Collaborative Inference via Ensembles on the Edge

Nir Shlezinger, Erez Farhan, Hai Morgenstern, and Yonina C. Eldar

ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9414740

***Abstract*—The success of deep neural networks (DNNs) as an enabler of artificial intelligence (AI) is heavily dependent on high computational resources. The increasing demands for accessible and personalized AI give rise to the need to operate DNNs on edge devices such as smartphones, sensors, and autonomous cars, whose computational powers are limited. Here we propose a framework for facilitating the application of DNNs on the edge in a manner which allows multiple users to collaborate during inference in order to improve their prediction accuracy. Our mechanism, referred to as *edge ensembles*, is based on having diverse predictors at each device, which can form a deep ensemble during inference. We analyze the latency induced in this collaborative inference approach, showing that the ability to improve performance via collaboration comes at the cost of a minor additional delay. Our experimental results demonstrate that collaborative inference via edge ensembles equipped with compact DNNs substantially improves the accuracy over having each user infer locally, and can outperform using a single centralized DNN larger than all the networks in the ensemble together.**

***Index terms*— Edge computing, deep ensembles, neural networks.**

I. Introduction

Deep learning systems have demonstrated unprecedented success in various applications, including computer vision and natural language processing [1]. Such deep neural networks (DNNs) consist of highly-parameterized models, trained using massive amounts of data. Consequently, deep learning is traditionally the domain of large scale computer servers, which posses the computational resources and the ability to aggregate the data required to store, train, and apply DNNs.

While DNNs are traditionally trained and maintained on powerful computer servers, the interface with the real-world is typically carried out using *edge devices*. The data used for inference, including images, text messages, and signal recordings, is obtained on the users side via devices such as smartphones, tablets, sensors, and autonomous vehicles. This motivates the implementation of DNNs on edge devices [2]. Applying DNNs as a form of mobile edge computing allows to infer on the same device where the data is collected, rather than having the samples sent to a centralized cloud server. Such DNN-aided edge device can operate at various connectivity conditions with reduced latency, as well as alleviate privacy issues and facilitate the personalization of artifical intelligence (AI) systems [3]. As a result, recent years have witnessed growing interest in implementing deep learning and applying DNNs on edge device [2]–[9].

One of the main challenges associated with implementing trained DNNs on edge devices stems from their limited computational resources [10]. A leading strategy to tackle this challenge is to make DNNs more compact. This is achieved by compressing existing highly-parameterized DNNs via pruning [11], [12] and quantization [13], [14] schemes, or alternatively, by designing DNN-aided systems to utilize compact networks by incorporating statistical model-based domain knowledge [15]–[18]. Such strategies focus on a single edge user, and thus do not exploit the fact that while each device is limited in its hardware, multiple users can collaborate to benefit from their joint computational resources. An alternative approach which accounts for the ability of edge devices to collaborate is based on partitioning and dividing a highly-parameterized DNN among multiple devices, which jointly form the large DNN during inference via computation offloading [2], [3], [19]. The main drawback of this approach is that each user cannot infer on its own, and the complete set of devices among which the DNN is divided must be present, resulting in high dependence on connectivity and possibly increased latency. This motivates designing DNN-aided edge devices in a manner which allows each user to infer on its own while supporting collaboration between different neighboring devices.

In this work we propose a framework for implementing DNNs on edge devices by allowing multiple users to form a *deep ensemble*. Deep ensembles are scalable DNN architectures comprised of a multitude of diverse deep predictors, which carry out inference by aggregating the individual predictions. Deep ensembles were empirically shown to achieve high accuracy and generalization performance in a manner which improves with the number of individual DNNs [20]–[26]. Here, we show that the concept of deep ensembles gives rise to a collaborative inference method for AI-empowered edge devices, referred to as *edge ensembles*, which allows users to benefit from the presence of other devices to improve accuracy while being operable in limited connectivity environments.

