Globe2Train: A Framework for Distributed ML Model Training using IoT Devices Across the Globe

Bharath Sudharsan*, John G. Breslin*, and Muhammad Intizar Ali§
*Confirm SFI Research Centre for Smart Manufacturing, Data Science Institute, NUI Galway, Ireland
{bharath.sudharsan, john.breslin}@insight-centre.org
§School of Electronic Engineering, Dublin City University, Ireland, ali.intizar@dcu.ie

***Abstract*—Training a problem-solving Machine Learning (ML) model using large datasets is computationally expensive and requires a scalable distributed training platform to complete training within a reasonable time frame. In this paper, we propose a novel concept where, instead of distributed training within a GPU cluster, we train one ML model by utilizing the idle hardware of numerous resource-constrained IoT devices existing across the globe. In such a global setting, staleness and real-world network uncertainties like congestion, latency, bandwidth issues are proven to impact the model convergence speed and training scalability. To implement the novel concept, while simultaneously addressing the real-world global distributed training challenges, we present Globe2Train (G2T), a framework with two components named G2T-Cloud (G2T-C) and G2T-Device (G2T-D) that can efficiently connect together multiple IoT devices and collectively train to produce the target ML models at very high speeds. The evaluation results with analysis show how the framework components jointly *eliminate staleness* and *improve training scalability and speed* by tolerating the real-world network uncertainties and by reducing the communication-to-computation ratio.**

***Index Terms*—Distributed Machine Learning, IoT Devices, Scalable Model Training, Latency Tolerance.**

I. Introduction

Nowadays, IoT datasets are being published at an explosive rate since the modern ML libraries, tools, and frameworks enabled deep compression, deployment, and execution of autonomous data collection plus data annotation algorithms on IoT devices [6]–[8]. To train ML models in a reasonable time using large-high-quality datasets gathered over decades, the generally practiced approach of loading datasets and training a model within a GPU cluster or a data center is traditional, slow and might consume days or even weeks for the model to converge. Hence, we propose the concept of a scalable distributed training system that efficiently harnesses the hardware resource of millions of idle IoT devices [9]–[11] without disturbing their routine and collectively train to produce the target ML models at very high speeds.

The Globe2Train (G2T) framework that we present in this paper implements our novel concept by enabling the distributed training of one ML model on numerous geographically separated IoT devices. In this work, like in [12, 13], we do not aim to collectively train a model without centralizing the data, which is similar to Federated Learning (FL). We instead aim to improve model convergence speed and training scalability by providing G2T. However, existing large-scale federated, decentralized learning works do not scale well under networks with high network latency and congestions [1], which can be addressed by the G2T components.

We claim that our concept of distributed training of one ML model on the hardware of thousands of IoT devices will become the *future of ML and IoT* because; (i) In the GPUs to IoT devices ratio, the IoT devices hold a much greater proportion. i.e., market estimates show that roughly 50 billion MCU and small CPU chips were shipped in 2020, which far exceeds other chips like GPUs (only 100 million units sold); (ii) Every modern household does not compulsorily own a GPU, yet it roughly has around a dozen of IoT devices which when efficiently connected together using G2T can locally, within their home network train mid-sized ML models without depending on Cloud or GPU servers that can perform the same training task at very high speeds, but at an additional expense; (iii) In most real-life IoT scenarios, the training dataset that produces problem-solving models when used, can often be hard to source (GDPR and privacy concerns). In such cases, when G2T is deployed across the devices that have access to the same type of rich training data, it can collectively train a model without storing the live data. Thus, producing problem-solving models without voiding the privacy protection regulations; (iv) Training advanced ML models on a single GPU might consume days or even weeks to converge. Here, the G2T can tolerate high latency, low bandwidth constraints and collectively harness the idle hardware resource of thousands of mid-sized IoT devices and complete training at extremely high speeds. For example, at the time of writing, the latest GEFORCE RTX 2080 Ti GPU has 11 GB RAM but costs ≈1500 $. Whereas one Alexa smart speaker has 2 GB RAM and efficiently connecting 20 can collectively pool 40 GB of RAM. In this way, we can complete training faster than the expensive GPU and at a 0 $ investment since millions of IoT devices already exist globally, and most of them are idle. The contribution of this paper can be summarized as follows:

- We present G2T framework that brings into practice the novel concept of global distributed ML model training

TABLE I
COMPARISON OF GLOBE2TRAIN FRAMEWORK WITH THE MOST RELATED PAPERS.

Paper	Experimentation hardware	Training reliability	On-device algorithm	MCUs & small CPUs support	Staleness	Uncertainties toleration	Scalability	Parameters compression
[1]	Distributed AWS resources	Not Investigated (NI), server don't fail	Non Comparable (NC)	✗	Reduced	✓	Few distributed servers	Delayed updates
[2]	Multiple GPUs	NI, GPUs in cluster also don't fail	NC	✗	NC	✗	Hundreds of GPUs in a cluster	✗
[3]	Mobiles	NI	ML framework-based	✗	Reduced	✓	Millions of mobiles	Sparse updates
[4]	Edge servers, Rpis	Not Available (NA)	Cannot run on MCUs	✗	NI	✗	NC	Pruning, Quant
[5]	Laptops, Rpis	NA	Cannot run on MCUs	✗	NI	✗	NC	✗
G2T (ours)	MCUs that are widely used to design IoT devices	G2T-C counter balances for device/network failures	G2T-D is designed for resource-friendly training, transmission even in poor network	All chipsets in Arduino IDE, Atmel studio, ARM Keil MDK	G2T-C is designed not to face staleness	Tolerates un avoidable latency, congestion, bandwidth issues	Thousands of distributed heterogeneous IoT devices	Resource-friendly shrinking by G2T-D to reduce bandwidth

on the hardware of numerous, geographically separated, resource-constrained idle IoT devices. The characteristics of the G2T framework components enable the completion of ML model training within a reasonable time frame while addressing challenges raised due to staleness and real-world network uncertainties in the distributed global training settings.

