Hybrid Fusion Learning: A Hierarchical Learning Model For Distributed Systems

Anirudh Kasturi, Anish Reddy Ellore, Paresh Saxena, Chittaranjan Hota
{p20170403,f20160104,psaxena,hota}@hyderabad.bits-pilani.ac.in
BITS Pilani, Hyderabad, India

ABSTRACT

Federated and fusion learning methods are state-of-the-art distributed learning approaches which enable model training without collecting private data from users. While federated learning involves lower computation cost as compared to fusion learning, the overall communication cost is higher due to a large number of communication rounds between the clients and the server. On the other hand, fusion learning reduces the overall communication cost by sending distributions of features and model parameters using only one communication round but suffers from high computation cost as it needs to find the distributions of features at the client. This paper presents *hybrid fusion learning*, a system that leverages hierarchical client-edge-cloud architecture and builds a deep learning model by integrating both fusion and federated learning methods. Our proposed approach uses fusion learning between the client and the edge layer to minimise the communication cost whereas it uses federated learning between the edge and the cloud layer to minimise the computation cost. Our results show that the proposed hybrid fusion learning can significantly reduce the total time taken to train the model with a small drop of around 2% in accuracies as compared to the other two algorithms. Specifically, our results show that fusion and federated learning algorithms take up to 26.28% and 9.74% higher average total time to build the model, respectively, than the proposed hybrid fusion learning approach.

CCS CONCEPTS

• **Computing methodologies** → **Distributed computing methodologies**; *Neural networks*; *Machine learning approaches*; • **Networks** → *Network resources allocation*; • **Computer systems organization** → **Client-server architectures**.

KEYWORDS

Federated learning, fusion learning, edge computing, distributed machine learning

ACM Reference Format:
Anirudh Kasturi, Anish Reddy Ellore, Paresh Saxena, Chittaranjan Hota. 2020. Hybrid Fusion Learning: A Hierarchical Learning Model For Distributed Systems. *4th International Workshop on Embedded and Mobile Deep Learning (EMDL'20), September 21, 2020, London, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3410338.3412339

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
EMDL'20, September 21, 2020, London, United Kingdom
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-8073-7/20/09…$15.00
https://doi.org/10.1145/3410338.3412339

1 INTRODUCTION

Recent deep learning advances have revolutionized applications such as predictive analysis, risk detection, image and natural language processing, and video analytics [7]. Deep learning models are trained primarily on robust computing systems such as a cloud data center, which includes large centralized datasets. Therefore, it is required for end devices, such as smartphones and sensors, to send data to a central repository for model training. However, the transmission of data from end devices to the central repository raises serious privacy issues. This has resulted in a lot of attention for privacy-preserving distributed training [1].

Federated learning (FL) [11], [16], a recent approach in the field of distributed machine learning is capable of building a global model without transferring the data from the client nodes to the central server. This approach allows local devices to download a global model from the server. The clients then train the model using the locally available data and then upload the model weights to the server. However, the two key challenges [13] in federated learning include communication cost and computation cost due to a large number of devices involved in the process, as described below:

(1) The first key challenge is the communication cost involved to transfer the model parameters between clients and the server over multiple rounds. Hence, bandwidth is a major bottleneck in federated learning as many devices transmit their local values to the central server.
(2) The second major issue with a federated setup is the large number of devices involved in the process. Usually, a federated network consists of hundreds to thousands of devices. Such a large number of devices may overwhelm the server with too many requests, resulting in a centralized system's failure.