We present a decentralized collaborative inference protocol which allows each user to form a deep ensemble with its neighboring devices during inference. We analyze the delay associated with this procedure, showing that the parallel nature of deep ensembles in which all DNNs predict simultaneously results in a minimal additional delay compared to having each user infer locally at reduced accuracy. Our experimental study considers edge devices utilizing Mobilenet deep classifier [27] of a limited amount of weights operating with the CIFAR-10 data set. We numerically demonstrate that DNN-empowered edge devices can notably improve their accuracy by collaborating as an ensemble. Furthermore, we show that in some scenarios edge ensembles, in which each user has a limited amount of weights, can achieve improved accuracy over a single centralized DNN with more parameters than those of all the devices together.

The rest of this paper is organized as follows: In Section II, we detail the problem of DNN-based edge inference and review some basics in deep ensembles. Section III presents the edge ensembles mechanism and analyzes its latency, while experimental results are stated in Section IV. Finally, Section V provides concluding remarks.

Throughout the paper, we use boldface lower-case letters for vectors, e.g., $\boldsymbol{x}$; the ith element of $\boldsymbol{x}$ is written as $(\boldsymbol{x})_i$. calligraphic letters, such as $\mathcal{X}$, are used for sets, and $\mathcal{R}$ is the set of real numbers.

II. System Model

In this section we present the system model, under which our proposed edge ensembles protocol is derived in Section III. We begin by formulating the problem of DNN-based edge inference in Subsection II-A. Then we discuss the notion of collaborative inference and its associated requirements in Subsection II-B, and review some preliminaries in deep ensembles in Subsection II-C.

A. Problem Formulation

Consider a set of K AI-empowered mobile user devices. At a given time instance t, a user of index $i_t \in \{1,\ldots,K\} \triangleq \mathcal{K}$ gains access to data sample denoted $\boldsymbol{x}_{i_t}$, such as an image or an audio recording, and wishes to carry out inference, e.g., classification, based on this sample. In conventional deep learning, the user needs to communicate its sample to a centralized server, which maintains a pre-trained highly-parameterized DNN, which implements the inference task and

This project has received funding from the Benoziyo Endowment Fund for the Advancement of Science, the Estate of Olga Klein – Astrachan, the European Union's Horizon 2020 research and innovation program under grant No. 646804-ERC-COG-BNYQ, and from the Israel Science Foundation under grant No. 0100101. N. Shlezinger is with the School of ECE, Ben-Gurion University of the Negev, Beer-Sheva, Israel (e-mail: nirshl@bgu.ac.il). E. Farhan and H. Morgenstern are with BeyondMinds (e-mail: {erez.farhan; hai.morgenstern}@beyondminds.ai). Y. C. Eldar is with the Faculty of Math and CS, Weizmann Institute of Science, Rehovot, Israel (e-mail: yonina.eldar@weizmann.ac.il).

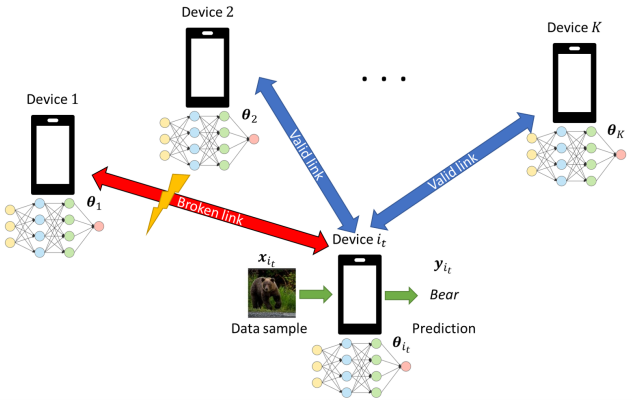

Fig. 1: DNN-based edge inference illustration.

conveys its result back to the user. The need to communicate with the centralized server gives rise to the challenges discussed in the introduction, such as latency, connectivity constraints, and privacy.

In mobile edge computing, the improved computational capabilities of modern edge devices are exploited to allow each user to maintain its local deep model. In particular, the user of index $i \in \mathcal{K}$ has access to a pre-trained model comprised of M parameters, represented by the vector $\boldsymbol{\theta}_i \in \mathcal{R}^M$. The users can communicate with each other, e.g., over a wireless communication channel. Due to the dynamic nature of the devices, the communication links may not be stable. Here, we adopt a simplified model for these links, allowing the channel between users $i, j \in \mathcal{K}$ to be either error-free or broken at each time instant t. Such setups represent, for example, AI-empowered vehicles which may operate in areas without connectivity to some centralized cloud server while being able to communicate in a device-to-device manner. An illustration of the resulting setup is depicted in Fig. 1.

