Asynchronous Edge Learning using Cloned Knowledge Distillation

Anonymous authors
Paper under double-blind review

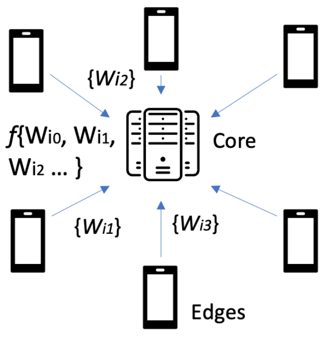
Abstract

With the increasing demand for more and more data, the *federated learning* (FL) methods, which try to utilize highly distributed on-device local data in the training process, have been proposed. However, fledgling services provided by startup companies not only have limited number of clients, but also have minimal resources for constant communications between the server and multiple clients. In addition, in a real-world environment where the user pool changes dynamically, the FL system must be able to efficiently utilize rapid inflow and outflow of users, while at the same time experience minimal bottleneck due to network delays of multiple users. In this respect, we amend the conventional federated learning scenario to a more flexible *asynchronous edge learning*. To solve the aforementioned learning problems, we propose an asynchronous communication method with knowledge distillation. In particular, we dub our knowledge distillation scheme as "cloned distillation" and explain how it is different from other knowledge distillation methods. In brief, we found that in knowledge distillation between the teacher and the student, there exist two contesting traits in the student: to attend to the teacher's knowledge or to retain its own knowledge exclusive to the teacher. And in this edge learning scenario, the attending property should be amplified rather than the retaining property, because teachers are dispatched to the users to learn from them and re-collected at the server to teach the core model. Our asynchronous edge learning method can elastically handle the dynamic inflow and outflow of users in a service with minimal communication cost, operate with essentially no bottleneck due to user delay, and protect user's privacy. Also we found that it is robust to users who behave abnormally or maliciously.

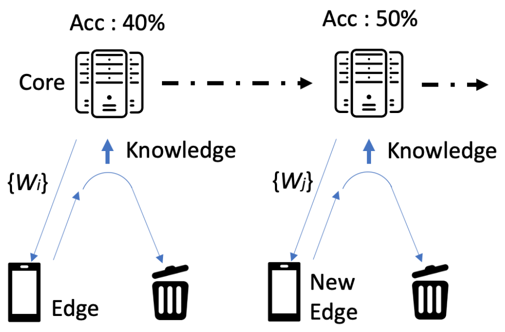
1 Introduction

Deep learning based on convolution neural networks (CNN) is considered as one of the most successful methods in modern machine learning and computer vision fields. It generally requires a large amount of data samples for real world applications, yet there exist physical limitation and privacy issues in acquiring more and more data. Meanwhile, recent researches suggest that utilizing more data is crucial for better performance. Some recent researches (Kolesnikov et al., 2019; Xie et al., 2020) show that just giving an additional big stream of data outperforms a model designed by a machine (AutoML) (Zoph et al., 2018), or carefully hand-crafted models (Tan & Le, 2019).

Federated learning (FL) (Yang et al., 2019; Ludwig et al., 2020), a recently suggested concept, is one of the many solutions to the challenge of acquiring vast amount of data without invasion of privacy. FL assumes that many distributed devices exist for their own data collection. The main goal is the parallel or distributed training of those device-specific datasets by one central model. Since its proposal with deep neural nets, many works have focused on model averaging of the client models to acquire a centralized server model. However, this standardized practice of model averaging necessitates communicating with all or subsets of clients for a single update of the centralized model, which may lead to a bottleneck in case of a network delay. Furthermore, this method of heavy aggregation also entails communication costs that linearly increases with the number of aggregated clients. These two main assumptions of the current FL formulation restrict its application to a real-world environment that fledgling services face. Namely, the environment is characterized by dynamic inflow and outflow of users, limited amount of users, and scarce resources allocated for communication. Thereby, in this paper we propose an amended formulation of FL, called *asyn-*



(a) Heavy aggregation of FL



(b) Our lightweight edge learning (EL)

Figure 1: Schematic comparison of heavy aggregation of the conventional FL and our proposed edge learning scenario.

chronous edge learning, which intends to extract as much information from a single interaction with a user as possible. In addition, we propose an effective method that can asynchronously update the centralized model without any bottleneck via knowledge distillation (Hinton et al., 2015).