In this paper we propose a new approach, *Hybrid fusion learning*, that addresses the above mentioned key challenges in federated learning. Our proposal includes a multi-layer hierarchical system with a client, an edge, and a cloud layer. The edge layer is introduced between the client and the cloud layer to distribute the central cloud server load with several edge servers. Further, clients in the client layer are divided into different clusters. All clients in each cluster communicate with one of the servers at the edge layer. To reduce the communication rounds between the client layer and the edge server, we propose the integration of a recently proposed fusion learning algorithm [9] with our multi-layer hierarchical system. The use of the fusion learning approach reduces the number of communication rounds to just one round between the clients and the edge server by transmitting the distribution of features along

with the model weights. The edge layer generates sample points from the client's input, and a machine learning model on these points is built at each edge server. The servers at the edge layer communicate with the cloud server using federated learning. The edge layer ensures that the load on the cloud server is distributed between several edge servers. Our results show that the hybrid fusion learning can significantly reduce the total time taken to build the model. Specifically, our results show that the two state-of-the-art fusion and federated learning approaches take up to 26.28% and 9.74% higher average total time, respectively, than the proposed hybrid fusion learning approach.

2 RELATED WORK

Several approaches, including [13], [16], have recently been studied in the literature to address the challenge of rising communication cost in federated learning. One way to reduce these costs is to reduce the number of communication rounds. This can be achieved by performing more local updates on the clients before sending the updates to a central server [8], [12]. Further, quantization and compression techniques are proposed in distributed learning for both master-workers and masterless topologies [10] to reduce communication cost. However, the performance of these approaches also depends on the number of clients. As the number of clients increases, the communication cost increases as well.

In federated learning, a centralized server plays a significant role in the training process. Therefore, the failure of a server jeopardizes the entire federated learning setup [18]. The server can also become a computational bottleneck when the number of clients is very large [14]. Decentralized approaches have been proposed to overcome this challenge where the server communication is replaced with peer to peer communication clients [4]. In such approaches, each client shares its model parameters with its neighbors instead of sending its update to a central server. These approaches reduce the load on the server; however, they significantly increase the communication cost.

Researchers have also started looking into the edge-based systems using federated learning with the recent emergence of edge computing platforms [17, 19]. For edge-based FL, the nearest edge server is the parameter server and acts like a base station, where clients collaborate on training a deep learning model within their range. Although, cloud-based and edge-based FL systems use the same FedAvg [16] algorithm, the two systems are fundamentally different with the number of clients in a cloud-based FL system being very large [3]. Moreover, the communication link between the client and the cloud server can be unreliable due to network congestion. This results in inefficient training of the model [3]. The communication latency between the clients and the edge server is also smaller than the communication latency between the clients and the cloud server. Hence, a better balance between computation and communication is needed to address the growing concern of increased client devices in a federated setup.

Our proposed multi-layer hybrid fusion learning approach tackles these issues by incorporating a fusion learning algorithm at the client layer that reduces the number of communication rounds to one and at the same time introduces several edge devices at edge layer which provide an interface between the client layer and the cloud layer to ease the load on the cloud server.

3 PRELIMINARIES

In this section, we review the fundamentals of fusion learning and federated learning, which are the two major techniques used in our proposed method.

3.1 Fusion learning

Fusion learning has been recently introduced to reduce the communication cost associated with the federated learning [9]. The main idea behind this approach is to send model parameters and distribution of features so that the central server can regenerate data points and build a global model. It is achieved by finding the distribution of each feature. The distribution parameters are communicated to the server, along with the parameters of the model that is built locally. The server regenerates sample data points from each client using these distribution parameters. The regenerated data of all the clients is merged to form a large corpus, and a global machine learning model is built. The parameters of this model are then passed back to the client. This approach needs only one round of communication, unlike several rounds of communication in a traditional federated learning algorithm.

Although the fusion learning approach reduces the communication rounds between the client and the cloud server, it increases the load on the cloud server. The cloud server is required to generate points from the distributions received from each client, and then a machine learning model needs to be built from these generated points. The cloud server's load is directly proportional to the number of devices participating in the learning process. As the number of devices increases, the computing power of the cloud server also needs to be scaled to achieve quick results.