Our goal is to characterize a mechanism and the corresponding DNN architecture for facilitating inference based on the sample $\boldsymbol{x}_{i_t}$, focusing on schemes which do not require connectivity with a central server. We use $\boldsymbol{y}_{i_t}$ and $\hat{\boldsymbol{y}}_{i_t}$ to denote the label value associated with $\boldsymbol{x}_{i_t}$ and its estimate produced by the set of AI-empowered mobile devices, respectively, while $\mathcal{L}(\cdot;\cdot)$ is the loss measure. Using these notations, we aim to design the DNNs such that at each time instance t, the user of index i_t recovers $\hat{\boldsymbol{y}}_{i_t}$ based on the loss $\mathcal{L}(\boldsymbol{y}_{i_t};\hat{\boldsymbol{y}}_{i_t})$.

B. Collaborative Inference Strategies

A straight-forward design is to train a single DNN with M parameters and have each device use it as its local model. This is done, e.g., when learning in a federated manner [7], where a single centralized DNN is trained using data available at a set of edge devices. Letting $\boldsymbol{f}_{\boldsymbol{\theta}_i}(\cdot)$ be the mapping carried out by the deep model $\boldsymbol{\theta}_i$, the estimate produced here is given by $\hat{\boldsymbol{y}}_{i_t} = \boldsymbol{f}_{\boldsymbol{\theta}_{i_t}}(\boldsymbol{x}_{i_t})$. This approach allows each user to infer without requiring any form of connectivity with the remaining devices, and avoids privacy issues as the users do not need to send their data to a centralized server for inference. The drawback of this approach is that while modern edge devices have improved computational capabilities, their compute resources are still limited compared to powerful centralized servers, and thus large networks typically need to be pruned or quantized to be applied on edge devices [11]–[14]. As a result M is typically small compared to the amount of parameters used in conventional centralized DNN, implying that the inference accuracy may be degraded.

An alternative strategy which exploits the ability of the users to collaborate during inference is based on partitioning a pre-trained highly-parameterized DNN among multiple users. Here, the computation of $\hat{\boldsymbol{y}}_{i_t}$ is divided among all the users, such that the resulting estimate corresponding to the output of a highly-parameterized DNN with $K \cdot M$ weights. For instance, $\boldsymbol{\theta}_i$ can be set to be the ith layer of a DNN with K layers, and thus setting $\hat{\boldsymbol{y}}_{i_t} = \left(\boldsymbol{f}_{\boldsymbol{\theta}_K} \circ \cdots \circ \boldsymbol{f}_{\boldsymbol{\theta}_1}\right)(\boldsymbol{x}_{i_t})$ implements highly-parameterized DNN-based inference of potentially improved performance [2, Sec. IV]. Nonetheless, this strategy, referred to as *sequential offloading*, involves increased delay, due to the need to operate in a sequential manner, and requires connectivity among all users. Going back to the AI-aided vehicles example, this implies that the same group of vehicles must be located in the same physical area whenever one has to carry out inference.

These relatively intuitive approaches reveal some key guidelines: First, in order to allow operation without any connectivity, each model $\boldsymbol{\theta}_i$ must correspond to a dedicated DNN capable of inferring on its own. Nonetheless, in order to benefit from collaboration among devices during inference, these individual DNNs should be diverse, e.g., if $\boldsymbol{\theta}_i = \boldsymbol{\theta}_j$ for some $i \neq j$ then users i and j have no added value in collaborating. This motivates utilizing architectures based on deep ensembles [21], briefly reviewed in the following subsection.

C. Deep Ensembles

Ensemble methods utilize multiple models whose outputs are combined to achieve improved results [28]. Deep ensembles utilize DNNs as the individual models. Here, during inference, the input sample is processed by each of these DNNs in parallel, and their outputs are aggregated into a single prediction [21]. Various techniques are proposed in the literature for ensemble aggregation, depending on the overall task, including averaging for regression and classifiers with soft outputs, as well as majority vote for hard-decision classifiers [21]. Since these aggregation methods can be applied with different numbers of models, deep ensembles are inherently scalable [20], which is desirable for edge-based collaborative settings.