In our edge learning, we call the centralized service provider as the "core" and the customer's device as the "edge". Edge learning characteristically assumes that the service provider has at most one round of interaction with a given edge and that only a single edge is used to update the centralized core model to minimize the communication cost. These assumptions resemble the realistic environment with dynamic inflow and outflow of users and minimal resource constraint to communicate with users. Thus, the main goal is to extract as much information from a single interaction with a given edge, while preventing catastrophic forgetting of the core model. To achieve this, we utilize knowledge distillation (KD) as a communication method between the core and edges and devise a concept called *prediction consensus score* to explain when KD is appropriate. Few works (Xie et al., 2020) have exploited KD in a similar manner to ours in different fields of study. We call this variant of KD as "cloned distillation" and explain when this type of distillation is appropriate with the prediction consensus score. Note that although the restrictions of single core-edge interaction followed by edge churning may be too rigorous, it nevertheless sets a lower bound of a real-world environment. Thus, demonstrating feasibility in this environment is meaningful.

Our contributions are as follows:
(1) We reformulate the federated learning problem to a more realistic problem of dynamic user pool change called asynchronous edge learning.
(2) We define "cloned distillation" as a variant of KD and explain when and why this is useful for a certain class of problems.
(3) In utilizing cloned knowledge distillation for asynchronous edge learning, we suggest a method that can effectively learn from the decentralized edge datasets and alleviate the catastrophic forgetting problem of the core model.

2 Related Works

With the prevalence of mobile devices that can collect rich variety of data, the concept of training a robust model with decentralized data has emerged as an important research area in both industry and academia (Ludwig et al. (2020); Wang et al. (2020); Chen et al. (2019)). The first recent proposal (McMahan et al. (2017)) applied to deep networks used model averaging of the client models to acquire a centralized model. Since then, many variants of model averaging in the aggregation step have been proposed. Li et al. (2020) add a proximal penalty between the client model and the core model. Yurochkin et al. (2019) solve the permutation invariant property of neural networks that causes a detrimental effect in model averaging by matching neurons of the client models. Wang et al. (2020) further extend this idea to deep neural networks by layer-wise matching and averaging. While all the aforementioned methods improve upon the standardized aggregation method of model aver-

Algorithm 1: Asynchronous Edge Learning

```
Result: Incrementally increasing core model performance
$K$: the number of currently available edges;
$i$: the index of round, initially set to 0;
PHASE 0: Core initialization – core is trained with the core dataset $C$ with $L_{core}$;
while edge $E_{i+1}$ exists do
    $i$++;
    DOWNLINK: Preparation of an edge model – send the core model to the edge;
    while edge model has not reached a learning threshold do
        PHASE 1: Expedition and learning on edge – edge model is trained with the edge
          dataset $E_i$ with $L_{edge}$;
    end
    UPLINK: Re-collection of the edge model – send the edge model to the core;
    while knowledge distillation has not converged do
        PHASE 2: Knowledge distillation – transfer edge's knowledge to the core with $L_{distill}$;
    end
    if $i = K$ then
        wait until another edge is appended;
    end
end
```

aging, this aggregation practice is only applicable when multiple client models are simultaneously available for communication. Meanwhile, Li & Wang (2019) approach the task by using knowledge distillation between the averaged model and the client model.

3 Proposed Method: Asynchronous Edge Learning

In the standard federated learning problem, a single round, which is characterized by a heavy communication with multiple sub-sampled or entire clients, is iterated for a server-specified number of iterations. In contrast, rather than a server-specified number of rounds, our edge learning assumes that an edge communicates with the core at most once. Single communication between the core and the edge constitutes a round. This difference is depicted in Figure 1. Under this condition, our solution is to send a certain deep model as a local learner to each edge device, learn from it, and re-collect it as a teacher to the core model in parallel. The re-collected teacher performs knowledge distillation with a small, pre-prepared core dataset. The model sent to each edge device is called as an "edge model". From now on, consider the core dataset C whose size is N_C, and K edge datasets $\{E_k\}_{k=1}^{K}$, each with the size of N_{E_k}. For a data sample x, the output logit vector of the core model is expressed as $F(x)$ and that of the k-th edge model is denoted as $f_k(x)$.