- We provide G2T-C as the cloud-level component designed to connect numerous IoT devices and efficiently handle the entire distributed training process by decomposing heavy ML training workloads into multiple lightweight tasks that can be comfortably handled by training involved IoT devices. G2T-C ensures *reliable and timely training completion* while *eliminating the staleness issue*.
- We provide G2T-D as the device-level, resource-friendly component designed to efficiently transmit the IoT device trained weights even under poor network conditions. G2T-D *improves training scalability, speed* while addressing the network uncertainty issues in distributed global training by *reducing the communication-to-computation ratio and tolerating the real-world latency, bandwidth issues*.

Outline. Section II briefs the essential concepts, highlights research challenges and related studies. Section III presents the Globe2Train (G2T) framework. Section IV evaluates G2T and discusses how it addresses distributed global training research challenges. Section V concludes with an outline of future work.

II. BACKGROUND AND RELATED WORK

In Table I, we compare G2T with the most related work. In the remainder, we briefly present essential concepts, research challenges, then compare ours with state-of-the-art approaches.

A. Machine Learning using Decentralized Data

Modern IoT devices generate and have access to a wealth of data that can be used to produce powerful models. Often, such rich data are large in quantity, privacy-sensitive, or most time both, thus restricting transmitting them to data centers and training using advanced ML frameworks on GPU clusters [14]. As the awareness of data privacy is growing rapidly and also since IoT apps are constantly being monitored for GDPR compliance, companies are following the FL approach where models are trained close to the data source. A few popular examples of training without data centralization are: data across numerous hospitals were used to train models for medical treatments [12], patient survival situations across 3 countries were analyzed [13]. However, the FL advancements are not of much use since the scalability is poor in high latency networks [15].

Decentralized learning is an orthogonal exploration where methods like D2 [16], AD-PSGD [17] perform partial synchronization in each update to escape latency issues. Such large-scale training takes advantage of data parallelism and by increasing the count of contributing devices, but at the cost of transmitting parameters, which is expensive, especially when multiple devices are pooled. This results in dwarfing the savings in computation time and producing a low computation-to-communication ratio [18]. However, again, such learning approaches do not scale under high network latency [19]. Also, face other critical problems due to lower network bandwidth, expensive mobile data plan, intermittent network connection, which is all common in the distributed, global ML scenario. We designed G2T components to jointly work and ensure the real-world network issues do not become a significant bottleneck when scaling up distributed training.

B. Parameter Quantization/Compression

When the parameters like gradients, weights transferred during distributed training are reduced, the communication bandwidth will also reduce [20]. The 1-bit gradient [21] achieved a 10× speedup by reducing the data transfer size. The QSGD approach [22] provides a method to trade off accuracy for gradient precision, and similarly, Terngrad [18] use 3-level gradients based approach. Also, attempts have been made to quantize the entire model including the parameters. e.g., DoReFa-Net [23] uses 2-bit gradients and 1-bit weights.

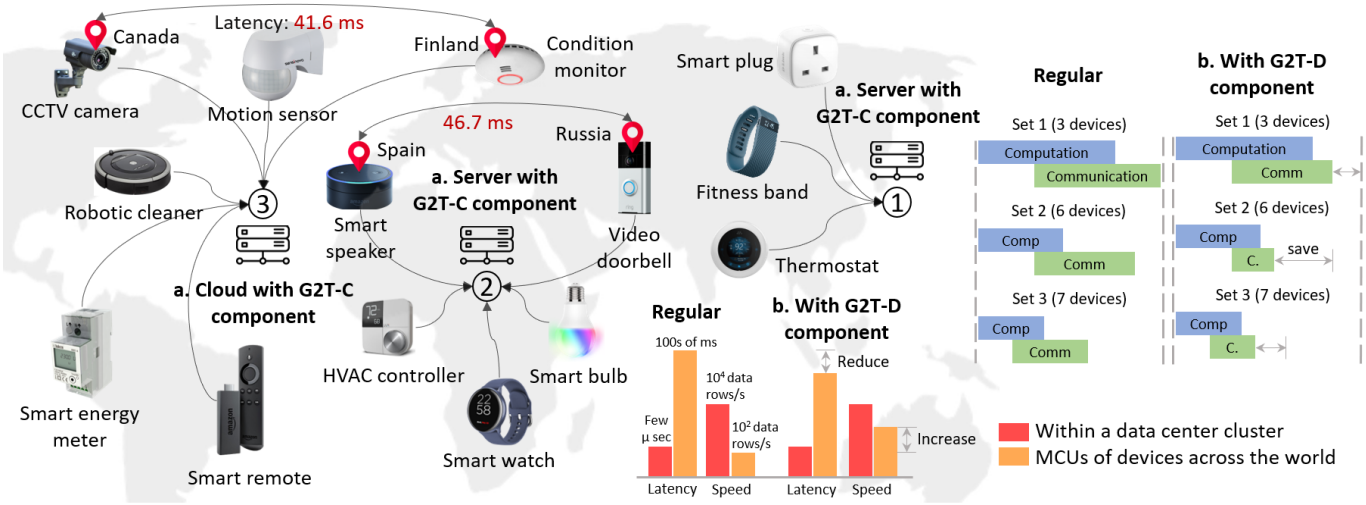

Fig. 1. The Globe2Train (G2T) framework enables distributed ML model training using IoT devices across the globe: a. The framework's G2T-Cloud (G2T-C) component is deployed on a central server/cloud. b. The G2T-Device (G2T-D) component is deployed on training involved IoT devices.