Deep ensembles require the individual models to be diverse, namely, while each predictor is designed for the same task, their individual mapping is different. Various measures have been proposed for quantifying such diversity [29], [30], and different learning algorithms have been suggested for training diverse models, see, e.g., [23], [24], [26], [31]. When such diversity holds, ensemble methods are known to achieve improved accuracy over that of the individual models, in a manner which increases with the number of models [21].

III. Edge Ensembles

Here, we present the proposed strategy of edge ensembles for collaborative inference. Unlike offloading-based schemes, our approach is scalable and allows local inference, while exploiting the presence of collaborative devices to improve accuracy. We begin by presenting the proposed inference mechanism and its underlying rationale in Subsection III-A, after which we characterize its associated delay and provide a discussion in Subsections III-B-III-C, respectively.

A. Edge Ensemble Inference Protocol

The core idea of edge ensembles is to provide each user the ability to carry out inference on its own, while allowing to benefit from collaboration with neighboring devices by forming together an ensemble of DNNs. This is achieved by setting the set of local models $\{\boldsymbol{\theta}_i\}$ to represent a diverse deep ensemble. Under such a setup, each individual user can utilize its local model for inference, achieving possibly limited accuracy due to the fact that it is comprised of a compact network with a relatively small number of parameters. Nonetheless, multiple users which are capable of communicating with each other can now collaborate in inference as a form of AI-based wisdom of crowds [32]. In particular, the scalable nature of deep ensemble aggregation functions, which can operate with varying number of participating predictors, allows edge ensembles to operate in dynamic and possibly unstable communication conditions, without being dependent on the participation of any specific device in the inference procedure. This operation allows each user to operate in

limited-connectivity environments. Furthermore, such edge ensembles translate the presence of neighboring devices into substantial accuracy improvements over using a single model, as we numerically demonstrate in Section IV, while having each device maintain a local compact DNN with a limited number of parameters.

To formulate the inference procedures of edge ensembles, we define the set of random variables (RVs) $\{C_{i,j}^t\}$ representing the status of the communication links between the users at time instance t. In particular, $C_{i,j}^t$ which equals one when users i and j can communicate at time instance t and zero when they cannot. We assume reciprocal channels, such that $C_{i,j}^t = C_{j,i}^t$, and set $C_{i,i}^t \equiv 1$. We further define $\mathcal{S}_i^t$ as the set of users with which user i can reliably communicate with at time t, i.e.,

$$\mathcal{S}_i^t \triangleq \{j \in \mathcal{K} | C_{i,j}^t \neq 0\}. \tag{1}$$

By definition, $i \in \mathcal{S}_i^t$, and thus $|\mathcal{S}_i^t| \in \{1, \ldots, K\}$. The set $\mathcal{S}_i^t$ represents the edge ensemble model available to user i at time instance t, as we detail in the following.

As stated in the problem formulation in Subsection II-A, the inference stage begins at a given time instance t, when all deep models are pre-trained, and a user of index i_t observes a data sample $\boldsymbol{x}_{i_t}$, to be used for inference. The user i_t then broadcasts the sample $\boldsymbol{x}_{i_t}$, resulting in it being available to all users in the set $\mathcal{S}_i^t$. Next, each user in $\mathcal{S}_i^t$ applies its local model to that sample, and conveys the resulting $\boldsymbol{f}_{\boldsymbol{\theta}_j}(\boldsymbol{x}_{i_t})$ back to user i_t, which aggregates them into the predicted $\hat{\boldsymbol{y}}_{i_t}$. In particular, for classification tasks with N different labels, the output of each individual network $\boldsymbol{f}_{\boldsymbol{\theta}_j}(\boldsymbol{x}_{i_t})$ is an $N \times 1$ vector whose entries are an estimate of the conditional distribution of each label given $\boldsymbol{x}_{i_t}$, and the predicted $\hat{\boldsymbol{y}}_{i_t}$ can be obtained as the label which maximizes the averaged conditional distribution [24], i.e.,

$$\underset{n=1,\ldots,N}{\arg\max} \frac{1}{|\mathcal{S}_i^t|} \sum_{j \in \mathcal{S}_i^t} \left(\boldsymbol{f}_{\boldsymbol{\theta}_j}(\boldsymbol{x}_{i_t})\right)_n. \tag{2}$$

The resulting inference procedure is summarized as Algorithm 1.

