ElastiQuant: Elastic Quantization Strategy for Communication Efficient Distributed Machine Learning in IoT

Bharath Sudharsan
Data Science Institute
NUI Galway, Ireland
b.sudharsan1@nuigalway.ie

John G. Breslin
Data Science Institute
NUI Galway, Ireland
john.breslin@nuigalway.ie

Muhammad Intizar Ali
School of Electronic Engineering
Dublin City University, Ireland
ali.intizar@dcu.ie

Peter Corcoran
School of Engineering
NUI Galway, Ireland
peter.corcoran@nuigalway.ie

Rajiv Ranjan
Newcastle University
Newcastle upon Tyne, U.K
raj.ranjan@ncl.ac.uk

ABSTRACT

In training distributed machine learning, communicating model updates among workers has always been a bottleneck. The magnitude of impact on the quality of resultant models is higher when distributed training on low hardware specification devices and in uncertain real-world IoT networks where congestion, latency, bandwidth issues are common. In this scenario, gradient quantization plus encoding is an effective way to reduce cost when communicating model updates. Other approaches can be to limit the client-server communication frequency, adaptive compression by varying the spacing between quantization levels, reusing outdated gradients, deep compression to reduce transmission packet size, and adaptive tuning of the number of bits transmitted per round. The optimization levels provided by such and other non-comprehensive approaches do not suffice for high-dimensional NN models with large size model updates.

This paper presents ElastiQuant, an elastic quantization strategy that aims to reduce the impact caused by limitations in distributed IoT training scenarios. The distinguishable highlights of this comprehensive work are: (i) theoretical assurances and bounds on variance and number of communication bits are provided, (ii) worst-case variance analysis is performed, and (iii) momentum is considered in convergence assurance. ElastiQuant experimental evaluation and comparison with top schemes by distributed training 5 ResNets on 18 edge GPUs over ImageNet and CIFAR datasets show: improved solution quality in terms of ≈ 2-11 % training loss reduction, ≈ 1-4 % accuracy boost, and ≈ 4-22 % variance drop; positive scalability due to higher communication compression resulting in saving bandwidth and ≈ 4-30 min per epoch training speedups.

CCS CONCEPTS

• **Computing methodologies** → **Distributed computing methodologies**; **Machine learning algorithms**; **Learning settings**;

KEYWORDS

Distributed Machine Learning, IoT Devices, Scalable Model Training, Quantization, Gradient Compression.

ACM Reference Format:
Bharath Sudharsan, John G. Breslin, Muhammad Intizar Ali, Peter Corcoran, and Rajiv Ranjan. 2022. ElastiQuant: Elastic Quantization Strategy for Communication Efficient Distributed Machine Learning in IoT. In *The 37th ACM/SIGAPP Symposium on Applied Computing (SAC '22), April 25–29, 2022, Virtual Event, .* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3477314.3507135

1 INTRODUCTION

To train problem-solving Machine Learning (ML) models within a reasonable time frame using large historical datasets, distributed training schemes from the High Performance Computing (HPC) domain are employed on a data center grade e.g. CPU-GPU cluster that contains numerous worker nodes. Whereas for privacy-sensitive yet valuable data, the problem-solving ML models are trained directly on ubiquitous devices (workers) in decentralized IoT settings.

In both scenarios, distributed training of one ML model on numerous devices/clients/processors using distributed versions of stochastic gradient descent (SGD) has gained attention due to its higher scalability characteristics. For example, data-parallel schemes such as QSGD [1], Buckwild [5], TernGrad [32], SignSGD [2] in a setting with K distributed devices that split a large dataset among themselves. Each client keeps a private copy of the model parameters while having access to the global function's stochastic gradients that need to be minimized. Each device, at each training round, privately computes its stochastic gradient using the local data it sees. This learned information is then broadcasted/synchronized to other training-involved devices, using which aggregation is performed at each device to obtain the updated model parameters.

Scalable distributed training on K devices, using parallel SGD significantly reduces convergence time and computational costs, but the communication costs to transmit and synchronize large model updates swiftly increase as K grows - thwarting the expected benefits of reducing computational costs. In reality, with data and

Table 1: ElastiQuant comparison with related papers. Not Applicable (NA), Not Investigated (NI), No Guarantee/assurance (NG).

Paper	Test Setup	Gradients; Momentum	Variance	Com bits	Scalability	Worst-case	Highlight
ATOMO [30]	AWS EC2 cloud	Atomic sparsification; NI	NG	Bounds	Within cluster nodes	NI	Sparsification to minimize variance
Terngrad [32]	128-node each with 4 Nvidia P100	Quantize to ternary levels; Yes	NI	NI	Within cluster nodes	NI	Need only three levels to aggressively reduce communication time
Globe2Train [20]	MCUs, CPUs	NA	NI	NI	Global IoT devices	NI	Latency, congestion tolerance
AD-PSGD [13]	32-node each with 4 Nvidia P100	NI	Bounds	NI	Within cluster nodes	NI	Robust to heterogeneity
QSGD [1]	AWS EC2 cloud	Lossy compression; NI	Bounds	Bounds	Within cluster nodes	NI	Good practical performance
NUQSGD [15]	8 NVIDIA 2080 Ti GPUs	Nonuniform quantization; Yes	Assurance, bounds	Bounds	Within cluster nodes	Yes	Stronger guarantees, higher empirical performance
D2 [27]	16 workers	NI	Assurance, bounds	NI	Within cluster nodes	NI	Much improve convergence rate, robust to data variance
EF-SignSGD [11]	Multiple workers	Error-feedback; Yes	NI	NI	Within cluster nodes	NI	Simply add EF to recover performance
DGC [14]	64-node each with 4 Nvidia Titan XP	Deep compression; Yes	NI	NI	Within cluster nodes, mobiles	NI	270 x - 600 x gradient compression ratio without losing accuracy
PowerSGD [29]	8-node each with 2 Nvidia Titan X	Low-rank compression; Yes	NI	NI	Within cluster nodes	NI	Consistent wall-clock speedups, test performance on par with SGD
ElastiQuant	18 IoT boards, edge GPUs	Elastic quantization; Yes	Assurance, bounds	Bounds	IoT boards, mobiles, edge GPUs	Yes	Higher solution quality, scalability - assurance with results

computation resources globally distributed, communication time consumed to share stochastic gradients is the major performance bottleneck [23] - to reduce which gradient quantization is used to transmit fewer communication bits per training iteration.