Algorithm 1 is used to train the core model using the core dataset without directly seeing the edge datasets. However, the core model learns from the returned edge models that had previously been sent to the edge devices to learn from the corresponding edge dataset. The algorithm is composed of 3 phases of which Phase 1 and Phase 2 are iterated. This iteration constitutes a round. In detail, in Phase 0 the core model learns from the core dataset, and each round is executed for each edge that participates in the core service. Note that the rounds could run asynchronously, not sequentially. The effect of lagged response is discussed in the experiment section.

In Algorithm 1, the learning in each phase (Phase 0 through 2) can be expressed as follows:

$$\begin{gathered} L_{core} = \sum_{i=1}^{N_C} L(F(x_i), y_i), \quad L_{edge} = \sum_{k=1}^{K} \sum_{j=1}^{N_{E_k}} L(f_k(x_j), y_j), \\ L_{distill} = L_{core} + t^2 \sum_{i=1}^{N_C} \mathrm{KL}(F(x_i), f_k(x_i)/t). \end{gathered} \tag{1}$$

Here, $L(\cdot,\cdot)$ and $KL(\cdot,\cdot)$ are the cross entropy loss and KL divergence, respectively. Note that C and E_k are fully disjoint sets, and each of those functions must be performed only in the specified

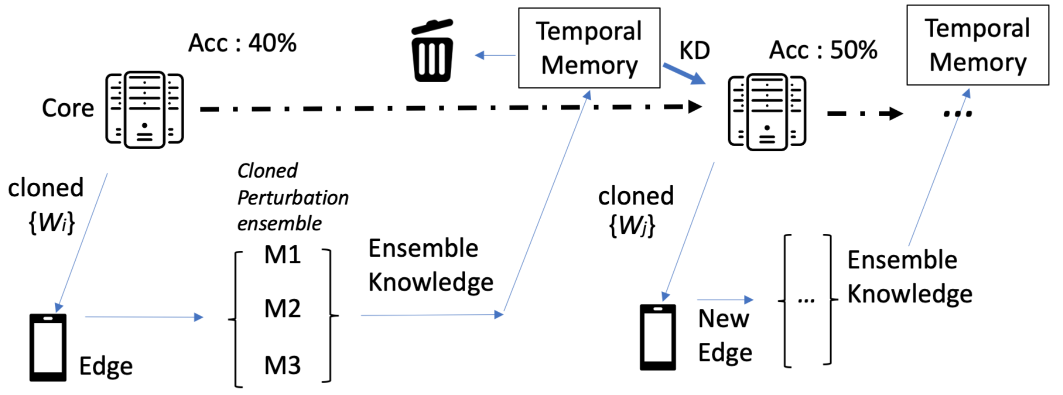


Figure 2: We incorporate model ensemble at each edge node and temporal memory at the core to store maximum m proceeding edge models. In Phase 1, to obtain an ensemble of edge models from one identical copy of the core model, a different input batch sampling is applied for each edge model (M1 to M3). In the KD phase (Phase 2), the returned ensemble of edge models and the edge models from the previous (M-1) rounds stored in the memory are used as teachers.

phase, i.e, in Phase 0, L_{core} is used, while L_{edge} and $L_{distill}$ are used in Phase 1 and Phase 2 respectively.

4 Detailed Methodology

To boost the performance of the proposed asynchronous edge learning, as shown in Figure 2, we incorporate model ensemble and temporal memory in the edge model training (Phase 1) and knowledge distillation (Phase 2), respectively. The details are described in the following subsections. Before moving on, we introduce the unique property of cloned knowledge distillation that constitutes the gist of our method, which is different from the conventional knowledge distillation.

4.1 Property of cloned Knowledge Distillation

In the previous section, we mentioned that the core model is dispatched to the edge and trained using the edge dataset. Once the training is done, the edge model becomes the teacher model for the training of the core model. Using the cloned core model as the initial point of edge learning rather than using an independent random initial point highlights a fundamental difference between cloned KD and vanilla KD (independent KD hereafter).