Algorithm 1: Collaborative Inference of Edge Ensembles

Input: Data sample $\boldsymbol{x}_{i_t}$ observed at time t at user i_t.

1 User i_t broadcasts $\boldsymbol{x}_{i_t}$ to its neighboring devices;

2 **for** *each* $j \in \mathcal{S}_{i_t}^t$ **do**

3 User j computes $\boldsymbol{f}_{\boldsymbol{\theta}_j}(\boldsymbol{x}_{i_t})$;

4 User j sends the result to user i_t;

5 **end**

6 User i_t aggregates $\{\boldsymbol{f}_{\boldsymbol{\theta}_j}(\boldsymbol{x}_{i_t})\}$ into $\boldsymbol{y}_{i_t}$ via, e.g., (2);

Output: Predicition $\hat{\boldsymbol{y}}_{i_t}$.

B. Inference Latency Analysis

Latency is one of the motivations for utilizing AI-empowered edge devices over conventional cloud server-based DNNs. Since our proposed collaborative inference algorithm obviously comes at increased delay compared to having the users infer only locally, due to the communication overhead and system heterogeneity in the devices, we next analyze the statistical properties of the overall inference latency of Algorithm 1. Due to page limitations, our results are given here without proofs.

To that aim, we first introduce the following symbols: Let $\{T_{i,j}^t\}$ be a set of RVs representing the *communication delay* between a pair of distinct users. Namely, $T_{i,j}^t$ is the sum of the duration needed for communicating a sample $\boldsymbol{x}$ from user i to user j, and that required for conveying $\boldsymbol{f}_{\boldsymbol{\theta}_j}(\boldsymbol{x})$ back to user i at time slot t. By definition, $T_{i,i}^t = 0$ with probability one. Further, we use τ_j to denote the computation time required by user j to apply its local model, i.e., evaluate $\boldsymbol{f}_{\boldsymbol{\theta}_j}(\boldsymbol{x})$. Since the computation times $\{\tau_j\}$ are dictated by the hardware of each device, which is static, they are assumed here to be deterministic quantities independent of the time instance t, as opposed to the random communication delays $\{T_{i,j}^t\}$. Using these notations, we can define the overall delay induced by the collaborative inference procedure at time t in Algorithm 1 as the RV Δ^t, given by

$$\Delta^t \triangleq \max_{j \in \mathcal{K}} \left(C_{i_t,j}\left(\tau_j + T_{i_t,j}^t\right)\right). \tag{3}$$

The formulation of the inference delay in (3) allows us to characterize its distribution for i.i.d. links, as stated in the following theorem:

Theorem 1. *When the communication links RVs $\{C_{i,j}^t\}_{i \neq j}$ are i.i.d. with $p \triangleq \Pr(C_{i,j}^t = 1)$, and the communication delays $\{T_{i,j}^t\}_{i \neq j}$ are i.i.d. with cumulative distribution function given by $F_T(\epsilon) \triangleq \Pr(T_{i,j}^t < \epsilon | C_{i,j}^t = 1)$, then the overall delay RV Δ^t satisfies*

$$\Pr\left(\Delta^t < \epsilon\right) = \begin{cases} 0 & \epsilon \leq \tau_{i_t}, \\ \prod_{j \neq i_t}\left(1 - p\left(1 - F_T(\epsilon - \tau_j)\right)\right) & \epsilon > \tau_{i_t}. \end{cases} \tag{4}$$

Theorem 1 reveals some of the statistical properties of the overhead induced by collaborative inference via Algorithm 1. For instance, consider the case in which the communication delays are upper bounded, i.e., there exists a finite $T_{\max}$ such $F_T(T_{\max}) = 1$ and $F_T(\epsilon) < 1$ for each $\epsilon < T_{\max}$. Here, it immediately follows that the delay Δ^t is not larger than $T_{\max} + \max_j \tau_j$, which indicates that the resulting delay is smaller than that obtained using edge computing based on sequential offloading with [3]. This follows from the fact the computations of each of the DNNs in a deep ensemble is carried out in parallel, without requiring any of the users to wait for another device to finish its local computation before it begins. Nonetheless, Theorem 1 also indicates that the delay is likely to increase as the number of users grow, as noted in the following corollary:

Corollary 1. *When the computation time is identical among all the devices, i.e., $\tau_j \equiv \tau$ for each $j \in \mathcal{K}$, and the communication delays are upper bounded by $T_{\max}$, and $p > 0$, then for each $\epsilon < \tau + T_{\max}$,*

$$\lim_{K \to \infty} \Pr\left(\Delta^t < \epsilon\right) = 0. \tag{5}$$

Corollary 1 implies that the aforementioned upper bound on the overall latency of $T_{\max} + \max_j \tau_j$ becomes tight as the number of users grow under identical computation times. Still, this upper bound is typically smaller than the overall delay induced when utilizing sequential offloading, which involves more computation and communication iterations for carrying out inference. Specifically, since the duration of inferring locally, which involves the smallest latency, is τ, Corollary 1 suggests that the only additional latency in that case is the maximal one induced by the single communication round in Steps 1 and 3 of Algorithm 1, i.e., $T_{\max}$.

C. Discussion

Edge ensembles facilitates the implementation of deep learning on edge devices by accounting for their unique properties and requirements. In particular, Algorithm 1 is tailored to account for the limited hardware and computation capabilities of such devices, combined with their mobile dynamic nature and ability to collaborate in a decentralized manner. In fact, edge ensembles treats AI-empowered mobile devices as a crowd of diverse intelligent individuals: each agent is capable of inferring on its own, and can thus operate solely without any connectivity. However, a group of users can infer more reliably in a collaborative manner as a form of (artificial) wisdom of crowds [32]. Specifically, this collaboration boils down to an equivalent deep ensemble, which is an established concept in the deep learning literature [21] known to improve accuracy and robustness.

Collaboration based on deep ensembles is fundamentally different from task offloading mechanisms, which facilitate complex DNN-based computations over a set of edge devices by partitioning a DNN

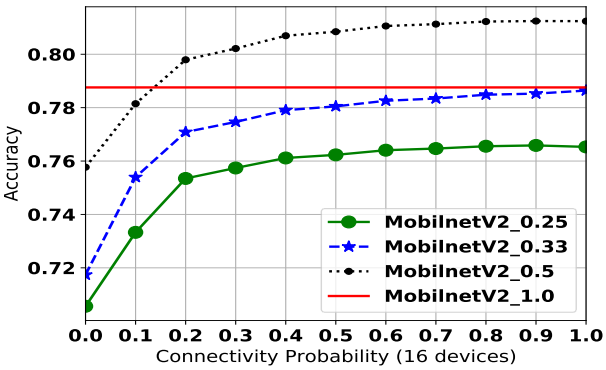

Fig. 2: Accuracy versus connectivity probability.

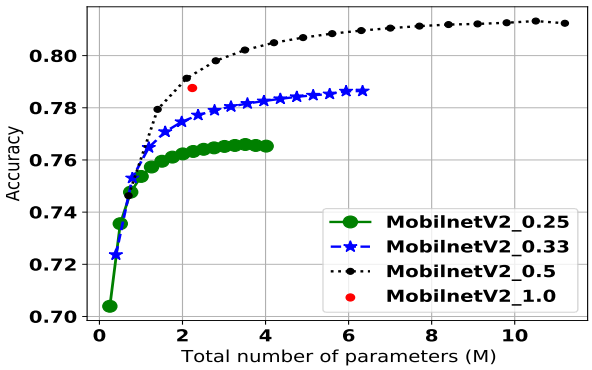

Fig. 3: Accuracy versus overall parameters.

and providing each user with a different partition [3]. Specifically, task offloading requires all the users to participate in order to carry out inference, and thus requires constant connectivity, while typically inducing increased latency when the DNN partitions must be applied in a sequential manner, e.g., when each partition is a set of layers in a feed forward network [2, Sec. IV]. Edge ensembles do not rely on connectivity, as each user can infer on its own, and collaboration is applicable with arbitrary number of users.