Standard approaches [5, 17, 29, 30, 32, 33] achieve compression levels by quantizing devices learned parameters to a uniform grid (from 2-bit to 16-bit) using randomized, lossy, or other methods that do not provide comprehensive assurances. For examples, in Table 1, atomic sparsification [30] provides bounds on communication bits, but not on variance. Whereas AD-PSGD [13] provides bounds on variance, but no analysis is performed on communication bits. Orthogonally, the communication-compression schemes of interest to this study such as QSGD [1], EF-SignSGD [11], and others [3, 6, 25, 31] have established several theoretical guarantees, can adapt the number of quantization levels (variance determined by this), enabling users to trade communication bandwidth with convergence time. As an improvement, a QSGD variant (QSGD-V) [1] was introduced. We created a real-world distributed training setup using K wirelessly interconnected edge GPUs. Here, when empirically evaluating QSGD, there was a substantial amount of quantization-induced variance, making the practical performance far from that of the original SGD and QSGD-V. Although QSGD-V has a stronger empirical performance, it cannot establish theoretical guarantees in the case when there is a need to vary the number of bits transmitted - an essential feature when few of the K devices are in bandwidth-constrained IoT networks. So, naturally, there is a requirement of an approach with distributed training performance close to SGD, and strong empirical performance as QSGD-V, along with the capability to adaptively alter the number of quantization levels. This paper presents Elastic Quantization (ElastiQuant), a strategy with comprehensive analysis and extensive evaluation whose contribution can be summarised as follows:

Analysis. ElastiQuant aims to achieve stronger theoretical assurances to offer benefits such as: Communication compression by reducing the number of bits communicated, enabling training even in low-bandwidth IoT networks; Reducing quantization induced variance improving optimization performance of models in terms of test accuracy and training loss; Reduced computational, encoding and transmission cost to achieve positive scalability and training speedup in every epoch.

Main Results. ElastiQuant is evaluated on distributed training for large-scale image classification tasks using CIFAR and ImageNet datasets. Results demonstrate that the real-world ElastiQuant performance surpasses QSGD-V and EF-SignSGD in terms of communication compression and training time to produce a central model without accuracy drops. Consuming the same number of bits per training iteration, ElastiQuant has a smaller variance than QSGD, translating to improved model optimization performance.

2 BACKGROUND AND RELATED WORK

This section covers essential concepts and prior work. Table 1 presents ElastiQuant closer comparison with related papers.

2.1 Gradient Compression

The gradient quantization in this work is not the same as model weight optimization by quantization and sparsification. When the gradients transferred during distributed training are reduced, the communication bandwidth will also reduce. Approaches exist to adapt compression such as varying the spacing between quantization levels [6], reusing outdated gradients [25], adaptive tuning the number of local updates or the communication frequency [31]. ElastiCL [24] dynamically alters the number of quantization levels used to represent a model update and achieves a low error floor as well as communication/transmission efficiency. Buckwild [5] is a lossy compressed SGD with convergence guarantees provided with bounds on the error probability of SGD. TernGrad [32] uses a 3-level gradients-based approach to significantly reduce communication costs with a small performance degradation compared to FP-SGD.

Table 2: Summary of notations used during ElastiQuant design.

Symbols	Descriptions
s, $\mathcal{L}$, $\hat{\mathcal{L}}$	Number of internal quantization levels, sequence of quantization levels, $\mathcal{L}$ for special case
l, r, i, b, C, n, W, p, j, τ, ϵ_Q, $\hat{\epsilon}_Q$, $\mathbf{w}$	Parameters and variables with values in range as defined
$Q_s(\mathbf{v})$; $\tilde{s}(r)$, $p(r)$, $l_{\tilde{s}(r)}$, $\tau(r)$, $F(\mathbf{w})$	Elastic quantization of vector $\mathbf{v}$; functions with variables defined for ElastiQuant design
$h_i(\mathbf{v}, s)$, $\mathbb{E}$, $\mathbb{R}$, $\mathbb{1}$	Independent random variables, expected value operator, real numbers, indicator function
v, ρ, $\mathbf{h}$; η, B	Coordinates, vector of signs of v_i, quantizations of normalized coordinates; second moment bound
N_Q, t_r, $g(\mathbf{w})$	Bound on communication bits, last coordinates to transmit, stochastic gradient with η
w_t, $\mathbf{w}_0$, $\mathbf{w}^*$, g, μ, α, V, T, $\hat{\mathbf{w}}_T$; $\mathcal{S}_j$	Parameters and variables in the ElastiQuant with momentum design; coordinates of $\mathbf{v}$
K, $\mathcal{D}$, ξ, t	Number of training involved devices, local datasets, mini-batch, training iterations

Also, attempts have been made to quantize the entire model - DoReFa-Net [33] uses 2-bit gradients and 1-bit weights. The current state of literature presents schemes applicable to guarantee the convergence of CNNs in TernGrad, DoReFa-Net; RNNs in QSGD; distributed ML classifier training in Globe2Train [20]. The theoretical quantization limit cannot exceed 32, to overcome which gradient sparsification techniques exist. The user predefined threshold-based sparsification [18], the adaptive compression ratio [3] shows that, with only negligible performance degradation, 99% gradients can be pruned. In [16], quantization, compression are combined to achieve next-level optimization ratio. The SignSGD (also known as 1-bit gradient) [17] achieved a 10× speedup by reducing the data transfer size. Here, each gradient component gets quantized to two values. The latter variant of SignSGD [2] established convergence guarantees. The number of quantization levels is fixed in SignSGD and is not unbiased. SignSGD does not consider the trade-off between convergence speed and communication costs. EF-SignSGD [11] is another improved variant of SignSGD.

ElastiQuant is motivated by the recent NUQSGD [15], ElastiCL [24], Globe2Train [20], and the standard pulse code modulation concept [28] to use unbiased nonuniform logarithmic quantization method [7]. This concept has been inherited for LSTM network compression [8], Logarithmic encodings to represent model weights and activations [12].

2.2 Distributed Machine Learning

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 of the time both, thus restricting transmitting them to data centers and training using advanced ML frameworks on GPU clusters. 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 collaborative learning 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 [9], patient survival situations across 3 countries were analyzed [10]. However, the advancements are not of much use as the scalability and bandwidth are poor in IoT networks [4, 23].

Decentralized learning is an orthogonal exploration where methods like D2 [27], AD-PSGD [13] 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 communicating model updates, 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 [32]. However, again, such learning approaches do not scale under high network latency [23]. Also, face other critical problems due to lower network bandwidth, expensive mobile data plan, intermittent network connection, which are common when distributed training.

ElastiQuant is compatible and contributes to decentralized techniques such as federated learning, split learning, distributed ensemble learning by providing applications the ability to balance communication savings with variance. Also, this is a distinctive study that evaluates top schemes on off-the-shelf IoT edge GPUs boards rather than within HPC data center grade CPU-GPU clusters (see Table 1).

3 ELASTIQUANT DESIGN

At each iteration of distributed training, to accelerate training while reducing communication costs, each device broadcasts an encoding of its privately calculated compressed gradient, decodes the gradients received from other devices, and produces a stochastic gradient by summing all the quantized vectors. The existing schemes that compress gradients before encoding does not take into consideration the properties of gradient vectors. This leads to slowing overall convergence as the gradient variance would have substantially increased. Also, it is necessary for the quantization to be stochastic, to not introduce bias. To optimize overall performance, the communication savings should be balanced with variance.

ElastiQuant elastically distributes quantization levels in the unit interval, making the quantized gradients remain unbiased and suitable for the original SGD during distributed training. ElastiQuant uses a custom parameterized generalization of the unbiased quantization scheme to control communication cost and gradient variance. Also, ElastiQuant reduces quantization error and variance as it can match the properties of gradient vectors - it increases the number of quantization levels near zero to obtain a stronger variance bound. Table 2 shows all the notations used during ElastiQuant design in the upcoming subsections.