3.2 Federated learning

A federated learning [16] system's aim is to minimize a loss function $l(\theta)$ in a distributed fashion given by Equation 1.

$$\min\ l(\theta) = \sum_{c=1}^{C} \frac{n_c}{n} L_c(\theta), \quad where \quad L_c(\theta) = \frac{1}{n_c} \sum_{i \in D_c} l_i(\theta) \tag{1}$$

where C is the total number of clients, $L_c(\theta)$ is the accuracy loss of c^{th} client, θ denotes the model vector and D_c denotes the dataset of c^{th} client with length $n_c = |D_c|$ and $n = \sum n_c$. Note that minimizing the overall loss is equal to minimizing the weighted loss of individual clients. Every client receives the parameters of the global model θ_t from the central server and then uses their own data to train their local models where θ_t is the model vector at time t. Each local device sends its trained local parameters to the server after training locally θ_t. The server aggregates the models received from all the clients and sends modified global model for the next iteration θ_{t+1}. The two key challenges with the federated learning, i.e., the communication and computations costs are discussed in Section 1.

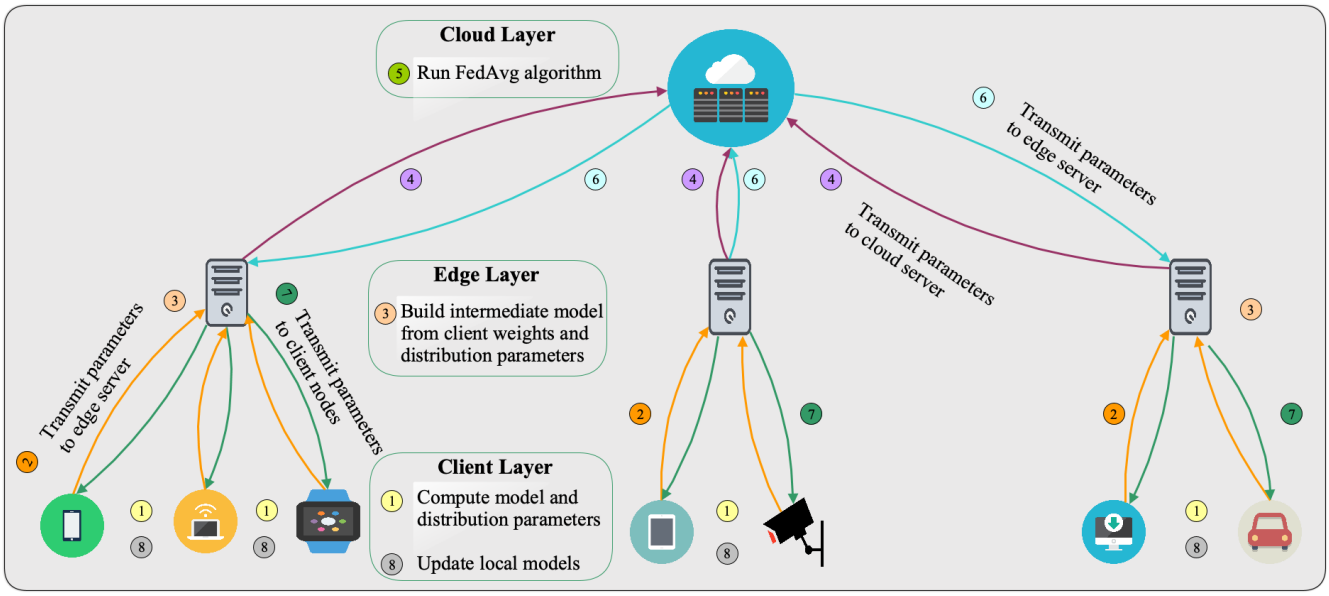


Figure 1: Architectural diagram of a hybrid fusion learning system consisting of three layers, Client, Edge and Cloud.