The current state of literature presents schemes that are applicable to guarantee the convergence of only NNs. i.e., TernGrad and DoReFa-Net studied CNNs and QSGD studied the RNNs training loss. The theoretical quantization limit cannot exceed 32. To overcome this, parameter sparsification techniques are proposed. The user predefined threshold-based sparsification [24], the adaptive compression ratio [25] shows that, with only negligible performance degradation, 99% gradients can be pruned. In [26], quantization, compression are combined to achieve next-level optimization ratio. However, such advancements cannot solve the latency issue, especially when the contributing devices are geographically separated [19]. When comparing G2T with the above, we are the first to present a method to distributedly train a multi-class classifier (unlike NNs) on a variety of globally deployed IoT devices (unlike within data centers) while tolerating latency, conserving bandwidth by shrinking parameters, and setting transmission frequency depending on network quality.

C. Distributed Global Training: Research Challenges

The global training scenarios/setups can be impacted by real-world network uncertainties and staleness.

Network Uncertainties: Congestion and Latency Variance. The access links that connect IoT devices to the parameter server show uncertainties such as network unavailability, time-varying connectivity, etc. For instance, consider the use of SSGD during a distributed training process, where only one gradient transmission occurs in one iteration. This aspect can worsen with an increase in the number of transmissions and if the previously sent gradients arrive late along with recent gradients (due to network congestion). The second expected issue is the large variance in latency, which is common in real-world IoT networks. While we can aim to maintain a low average latency by choosing and involving only IoT devices with a stable internet connection, changes in device network connectivity due to mobility (e.g., when the IoT device is placed in a car) can cause variable latency.

Staleness Effect. Most popular distributed model training algorithms adopt non-synchronous execution for alleviating the network communication bottleneck that produces *stale* parameters. i.e., the model parameters arrive late, not reflect the latest updates. Staleness slows down convergence and also degrades model performance. Despite the notable contributions [3, 27] in the large-scale ML domain, the effects of staleness during training does not lead to a firm conclusion [15] because it is practically not feasible to monitor and control staleness in the current complex distributed environments containing heterogeneous devices with varying network landscapes.

In [19], we presented in detail the other bottlenecks such as expensive dataset loading I/O, high FLOPs consumption, variable model training, and convergence speed, etc.

D. Machine Learning Model Training on IoT Devices

Training models on GPUs and data center servers are much efficient and provide high-quality models, making MCUs the least preferred choice for offloading any ML-related tasks. Practicing decentralized learning close to the data source has become a common approach to preserve privacy and comply with the GDPR. Since sensitive personal data is generated by the billions of daily used tiny IoT devices [28], there is a pressing need to train models on their resource-constrained hardware, then transmit the learned information (model weights). The existing frameworks like Tensorflow Micro, Keras, Edge-ML, Open-NN, etc. do not yet provide methods to enable training models on MCUs of IoT devices [29]. Currently, to achieve resource-efficient training on MCUs, authors have optimized training algorithms to run on various

resource-constrained setups, which we describe in the rest of this section.

In [30], a Gaussian Mixture Model was executed on an embedded board aiming to re-train an ML algorithm at the edge level. Articles [31, 32] present optimized methods to enable training models on smartphones. Multiple Federated Learning algorithms [33, 34] enable fine-tuning global models offline, at the edge level using local datasets. SEFR, a low-power classifier [35] is the most recent work to enable a binary classifier training and inference on MCUs. However, thus outlined and other impactful algorithms [36, 37] are tailored for specific applications and do not enable tiny IoT devices to self learn/train from a wide range of IoT use-case data. The AIfES library is a C-based platform-independent tool for generating NNs compatible with a range of open-source MCU boards. AIfES can be used with Windows and embedded Linux platforms by producing efficient code in the form of Dynamic Link Library (DLL). Similar to ML-MCU [29], Edge2Train [38] and TinyOL [39], Train++ [40] that present resource-friendly ML model training algorithms, AIfES permits to implement ML model training process on the embedded devices. Cartesiam NanoEdge AI Studio [7] enables the creation of ML static libraries to embed them in ARM Cortex-M MCUs. It allows integrating the training process within the constrained device. In addition, it also can perform unsupervised algorithm training on MCUs.

III. Globe2Train (G2T) Framework Design

In this section, we present the G2T framework, using which we aim to perform distributed model training on IoT devices like video doorbells, smart meters, thermostats, etc., that exists across the globe. As shown in Fig. 1, at the central server/cloud, we first need an algorithm that should efficiently decompose heavy workloads into multiple lightweight tasks which can be comfortably handled by tiny IoT devices. Second, we need a device-level algorithm that can make the training involved devices complete their given tasks, followed by the timely transfer of the learned information to the server even when the connected network is highly uncertain. In the following, we present G2T components, which are the thus required algorithms that can jointly enable distributed machine learning by training one model on numerous global IoT devices.

