DAdaQuant: Doubly-adaptive quantization for communication-efficient Federated Learning

Robert Hönig RH723@CL.CAM.AC.UK
Yiren Zhao YIREN.ZHAO@CL.CAM.AC.UK
Robert Mullins ROBERT.MULLINS@CL.CAM.AC.UK
University of Cambridge, United Kingdom

Abstract

Federated Learning (FL) is a powerful technique to train a model on a server with data from several clients in a privacy-preserving manner. FL incurs significant communication costs because it repeatedly transmits the model between the server and clients. Recently proposed algorithms quantize the model parameters to efficiently compress FL communication. We find that dynamic adaptations of the quantization level can boost compression without sacrificing model quality. We introduce DAdaQuant as a doubly-adaptive quantization algorithm that dynamically changes the quantization level across time and different clients. Our experiments show that DAdaQuant consistently improves client→server compression, outperforming the strongest non-adaptive baselines by up to $2.8\times$.

1. Introduction

Federated Learning (FL) has emerged as a privacy-preserving technique to train models on data from many devices [9, 11]. It assumes a client-server topology with a set $\mathbb{C} = \{c_i | i \in \{1, 2...N\}\}$ of N clients that are connected to a single server. Each client c_k has a local dataset D_k from the local data distribution $\mathcal{D}_k$. Given a model M with parameters $\boldsymbol{p}$, a loss function $f_{\boldsymbol{p}}(d \in D_k)$ and the local loss $F_k(\boldsymbol{p}) = \frac{1}{|D_k|} \sum_{d \in D_k} f_{\boldsymbol{p}}(d)$, FL seeks to minimize the global loss $G(\boldsymbol{p}) = \sum_{k=1}^{N} \frac{|D_k|}{\sum_l |D_l|} F_k(\boldsymbol{p})$.

To this end, the server initializes the model with parameters $\boldsymbol{p}_0$. In round t, the server sends $\boldsymbol{p}_t$ to a random subset $\mathbb{S}_t$ of K clients to optimize their local objectives $F_k(\boldsymbol{p}_t)$ and send the updated model parameters $\boldsymbol{p}_{t+1}^k$ back to the server. The server accumulates all parameters into the new global model $\boldsymbol{p}_{t+1} = \sum_{k \in \mathbb{S}_t} \frac{|D_k|}{\sum_j |D_j|} \boldsymbol{p}_{t+1}^k$ and starts the next round. This procedure enables the server to train a model without accessing any local datasets [11]. FL communication is expensive and can account for over 70% of its energy consumption [14]. Reducing FL communication is an attractive optimisation because it lowers bandwidth usage, energy consumption and training time.

Communication in FL happens in two phases: Sending parameters from the server to clients (*downlink*) and sending updated parameters from clients to the server (*uplink*). Uplink bandwidth usually imposes a tighter bottleneck than downlink bandwidth because of connection asymmetry [19] and cheap downlink broadcasting [2, 15]. Therefore, we compress uplink communication.

FL research has explored several approaches to reduce communication. Algorithms like FedAvg [11] and FOLB [12] try to converge in fewer rounds. PruneFL [8] and AFD [4] prune the model or only train on part of the whole model to reduce the number of parameters that have to be communicated. Finally, it is possible to directly compress FL training communication. To this end,

techniques such as top-k sparsification [10, 16] and quantization [15, 18] are commonly applied to parameter updates, optionally followed by lossless compression.

Our work applies to quantization-based compression algorithms for FL that apply some lossy quantizer $\mathtt{Q}$ to the model parameters. $\mathtt{Q}$ usually provides a "quantization level" hyperparameter q to control the coarseness of quantization. When q is kept constant during training, we speak of *static quantization*. When q changes, we speak of *adaptive quantization*.

Adaptive quantization can exploit asymmetries in the FL framework to minimize communication. One such asymmetry lies in FL's training time, where we observe that with *time-adaptive quantization*, early training rounds can use a lower q without affecting convergence (see Figure 4). Another asymmetry lies in FL's client space, because most FL algorithms weight client contributions to the global model proportional to their local dataset sizes. Intuitively, FL clients with greater weighting should have a greater communication budget and our proposed *client-adaptive quantization* achieves this in a principled way (see Figure 3).

Adaptive quantization has been explored by Shen et al. [17] who adapt the quantization level for different model parameters. FracTrain [6] combines this approach with time-adaptive quantization. However, both algorithms are for non-distributed training. AdaQuantFL [7] introduces time-adaptive quantization to FL, but is impractical for realistic FL settings because its communication costs grow linearly with the number of clients (see Appendix C.6 for an empirical validation). In contrast, DAdaQuant scales to arbitrarily many clients.