4 PROPOSED APPROACH

Our proposed approach comprises a hierarchical multi-layer system with a client layer, an edge layer, and a server layer, as shown in Figure 1. Our proposed approach uses fusion learning between the client and the edge layer to minimise the communication cost whereas it uses federated learning between the edge and the cloud layer to minimise the computation cost. The addition of edge layer between the clients and the cloud sever also decreases expensive connectivity from the clients to the cloud, as compared to client-based FL, resulting in a substantial reduction in both runtime and the number of local iterations. The overall architecture can be divided into 3 layers as shown in Figure 1. The first layer is the client layer where all the client nodes reside. The second layer is the edge layer where we have the edge servers and finally the cloud layer where the cloud server is deployed. Figure 1 also illustrate steps involve in the hybrid fusion learning. The layer wise functionality is as follows.

- *Client Layer*: A group of client nodes communicates to the edge layer using fusion learning, where each client computes its local model parameters along with the distribution parameters Ψ. The first step in this process is to find out what distribution each feature follows. In order to do so, Kolmogorov-Smirnov ($K-S$) test [5] is used to compare the sample data with reference probability distributions and computes the probability (p) value for each distribution. The value of p indicates the similarity of sample data with a given distribution. The probability distribution with the highest p value is selected. All the clients build a local model and generate the model vector. The model parameters along with the distribution parameters are transmitted to their corresponding edge server as described in Algorithm 1.
- *Edge Layer*: The edge server receives the distribution and the model parameters from its clients. It uses the distribution parameters to generate sample data points and the weights to compute the predicted outcomes of the generated data. Once the edge server generates the points from each client, it merges all these points to create a large dataset. This large dataset is now used to create an intermediate machine learning model. Such models are created across all edge servers. These edge servers participate in federated learning by transmitting their intermediate model parameters to the cloud server. This process is repeated for every epoch as detailed in Algorithm 2. Since the communication latency from the clients to the edge server is smaller than that to a cloud server, the transmission cost is significantly reduced. Besides, the computation load on a single cloud server is also reduced by dividing the computation among different edge servers.
- *Cloud Layer*: The cloud server's primary purpose is to aggregate the model weights received from the edge server and transmit them back, as shown in Algorithm 3. It uses FedAvg algorithm to aggregate the model parameters. After aggregation, the final values are passed back to the edge nodes. This process is repeated until the desired accuracy is achieved.

5 EXPERIMENTAL RESULTS

This section presents experimental results and highlights the benefits and trade-off of using hybrid fusion learning.

5.1 Experimental Settings

Our proposed system model consists of a single cloud server with multiple edge servers, and each edge server has several clients evenly distributed across each server. For our experiments, we have

Algorithm 1 Hybrid fusion learning: Client Update

Client Update:
for $i \in \{1\ to\ F\}\forall\ features$ **do**
  a. calculate the p value for each distribution using K-S test
  b. find the distribution ψ_i, with max 'p' value
end for
for $e \in \{1\ to\ E\}\forall\ epochs$ **do**
  for $x \in \{1\ to\ X\}\forall\ inputs$ **do**
    Update weights given by:
    $\theta^{e+1} = \theta^e - \eta\nabla L(\theta^e)$
    where θ = weight vector, η = learning rate, and
    L = Empirical loss function
  end for
end for
send $[\Psi, \theta^{e+1}]$ to server

Algorithm 2 Hybrid fusion learning: Edge Update

for $i \in \{1\ to\ C\}\forall\ clients$ **do**
  a. generate points from feature distribution
  b. find predicted value for the generated points
  using θ_i
end for
$D_s = \bigcup_{i=1}^{C} D_i$ //*merge data points from all clients*
for $e \in \{1\ to\ E\}\forall\ epochs$ **do**
  for $d \in \{1\ to\ s\}\forall\ D_s$ **do**
    Update weights given by:
    $\theta^{e+1} = \theta^e - \eta\nabla L(\theta^e)$
    where θ = weight vector, η = learning rate, and
    L = Emperical loss function
  end for
  transmit θ^{e+1} to cloud
end for

Algorithm 3 Hybrid fusion learning: Cloud Update

for $e \in \{1\ to\ E\}\forall\ epochs$ **do**
  for $k \in \{1\ to\ K\}\forall\ EdgeServers$ **do**
    $$\theta^{avg} = \frac{1}{K}\sum_{k=1}^{K}\theta_k$$
    //*avg weights from all edge servers*
  end for
  transmit θ^{avg} to edge server
end for

considered the total number of clients to be 100. The experiments have been performed on three different scenarios where the number of edge servers are varied between 2, 5 and 10 and the corresponding number of clients at each edge server are 50, 20 and 10 respectively. It is also assumed that the edge servers are located closer to the clients resulting in lower communication latencies between clients and the edge server.