A. Globe2Train-Cloud (G2T-C) Component

On the central server/cloud shown in Fig. 1. a, we propose to deploy the framework component named G2T-C that interacts with the training involved IoT devices, assign workloads depending on their hardware specifications, and collectively train a high-quality ML model. Using Fig. 2, we explain the G2T-C. Here, in the first step circled ①, we take a multi-class problem with a dataset. Next, in step circled ②, our algorithm decomposes this multi-class problem into $k(k-1)/2$ binary problems because training binary classifiers

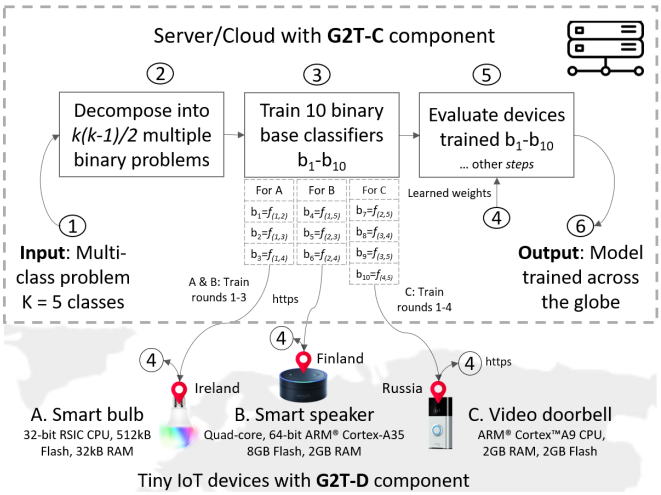

Fig. 2. G2T-C component decomposes one multi-class problem into multiple binary problems, which IoT devices solve and updates back the weights.

on resource-constrained IoT devices like smart speakers and video doorbells are more feasible. Here, the algorithm employed $k(k-1)/2$ binary classifiers, trains $k(k-1)/2$ classifiers. i.e., $f_{(1,2)}, f_{(1,3)}, \ldots, f_{(k-1,k)}$ on the available IoT devices. The IoT device trained binary classifier $f_{(1,2)}$ is capable to classify 1 from 2, the $f_{(1,3)}$ can classify 1 from 3, and finally the $f_{(k-1,k)}$ can classify $k-1$ from k. In our case, we have chosen a 5 class dataset (5 class problem) that requires a multi-class classifier to be trained to solve this. As explained, in step circled ②, our algorithm produces 10 base binary classifiers b_1 to b_{10} as a result of decomposition. In step circled ③, the data corresponding to each of the base classifiers b_i is sent to the deployed IoT devices via HTTPS. So, the smart speaker named B needs to train 3 binary classifiers, i.e., b_4 (to differentiate class 1 data from class 5 data), b_5 (class 2 from 3) and b_6 (2 from 4). Similarly, the other shown devices, Smart bulb (named A) and Video doorbell (named C), are assigned binary classifiers that need to be trained by them. Here, each device trains one classifier per round. For instance, device C trains classifiers b_7 to b_{10} in four rounds.

To train these base classifiers on MCUs or small CPUs based IoT devices, any of the base learners like LDA or the Opt-SGD [29] can be used. When there is a large number of rows or high features in the input dataset, memory overflow issues will arise at the device end. When Opt-SGD is used, the device becomes capable of handling high data volume, velocity situations due to the incremental data loading and training characteristics of Opt-SGD. i.e., in each round, the train set will be streamed to each IoT device by central G2T-C, and the Opt-SGD algorithm executing on the devices will incrementally load this data stream and update the weights without losing information it learned from the initial part of the data stream.

Algorithm 1 G2T-D to efficiently transmit the IoT device trained weights even under poor network conditions.

1: **Input**: Available weights $w_0, \ldots, w_n$ of IoT device trained base binary classifiers $b_1, \ldots, b_n$
2: **Output**: Efficiently transmit shrunken weights
3: $w_{s0}, \ldots, w_{sn}$ to a central server with G2T-C component
4: Set weights threshold $W_t \leftarrow 1/\propto$ network quality
5: **if** n in $w_n > W_t$
6: **for** i = 0, 1, 2, …, n **do**
7: $w_{si} \leftarrow$ shrink the weight w_i. Use *encode* (w_i)
8: **end for**
9: Transmit $w_{s0}, \ldots, w_{sn}$. Use HTTPS
10: **else**
11: Locally accumulate weights till W_t

After each training round, in the step circled ④, the IoT devices transmit the learned information (weights) to the server where G2T-C would be waiting to evaluate the thus trained base binary classifiers. At this stage, each device-created model b_i will be capable of producing a binary output $\in \{-1, +1\}$ for each input vector from the test set of the input dataset. Then, all the base classifiers b_1 to b_{10}, trained on the 3 IoT devices (smart bulb, smart speaker, and video doorbell), are evaluated in step circled ⑤. The remainder processing, starting from creating the correlation matrix until creating the probability table, is done by the G2T-C (follows steps from [29]), using the model weights provided by the training involved devices. After processing, in the last step circled ⑥, a multi-class classifier model distributedly trained across the world is obtained.

In Fig. 2, in step circled ①, when a classifier needs to be trained using the less class count datasets like the MNIST Fashion, Digits, or Cifar 10, then the IoT devices can train base-classifiers and transmit its weights within few rounds. But most ML datasets like Imagenet (200 classes) and COCO (80 classes) datasets are large. In such cases, multiple rounds of training data and learned weights get exchanged between the cloud G2T-C and devices across the world via HTTPS. For instance, when we want to train a classifier using the COCO dataset, the G2T-C in step circled ② breaks down the multi-class problem into 3160 binary problems and let us assume there are 100 IoT devices available to train. So, each device has to interact with the cloud over 32 times to transmit the learned binary base-classifier weights. In such a bandwidth-demanding scenario, since the real-world networks suffer from higher latency, lower throughput, and intermittent poor connections, there is a pressing need to optimize the transmitted information in multiple aspects in order to reduce latency, improve transmission speeds and increase scalability (more devices can contribute to training). To tolerate such real-world challenges and amortize latency issues, we provide a resource-friendly G2T-D component in the upcoming section that needs to execute on tiny IoT devices when they are distributedly contributing to produce a central multi-class classifier model.