In this paper, we first create Federated QSGD as an adaption of the gradient compression algorithm QSGD [1] that works with FL. We then introduce the concept of client-adaptive quantization and develop algorithms for client- and time-adaptive quantization that are computationally efficient, empirically Pareto optimal and compatible with arbitrary FL quantizers. Finally, we combine time- and client-adaptive quantization into DAdaQuant and show that it significantly boosts the compression of Federated QSGD, outperforming competing FL compression algorithms.

2. The DAdaQuant method

Federated QSGD While DAdaQuant can be applied to any quantizer with a configurable quantization level, it is optimized for fixed-point quantization. Fixed-point quantization uses a quantizer $\mathtt{Q}_q$ with quantization level q that splits $\mathbb{R}_{\geq 0}$ and $\mathbb{R}_{\leq 0}$ into q intervals each. $\mathtt{Q}_q(p)$ then returns the sign of p and $|p|$ rounded to one of the endpoints of its encompassing interval. $\mathtt{Q}_q(\boldsymbol{p})$ quantizes the vector $\boldsymbol{p}$ elementwise.

QSGD combines fixed-point quantization with 0 run-length encoding and Elias ω coding into a competitive fixed-point quantizer. QSGD has been designed specifically for quantizing gradients. This makes it not directly applicable to parameter compression. To overcome this limitation, we apply difference coding to uplink compression, first introduced to FL in [15]. Each client c_k applies $\mathtt{Q}_q$ to the *parameter updates* $\boldsymbol{p}_{t+1}^k - \boldsymbol{p}_t$. The server accumulates the quantized parameter updates into the new parameters as $\boldsymbol{p}_{t+1} = \boldsymbol{p}_t + \sum_{k \in \mathbb{S}_t} \frac{|D_k|}{\sum_l |D_l|} \mathtt{Q}_q(\boldsymbol{p}_{t+1}^k - \boldsymbol{p}_t)$. We find that QSGD works well with parameter updates, which can be regarded as an accumulation of gradients over several training steps. We call this adaptation of QSGD *Federated QSGD* and use it with DAdaQuant in our experiments.

Time-adaptive quantization Time-adaptive quantization uses a different quantization level q_t for each round t of FL training. DAdaQuant chooses q_t to minimize communication costs without

Client	Samples	Round 1	2	3	4	5
A	1				8	
B	2	8	8	8	8	
C	3	8		8		8
D	4		8			8

(a) Static quantization.

Client	Samples	Round 1	2	3	4	5
A	1				4	
B	2	1	2	2	4	
C	3	1		2		8
D	4		2			8

(b) Time-adaptive quantization.

Client	Samples	Round 1	2	3	4	5
A	1				6	
B	2	7	6	7	9	
C	3	9		9		7
D	4		9			9

(c) Client-adaptive quantization.

Client	Samples	Round 1	2	3	4	5
A	1				3	
B	2	1	1	2	5	
C	3	1		2		7
D	4		1			9

(d) Time-adaptive and client-adaptive quantization.

Figure 1: Examples of static and adaptive quantization level assignments for four FL clients (A, B, C, D) that train over five rounds. Each round, two clients are selected for training.

sacrificing accuracy. To this end, we find that lower quantization levels suffice to initially reduce the loss, while partly trained models require higher quantization levels to further improve (as illustrated in Figure 4). FracTrain is built on similar observations for non-distributed training. Therefore, we design DAdaQuant to mimic FracTrain in monotonically increasing q_t as a function of t and using the training loss to inform increases in q_t.

When q is too low, FL converges prematurely. Like FracTrain, DAdaQuant monitors the FL loss and increases q when it converges. Unlike FracTrain, there is no single centralized loss function to evaluate and accumulating the global loss $G(\boldsymbol{p}_t)$ like AdaQuantFL would dramatically increase communication (see Appendix C.6). Instead, we estimate $G(\boldsymbol{p}_t)$ as the average local loss $\hat{G}_t = \sum_{k \in \mathbb{S}_t} \frac{|D_k|}{\sum_l |D_l|} F_k(\boldsymbol{p}_t)$ where $\mathbb{S}_t$ is the set of clients sampled at round t. Since $\mathbb{S}_t$ typically consists of only a small fraction of all clients, $\hat{G}_t$ is a very noisy estimate of $G(\boldsymbol{p}_t)$. This makes it unsuitable for convergence detection. Instead, DAdaQuant tracks a running average loss $\hat{\hat{G}}_t = \psi \hat{\hat{G}}_{t-1} + (1-\psi)\hat{G}_t$.

