac{P_m(d_k S_d + S_m)}{B_k^R log_2(1 + \frac{P_k^R h_k}{N_o})} \tag{8}$$

III. Problem Formulation

The goal of this work is to minimize the global loss function in each global cycle by maximizing the number of local cycle in each global aggregation cycle, which should typically result in the maximum possible learning accuracy at the end of this global cycle [12]. This goal will be achieved jointly optimizing the tasks allocated to each learner (i.e., $d_k \ \forall \ k$) and its assigned physical resources (i.e., $f_k, B_k^F, B_k^R, P_k^F, P_k^R \ \forall \ k$) so as to maximize τ. Thus, the general form of this optimization problem can be expressed as follows:

$$\max_{\substack{\tau, d_k, f_k, B_k^F, B_k^R, \\ P_k^F, P_k^R, \ \forall \ k}} \quad \tau \tag{9a}$$

$$\text{s.t.} \quad t_k^S + t_k^C + t_k^R \leq T, \quad \forall \ k \tag{9b}$$

$$\sum_{k=1}^{K} d_k = D \tag{9c}$$

$$\sum_{k=1}^{K} B_k^F \leq B \tag{9d}$$

$$\sum_{k=1}^{K} P_k^F \leq P \tag{9e}$$

$$0 \leq f_k \leq f_k^{max}, \quad \forall \ k \tag{9f}$$

$$0 \leq B_k^R \leq B_k^{R,max}, \quad \forall \ k \tag{9g}$$

$$0 \leq P_k^R \leq P_k^{R,max}, \quad \forall \ k \tag{9h}$$

The constraints in (9b) guarantee that the total time of the three process steps will not exceed the preset global cycle duration T for any of the learners. Constraint (9c) ensures that the total no. of samples analyzed by all learners conforms with the bound D set by the orchestrator for each global cycle. Constraints (9d) and (9e) assures that the total forward bandwidths and powers used by the orchestrator to complete Step 1 do not exceed its total bandwidth B and power budgets (denoted by B and P, respectively). Finally, the K constraints in (9f) ensure that each learner does not exceed its maximum flop speed given its computational capabilities or allowance given its other loads. Similarly, the constraints in (9g) and (9h) ensure that each learner does not exceed its maximum reverse bandwidth nor transmit power, respectively, when returning its local model parameter vector to the orchestrator.

As per the above description, the considered problem is a linear integer program with nonlinear constraints (NLCLP), which is well known to be NP-Hard [13]. Thus, solving this optimization problem is challenging, even when using numerical solvers, and thus require some simplification or reduction of variables. As in several prior works [4], [14], the optimal values of several of the optimization parameters can be directly obtained from the formulation. For instance, maximizing τ is directly impacted by setting any variable so as to minimize each of the time expressions t_k^S, t_k^C, and t_k^R in

[1]This expression is for PL. For FL, the value of d_k in the first term of the numerator is simply set to zero.

Constraints (9b). This simple fact can result in the following determinations of the optimal values of some variables, thus eliminating them and their constraints from the problem:

- By examining the expression of t_k^C in (7), we can clearly see that it is minimized for every Learner k by setting its f_k to its maximum possible value. By looking at the constraints in (9f), we can clearly conclude that the optimal value for f_k is to set it to $f_k^{max} \ \forall \ k$.
- By examining the expression of t_k^R in (8), it can be clearly inferred that it is minimized for every Learner k by setting its B_k^R and P_k^R to their maximum possible values, which are defined in Constraints (9g) and (9h) to be $B_k^{R,max}$ and $P_k^{R,max}$, respectively.

As mentioned above, the above facts enables the removal of these parameters from the set of optimization variables of (9) as well as the set of constraints in (9f), (9g), and (9h). This decreases the size of the problem as follows:

$$\max_{\substack{\tau, d_k, B_k^F, \\ P_k^F \ \forall \ k}} \quad \tau \tag{10a}$$

$$\text{s.t.} \quad t_k^S + t_k^C + t_k^R \leq T, \quad \forall k \tag{10b}$$

$$\sum_{k=1}^{K} d_k = D \tag{10c}$$

$$\sum_{k=1}^{K} B_k^F \leq B \tag{10d}$$

$$\sum_{k=1}^{K} P_k^F \leq P \tag{10e}$$

clearly, Problem (10) represents the joint optimization of allocated tasks to learners and their assigned resources from the orchestrator to achieve the maximum possible τ. Constraints (10b), (10c), (10d), and (10e) are the immediate equivalent to those in (9b), (9c), (9d), and (9e), respectively.