B. Globe2Train-Device (G2T-D) Component

The G2T-D framework component that we present in this section is a resource-friendly logic (can be implemented in a few lines of code), that when deployed on IoT devices, will improve communication aspects during distributed training. In the considered training across the globe scenario, the network landscape is dynamic, and the heterogeneous daily IoT devices contributing to the distributed training have diverse hardware specifications [41, 42] and internet conditions [43]–[45]. For instance, when considering the 3 devices shown in Fig. 2, the video doorbell has FPU, KPU capability to process image frames, hence could train the G2T-C component assigned base classifiers efficiently, at higher speeds. The next smart speaker might have 1 GB of RAM to run DSP algorithms hence it can also train faster. In the same group of devices, if a smart bulb is contributing, it has a comparatively weak hardware specification (e.g., Philips Hue has ≈32KB RAM) but might have excellent network quality.

Such situations can be handled by the G2T-D component when deployed on all training involved IoT devices. We present G2T-D in Algorithm 1. Here, when the weights $w_0, \ldots, w_n$ of IoT device trained base binary classifiers $b_1, \ldots, b_n$ are fed to G2T-D algorithm, even when the device connected internet suffers high latency, low bandwidth, congestion issues, it can still manage to transmit the weights back to the G2T-C server component (which assigned the device with the two-class problems to solve) because: (i) In line 4, it intelligently sets the weight threshold W_t high for the training involved devices that have a poor internet connection. Thus, this step reduces the frequent transmission of weights, reducing the network bandwidth. i.e., weights are locally accumulated till it reaches W_t and then transmitted. Then in lines 6 to 8, it shrinks all the accumulated weights using the *encode()* function, which packs the non-zero weight values similar to how Vanilla, Nesterov momentum SGD encodes the parameters/gradients.

IV. GLOBE2TRAIN EVALUATION

Since the majority of IoT devices are MCU-based, we use three popular MCUs for evaluation, where MCU1 is an nRF52840 Adafruit Feather, MCU2 is a STM32f103c8 Blue Pill, MCU3 is a ATSAMD21G18 Adafruit METRO. Then, we take the 22 features, 95 classes Australian Sign Language signs dataset [46] and aim to distributedly train (on MCUs 1-3) a multi-class classifier model that can identify 95 Auslan signs such as *alive, all, answer, more*, etc.

A. Distributed Training on MCUs

As explained in Section III-A, we initiate the distributed training from G2T-C component, where it decomposes the

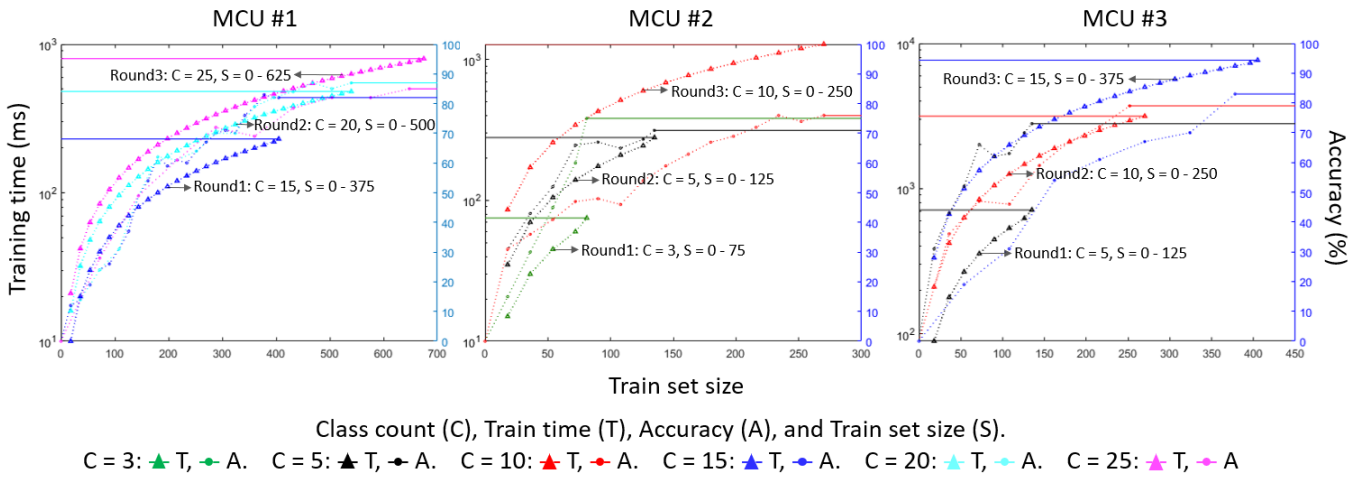

Fig. 3. Distributed training of a multi-class classifier on 3 MCUs using G2T: Onboard accuracy and training time consumed for each training round.

given multi-class problem with 95 classes into 4465 binary problems (i.e., $K = 95$ in $K(K-1)/2$), where b_0 to b_{4465} binary classifiers need to be trained by MCUs 1-3 in numerous rounds and in each round report back the calculated weights of a bunch of trained base classifiers. The G2T-C allocates training workloads depending on the hardware specification of involved devices. So, MCU1 is assigned to learn using the data of the first 60 classes. Similarly, the next better spec MCU3 is assigned to train base classifiers for the next 30 classes, and the remainder 18 classes are assigned to MCU3. After this allocation, the training on each MCU is carried out using the Opt-SGD algorithm [29]. We show the training results in terms of training time, accuracy in Fig. 3. Here, in the first round, MCU1 trains using data from 15 classes, 3 classes on MCU2, and parallelly MCU3 trains using data from 5 classes. Similarly, the results for the subsequent training rounds are shown in the same Fig. 3. After completing each round, the accumulated weights of thus trained base classifiers are shrunken and transmitted to the server by the G2T-D component. Finally, when all 4465 base classifiers weights were sent to G2T-C by MCUs, the base models are combined and a central model is produced.