We initialize $q_1 = q_{\text{min}}$ for some $q_{\text{min}} \in \mathbb{N}$. DAdaQuant determines training to converge whenever $\hat{\hat{G}}_t \geq \hat{\hat{G}}_{t+1-\phi}$ for some $\phi \in \mathbb{N}$ that specifies the number of rounds across which we compare $\hat{\hat{G}}$. On convergence, DAdaQuant sets $q_t = 2q_{t-1}$ and keeps the quantization level fixed for ϕ rounds to enable reductions in G to manifest in $\hat{\hat{G}}$. Eventually, the training loss converges regardless of the quantization level. To avoid unconstrained quantization level increases, we limit the quantization level to q_{max}. The following equation summarizes DAdaQuant's time-adaptive quantization:

$$q_t \longleftarrow \begin{cases} q_{\text{min}} & t = 0 \\ 2q_{t-1} & t > 0 \text{ and } \hat{\hat{G}}_{t-1} \geq \hat{\hat{G}}_{t-\phi} \text{ and } t > \phi \text{ and } 2q_{t-1} \leq q_{\text{max}} \text{ and } q_{t-1} = q_{t-\phi} \\ q_{t-1} & \text{else} \end{cases}$$

Client-adaptive quantization FL algorithms typically accumulate each parameter p_i over all clients into a weighted average $p = \sum_{i=1}^{K} w_i p_i$ (see Algorithm 1). Quantized FL communicates and accumulates quantized parameters $\mathtt{Q}_q(p) = \sum_{i=1}^{K} w_i \mathtt{Q}_q(p_i)$ where q is the quantization level. We define the quantization error e_p^q as $e_p^q = |p - \mathtt{Q}_q(p)|$. We find that $\mathbb{E}_{p_1 \ldots p_K}[\text{Var}(\mathtt{Q}_q(p))]$ is a useful statistic of the quantization error because it strongly correlates with the loss added by quantization. For a stochastic, unbiased fixed-point compressor like Federated QSGD, $\mathbb{E}_{p_1 \ldots p_K}[\text{Var}(\mathtt{Q}_q(p))]$ equals $\mathbb{E}_{p_1 \ldots p_K}[\text{Var}(e_p^q)]$ and can be evaluated analytically.

We observe in our experiments that communication per client is roughly a linear function of Federated QSGD's quantization level q. This means that communication per round is proportional to $Q = Kq$. We call Q the communication budget and use it as a proxy measure of communication.

Client-adaptive quantization dynamically adjusts the quantization level of each client. This means that even within a single round, each client c_k can be assigned a different quantization level q_k. The communication budget of client-adaptive quantization is then $Q = \sum_{k=1}^{K} q_k$ and $\mathtt{Q}_q(p)$ generalizes to $\mathtt{Q}_{q_1 \dots q_K}(p) = \sum_{i=1}^{K} w_i \mathtt{Q}_{q_i}(p_i)$. We devise an algorithm that chooses q_k to minimize Q subject to $\mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^{q_1 \dots q_K})] = \mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^{q})]$ for a given q. Thus, our algorithm effectively minimizes communication costs while maintaining a quantization error similar to static quantization. Theorem 1 provides us with an analytical formula for quantization levels $q_1 \dots q_K$.

Theorem 1 *Given parameters $p_1 \dots p_k \sim \mathcal{U}[-t, t]$ and quantization level q, $\min_{q_1 \dots q_K} \sum_{i=1}^{K} q_i$ subject to $\mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^{q_1 \dots q_K})] = \mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^{q})]$ is minimized by $q_i = \sqrt{\frac{a}{b}} \times w_i^{2/3}$ where $a = \sum_{j=1}^{K} w_j^{2/3}$ and $b = \sum_{j=1}^{K} \frac{w_j^2}{q^2}$.*

DAdaQuant applies Theorem 1 to lower communication costs while maintaining the same loss as static quantization does with a fixed q. To ensure that quantization levels are natural numbers, DAdaQuant approximates the optimal real-valued solution as $q_i = \max(1, \mathrm{round}(\sqrt{\frac{a}{b}} \times w_i^{2/3}))$. Appendix D gives a detailed proof of Theorem 1. To the best of our knowledge, DAdaQuant is the first algorithm to use client-adaptive quantization.

Doubly-adaptive quantization (DAdaQuant) DAdaQuant combines the time-adaptive and client-adaptive quantization algorithms described above. At each round t, time-adaptive quantization determines a preliminary quantization level q_t. Client-adaptive quantization then finds the client quantization levels q_t^k, $k \in \{1, \dots, K\}$ that minimize $\sum_{k=1}^{K} q_t^k$ subject to $\mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^{q_t^1 \dots q_t^K})] = \mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^{q_t})]$. Algorithm 1 lists DAdaQuant in detail. Figure 1 gives an example of how our time-adaptive, client-adaptive and doubly-adaptive quantization algorithms set quantization levels.