In our method, a specific form of cloned KD is applied. In detail, a cloned core model that has been trained on the edge dataset teaches the core model with the initial high learning rate of the original learning schedule. In essence, cloned KD performs knowledge distillation without deviating significantly from the solution that was obtained by training with the first learning rate schedule. In contrast, the independent knowledge distillation involves teacher and student models trained with two different initial seeds. Accordingly, no restrictions exist in the proximity of the two models in the parameter space, which is depicted in Figure 3. The motivation of maintaining a desirable initial point as in cloned distillation is not a newly suggested concept in the fields of deep learning. Xie et al. (2020) utilized a similar idea for self-training by using an identical teacher model trained with perturbation. Lottery Ticket Hypothesis (Frankle & Carbin, 2018) is also an effort to take advantage of a good initial point referred to as the lottery ticket, although KD is not used. Additionally, many federated learning methods operate in a similar way (even if they do not use knowledge distillation), because there is no need to train another core model or edge model from scratch.

Then why is cloned KD useful in our scenario? In KD, the student model receives the learned knowledge by imitating the teacher model. But at the same time, the student attempts to retain its own knowledge exclusive to the teacher. Thus, a trade-off exists between two traits of the student: to attend to the teacher or to retain its exclusive knowledge. General research on KD will be interested in both traits rather than one particular trait, because it needs to boost accuracy. However, we need

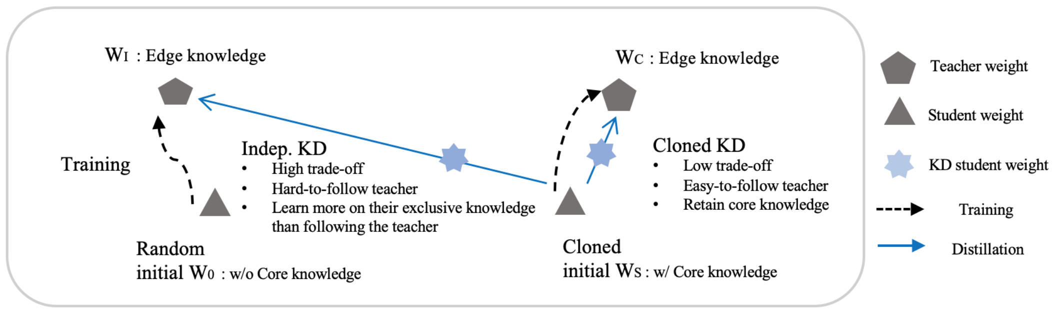
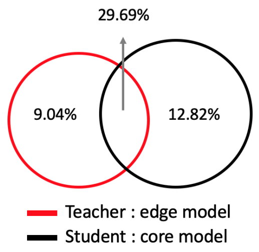
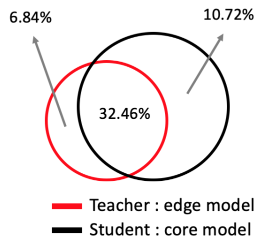


Figure 3: Intuitive interpretation of the difference between cloned KD and independent KD on an imaginary 2-dimensional parameter space. The properties of the two types of distillation are written in text.



(a) Independent distillation



(b) Cloned distillation

Figure 4: Venn diagrams of correctly classified samples after KD with a fixed teacher. In both cases (a) and (b), the teacher model was dispatched to and re-collected from the edge. Results on CIFAR-100 with ResNet-32.

a way to focus more on the attending trait for our scenario, because only through the dispatched teacher model can the student model in the core learn about the edge dataset. Cloned distillation does this by preparing a teacher that shares the same knowledge with the student, which is done by initializing the teacher with the student's weight, then training it with its exclusive edge dataset. This allows the student to focus more on the exclusive side of the teacher's knowledge learned on the edge dataset. Fig. 3 compares the difference between cloned KD and independent KD. In the parameter space, consider the two initial settings of the teacher model W_0 and W_S for independent and cloned teacher respectively. After learning the edge data, the final teacher model is denoted by the pentagons (W_I and W_C) in the figure. In the distillation stage at the core, the distance between the student (W_S) and the teacher for independent KD (W_I) is typically so large that the final student's weight after KD will be located somewhere in the middle between W_S and W_I. On the other hand, in cloned KD, the teacher-student distance ($||W_C - W_S||$) is small and the resultant student's weight after KD will be close to the original student's weight (W_S). This means that the student did not forget about the original knowledge, while also absorbing teacher's new knowledge on the edge dataset. This explains the larger intersection area in Fig. 4(b) compared to that of Fig. 4(a).