3.1 ElastiQuant Quantization Strategy

Let $s \in 1, 2, \cdots$, and $\mathcal{L} = (l_0, l_1, \cdots, l_{s+1})$ with $l_0 = 0 < l_1 < l_{s+1} = 1$. For $r \in [0, 1]$, let $\tilde{s}(r)$ satisfy $l_{\tilde{s}(r)} \leq r \leq l_{\tilde{s}(r)+1}$ and $p(r)$ satisfy $r = (1 - p(r))l_{\tilde{s}(r)} + p(r)l_{\tilde{s}(r)+1}$. Here $\tau(r)$ can be given as

$l_{\tilde{s}(r)+1} - l_{\tilde{s}(r)}$ with $\tilde{s}(r) \in 0, 1, \cdots, s$. In this setup, the ElastiQuant quantization strategy of $\mathbf{v} \in \mathbb{R}^d$ is

$$\begin{aligned} Q_s(\mathbf{v}) &\triangleq [Q_s(v_1), \cdots, Q_s(v_d)]^T \\ \text{where } Q_s(v_i) &= \|\mathbf{v}\| \cdot \operatorname{sign}(v_i) \cdot h_i(\mathbf{v}, s) \end{aligned} \tag{1}$$

where, letting $r_i = |v_i| / \|\mathbf{v}\|$, the $h_i(\mathbf{v}, s)$'s are independent random variables such that $h_i(\mathbf{v}, s) = l_{\tilde{s}(r_i)}$ with probability $1 - p(r_i)$ and $h_i(\mathbf{v}, s) = l_{\tilde{s}(r_i)+1}$ otherwise. We note that the distribution of $h_i(\mathbf{v}, s)$ satisfies $\mathbb{E}[h_i(\mathbf{v}, s)] = r_i$ and achieves the minimum variance over all distributions that satisfy $\mathbb{E}[h_i(\mathbf{v}, s)] = r_i$ with support $\mathcal{L}$. We first focus on a special case of ElastiQuant's elastic quantization with $\hat{\mathcal{L}} = (0, 1/2^s, \cdots, 2^{s-1}/2^s, 1)$ as the quantization levels.

In reality, it is not common for ML models to have large r_i in their stochastic gradient vectors - instead they are dense vectors. Hence, ElastiQuant uses fine intervals for small r_i values to reduce quantization error and control the variance. After quantizing the stochastic gradient with a small number of discrete levels, each training involved device must encode its local gradient into a binary string for broadcasting using the method presented in the below subsection.

3.2 ElastiQuant Efficient Encoding

The quantized gradient $Q_s(\mathbf{v})$ from Eqn (1) is determined by $\boldsymbol{\rho}, \mathbf{h}, \|\mathbf{v}\|$. Here, $\rho \triangleq [\operatorname{sign}(v_1), \cdots, \operatorname{sign}(v_d)]^T$ is the vector of the coordinate signs v_i, $\mathbf{h} \triangleq [h_1(\mathbf{v}, s), \cdots, h_d(\mathbf{v}, s)]^T$ are quantization of normalized coordinates, and $\|\mathbf{v}\|$ is norm of the gradient. The *Encode* function that encodes the elastic quantized values use $\|\mathbf{v}\|, \boldsymbol{\rho}, \mathbf{h}$ and the encode-decode method ERC for encoding/decoding positive integers such that ERC: $1, 2, \cdots, n \to 0, 1^*$ and ERC^{-1}: $0, 1^* \to 1, 2, \cdots, n$. When stochastic gradient is passed to ElastiQuant *Encode* function: The norm $\|\mathbf{v}\|$ is encoded using b bits floating point encoding (b set to 4 produces 4-bit-ElastiQuant). Then the processing is performed in rounds $r = 0, 1, \cdots, n$. On round r, after transmitting all nonzero coordinates including t_r, the ERC(i_r) is transmitted where $t_{r+1} = t_r + i_r$ is either: (i) Index of the first nonzero coordinate of $\mathbf{h}$ after t_r (with $t_0 = 0$). Here, the one bit encoding of sign $\rho_{t_{r+1}}$ and ERC$(\log(2^{s+1} h_{t_{r+1}}))$ are transmitted before the next round. (ii) Index of the last nonzero coordinate. Here, $\rho_{l_{r+1}}$ and ERC are transmitted and encoding is complete. In rounds, the *Decode* function reads b bits, uses ERC^{-1} to reconstruct $\|\mathbf{v}\|$.

3.3 Theoretical Assurance

Here, theoretical assurance for ElastiQuant is provided in terms of bounds on the variance and communication bits.

3.3.1 ElastiQuant Variance Bound. When training using DGC, QSGD, or even ElastiQuant, if models show high variance, such models fail to generalize for unseen data. This subsection presents the variance bound of ElastiQuant.

Let $\mathbf{v} \in \mathbb{R}^d$. The elastic quantization of $\mathbf{v}$ satisfies $\mathbb{E}[Q_s(\mathbf{v})] = \mathbf{v}$. Then we have

$$\begin{aligned} \mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] \leq \epsilon_Q \|\mathbf{v}\|^2. \text{ Here } \epsilon_Q = (1/8 + 2^{-2s-2} d) \\ \mathbb{1}\{d < 2^{2s+1}\} + (2^{-s}\sqrt{d} - 7/8)\mathbb{1}\{d \geq 2^{2s+1}\} \end{aligned} \tag{2}$$

and $\mathbb{1}$ denotes the indicator function Provided that $s \leq \log(d)/2$, then it also holds that $\mathbb{E}[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2] \leq \hat{\epsilon}_Q \|\mathbf{v}\|^2$ where $\hat{\epsilon}_Q = \min 2^{-2s}/4(d - 2^{2s}), 2^{-s}\sqrt{d - 2^{2s}} + O(s)$

This result implies that if $g(\mathbf{w})$ is a stochastic gradient with a second moment bound η, then $Q_s(g(\mathbf{w}))$ is a stochastic gradient with a variance upper bound $\epsilon_Q \eta$. Note that the variance upper bound decreases with the number of quantization levels. In the range of $s = o(\log(d))$, $\hat{\epsilon}_Q$ decreases with s, which is because the first term in the upper bound decreases exponentially fast in s. To obtain $\hat{\epsilon}_Q$, we establish upper bounds on the number of coordinates of $\mathbf{v}$ falling into intervals defined by $\hat{\mathcal{L}}$.

ElastiQuant bound is tighter than related schemes - demonstrated in Section 4.2, by training ResNets on CIFAR and ImageNet datasets. Here, variance vs training iterations is recorded and compared.

3.3.2 ElastiQuant Bound on Number of Communication Bits. When communicating (quantized gradients in our case) in low-bandwidth IoT networks, for compression, the source data is mapped to a variable number of bits. This subsection presents the number of communication bits bound of ElastiQuant.