Each client builds a local model using a simple multi-layer perceptron with two hidden layers that use ReLu as its activation function. Each hidden layer consists of 10 hidden nodes. We have used Adam Optimizer and Sparse categorical cross-entropy to calculate the loss [2]. The size of the batch is set to 32. Each client sends the model parameters and their feature distributions to respective edge servers. Each edge server generates 1000 random points from the distribution parameters sent by each client. The edge server then merges all these points to create a global dataset and an intermediate model is created at the edge server. All edge servers send their model parameters to a single cloud server. The weights are exchanged for every epoch.

Table 1: Dataset description

Dataset	Instances	Features	Distributions
Credit Card	30000	24	22 - Normal, 1 - Beta, 1 - Erlang
Adult	48842	14	12 - Normal, 1 - Beta, 1 - Pearson3
Bank Marketing	45211	17	15 - Normal, 1 - Lognormal, 1 - Exponential Weibull

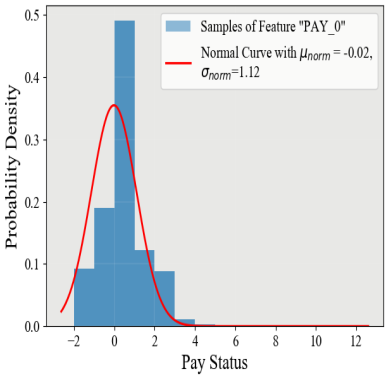
We have used three datasets in this paper: Credit Card, Bank Marketing, and Adult Datasets from UCI Repository [6]. The Credit Card dataset consists of 30,000 instances and 24 features. The distribution of the individual features of the data set is calculated using the Fusion algorithm [9]. Figure 2 illustrates the distributions of three features, *Pay_0*, *AGE* and *Limit Balance* from Credit Card dataset. Overall, 22 features follow a normal distribution, whereas the remaining two features follow Erlang and beta distributions [5] respectively. The Adult dataset consists of 48,842 instances with 14 features, 12 of which follow normal distribution while the other two follow beta and Pearson3. The final Bank Marketing dataset consists of 45,211 instances with 17 features, with 15 features following normal distribution while the remaining two follow lognormal and exponential weibull distribution. This information has been consolidated in Table 1.

5.2 Performance Metrics

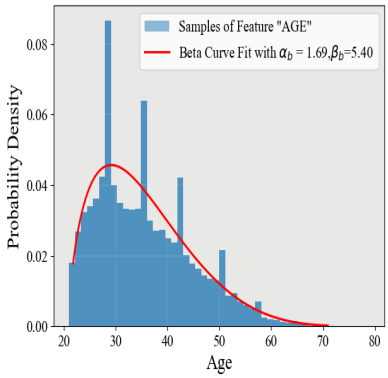
We use two performance metrics: (i) accuracy and (ii) total time taken. We compare hybrid fusion learning with the existing state-of-the-art federated and fusion learning. The total time taken is defined as the summation of transmission time and the computation time. For a federated learning, though the computation time at the server is relatively low, the average total time taken to transmit data from the clients to the cloud server increases as the number of clients increases. Similarly, for a fusion algorithm, the computation time increases when the number of clients increases. In our proposed hybrid fusion learning, the total time taken depends on the number of edge servers and the clients per edge server. Hence it is important to select a right configuration that results into the minimum total time taken.