To quantify this underlying effect, we propose a measurement that we name "prediction intersection", which can be seen in the Venn diagram of Figure 4. This is represented by the intersection ($\cap$) in the Venn diagram of the correct prediction result for an untrained dataset such as a test set. It also yields a score called prediction consensus, which is defined as the intersection over union. The teacher-student intersection area and the student's exclusive area are the regions of interest referred to as mutual areas and exclusive areas, respectively. As long as the prediction consensus is less then 1, knowledge distillation will have an effect on the student, and both the mutual area and the exclusive area may change. That is, if (Δmutual + Δexclusive) is positive after distillation, there is a performance improvement. For the case of cloned distillation, the exclusive area is almost unchanged, while the mutual area is enlarged. With this interpretation, we can get an intuition as to why there is an improvement in accuracy after KD, even when an equivalent-sized or smaller model is used as a teacher.

Experiment	Edge Model Preparation	Phase1	Phase2	Final Acc.
Independent KD (dashed line)				
IsPP	scratch	-	-	39.99
IPP	indep.	-	-	44.97
IEP	indep.	ensemble	-	47.04
IPM	indep.	-	memory	49.05
Cloned KD (solid line)				
CPP	clone	-	-	53.37
CEP	clone	ensemble	-	54.26
CPM	clone	-	memory	55.11
CEM (Ours)	clone	ensemble	memory	**56.67**

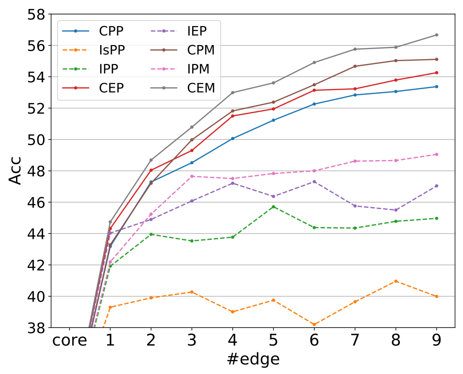


Figure 5: The left table shows four variants of independently trained edge model (I–) and variants of cloned edge model (C–). The next two letters indicate whether proposed methods are applied or not with 'P' (plain) indicating no method is applied. The final accuracy of the table and the accuracy curve all indicate that cloned distillation (solid line) is crucial. Refer to Section 5.1 for more details. Results on CIFAR-100 with ResNet-32.

4.2 CLONED PERTURBATION ENSEMBLE

As mentioned earlier, we need a way to maximize efficiency with only one round of interaction with an edge. So to boost the performance at the edge, we propose an ensemble methodology. In order to obtain a meaningful ensemble of models, individual models have to provide exclusive domains of knowledge not learned by the other models. In this paper, we achieve this by enforcing a small perturbation to induce the models to yield different solutions. By simply performing different batch sampling, the cloned models are trained differently and their ensemble becomes meaningful. These ensembled edge models (M1, M2, M3 in Fig. 2) are re-collected and passed to Phase 2.

4.3 TEMPORAL MEMORY

As mentioned above, in our scenario each edge is used once and discarded. If catastrophic forgetting happens, this may be a serious problem as no edge is left. To prevent this, we propose a method of storing up to M number of edge models that are nearest to the current round and use them together. We call this temporal memory. At this time, $L_{distill}$ in Eq. (1) for Phase 2 is converted to:

$$L_{distill} = L_{core} + t^2 \sum_{m=1}^{M} \sum_{i=1}^{N_C} \mathrm{KL}(F(x_i), f_m(x_i)/t). \tag{2}$$