Let $\mathbf{v} \in \mathbb{R}^d$. Provided d is large enough to ensure $2^{2s} + \sqrt{d}2^s \leq d/e$, the expectation $\mathbb{E}[|\text{ENCODE}(\mathbf{v})|]$ of the number of communication bits needed to transmit $Q_s(\mathbf{v})$ is bounded above by

$$\begin{aligned} N_Q = C + 3n_{s,d} + (1 + o(1))n_{s,d} \log(d/n_{s,d}) \\ +(1 + o(1))n_{s,d} \log\log(8(2^{2s} + d)/n_{s,d}) \end{aligned} \tag{3}$$

where $C = b - (1 + o(1))$ and $n_{s,d} = 2^{2s} + 2^s\sqrt{d}$.[2] The results provide a bound on the expected number of communication bits to encode the quantized stochastic gradient. Note that $2^{2s} + \sqrt{d}2^s \leq d/e$ is a mild assumption in practice. As one would expect, the bound Eqn (3) increases monotonically in d and s. In the sparse case, if we choose $s = o(\log d)$ levels, then the upper bound on the expected code-length is $O(2^s\sqrt{d}\log(\sqrt{d}/2^s))$.

The efficient encoding method of ElastiQuant helps provide tighter bounds on the code-length of the gradients transmitted during distributed training. Also, when the ElastiQuant variance bound is combined with its code-length bound, a bound on the total communication costs can be obtained, which shows suboptimal communication savings compared to other schemes.

3.4 ElastiQuant with Momentum

Momentum helps accelerate gradients consistently in the right directions and also dampens oscillations, thus leading to faster convergence. This section presents the convergence assurance when distributed training using ElastiQuant with momentum.

The update rule for FP-SGD (unquantized version) with momentum is: $\mathbf{y}_{t+1} = \mathbf{w}_t - \alpha g(\mathbf{w}_t)$; $\mathbf{y}^l_{t+1} = \mathbf{w}_t - l\alpha g(\mathbf{w}_t)$; and $\mathbf{w}_{t+1} = \mathbf{y}_{t+1} + \mu(\mathbf{y}^l_{t+1} - \mathbf{y}^l_t)$. Here, $\mathbf{w}_t$ is the current parameter input and $\mu \in [0, 1)$ is the momentum parameter. When substituting $l = 0$, the heavy-ball method is obtained, and Nesterov's accelerated gradient method when substituted with $l = 1$. Momentum can be added to training algorithms (like SGD, QSGD, ElastiQuant) by substituting the above update rules as appropriate.

For convex optimization, there can be only one globally optimal solution. For nonconvex optimization, there can exist multiple locally optimal points, requiring extra computation to identify the global solution (cannot be guaranteed). Given the importance of momentum, in the following, the convergence assurance for momentum ElastiQuant is established for both optimizations.

3.4.1 ElastiQuant with Momentum for Convex Optimization. Let $f : \mathbb{R}^d \to \mathbb{R}$ denote a convex function with $\|\nabla f(\mathbf{w})\| \leq V$ for all $\mathbf{w}$. Let $\mathbf{w}_0$ denote an initial point, $\mathbf{w}^* = \arg\min f(\mathbf{w})$, $\hat{\mathbf{w}}_T = 1/T \sum_{t=0}^{T} \mathbf{w}_t$, and ϵ_Q be defined as in Eqn (2). Suppose that ElastiQuant with momentum is executed for T iterations with a learning rate $\alpha > 0$ on K processors, each with access to independent stochastic gradients of f with *a.* second-moment bound B. Then ElastiQuant with momentum satisfies

$$\mathbb{E}\left[f\left(\hat{\mathbf{w}}_T\right)\right] - \min_{\mathbf{w}\in\Omega} f(\mathbf{w}) \leq \frac{\alpha(1+2l\mu)\left(V^2 + \left(1+\epsilon_Q\right)B/K\right)}{2(1-\mu)} + \frac{\mu\left(f\left(\mathbf{w}_0\right) - f\left(\mathbf{w}^*\right)\right)}{(1-\mu)(T+1)} + \frac{(1-\mu)\left\|\mathbf{w}_0 - \mathbf{w}^*\right\|^2}{2\alpha(T+1)}$$

On nonconvex problems, (weaker) convergence assurances can be established for ElastiQuant with momentum. In particular, ElastiQuant with momentum is guaranteed to converge to a local minima for smooth general loss functions.

3.4.2 ElastiQuant with Momentum for Smooth Nonconvex Optimization. Let $f : \mathbb{R}^d \to \mathbb{R}$ denote a possibly nonconvex and β-smooth function with $\|\nabla f(\mathbf{w})\| \leq V$ for all $\mathbf{w}$. Let $\mathbf{w}_0$ denote an initial point, $\mathbf{w}^* = \arg\min f(\mathbf{w})$, and ϵ_Q be defined as in Eqn (2).

Suppose that ElastiQuant with momentum is executed for T iterations with $\alpha = \min\{(1-\mu)/(2\beta), C/\sqrt{T+1}\}$ for some $C > 0$ on K processors, each with access to independent stochastic gradients of f with a second-moment bound B. Then ElastiQuant with momentum satisfies

$$\min_{t=0,\cdots,T} \mathbb{E}[\|\nabla f(\mathbf{w}_t)\|^2] \leq \frac{2(f(\mathbf{w}_0) - f(\mathbf{w}^*))(1-\mu)}{\alpha(T+1)} + \frac{C}{(1-\mu)^3\sqrt{T+1}}\tilde{V}. \text{ Here } \tilde{V} = \beta(\mu^2((1-\mu)l-1)^2 + (1-\mu)^2)(V^2 + (1+\epsilon_Q)B/K)$$

In next subsection, this analysis is extended to decentralized settings.

3.5 ElastiQuant for Decentralized Training

In distributed training (setup in Section 4.1 can be an example), all networked devices cannot guarantee low latency and high bandwidth networks for gradient communication. Here we show ElastiQuant integration with communication-efficient variants of decentralized parallel SGD for a promising solution to train deep networks in constrained networked systems. In a decentralized optimization scenario, the following problem can be considered

$$\min_{\mathbf{w}\in\mathbb{R}^d} F(\mathbf{w}) = \frac{1}{K}\sum_{i=1}^{K} f_i(\mathbf{w})$$

where $f_i(\mathbf{w}) = \mathbb{E}_{\xi\sim\mathcal{D}_i}[f(\mathbf{w};\xi)]$, $\mathcal{D}_i$ is the local datasets stored in edge GPU i, and $f(\mathbf{w};\xi)$ is the loss of a model described by $\mathbf{w}$ on mini-batch ξ. At iteration t of decentralized algorithms (such as D2 [27], AD-PSGD [13], D-PSGD [26]), each edge GPU i computes its local stochastic gradient $g_i(\mathbf{w}_t^{(i)})$ with $\mathbb{E}_{\varepsilon_t^{(i)}\sim\mathcal{D}_i}[g_i(\mathbf{w}_t^{(i)})] = \nabla f_i(\mathbf{w}_t^{(i)})$ where $\mathbf{w}_t^{(i)}$ local parameter vector and local dataset $\xi_t^{(i)}$. Then, edge GPU i fetches its neighbours' parameter vectors and updates its local parameter vector using $\mathbf{w}_{t+1/2}^{(i)} = \sum_{j=1}^{K} W_{i,j}\mathbf{w}_t^{(j)}$ where $W \in \mathbb{R}^{K\times K}$ is a symmetric doubly stochastic matrix (for all i $W = W^T$ and $\sum_{j=1}^{K} W_{i,j} = 1$). Note that $W_{i,j} \geq 0$ in general and $W_{i,j} = 0$ means that edge GPUs i and j are not connected. Finally, edge GPU i updates its local parameter vector using the update rule $\mathbf{w}_{t+1}^{(i)} \leftarrow \mathbf{w}_{t+1/2}^{(i)} - \alpha g_i(\mathbf{w}_t^{(i)})$.