5.3 Results

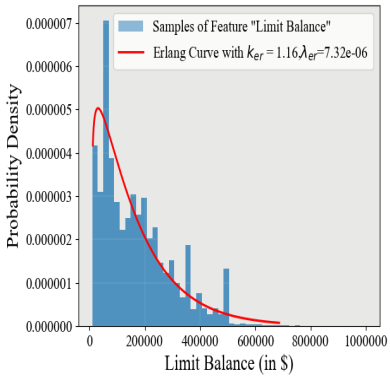
Table 2 summarizes the total time taken and the accuracies of the proposed hybrid fusion learning with different edge servers. The



(a) The normal distribution of feature (Pay_0) from Credit Card dataset with $\mu_{norm} = -0.02$ and $\sigma_{norm} = 1.12$. Overall 22 features from Credit Card dataset follow the normal distribution.



(b) The beta distribution of feature (AGE) from Credit Card dataset with $\alpha_b = 1.69$ and $\beta_b = 5.4$. Overall 1 feature from Credit Card dataset follows the beta distribution.



(c) The Erlang distribution of feature (Limit Balance) from Credit Card dataset with $k_{er} = 1.16$ and $\lambda_{er} = 7.32e{-}06$. Overall 1 feature from Credit Card dataset follows the Erlang distribution.

Figure 2: Distributions of three features of Credit Card dataset.

Table 2: Total time taken and accuracy with hybrid fusion, fusion, and federated learning.

Datasets	Credit Card		Adult		Bank Marketing	
	Total Time (s)	*Accuracy* (%)	*Total Time* (s)	*Accuracy* (%)	*Total Time* (s)	*Accuracy* (%)
C=50, K=2	302	80.9	321.8	90.2	352.32	82.2
C=20, K=5	**201.6**	80.6	**223.8**	90.1	**251.12**	81.9
C=10, K=10	260.3	76.9	283.2	86.1	310.02	76.6
C=100, K=0 (*Fusion*)	254.6	81.8	272.4	90.4	304.02	83.8
C=100, K=0 (*Fed*)	215.6	82.4	245.6	91.5	265.02	84.6

results are compared with federated and fusion learning techniques. In this table, K indicates the number of edge servers and C indicates the number of clients per edge server. In both federated and fusion learning, the number of edge servers is zero as the parameters are transmitted directly to the cloud and hence $C = 100$. We assume that the the communication latency between the client and the cloud server is approximately ten times larger than that to the edge server [15].

Our results show that the total time taken is substantially reduced by using hybrid fusion learning when $K = 5$. Specifically, fusion learning takes overall 26.28%, 21.71% and 21.06% higher total time than hybrid fusion learning for Credit Card, Adult and Bank Marketing datasets, respectively. Similarly, federated learning takes overall 6.94%, 9.74% and 5.53% higher total time than hybrid fusion learning for Credit Card, Adult and Bank Marketing datasets, respectively.

The reason for the above results is as follows. The total time includes computation time at each layer and the transmission time to transmit parameters from client to edge, edge to cloud, and vice versa. For the fusion learning, the total computation time is higher as the clients are required to perform more tasks in order to find the distribution parameters along with the model parameters. Further, edge servers are required to generate sample data from each client and build a machine learning model from these points. For the federated learning, the computation time is smaller but the transmission time to send the parameters from the client to the cloud server is higher as a higher number of communication rounds is necessary to achieve the required accuracy. The hybrid fusion learning introduces multiple edge servers that reduce the load from the central server. Since the clients are distributed across different edge servers, the computations are also done in parallel. At the same time, the communication latency between the clients and the edge server is significantly smaller than the latency between the clients and the cloud server.

Our results also show that it is important to select the correct configuration (total number of edge servers and clients per edge server) for the proposed hybrid fusion learning. For example, when the number of edge servers increases, the total time also increases because more number of edge servers participate in a federated setup and therefore require more communication rounds to converge to a required accuracy level.