B. Results Analysis

Here we analyze the distributed training on MCUs results shown in Fig. 3. The total time consumed for three training rounds by MCU 1-3 is 1.5 sec, 1.6 sec, 11.4 sec respectively. The slowest MCU3 completed training in 7.6 sec using 15 class data, and parallelly the fastest MCU1 trained for data belonging to other 15 classes in 0.23 sec. The total distributed training time on MCUs 1-3 was $\approx$ 15 sec, which is only $\approx$ 3 sec slower than training on one Intel Core i7-8650U @ 1.90 GHz CPU laptop with Windows 10 Python scikit-learn setup. This CPU setup is $\approx$ \$700, $\approx$ 32 x times more expensive than 3 MCUs, and consumes $\approx$ 170x times more energy for the same training task.

During this process, to maintain the distributed training reliability and ensure timely completion, G2T-C made MCU2 perform redundant training using data from 13 classes. Thus, in a real scenario, G2T-C can counterbalance when a training involved device fails to complete training due to network errors, resources occupied by the routine device functionalities, etc. Here, since the G2T-C efficiently split the workload and made the MCUs complete training within 3 rounds and since the G2T-D accumulates the weights, then shrinks and sends it to the cloud only once per round, overall, *there is less bandwidth usage and reduced computation to communication ratio*. Similar to how we involved just 3 MCUs and accomplished distributed training, a higher number of IoT hardware can be efficiently connected by G2T to pool GBs of RAM and other hardware resources. In this way, we can complete training faster than training on an expensive GPU, but at a 0 \$ investment since millions of IoT devices exist, and most of them are idle. In the upcoming subsections, we present the individual contributions (benefits) of G2T components and describe how it addresses the global training challenges presented in Section II-C.

C. Contributions of the G2T-C Framework Component

Here, we comparatively show how the characteristics of G2T-C can add value when deployed in a central server to handle the entire distributed training process.

Overcoming the Staleness Effect. Asynchronous SGD (ASGD) [47] shows better latency and fault tolerance since, unlike synchronous SGD (SSGD), ASGD does not follow synchronization and trains models inconsistently. Most of the popular applications like HOGWILD [48], BUCKWILD [49] are powered by ASGD are implemented using a parameter server [50], leading to communication and resource congestion problems when scaling the distributed learning by adding more devices. Scalability is the mandatory characteristic required for us since we aim to train one model using thousands of

IoT devices upon their availability. Moreover, the inconsistent training strategy of ASGD leads to unpredictable behaviors when compared with standard training on a single GPU cluster. Although studies show ASGD to have convergence similar to SSG [51], SSGD is preferred in practice due to its consistent behaviors even when devices increase.

The G2T-C is designed not to face staleness and is more efficient than SSGD and ASGD since, as explained in Section III-A, the step circled ② in Fig. 2, intelligently splits the tasks that execute in multiple rounds on the IoT devices. As shown, the smart bulb named A is assigned to solve the two-class problems b_1 to b_3 and update back the learned weights. Here, if this device solves only b_1 and goes offline, the remaining tasks b_2 to b_3 will be assigned to other better-resourced devices that have completed their assigned task faster. Unlike others, the G2T-C approach does not wait long/depend on gradient/weight updates from slow or poor network devices in order to keep training. Similarly, any of the IoT devices from the distributed training network can perform their assigned tasks without any dependence on the cloud or other devices involved in training. Also, our approach does not require a parameter server.

D. Contributions of the G2T-D Framework Component

In the global distributed training scenario, we model the communication time as $t_c = latency + (modelsize/bandwidth)$. Scalability is essential when connecting a large number of devices. To improve scalability, we need to greatly reduce communication frequency, which is determined by network bandwidth and latency (see above equation). All conventional studies focus only to reduce the bandwidth requirements as the latency between GPUs inside a GPU cluster or servers inside a data center is usually low. In contrast, in our use case, since we perform the same training but on the IoT device hardware that is globally distributed, the latency still remains an issue due to physical device separation. In the remainder, we explain how the resource-friendly G2T-D Algorithm 1 improves training scalability, speed by tolerating the real-world latency, bandwidth issues.

Congestion and Latency Toleration. Since the G2T-D locally keeps accumulating IoT device learned weights till it reaches the weight threshold (intelligently set by considering the current network quality), it reduces the weights synchronization frequency (less usage of congested network) by not allowing to transmit weights frequently or based on its availability. So, as shown in Fig. 1. b (left), the training process used in across the globe setting *gains the ability to tolerate latency* (does not reduce the dynamic real-world latency since it is practically not possible). This latency tolerating property of G2T-D also increases the training scalability, thus enabling the participation of more IoT devices to complete training at higher speeds.

Improved Training Scalability. After the accumulated weights cross the calculated threshold, in order to conserve bandwidth, the G2T-D algorithm shrinks the weights (not quantizing like previous works), then transmit them to the parameter server. As G2T-D sends the weights at intervals that depend on the network condition, as shown in Fig. 1. b (right), this process *improves scalability by reducing the communication-to-computation ratio*.