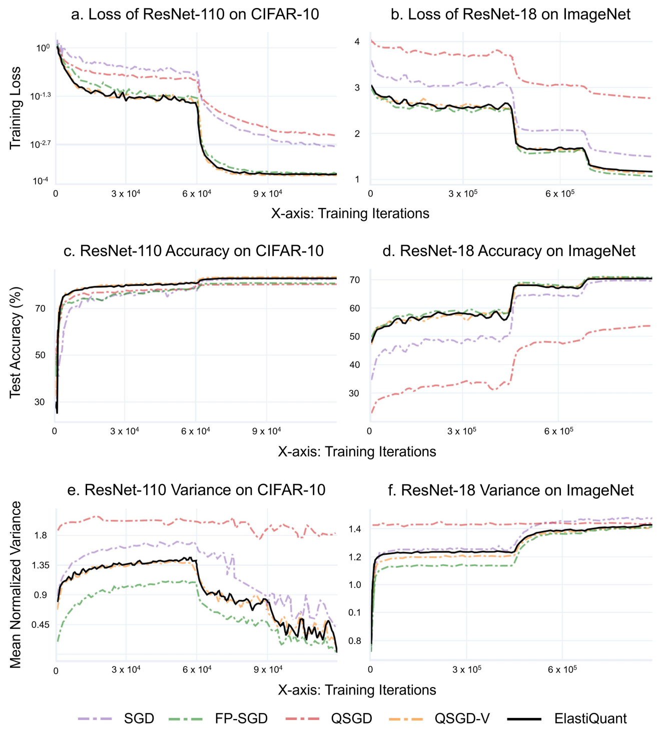

Figure 1: ElastiQuant evaluation and performance comparison by distributed training on edge GPUs: Training loss, test accuracy, and mean normalized variance.

3.6 Worst-case Variance Analysis

Generally, a worst-case analysis can be incorporated into any solution selection process for a robust system design. Here, an upper bound for tight worst-case variance is established by optimization over the distribution of normalized coordinates for an arbitrary sequence of quantization levels. Worst-case analysis helps gain insights on the behavior of the variance upper bound - by performing which, this section shows that ElastiQuant is nearly optimal in the worst-case. This section also extends the elastic quantization strategy from Section 3.1 that focuses on a special case of elastic quantization levels to the below arbitrary sequence of levels.

3.6.1 Generally Spaced Level. Space is a collection of elements from a set ($\mathcal{L}$ here) endowed with some features/structure that can be generally or exponentially spaced.

Let $\mathcal{L} = (l_0, l_1, \cdots, l_s, l_{s+1})$ denote an arbitrary sequence of quantization levels where $l_0 = 0 < l_1 < \cdots < l_{s+1} = 1$. Recall that in Section 3.1, for $r \in [0,1]$, $\tilde{s}(r)$ was defined to satisfy $l_{\tilde{s}(r)} \leq r \leq l_{\tilde{s}(r)+1}$,

Table 3: Evaluating scalability performance of ElastiQuant by distributed training on 2 to 7 devices, and comparison with SGD: Calculating speedup (*Sp*) and total time (*T*) per epoch in minutes - sum of computation (*Cp*), encoding (*En*), transmission (*Tx*).

Network, Dataset	Scheme	2 Edge GPUs				4 Edge GPUs					7 Edge GPUs				
		Cp	*En*	*Tx*	*T*	*Cp*	*En*	*Tx*	*T*	*Sp*	*Cp*	*En*	*Tx*	*T*	*Sp*
ResNet-34, ImageNet	SGD	16.06	NA	17.93	33.99	10.47	NA	20.62	31.09	-2.9 ↑	15.34	NA	26.84	42.18	11.09 ↓
	8-bit-ElastiQuant	14.61	4.15	16.16	34.92	7.98	4.04	11.92	23.94	-10.98 ↑	10.05	4.04	10.26	25.35	1.41 ↓
	4-bit-ElastiQuant	15.75	0.93	15.55	32.23	11.19	1.25	9.32	21.76	-10.47 ↑	9.64	1.34	7.47	18.45	-3.31 ↑
	E-ElastiQuant	14.72	1.34	15.24	31.3	10.57	1.55	8.29	20.41	-10.89 ↑	8.91	1.35	6.01	16.27	-4.14 ↑
ResNet-50, ImageNet	SGD	177.23	NA	222.53	399.76	167.9	NA	265.17	433.07	33.31 ↓	85.28	NA	465.06	550.34	117.27 ↓
	8-bit-ElastiQuant	179.21	38.65	190.55	409.09	145.25	39.97	145.25	330.49	-78.62 ↑	99.94	38.64	183.89	322.47	-8.02 ↑
	4-bit-ElastiQuant	179.89	9.33	183.89	373.11	142.58	8.12	118.59	269.17	-103.94 ↑	110.6	6.66	126.59	243.85	-25.32 ↑
	E-ElastiQuant	169.23	21.32	171.9	362.45	126.59	18.66	103.93	249.18	-113.27 ↑	119.93	19.99	78.62	218.54	-30.64 ↑

and $p(r)$ to satisfy $r = (1 - p(r))l_{\tilde{s}(r)} + p(r)l_{\tilde{s}(r)+1}$. Here, $\tau(r)$ is defined as $l_{\tilde{s}(r)+1} - l_{\tilde{s}(r)}$, with $\tilde{s}(r) \in 0, 1, \cdots, s$. So, $h_i(\mathbf{v}, s)$ can be defined in two cases depending on the quantization interval of r_i:

$$\text{If } r_i \in [0, l_1]\,;\, h_i(\mathbf{v}, s) = \begin{cases} 0 & \text{probability: } 1 - p_1\left(r_i, \mathcal{L}\right) \\ l_1 & \text{otherwise} \end{cases}$$
$$\text{Here } p_1(r, \mathcal{L}) = r/l_1$$

$$\text{If } r_i \in \left[l_{j-1}, l_j\right]\,;\, h_i(\mathbf{v}, s) = \begin{cases} l_{j-1} & \text{probability: } 1 - p_2\left(r_i, \mathcal{L}\right) \\ l_j & \text{otherwise} \end{cases}$$
$$\text{Here } j = 1, \cdots, s+1.\ p_2(r, \mathcal{L}) = \left(r - l_{j-1}\right)/\tau_{j-1}$$

When the elements from coordinates of vector $\mathbf{v}$ defined as $\mathcal{S}_j$ fall into the $(j+1)$ bin, then for $j = 0, \cdots, s$, $\mathcal{S}_j \triangleq \{i : r_i \in \left[l_j, l_{j+1}\right]\}$. Let $d_j \triangleq \left|\mathcal{S}_j\right|$. Following the variance bound Eqn (2), it can be shown that

$$\mathbb{E}\left[\|Q_s(\mathbf{v}) - \mathbf{v}\|^2\right] \leq \|\mathbf{v}\|^2 (\min \tau_0^2 d_0/4, \tau_0\sqrt{d_0}$$
$$+ \sum_{j=1}^{s} \min \tau_j^2 d_j/4, \tau_j(\sqrt{d_j} - l_j d_j))$$

Above is the ElastiQuant variance upper bound for generally spaced levels.