Though this problem is simpler than the one in (9), it is still very combinatorial in nature. Finding closed-form expressions or approximate methods was not feasible for even simpler version of the problem in which only resources were optimized [5]. We thus use a numerical solver, namely the OPTI solver [15], to find the solution of the above problem and identify its gains compared to the two most recent related works optimizing resources allocation only and task allocation only.

IV. Simulation Results

In this section, we illustrate the simulation results for our joint task and resource allocation problem of interest in heterogeneous mobile edge environments. We also compare its performance with the two following recent optimizations proposed in the literature.

TABLE I
SIMULATION PARAMETERS

Parameter	Value
System Bandwidth B	100 MHz
Node Bandwidth B_k^R	5 Mhz
Maximum B_k^F	$5K$ Mhz
Device Proximity	50 m
Node Power	23 dBm
Maximum P_k^F	$23K$ dBm
Noise Power Density N_o	-174 dBm/Hz
Attenuation Model	7+2.1log(R)dB[]
Computation Capability f_k	2.4 GHz and 700 MHz
MNIST dataset size D	60,000 images
MNIST dataset Features F	784 features

- **Task-Only**: In this scheme studied in [7], only the task allocation to different learners is optimized to achieve the maximum possible τ while assuming fixed resource allocation.
- **Resource-Only**: In this scheme studied in [5], only the resource allocation to different learners is optimized to achieve the maximum possible τ while assuming fixed allocation of tasks.

The two figures of merit that we use in our comparisons are the maximum achievable number of local cycles per global aggregation cycle (i.e., τ) and the achieved model accuracy at the end of each global aggregation cycle.

The dataset chosen to test our proposed scenarios is the MNIST [16] dataset which consists of 60,000 images where each image consist of 784 features. The employed neural network consists of 3 hidden layers with the following configuration [784, 300, 124, 60, 10] For this network, the model size was calculated to be 8,974,080 bits [17] and the required floating-points operations were 1,123,736 [18]. To ensure a fair comparison between all three schemes, the neural network was constructed from scratch in the simulation environment without using any predefined functions. This guarantees our ability to control different parameters and obtained the most accurate results for each of the three schemes without any impacts from any hidden settings or variables.

From the physical perspective, the edge learners were divided into two groups, one simulating the computational capabilities of portable computing devices and the other simulating those of commercial micro-controllers. In addition, random distances (with a maximum distance of 50m) and fading conditions were generated for each of the learners with respect to the orchestrator. The employed channel model was chosen to emulate 802.11 links between the learners and the

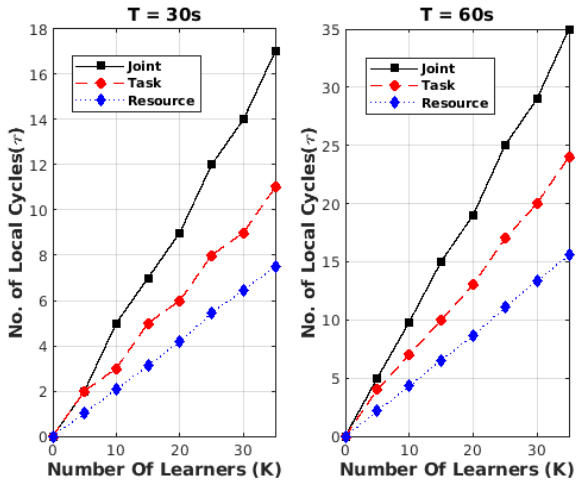

Fig. 1. Number of local cycles for all schemes against K for T = 30 and 60s.

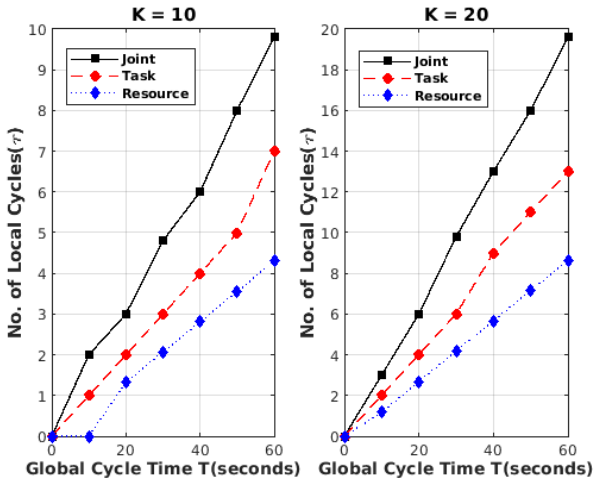

Fig. 2. Number of local cycles for all schemes against T for K = 10 and 20.

orchestrator. Table I summarizes the simulation parameters for both the physical resources/setup and the employed data set.