IoT Hardware Friendliness. When following approaches like the SSDS and ASGD, the IoT devices need to use any of the techniques such as gradient sparsification (transmit only important information), temporally sparse updates, gradient quantization/compression in order to tolerate extreme network conditions by reducing the data to be transferred. In IoT devices, accommodating such techniques adds computation strain while consuming the limited memory that is sufficient only for training models and executing the device's routine functionalities. The presented G2T-D framework component is highly resource-friendly (implementation is only a few lines of code), which can shrink and efficiently transmit trained weights without straining the training involved IoT hardware.

V. Conclusion: Discussion and Future Work

In this paper, we presented Globe2Train, a framework for training ML models on idle IoT devices, millions of which exist across the globe. We showed how the proposed G2T-C and G2T-D framework components can improve distributed training scalability, speed while reducing communication frequency and tolerating network latency. Since the G2T-D can significantly compress gradients/parameters during distributed training, it can be used in federated learning, split learning, distributed ensemble learning approaches, thereby providing the basis for a broad spectrum of decentralized and collaborative learning applications. Similarly, since the G2T-C can decompose one multi-class problem into multiple binary problems, it can be used to decompose a high resource-demanding problem (can run only on GPU clusters/servers with TensorFlow, PyTorch) into multiple resource-friendly tiny parts that can distributedly execute on numerous idle IoT devices across the globe.

We see the lack of comprehensive real-world experimental evaluation of G2T as the major limitation. Hence, in future work, we plan to implement the following steps, observe and report the framework behavior with its performance; (i) Deploy the G2T-D component on a few IoT devices that are geographically separated and connected to networks with poor to good conditions. Then deploy the G2T-C component on an Amazon AWS server and establish communication with IoT devices; (ii) Take a multi-class classification dataset, define an ML model on the server, then instruct the G2T-C to decompose the given ML multi-class problem into multiple binary problems and assign it to the connected IoT devices; (iii) Make the IoT devices solve the binary problems (by

performing model training), then using G2T-D report back the calculated weights; (iv) Now, we should have a full ML model, that was distributedly trained by multiple IoT devices and capable to solve a multi-class problem. We evaluate this model using accuracy, F1-score metrics, compare, report, and investigate the obtained results with the results of a single GPU trained model trained using the same dataset.

ACKNOWLEDGEMENT

This publication has emanated from research supported in part by a research grant from Science Foundation Ireland (SFI) under Grant Number SFI/16/RC/3918 (Confirm) and also by a research grant from SFI under Grant Number SFI/12/RC/2289_P2 (Insight), with both grants co-funded by the European Regional Development Fund.

REFERENCES

[1] L. Zhu, Y. Lu, Y. Lin, and S. Han, "Distributed training across the world," 2019.

[2] J. Lin, C. Gan, and S. Han, "Training kinetics in 15 minutes: Large-scale distributed training on videos," *arXiv preprint*, 2019.

[3] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint*, 2017.

[4] W. Xu, W. Fang, Y. Ding, M. Zou, and N. Xiong, "Accelerating federated learning for iot in big data analytics with pruning, quantization and selective updating," *IEEE Access*, 2021.

[5] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *IEEE Conference on Computer Communications (INFOCOM)*, 2018.

[6] "Tinyml ai inference library," 2021. [Online]. Available: https://github.com/uTensor/uTensor

[7] "Empower your teams to quickly, easily and cost-effectively integrate ai into your projects," 2021. [Online]. Available: https://cartesiam.ai/

[8] B. Sudharsan and P. Patel, "Machine learning meets internet of things: From theory to practice," in *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2021.

[9] B. Sudharsan, P. Corcoran, and M. I. Ali, "Smart speaker design and implementation with biometric authentication and advanced voice interaction capability." in *27th AIAI Irish Conference on Artificial Intelligence and Cognitive Science (AICS)*, 2019.

[10] B. Sudharsan, S. Malik, P. Corcoran, P. Patel, J. G. Breslin, and M. I. Ali, "Owsnet: Towards real-time offensive words spotting network for consumer iot devices," in *IEEE 7th World Forum on Internet of Things (WF-IoT)*, 2021.

[11] B. Sudharsan, D. Sundaram, J. G. Breslin, and M. I. Ali, "Avoid touching your face: A hand-to-face 3d motion dataset (covid-away) and trained models for smartwatches," in *10th International Conference on the Internet of Things Companion*, 2020.

[12] A. Jochems *et al.*, "Distributed learning: developing a predictive model based on data from multiple hospitals without data leaving the hospital–a real life proof of concept," *Radiotherapy and Oncology*, 2016.

[13] A. Jochems, T. M. Deist *et al.*, "Developing and validating a survival prediction model for nsclc patients through distributed learning across 3 countries," *Journal of Radiation Oncology Biology Physics*, 2017.

[14] B. Sudharsan, P. Yadav, J. G. Breslin, and M. I. Ali, "An sram optimized approach for constant memory consumption and ultra-fast execution of ml classifiers on tinyml hardware," in *IEEE International Conference on Services Computing (SCC)*, 2021.

[15] W. Dai, E. P. Xing *et al.*, "Toward understanding the impact of staleness in distributed machine learning," *arXiv preprint*, 2018.

[16] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu, "D2: Decentralized training over decentralized data," *arXiv preprint*, 2018.

[17] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Adpsgd asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning (ICML)*, 2018.

[18] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," in *Neural Information Processing Systems (NIPS)*, 2017.

[19] B. Sudharsan, O. Rana, P. Patel, J. Breslin, M. Intizar Ali, K. Mitra, S. Dustdar, P. Prakash, and R. Ranjan, "Towards distributed, global, deep learning using iot devices," *IEEE Internet Computing*, 2021.