3.6.2 Exponentially Spaced Levels. In the case of exponentially spaced collection of levels $\mathcal{L}_p = \left(0, p^s, \cdots, p^2, p, 1\right)$ for $p \in (0, 1)$ with s as number of quantization levels in integer. Here, $\tau_0 = p^s$ and $\tau_j = (1-p)p^{s-j}$ for $j = 1, \cdots, s$. For any given s and d, roughly similar to the above subsection, the worst-case variance bound can be found. When the optimal value of p is searched, the worst-case variance is reduced for any given s and d.

ElastiQuant is analyzed by obtaining variance bound values by varying p in two setups. In the first setup, d is kept fixed at 10^4, s varying from 2 to 8. Here it was observed that variance upper bound decreases as s increases - the optimal p value shifts to the left as s increases. In the second setup, s is fixed at 6, varying d from 10^2 to 10^6. Here, variance upper bound increases as d increases - the optimal p value shifts to the right as d increases.

4 EXPERIMENTAL EVALUATION

In this section, we evaluate ElastiQuant that aims to:

- Investigate the effect of elastic quantization on solution quality by examining distributed training performance. Perform this by measuring variance, loss, speedups, accuracy for networks from the ResNet family when distributed training using ImageNet and CIFAR datasets.
- Justify the theoretical analysis by examination of the variance induced by ElastiQuant and relevant schemes in a decentralized IoT setting. Also, study their convergence in terms of loss and accuracy as the training progresses in iterations.
- Examine the distributed training in real-world scalability and performance of ElastiQuant. Check for improvements by comparing results with SGD variants, QSGD-V, Atomo, DGC, TernGrad, and others.

4.1 Setup

For experiments, to replicate the real-world heterogeneous IoT [22], 18 development boards ($K = 18$) were set up: 7 Jetson Xaviers, 4 Jetson Nanos inserted on a Jetson Mate carrier board, 3 accelerated Google Coral Dev boards, 4 Intel Movidius NCS accelerated Raspberry Pi 4. ElastiQuant evaluation, performance, and characteristics comparison with related schemes are performed by distributed model training on these 18 IoT edge GPU development boards. Portions of ImageNet, CIFAR-10, CIFAR-100 datasets are supplied to these boards for distributed training of ResNet family networks such as ResNet-18, ResNet-20, ResNet-34, ResNet-50, Resnet-110. ElastiQuant implementation is in TensorFLow since the TFLite version is well suited for edge GPUs and supports the used hardware accelerators. More resource-constrained small CPUs and AIoT boards can also be involved in learning by employing IoT hardware-friendly training algorithms like Train++ [19], ML-MCU [21]. The setup in the original papers of the related schemes uses cluster machines with slots containing few standard NVIDIA GPUs (see Table 1). Their setups are orthogonal to the ElastiQuant evaluation setup consisting of edge GPUs wirelessly interconnected using a local access point. Hence, the results from the original paper are not inherited. Instead, their proposed schemes are re-evaluated alongside ElastiQuant in the distributed setup with wireless edge GPUs. For improved readability, all graphs contain sampled and smoothed data.

4.2 Solution Quality: Loss, Accuracy & Variance

All selected schemes, including ElastiQuant are trained by quantizing and de-quantizing gradients shared by 18 edge GPUs. The FP-SGD is used as a baseline as it performs plain training without

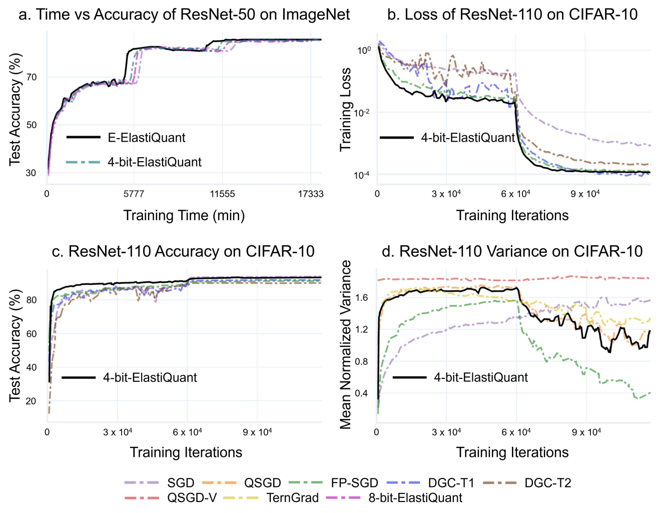

Figure 2: ElastiQuant variants evaluation and comparison of 4-bit-ElastiQuant with DGC, momentum tuned DGC variants, TernGrad, and other schemes.

quantization making it impractical in limited bandwidth IoT networks. SGD is shown to spotlight the significance of performance gaps between schemes. E.g., the gap between QSGD and QSGD-V that lack theoretical assurances.

Training Loss. Figure 1 a-b shows the training loss with 18 GPUs for both datasets - the following can be observed: On ImageNet, ElastiQuant, QSGD-V has lower loss compared to QSGD; On CIFAR-10, there is a significant gap in training loss which grows as training progresses.

Test Accuracy. Figure 1 c-d shows the test accuracy results for training ResNet from random initialization up to convergence - following can be observed: Similar to the training loss, performance gaps in test accuracy exist. Here, unlike ElastiQuant, QSGD does not achieve the accuracy of FP-SGD; For both datasets, ElastiQuant and QSGD-V outperform QSGD; The gap between ElastiQuant and QSGD is significant on ImageNet. This is because ElastiQuant and QSGD-V show lower variance in comparison to QSGD, which in fact can benefit both training loss and generalization error.

Quantization induced Variance. Variance of the gradient versus the training iterations is given in Figure 1 e-f. For both datasets, FP-SGD shows the lowest variance as it does not quantize gradients. For CIFAR-10, the variance of ElastiQuant and QSGD-V is lesser than SGD throughout the training, and it decreases at higher training iterations where the learning rate is dropped. For ImageNet, at reduced learning rates, the variance of ElastiQuant grows lesser in comparison to SGD. This observation shows that, in some cases, ElastiQuant can improve solution quality by achieving lower variance than the unrealistic baseline set by FP-SGD.