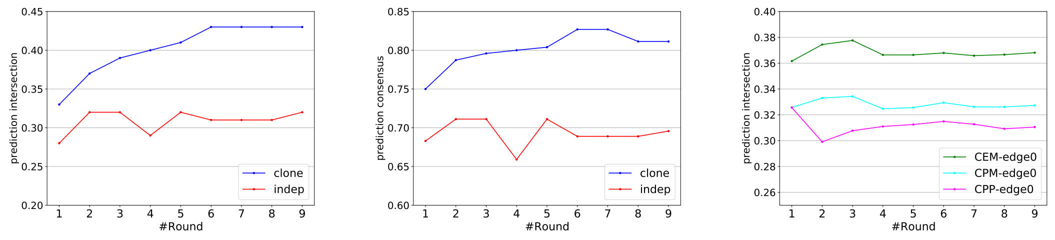
We will show in the experiment that using the re-collected edge models from the past benefits the training of the core model.

5 EXPERIMENTS

In all our experiments, we used the CIFAR-100 dataset with the ResNet-32 (He et al., 2016) architecture and assumed 9 edges (clients) and one core. The 50K training samples were evenly split into ten subsets, each of which consisting of 50±6.9 samples per class. All the hyper-parameters such as the training epochs of the edge and core models were not fine-tuned in our experiment. Instead, we maintained the original ResNet schedule for both edge and core models.

5.1 CLONED DISTILLATION FOR ASYNCHRONOUS EDGE LEARNING

The main purpose of this experiment is to demonstrate the promising effect of cloned distillation in incrementally enhancing the accuracy of the core model as opposed to independent distillation. In Figure 5, we compare several variants of independent KD (Baselines) and cloned KD (Our method) by applying cloned perturbation ensemble in Phase 1 and temporal memory in Phase 2. For instance,



(a) Prediction intersection between edge model and core model ($T \cap S$)

(b) Prediction consensus between edge model and core model $(T \cap S)/(T \cup S)$

(c) Prediction intersection between edge-1 model and core model ($T \cap S$)

Figure 6: Scores that show cloned distillation is more easy-to-follow than independent knowledge distillation. Results on CIFAR-100.

the first letter I in the table indicates that the edge model was initialized independently and trained with the edge dataset. In other words, it is trained by scratch (Is-) at the edge or trained at the core (I-) using different seeds with the core model, and sent to the edge for local training. On the other hand, cloned distillation (C-) is our method, which sends the cloned core model to the edge.

We first compare plain variants indicated by the first and the second lines of Independent KD (IsPP, IPP) and the first line of cloned KD (CPP). The final accuracies of models trained independently all fall short of CPP, even though the same amount of datasets and training resources was used for IPP. This result indicates that cloned distillation is crucial for edge learning. Also, from Figure 6, we can see the benefit of cloned KD compared to the independent KD.

Next, note the variants that applied *cloned perturbation ensemble* mentioned in Section 4 are indicated as IEP and CEP. Note that the individual network's accuracy is about 40%s, but their ensemble shows 43.8% performance. The ensembles were created by perturbing the batch sampling and sent to the core for Phase 2. These experiments demonstrate the effect of the improved edge models on the core model. Interestingly, this method contributes performance improvement for Cloned KD, whereas for Independent KD, the ensemble only improved the performance at the first round, but did not show sustained performance improvement. In other words, without cloned KD, even if the edge model is improved by the ensemble, the core model at Phase 2 cannot attend enough to the teacher to benefit from it.

We also compare IPM and CPM, which are the variants with *temporal memory*. The results demonstrate that Cloned KD benefits from temporal memory of past models. We actually observed that the prediction consensus between the core model and the first edge model (edge-1) during the ninth round was higher when temporal memory was used. Note that *cloned perturbation ensemble* method and *temporal memory* method show higher synergy when combined (CEM).

5.2 LAGGED EDGE MODEL

In this experiment, we show that our proposed method is robust to lagged response of the edges. Some delayed responses are still inevitable in a realistic federated learning scenario. For instance, some users may have a slower training process on the edge device or not be able to communicate with the server even after training has completed. Such delays may bottleneck the learning process if the federated learning scheme is seriously hampered by omitting the delayed responses. Suppose we wish to aggregate all the feedbacks from the edges at time t, but a feedback (edge model) from a certain edge is delayed and hence not available. After aggregation has already been completed, simply rejecting the delayed response would be wasteful, but using it may found to be detrimental to the core model.