Furthermore, we also present the trade-off of using hybrid fusion approach. Figure 3 shows the comparison of accuracy for the federated, fusion, and the hybrid fusion approaches where for the hybrid fusion learning, the number of edge servers is set to 5. The average values of accuracy are also shown in Table 2. We observe a small drop (less than 2%) in accuracy for hybrid fusion when $K = 5$ as compared to federated and fusion learning. We also observe that

(a) Credit Card Data. **(b) Bank Marketing Data.** **(c) Adult Data.**

Figure 3: Testing accuracy for hybrid fusion, fusion and federated learning for Credit Card, Bank Marketing and Adult datasets.

when the number of edge server increases, the accuracy further decreases. This is because the number of clients under each edge server decreases. With fewer clients, the data generated by the edge server using the distribution parameters is also reduced, resulting in a less accurate model. Overall hybrid fusion learning reduces the total time taken significantly within the accuracy drop of around 2%.

6 CONCLUSION

We have proposed a hybrid fusion learning approach that uses both fusion and federated learning methods for hierarchical client-edge-cloud architecture. Our experimental results show that by introducing the intermediate edge layer, the total time taken by hybrid fusion learning can be significantly reduced when compared to conventional cloud-based federated learning or cloud-based fusion learning approaches. We have also shown that the hybrid fusion learning achieves similar accuracy as compared to the accuracy achieved by both federated and fusion learning methods. Future work includes the investigation of hybrid fusion learning for different communication systems. Furthermore, we plan to explore and mathematically formulate the optimal selection of the configuration, i.e., the number of edge servers and clients per edge server for the hybrid fusion learning setup.

REFERENCES

[1] Naman Agarwal, Ananda Theertha Suresh, Felix Xinnan X Yu, Sanjiv Kumar, and Brendan McMahan. 2018. cpSGD: Communication-efficient and differentially-private distributed SGD. In *Advances in Neural Information Processing Systems*. 7564–7575.

[2] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg.

[3] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konečný, Stefano Mazzocchi, Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. 2019. Towards Federated Learning at Scale: System Design. In *Proceedings of Machine Learning and Systems 2019*. 374–388.

[4] Igor Colin, Aurélien Bellet, Joseph Salmon, and Stéphan Clémençon. 2016. Gossip dual averaging for decentralized optimization of pairwise functions. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*. 1388–1396.

[5] M.H. DeGroot and M.J. Schervish. 2012. *Probability and Statistics*. Addison-Wesley. https://books.google.co.in/books?id=4TlEPgAACAAJ

[6] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[7] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. 2016. Deep learning. vol. 1.

[8] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. 2014. Communication-efficient distributed dual coordinate ascent. In *Advances in neural information processing systems*. 3068–3076.

[9] Anirudh Kasturi, Anish Reddy Ellore, and Chittaranjan Hota. 2020. Fusion Learning: A One Shot Federated Learning. https://www.iccs-meeting.org/archive/iccs2020/. In *Proceedings of International Conference in Computational Sciences*. Springer.

[10] Anastasiia Koloskova, Sebastian Urban Stich, and Martin Jaggi. 2019. Decentralized Stochastic Optimization and Gossip Algorithms with Compressed Communication. *Proceedings of Machine Learning Research* 97, CONF (2019).

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

[12] Ching-Pei Lee and Dan Roth. 2015. Distributed box-constrained quadratic optimization for dual linear SVM. In *International Conference on Machine Learning*. 987–996.

[13] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[14] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. 2017. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*. 5330–5340.

[15] L Liu, J Zhang, SH Song, and KB Letaief. 2019. Client-edge-cloud hierarchical federated learning. *arXiv preprint arXiv:1905.06641* (2019).

[16] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Artificial Intelligence and Statistics*. 1273–1282.

[17] Nguyen H Tran, Wei Bao, Albert Zomaya, Nguyen Minh NH, and Choong Seon Hong. 2019. Federated learning over wireless networks: Optimization model design and analysis. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 1387–1395.

[18] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. 2017. Decentralized collaborative learning of personalized models over networks.

[19] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. 2019. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications* 37, 6 (2019), 1205–1221.