In Fig 1, τ is tested at different values of K for T=30s and T=60s. The first observation from both sub-figures is that the gain between the three schemes remains almost the same as K increases. For instance, at K=10, T=30s the joint scheme performs 5 updates, the task scheme performs 3 updates and the resource scheme performs 2 updates, resulting in a gain of 166% and 250% respectively for the joint scheme over the task-only and resource-only schemes. When K was increased to 20, the gains remained in the range of 150% and 225%, respectively. Same gain ranges were also obtained for T=60s. This shows the consistency of the gains of our joint scheme for different values for K. Another interesting observation is that the performance of the joint scheme at K=20 and T=30s exceeds the performance for the resource scheme at K=20 and T=60s, which means that the joint scheme can achieve better performance at less duration than the resource-only scheme.

In Fig 2, τ is tested at different values of T for K=10 and K=20. Similar to Fig 1, the gains between the different schemes is almost the same as T increases. One important observation is that at K=10 and T=10, the resource-only scheme wasn't able to perform even one local update, while the other two schemes were able to perform the same number of updates. Yet, the joint scheme outperforms the task-only scheme as T increases.

In Fig 3, the progression of learning accuracy achieved by all three schemes at the end of each global cycle are plotted for T=30s and K = 10 and 20. The figure shows higher accuracy for the joint scheme especially for low global cycle indices. As

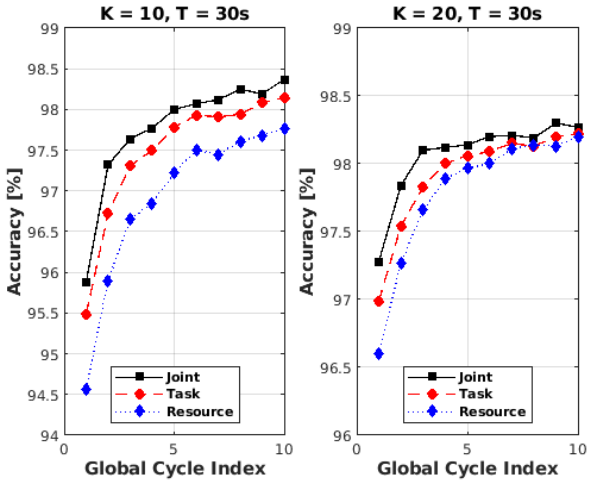

Fig. 3. Learning accuracy achieved at the end of each global cycle for T=30s and K = 10 and 20.

the learning progresses, the accuracy of all the schemes start to be equivalent when K=20, but not K= 10. Yet, even for $K = 20$ the joint, task-only, and resource-only schemes reach 98% accuracy after 3, 4, and 7 global cycles, thus achieving a reduction of 25% (i.e., 30s) and 58% (i.e., 120s) opposed to the other two schemes, respectively.

Finally, Fig 4 depicts the same progression of learning accuracy for K=20 and T = 12s and 30s. Again, the joint scheme achieves a better accuracy than the other schemes at low global indices for both cycle durations. It also converges to 98% faster than the other two schemes. For T=12s, the joint, task-only, and resource-only schemes exceeded 98% accuracy after 4, 7, and 9 cycles, resulting in a reduction

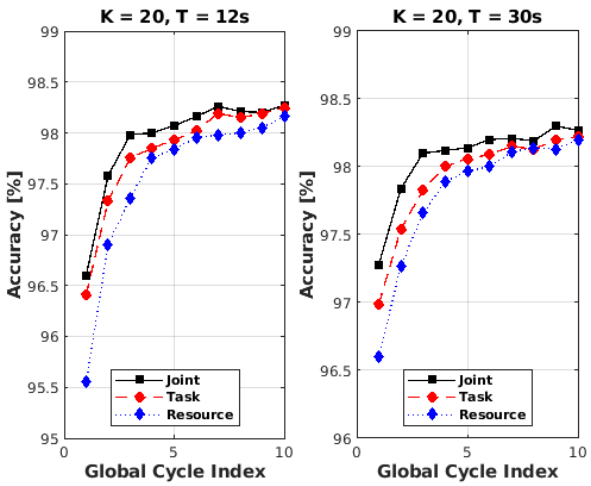

Fig. 4. Learning accuracy achieved at the end of each global cycle for K=20 and T = 12s and 30s.

of 43% (i.e., 36s) and 56% (i.e., 60s) for the joint scheme over the task-only and resource-only schemes respectively. More interestingly, the joint scheme needed 4 T=12s cycles to reach 98% accuracy whereas the task-only and resource-only schemes needed 4 and 7 T=30s cycles to reach the same accuracy. Thus, the joint scheme comes in much handier when the amount of available time for the learning is restricted.