We claim that since our method use knowledge distillation as the aggregation method, it is robust to such delayed responses as knowledge distillation has both the attending trait and the retaining trait. In Figure 7, we show the results of the core model for three scenarios : 1. Omitting the delayed response, 2. Normal learning without any delays, 3. Using the delayed response once it has reached the core. In Case 3, an edge model model was dispatched to Edge 2 at time t, but was unable to return at the specified time. Hence, at time $t + 1$ an edge model was dispatched to another edge,

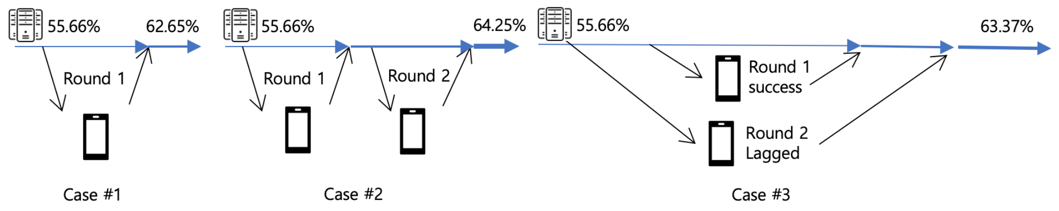
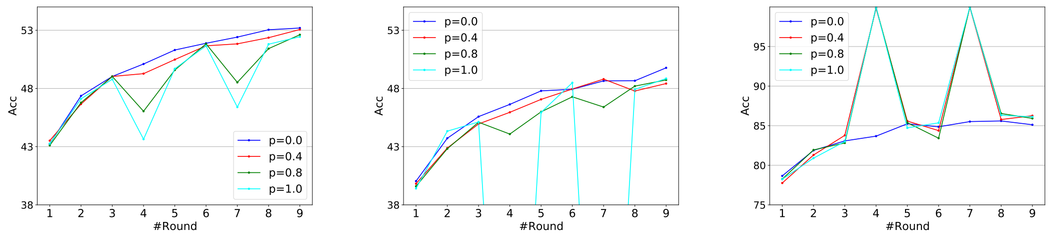


Figure 7: Diagram and results of using one core and two edges with lagged response on CIFAR-100. The results indicate that simply using the lagged response (63.37%) is beneficial than aborting it as in Case #1 (62.65%).



(a) Test accuracy of core model after # round

(b) Test accuracy of edge model after # round

(c) Train accuracy of core model after # round

Figure 8: Robustness of the method when noisy edge model is included on CIFAR-100.

Edge 1 and returned to the core model to distill knowledge. Then, the delayed response of Edge 2 returned and proceeded to the knowledge distillation process. Rather than having a detrimental effect, using the delayed response enhanced the core model's accuracy than omitting it (63.37 vs. 62.65), although it slightly fell short of the normal learning process (64.25).

5.3 Noisy Edge Problem

Anticipating malignant users that produce corrupted data is valid considering how much damage it can provoke to the federated learning process. Thereby in this experiment, we shuffle the input-label pairs of an entire edge batch with probability p. The edge model using such corrupted data and the subsequent core model, to which knowledge is distilled, will be harmed by this. The results shown in Figure 8 have noisy edge models on the 4th and 7th edge. As expected, we can observe from Fig. 8 (a,b) that the performance drastically drops for both the core model and the edge model after training from the corrupted labels. However, once a regular edge model joins the learning process to teach the core model, the core model of ours recovers its performance and nearly returns to the regular learning trajectory.

What is particularly interesting is the tendency of the *training accuracy* of the core sharply jumping when corrupted labels are used. These phenomena are typical cases of over-fitting to the corrupted labels. This implies that distilling false information induces the core model to memorize the inputs rather than learn semantic patterns via training, which can be supported by other findings (Zhang et al. (2016; 2017)). This observation hints at a possible scheme unique to our method to prevent malignant data from disturbing the federated learning process. Applying certain threshold to detect abnormally high training accuracy may be a possible key to detecting malignant dataset.