4.3 Scalability and Speedup Behavior

During practical distributed training, the advantage of ElastiQuant is it provides code-length bounds on the gradients transmitted, plus good accuracy. QSGD also provides such bounds, but it drops model quality in terms of accuracy (above subsection). The QSGD-V has lower loss and variance than QSGD but does not provide bounds on the expected code length. In addition to the 4-bit and 8-bit ElastiQuant variants, another variant named E-ElastiQuant is created for a detailed comparison of scalability and Speedup behavior during distributed training. E-ElastiQuant employs Huffman coding on gradients to encode the integer levels transmitted.

Scalability - Cost for Computation (*Cp*), Encoding (*En*), and Transmission (*Tx*). ResNet-34 and ResNet-50 are each trained three times on Jetson Xaviers (edge GPU) using the ImageNet dataset - the first time using 2 Xaviers, then 4 Xaviers, and finally using all 7 Xaviers. Scalability behavior can be quantified by first calculating the total distributed training time per epoch (T), which is the sum of Cp, En, Tx. Then, by subtracting T of schemes, the speedup (Sp) achieved due to scaling by adding extra edge GPUs can be obtained. Comparing scalability results of SGD, QSGD, QSGD-V with ElastiQuant variants, QSGD-V showed roughly similar performance as 4-bit-ElastiQuant, and QSGD shows under par convergence, so both omitted. The results are given in Table 3. In the Sp column of the table, the down-arrow represents negative scalability and positive for up-arrow. For SGD, negative scalability can be observed as, when device count increases from 4 to 7, instead of speedups, T increases from 31.09 min to 42.18 min for ResNet-34, and from 433.07 min to 550.34 min for ResNet-50; The 4-bit-ElastiQuant attains positive scaling for both ResNets. E.g., for ResNet-34, T is conserved by 10.47 min (1.48 times speedup) when devices scaled from 2 to 4, and 3.31 min (1.17 times speedup) when devices scaled from 4 to 7; The 8-bit-ElastiQuant faces a scalability stall (no speedups) when devices scaled from 4 to 7. This behavior is due to elevated encoding and communication costs; E-ElastiQuant shows top-of-the-class scalability and communication compression as T is conserved by 4.14 min for ResNet-34 and 30.64 min for ResNet-50 when devices scaled from 4 to 7.

Training Speedup and Accuracy. There is a minimal performance gap across all considered schemes in the 2 Xaviers setup and highest for 7 devices (see Table 3). So, the ideal training speedup and accuracy comparison approach is to use the 2 Xaviers setup for distributed training of ResNet-50 on ImageNet. The time versus accuracy results is given in Figure 2 a - the following can be observed: All ElastiQuant variants match the non-quantized ResNet-50 model accuracy, with speedups over the SGD (baseline). The QSGD-V result is not plotted as its performance overlaps 4-bit-ElastiQuant in the majority of points on the curve. EF-SignSGD needed intensive hyperparameter tuning to make it converge and bring up its accuracy close to other schemes that use standard hyperparameter settings. This tuning makes EF-SignSGD send additional scaling data, reducing parallelism and efficiency. Still, the speedup of tuned EF-SignSGD remains inferior to 8-bit ElastiQuant. In summary, E-ElastiQuant offers competitive performance while simultaneously providing convergence assurances, also additional communication bandwidth savings from its gradients encoding feature.

4.4 Results Comparison

Here we perform additional ElastiQuant evaluations and comparisons by including three related schemes.

Table 4: Accuracy comparison of ElastiQuant trained ResNets with models trained using DGC, compression ratio (CR) tuned DGC, Atomo, TernGrad, others. FP-SGD is baseline.

Network, Dataset	Scheme	Tune	Edge GPUs	Test Accuracy (%)
	Atomo			87.6
ResNet-20,	TernGrad	Default	2	No convergence
CIFAR-10	FP-SGD			90.5
	4-bit-ElastiQuant			86.3
		0.02 CR	2	91.25
	DGC		6	88.87
ResNet-18,		0.12 CR	2	90.08
CIFAR-10			6	87.36
	FP-SGD	Default	6	92.72
	4-bit-ElastiQuant		6	91.96
	DGC	0.02 CR	2	74.41
ResNet-18,			6	72.69
CIFAR-100	FP-SGD	Default	6	74.33
	4-bit-ElastiQuant			73.63
	SGD			89.76
	QSGD			89.22
ResNet-110,	QSGD-V	Default	6	90.10
CIFAR-10	FP-SGD			92.03
	TernGrad			91.33
	4-bit-ElastiQuant			90.80

ElastiQuant vs Deep Gradient Compression (DGC). DGC [14] is a sparsification method that leverages gradient clipping and momentum correction to improve accuracy. DGC was selected as it can show close to full-precision performance when carefully tuned. For schemes to have the same communication cost during distributed training, a closely similar compression ratio is set: 12 % for DGC, 4 bits for ElastiQuant. The result for ResNet-110 on CIFAR-10 is shown in Figure 2 b-c, to compare convergence and generalization of schemes. DGC needed careful hyperparameter tuning to not show noisy curves and get close to the full-precision performance. E.g., the learning rate is set to 0.06, and momentum tuned to 0.01 (to produce DGC-T1), 0.101 (to produce DGC-T2). For ElastiQuant, the hyperparameters from full-precision schemes can be reused with slight tuning. This easy implementation nature enables to achieve on-the-fly communication efficiency. Also, ElastiQuant can be viewed as a complementary strategy as it is compatible to be used as the encoding function for DGC and other schemes resulting in further reduction of communication costs.

The same schemes were also tested by varying the count of training involved devices and varying compression ratios (CR). ElastiQuant was set to use the same tuning of the FP-SGD baseline, and the CR hyperparameters for DGC were tuned to 2% (0.02) and 12% (0.12). The distributed training result in terms of accuracy is given in Table 4. Here, for ResNet-18 on both CIFAR-10 and CIFAR-100, unlike ElastiQuant, DGC accuracy degrades when training involved edge GPUs are scaled from 2 to 4. So, even if ElastiQuant could save less communication bandwidth than DGC, ElastiQuant is more practical due to its better scalability (see Table 3), enabling pooling more devices to complete training faster.

ElastiQuant vs ATOMO and TernGrad. For further comparison, ATOMO [30] and TernGrad [32] were considered for training ResNet-20, ResNet-110 on CIFAR-10 using standard parameters. This time the count of training involved devices was reduced from 18 to a maximum of 6 due to the inferior scalability characteristic (in the current wireless IoT setup) of new schemes. From the results in Table 4, for ResNet-20, although ATOMO shows slightly higher accuracy than ElastiQuant, ATOMO consumed higher time for each iteration as it computationally strains the edge GPUs set up for training. TernGrad convergence performance for ResNet-20 was under par for standard parameters, so additional tuning was performed to bring it close to ATOMO and ElastiQuant performance. For ResNet-110, TernGrad shows the closest performance to FP-SGD and slightly outperforms 4-bit-ElastiQuant. But from Figure 2 d, 4-bit-ElastiQuant show lower mean normalized variance than TernGrad and others.