As Algorithm 1 forms a deep ensemble, the more diverse users participate, the more accurate the prediction is expected to be [28]. This comes at the probable cost of increased latency, as noted in Subsection III-B. The accuracy can be thus further improved by allowing more users to participate, e.g., by extending Algorithm 1 to support multi-hop communications as in [33], again at the cost of increased latency. Furthermore, the communication delay can be reduced by compressing the samples, thus conveying smaller volumes of data. We leave the study of these extensions to future work.

Designing separate DNN-aided edge devices to form a deep ensemble gives rise to several challenges. For once, the individual DNNs have to be different from one another in order to benefit from collaboration [30]. This can be achieved by either training the DNNs jointly while boosting diversity, e.g., by using a regularized objective as in [24], [31] or different randomized initialization [26]. Alternatively, this can be achieved by modifying distributed learning algorithms to result in different individual networks [34]. An additional challenge stems from the need of each user to share its sample with other devices, giving rise to privacy concerns. Finally, the participation of multiple devices in inference induces security issues as malicious users may affect the prediction result, which can be possibly tackled using Byzantine-robust aggregation methods [35]. These challenges of edge ensembles are left for future research.

IV. Experimental Study

In this section we numerically evaluate the accuracy of edge ensembles collaborating via Algorithm 1. To that aim, we utilize the MobilenetV2 DNN architecture [27] for image classification based on the CIFAR-10 dataset. Our baseline centralized model is a MobilenetV2 DNN with width factor one, which has $2.23 \cdot 10^6$ trainable parameters and is trained from random initialization (no pre-training). For the edge devices, we train a set of diverse MobilenetV2 DNNs with width factors of $\frac{1}{4}$, $\frac{1}{3}$, and $\frac{1}{2}$, for which the number of parameters is $\{2.51, 3.96, 7\} \cdot 10^5$, respectively. The edge networks are trained using the same dataset and training algorithm, but with different random weight initialization in order to achieve diverse individual networks [26]. These individual edge devices then collaborate during inference via Algorithm 1 where the aggregation is computed via (2).

We first consider a scenario comprised of $K = 16$ devices in which $\{C^t_{i,j}\}$ are i.i.d. with connectivity probability $p \in [0, 1]$, i.e., $\Pr(C^t_{i,j} = 1) = p$ for each $i \neq j$. For this setup, we evaluate how the accuracy of edge ensembles grows with the connectivity probability p. The resulting accuracy versus number of devices compared to the baseline centralized full MobilnetV2 (abbreviated *MobilenetV2_1.0*) is depicted in Fig. 2. Observing Fig. 2 we note that collaboration among edge devices substantially improves accuracy, while allowing each device to infer solely. For example, having each device utilize a DNN with width factor $\frac{1}{2}$, which has roughly 30% the number of weights as MobilenetV2_1.0, allows each user to achieve inference accuracy of 75% (i.e., for $p = 0$), compared to 79% achieved by the full model. However, having multiple devices collaborate via Algorithm 1 improves the accuracy to 80% when the users can communicate with probability as low as 20%. It is also observed in Fig. 2 that even when each device has a model as compact as comprising of 6% of the overall number of parameters, which is the case for width factor of $\frac{1}{4}$, the resulting accuracy still improves when p grows, namely, by collaborating via Algorithm 1, though the collaborating users are unable to outperform the centralized full model, reaching an accuracy of less than 77% for $p = 1$.

Next, we evaluate how the accuracy scales with respect to the overall number of parameters used for inference, which dictates the capacity of the DNN. The results, depicted in Fig. 3, illustrate that collaborative edge ensembles can outperform a centralized model with more parameters than that used by all the participating users together. In particular, we observe that three edge devices equipped with diverse trained MobilnetV2 DNN with width factor $\frac{1}{2}$, whose overall number of weights is $2 \cdot 10^6$, outperform the centralized MobilenetV2_1.0, which has 10^5 additional parameters. These results indicate that collaborative inference of edge ensembles does not only facilitate inference of DNN-based hardware-limited edge devices, but in some scenarios can also allow to achieve improved accuracy with less parameters compared to deep centralized models.