6 Conclusion

In this paper, we formulate a modified federated learning scenario and propose the asynchronous edge learning which is more suitable for situations characterized by dynamic inflow and outflow of edges. As a solution, we define a variant of KD that we call cloned KD. We explain how it works and where it is appropriate, and show that this is suitable for edge learning. We proposed an improved edge learning method through a method of ensemble and a method of using knowledge of past edge models together.

References

Yang Chen, Xiaoyan Sun, and Yaochu Jin. Communication-efficient federated deep learning with asynchronous model update and temporally weighted aggregation. *arXiv preprint arXiv:1903.07424*, 2019.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Large scale learning of general visual representations for transfer. *arXiv preprint arXiv:1912.11370*, 2019.

Daliang Li and Junpu Wang. Fedmd: Heterogenous federated learning via model distillation, 2019.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks, 2020.

Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, Yi Zhou, Ali Anwar, Shashank Rajamoni, Yuya Ong, Jayaram Radhakrishnan, Ashish Verma, Mathieu Sinn, et al. Ibm federated learning: an enterprise framework white paper v0. 1. *arXiv preprint arXiv:2007.10987*, 2020.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282. PMLR, 2017.

Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=BkluqlSFDS`.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10687–10698, 2020.

Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications, 2019.

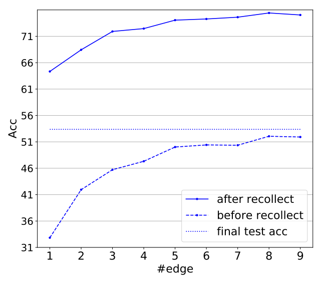
Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2016.

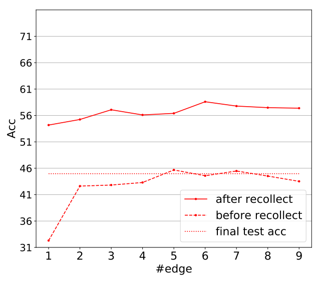
Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.

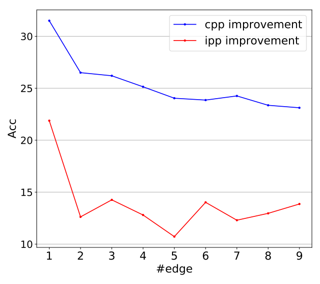
A Appendix



(a) CPP



(b) IPP



(c) Accuracy improvement after Phase 2 (after - before accuracy)

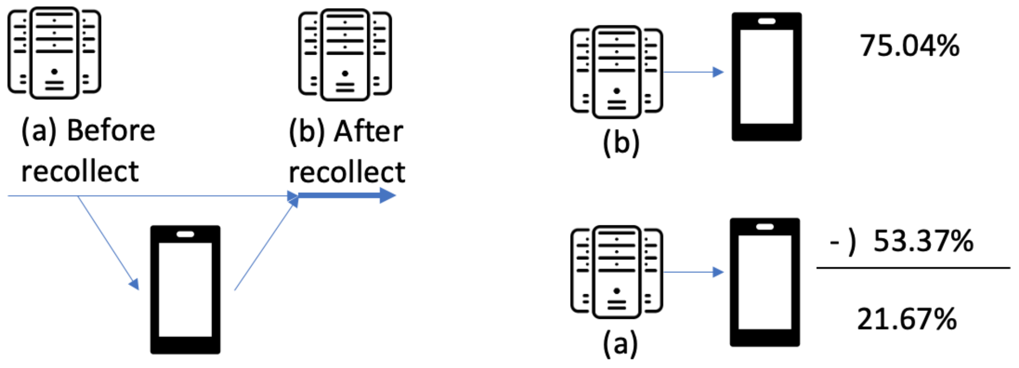


Figure 9: Experiment results quantifying how much the core model actually learns from the edge data information. The diagram below depicts how the score is computed. Those scores are calculated on the edge dataset. A core model shows improvement on edge dataset after re-collecting. Results on CIFAR-100 with ResNet-32.