5 CONCLUSION

This paper presented ElastiQuant to improve communication efficiency during distributed learning in IoT. During extensive evaluation by using ElastiQuant for distributed training of 5 ResNet variants on 18 wireless edge GPUs, ElastiQuant consistently demonstrated the following: Improved solution quality as the resultant ResNet models achieved lower loss and better accuracy; Higher training scalability and speedup due to reduced communication volume; Reduced quantization induced variance due to its elastic quantization approach; On-the-fly communication efficiency as ElastiQuant allows re-usage of parameters from full-precision schemes with only slight tuning. Such improvements on heterogeneous IoT hardware show ElastiQuant's robustness to system variability, which is vital to scaling distributed learning on ubiquitous resource-limited nodes in low bandwidth IoT networks. The future extended version of this paper plans to include proofs of the ElastiQuant's assurance on its bounds on variance and communication bits.

ACKNOWLEDGEMENT

Bharath is the corresponding author of this article. 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 European Regional Development Fund.

REFERENCES

[1] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. 2017. QSGD: Communication-efficient SGD via gradient quantization and encoding. *Neural Information Processing Systems (NIPS).*

[2] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. 2018. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning (ICML).*

[3] Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan. 2018. Adacomp: Adaptive residual gradient compression for data-parallel distributed training. In *AAAI Conference.*

[4] Wei Dai, Eric P Xing, et al. 2018. Toward understanding the impact of staleness in distributed machine learning. *arXiv preprint.*

[5] Christopher M De Sa, Ce Zhang, Kunle Olukotun, and Christopher Ré. 2015. Taming the wild: A unified analysis of hogwild-style algorithms. In *Neural Information Processing Systems (NIPS).*

[6] Fartash Faghri, Iman Tabrizian, Ilia Markov, Dan Alistarh, Daniel Roy, and Ali Ramezani-Kebrya. 2020. Adaptive Gradient Quantization for Data-Parallel SGD. In *Neural information processing systems (NIPS).*

[7] Samuel Horvath, Chen-Yu Ho, Ludovit Horvath, Atal Narayan Sahu, Marco Canini, and Peter Richtárik. 2019. Natural compression for distributed deep learning. *arXiv preprint*.

[8] Lu Hou and James T. Kwok. 2018. Loss-aware Weight Quantization of Deep Networks. In *6th International Conference on Learning Representations (ICLR)*.

[9] Arthur Jochems et al. 2016. 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*.

[10] Arthur Jochems, Timo M Deist, et al. 2017. Developing and validating a survival prediction model for NSCLC patients through distributed learning across 3 countries. *Journal of Radiation Oncology Biology Physics*.

[11] Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian Stich, and Martin Jaggi. 2019. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning (ICML)*.

[12] Edward H Lee, Daisuke Miyashita, Elaina Chai, Boris Murmann, and S Simon Wong. 2017. Lognet: Energy-efficient neural networks using logarithmic computation. In *IEEE ICASSP*.

[13] Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. 2018. ADPSGD Asynchronous decentralized parallel stochastic gradient descent. In *International Conference on Machine Learning (ICML)*.

[14] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. 2017. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint*.

[15] Ali Ramezani-Kebrya, Fartash Faghri, Ilya Markov, Vitalii Aksenov, Dan Alistarh, and Daniel M Roy. 2021. NUQSGD: Provably Communication-efficient Data-parallel SGD via Nonuniform Quantization. *Journal of Machine Learning Research*.

[16] Felix Sattler, Simon Wiedemann, and Wojciech Samek. 2019. Sparse binary compression: Towards distributed deep learning with minimal communication. In *International Joint Conference on Neural Network (IJCNN)*.

[17] Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 2014. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Interspeech*.

[18] Nikko Strom. 2015. Scalable distributed DNN training using commodity GPU cloud computing. In *International Speech Communication Association (ISCA)*.

[19] Bharath Sudharsan, Muhammad Intizar Ali, et al. 2021. Train++: An Incremental ML Model Training Algorithm to Create Self-Learning IoT Devices. In *18th IEEE International Conference on Ubiquitous Intelligence and Computing*.

[20] Bharath Sudharsan, John G Breslin, and Muhammad Intizar Ali. 2021. Globe2Train: A Framework for Distributed ML Model Training using IoT Devices Across the Globe. In *Proceedings of the 18th IEEE International Conference on Ubiquitous Intelligence and Computing (UIC)*.

[21] Bharath Sudharsan, John G Breslin, and Muhammad Intizar Ali. 2021. ML-MCU: A Framework to Train ML Classifiers on MCU-based IoT Edge Devices. In *IEEE Internet of Things Journal*.

[22] Bharath Sudharsan, Pankesh Patel, John G Breslin, and Muhammad Intizar Ali. 2021. Enabling Machine Learning on the Edge using SRAM Conserving Efficient Neural Networks Execution Approach. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*.

[23] Bharath Sudharsan, Omer Rana, Pankesh Patel, John Breslin, Muhammad Intizar Ali, Karan Mitra, Schahram Dustdar, Prem Prakash, and Rajiv Ranjan. 2021. Towards Distributed, Global, Deep Learning using IoT Devices. *IEEE Internet Computing*.

[24] Bharath Sudharsan, Dhruv Sheth, Shailesh Arya, Federica Rollo, Piyush Yadav, Pankesh Patel, John G Breslin, and Muhammad Intizar Ali. 2021. Elasticl: Elastic quantization for communication efficient collaborative learning in iot. In *The 19th ACM Conference on Embedded Networked Sensor Systems (SenSys)*.

[25] Jun Sun, Tianyi Chen, Georgios B Giannakis, and Zaiyue Yang. 2019. Communication-efficient distributed learning via lazily aggregated quantized gradients. In *Neural Information Processing Systems (NIPS)*.

[26] Hanlin Tang, Shaoduo Gan, Ce Zhang, Tong Zhang, and Ji Liu. 2018. Communication Compression for Decentralized Training. In *Neural information processing systems (NIPS)*.

[27] Hanlin Tang, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. 2018. D2: Decentralized Training over Decentralized Data. *arXiv preprint*.

[28] Kasturi Vasudevan. 2021. Pulse Code Modulation. In *Analog Communications*. Springer.

[29] Thijs Vogels, Sai Praneeth Karinireddy, and Martin Jaggi. 2019. PowerSGD: Practical low-rank gradient compression for distributed optimization. *Neural Information Processing Systems (NIPS)*.

[30] Hongyi Wang, Scott Sievert, Shengchao Liu, Zachary Charles, Dimitris Papailiopoulos, and Stephen Wright. 2018. ATOMO: Communication-efficient Learning via Atomic Sparsification. In *Neural Information Processing Systems (NIPS)*.

[31] Jianyu Wang and Gauri Joshi. 2019. Adaptive Communication Strategies for Best Error-Runtime Trade-offs in Communication-Efficient Distributed SGD. In *Neural Information Processing Systems (NIPS)*.

[32] Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2017. Terngrad: Ternary gradients to reduce communication in distributed deep learning. In *Neural Information Processing Systems (NIPS)*.

[33] Shuchang Zhou, Yuheng Zou, et al. 2016. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint*.