V. Conclusions

In this paper we proposed a DNN-based mobile edge collaborative inference scheme via the joint forming of deep ensembles. Our proposed edge ensemble mechanism allows AI-empowered edge devices to operate at limited connectivity setups, as well as achieve improved accuracy by collaboration with neighboring devices. We have characterized the inference latency of this strategy, showing that it results in a minor overhead compared to local inference. Our numerical tests demonstrate that such a collaboration allows a group of users equipped with compact DNNs to achieve improved accuracy by collaboration, indicating the potential of the proposed strategy in facilitating accurate inference on hardware-limited edge devices.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[2] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[3] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 2017.

[4] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[5] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 2019.

[6] H. B. McMahan, E. Moore, D. Ramage, and S. Hampson, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, 2020.

[8] N. Shlezinger, M. Chen, Y. C. Eldar, H. V. Poor, and S. Cui, "UVeQFed: Universal vector quantization for federated learning," *IEEE Trans. Signal Process.*, 2021.

[9] T. Sery, N. Shlezinger, K. Cohen, and Y. C. Eldar, "COTAF: Convergent over-the-air federated learning," in *Proc. IEEE GLOBECOM*, 2020.

[10] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y. Shi, "Scaling for edge inference of deep neural networks," *Nature Electronics*, vol. 1, no. 4, pp. 216–222, 2018.

[11] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

[12] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1389–1397.

[13] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Benini, and L. V. Gool, "Soft-to-hard vector quantization for end-to-end learning compressible representations," in *Advances in Neural Information Processing Systems*, 2017, pp. 1141–1151.

[14] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[15] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "ViterbiNet: A deep learning based Viterbi algorithm for symbol detection," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3319–3331, 2020.

[16] ——, "Learned factor graphs for inference from stationary time sequences," *arXiv preprint arXiv:2006.03258*, 2020.

[17] N. Farsad, N. Shlezinger, A. J. Goldsmith, and Y. C. Eldar, "Data-driven symbol detection via model-based machine learning," *arXiv preprint arXiv:2002.07806*, 2020.

[18] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," *arXiv preprint arXiv:2012.08405*, 2020.

[19] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, 2019.

[20] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in neural information processing systems*, 2017, pp. 6402–6413.

[21] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[22] H. Chen and A. Shrivastava, "Group ensemble: Learning an ensemble of convnets in a single convnet," *arXiv preprint arXiv:2007.00649*, 2020.

[23] H. Li, J. Y.-H. Ng, and P. Natsev, "Ensemblenet: End-to-end optimization of multi-headed models," *arXiv preprint arXiv:1905.09979*, 2019.

[24] B. Brazowski and E. Schneidman, "Collective learning by ensembles of altruistic diversifying neural networks," *arXiv preprint arXiv:2006.11671*, 2020.

[25] T. Raviv, N. Raviv, and Y. Be'ery, "Data-driven ensembles for deep and hard-decision hybrid decoding," *arXiv preprint arXiv:2001.06247*, 2020.

[26] S. Fort, H. Hu, and B. Lakshminarayanan, "Deep ensembles: A loss landscape perspective," *arXiv preprint arXiv:1912.02757*, 2019.

[27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[28] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[29] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural networks*, vol. 12, no. 10, pp. 1399–1404, 1999.

[30] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine learning*, vol. 51, no. 2, pp. 181–207, 2003.

[31] C. Shui, A. S. Mozafari, J. Marek, I. Hedhli, and C. Gagné, "Diversity regularization in deep ensembles," *arXiv preprint arXiv:1802.07881*, 2018.

[32] J. Surowiecki, *The wisdom of crowds*. Anchor, 2005.

[33] A. Cohen, N. Shlezinger, S. Salamatian, Y. C. Eldar, and M. Médard, "Serial quantization for sparse time sequences," *arXiv preprint arXiv:1907.01691*, 2019.

[34] N. Shlezinger, S. Rini, and Y. C. Eldar, "The communication-aware clustered federated learning problem," in *Proc. IEEE ISIT*, 2020.

[35] A. Portnoy and D. Hendler, "Towards realistic Byzantine-robust federated learning," *arXiv preprint arXiv:2004.04986*, 2020.