[20] B. Sudharsan, D. Sheth, S. Arya, F. Rollo, P. Yadav, P. Patel, J. G. Breslin, and M. I. Ali, "Elasticl: Elastic quantization for communication efficient collaborative learning in iot," in *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 2021.

[21] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns," in *Interspeech*, 2014.

[22] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Randomized quantization for communication-optimal stochastic gradient descent," *arXiv preprint*, 2016.

[23] S. Zhou, Y. Zou *et al.*, "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients," *arXiv preprint*, 2016.

[24] N. Strom, "Scalable distributed dnn training using commodity gpu cloud computing," in *International Speech Communication Association (ISCA)*, 2015.

[25] C.-Y. Chen, J. Choi, D. Brand, A. Agrawal, W. Zhang, and K. Gopalakrishnan, "Adacomp: Adaptive residual gradient compression for data-parallel distributed training," in *AAAI Conference*, 2018.

[26] F. Sattler, S. Wiedemann, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," in *International Joint Conference on Neural Network (IJCNN)*, 2019.

[27] I. Mitliagkas, C. Ré *et al.*, "Asynchrony begets momentum, with an application to deep learning," in *54th Allerton Conference*, 2016.

[28] B. Sudharsan, S. P. Kumar, and R. Dhakshinamurthy, "Ai vision: Smart speaker design and implementation with object detection custom skill and advanced voice interaction capability," in *11th IEEE International Conference on Advanced Computing (ICoAC)*, 2019.

[29] B. Sudharsan, J. G. Breslin, and M. I. Ali, "Ml-mcu: A framework to train ml classifierson mcu-based iot edge devices," *IEEE Internet of Things Journal*, 2021.

[30] J. Lee, M. Stanley, A. Spanias, and C. Tepedelenlioglu, "Integrating machine learning in embedded sensor systems for internet-of-things applications," in *IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, 2016.

[31] G. Huang, S. Liu, L. van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," *arXiv preprint*, 2017.

[32] M. Tan, B. Chen, V. Vasudevan, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," *arXiv preprint*, 2018.

[33] C. Briggs, Z. Fan, and P. Andras, "A review of privacy preserving federated learning for private iot analytics," *arXiv preprint*, 2020.

[34] Q. Li, Z. Wen, and B. He, "Federated learning systems: Vision, hype and reality for data privacy and protection," *arXiv preprint*, 2019.

[35] H. Keshavarz, M. S. Abadeh, and R. Rawassizadeh, "Sefr: A fast linear-time classifier for ultra-low power devices," *arXiv preprint*, 2020.

[36] G. Kamath, P. Agnihotri, M. Valero, K. Sarker, and W. Song, "Pushing analytics to the edge," in *Global Communications (GLOBECOM)*, 2016.

[37] A. Kumar, S. Goyal, and M. Varma, "Resource-efficient machine learning in 2 KB RAM for the internet of things," in *34th International Conference on Machine Learning (ICML)*, 2017.

[38] B. Sudharsan, J. G. Breslin, and M. I. Ali, "Edge2train: A framework to train machine learning models (svms) on resource-constrained iot edge devices," in *10th International Conference on the Internet of Things (IoT)*, 2020.

[39] H. Ren, D. Anicic, and T. Runkler, "Tinyol: Tinyml with online-learning on microcontrollers," *arXiv preprint*, 2021.

[40] B. Sudharsan, P. Yadav, J. G. Breslin, and M. I. Ali, "Train++: An incremental ml model training algorithm to create self-learning iot devices," in *Proceedings of the 18th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC 2021)*, 2021.

[41] B. Sudharsan, P. Patel, J. G. Breslin, and M. I. Ali, "Sram optimized porting and execution of machine learning classifiers on mcu-based iot devices: demo abstract," in *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems (ICCPS)*, 2021.

[42] B. Sudharsan, P. Patel, J. G. Breslin, and M. I. Ali, "Ultra-fast classification on iot devices without sram consumption," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2021.

[43] B. Sudharsan, D. Sundaram, P. Patel, J. G. Breslin, and M. I. Ali, "Edge2guard: Botnet attacks detecting offline models for resource-constrained iot devices," in *IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2021.

[44] B. Sudharsan, J. G. Breslin, and M. I. Ali, "Adaptive strategy to improve the quality of communication for iot edge devices," in *IEEE 6th World Forum on Internet of Things (WF-IoT)*, 2020.

[45] F. Rollo, B. Sudharsan, L. Po, and J. G. Breslin, "Air quality sensor network data acquisition, cleaning, visualization, and analytics: A real-world iot use case," in *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers*, 2021.

[46] M. Waleed, "Dataset," UCI. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Australian+Sign+Language+signs+(High+Quality)

[47] J. Tsitsiklis, D. Bertsekas, and M. Athans, "Distributed asynchronous deterministic and stochastic gradient optimization algorithms," *IEEE transactions on automatic control*, 1986.

[48] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Neural Information Processing Systems (NIPS)*, 2011.

[49] C. M. De Sa, C. Zhang, K. Olukotun, and C. Ré, "Taming the wild: A unified analysis of hogwild-style algorithms," in *Neural Information Processing Systems (NIPS)*, 2015.

[50] J. Dean, G. Corrado, K. Chen *et al.*, "Large scale distributed deep networks," in *Neural Information Processing Systems (NIPS)*, 2012.

[51] X. Lian, Y. Huang, Y. Li, and J. Liu, "Asynchronous parallel stochastic gradient for nonconvex optimization," in *Neural Information Processing Systems (NIPS)*, 2015.