V. Conclusion

In this paper, we investigated the problem of jointly allocating tasks and resources in mobile edge learning environments. The general joint problem was formulated as a nonlinear constrained integer-linear problem, which was then simplified by finding the trivial optimal solutions for some variables. Being still complex, we solved the problem numerically and quantified its gains in maximizing the number of local cycles and converging to a given accuracy compared to the task-only and resource-only schemes. Through extensive experimenting, the joint scheme was proven to outperform both schemes in less time and with less number of learners.

References

[1] Y. Arjevani and O. Shamir, "Communication complexity of distributed convex learning and optimization," in Advances in neural information processing systems, 2015, pp. 1756–1764.

[2] C. Ma, J. Kone˘cn'y, M. Jaggi, V. Smith, M. I. Jordan, P. Richt´arik, and M. Tak´a˘c, "Distributed optimization with arbitrary local solvers," Optimization Methods and Software, vol. 32, no. 4, pp. 813–848, 2017.

[3] J. Wang and G. Joshi. (Jan. 2019). "Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms." [Online]. Available: https://arxiv.org/abs/1808.07576

[4] Chen, Mingzhe, Zhaohui Yang, Walid Saad, Changchuan Yin, H. Vincent Poor, and Shuguang Cui. "A joint learning and communications framework for federated learning over wireless networks." arXiv preprint arXiv:1909.07972 (2019).

[5] Yang, Zhaohui, Mingzhe Chen, Walid Saad, Choong Seon Hong, and Mohammad Shikh-Bahaei. "Energy Efficient Federated Learning Over Wireless Communication Networks." arXiv preprint arXiv:1911.02417 (2019).

[6] S. Wang et al., "When Edge Meets Learning: Adaptive Control for Resource-Constrained Distributed Machine Learning," IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, Honolulu, HI, 2018, pp. 63-71, doi: 10.1109/INFOCOM.2018.8486403.

[7] U. Mohammad and S. Sorour, "Adaptive Task Allocation for Mobile Edge Learning," 2019 IEEE Wireless Communications and Networking Conference Workshop (WCNCW), Marrakech, Morocco, 2019, pp. 1-6, doi: 10.1109/WCNCW.2019.8902527.

[8] U. Mohammad and S. Sorour, "Adaptive Task Allocation for Asynchronous Federated Mobile Edge Learning," ArXiv Pre-prints. [Online]: https://arxiv.org/pdf/1905.01656.pdf

[9] J. Konečný, J. Liu, P. Richtárik and M. Takáč, "Mini-Batch Semi-Stochastic Gradient Descent in the Proximal Setting," in IEEE Journal of Selected Topics in Signal Processing, vol. 10, no. 2, pp. 242-255, March 2016, doi: 10.1109/JSTSP.2015.2505682.

[10] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep Learning for IoT Big Data and Streaming Analytics: A Survey," IEEE Communications Surveys and Tutorials, vol. 20, no. 4, pp. 2923 - 2960, Fourth Quarter 2018

[11] T. Tuor, S. Wang, T. Salonidis, B. J. Ko, and K. K. Leung, "Demo abstract: Distributed machine learning at resource-limited edge nodes," INFOCOM 2018 - IEEE Conference on Computer Communications Workshops, pp. 1–2, 2018.

[12] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. A. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large Deep Networks Nips2012," pp. 1–11, 2012. [Online]. Available: papers3://publication/uuid/3BA537A9-BC9A-4604- 95A4-E6F64F6691E7

[13] A. D. Pia, S. S. Dey, and M. Molinaro, "Mixed-integer Quadratic Programming is in NP," pp. 1–10, 2014.

[14] X. Cai, X. Mo, J. Chen, and J. Xu, "D2D-Enabled Data Sharing for Distributed Machine Learning at Wireless Network Edge," ArXiv, 2020.

[15] J. Currie and D. I. Wilson, "OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User," in Foundations of Computer-Aided Process Operations, N. Sahinidis and J. Pinto, Eds., Savannah, Georgia, USA, 2012

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based Learning Applied to Document Recognition," Proceedings of IEEE, vol. 86, no. 11, 1998.

[17] Bryan Longa "How do you calculate the size of a neural network in memory? - Quora," 2016. [Online]. Available: https://www.quora.com/How-do-you-calculate-the-size-of-a-neural-network-in-memory

[18] Brendan Shillingford, "What is the time complexity of backpropagation algorithm for training artificial neural networks? - Quora," 2016. [Online]. Available: https://www.quora.com/What-is-the-time-complexity-of-backpropagation-algorithm-for-training-artificial-neural-network