Reisizadeh et al. [15] prove the convergence of FL with quantization for convex and non-convex cases as long as the quantizer $\mathtt{Q}$ is (1) unbiased and (2) has a bounded variance. While this convergence result has been derived for a static quantization level, it readily extends to DAdaQuant's dynamic quantization with Federated QSGD: It suffices to show that (1) and (2) are satisfied for DAdaQuant's minimum quantization level $q = 1$ and conclude that (1) and (2) are still satisfied for $q > 1$ because the quantizer remains unbiased and decreases in variance.

The computational overhead of DAdaQuant is dominated by an additional evaluation epoch per round per client to compute $\hat{\tilde{G}}_t$, which is negligible when training for many epochs each round.

3. Experiments

3.1. Experimental details

Evaluation We use DAdaQuant with Federated QSGD to train different models with FL on different datasets for a fixed number of rounds. We monitor the test loss and accuracy at fixed intervals and measure uplink communication at every round across all devices. Appendix C.1 gives a full description of all hyperparameters, including the parameterization of DAdaQuant.

Models & datasets We select a broad and diverse set of five models and datasets to demonstrate the general applicability of DAdaQuant. To this end, we use DAdaQuant to train a linear model,

	Synthetic		FEMNIST		Sent140	
Uncompressed	78.3 ± 0.3	12.2 MB	77.7 ± 0.4	132.1 GB	69.7 ± 0.5	43.9 GB
Federated QSGD	-0.1 ± 0.1	$17\times$	$+0.7 \pm 0.5$	$2809\times$	-0.0 ± 0.5	$90\times$
FP8	$\mathbf{+0.1 \pm 0.4}$	$4.0\times$ ($0.23\times\!\!\times$)	-0.1 ± 0.4	$4.0\times$ ($0.00\times\!\!\times$)	-0.2 ± 0.5	$4.0\times$ ($0.04\times\!\!\times$)
FedPAQ (FxPQ)	-0.1 ± 0.1	$6.4\times$ ($0.37\times\!\!\times$)	$+0.7 \pm 0.5$	$11\times$ ($0.00\times\!\!\times$)	-0.0 ± 0.5	$4.0\times$ ($0.04\times\!\!\times$)
FxPQ + GZip	-0.1 ± 0.1	$14\times$ ($0.82\times\!\!\times$)	$+0.6 \pm 0.2$	$1557\times$ ($0.55\times\!\!\times$)	-0.0 ± 0.6	$71\times$ ($0.79\times\!\!\times$)
UVeQFed	-0.5 ± 0.2	$0.6\times$ ($0.03\times\!\!\times$)	-2.8 ± 0.5	$12\times$ ($0.00\times\!\!\times$)	$+0.0 \pm 0.2$	$15\times$ ($0.16\times\!\!\times$)
DAdaQuant	-0.2 ± 0.4	$\mathbf{48\times}$ ($\mathbf{2.81\times\!\!\times}$)	$+0.7 \pm 0.1$	$\mathbf{4772\times}$ ($\mathbf{1.70\times\!\!\times}$)	-0.1 ± 0.4	$\mathbf{108\times}$ ($\mathbf{1.19\times\!\!\times}$)
DAdaQuant$_{\text{time}}$	-0.1 ± 0.5	$37\times$ ($2.16\times\!\!\times$)	$\mathbf{+0.8 \pm 0.2}$	$4518\times$ ($1.61\times\!\!\times$)	-0.1 ± 0.6	$93\times$ ($1.03\times\!\!\times$)
DAdaQuant$_{\text{clients}}$	$+0.0 \pm 0.3$	$26\times$ ($1.51\times\!\!\times$)	$+0.7 \pm 0.4$	$3017\times$ ($1.07\times\!\!\times$)	$\mathbf{+0.1 \pm 0.6}$	$105\times$ ($1.16\times\!\!\times$)

Table 1: Top-1 test accuracies and total communication of all baselines, DAdaQuant, DAdaQuant$_{\text{time}}$ and DAdaQuant$_{\text{clients}}$. Entry $x \pm y \quad p\times$ ($q\times\!\!\times$) denotes an accuracy difference of x% w.r.t. the uncompressed accuracy with a standard deviation of y%, a compression factor of p w.r.t. the uncompressed communication and a compression factor of q w.r.t. Federated QSGD. Appendix C.3 shows the full results, including the datasets Celeba and Shakespeare.

CNNs and LSTMs of varying complexity on a federated synthetic dataset (Synthetic), as well as two federated image datasets (FEMNIST and CelebA) and two federated natural language datasets (Sent140 and Shakespeare) from the LEAF [5] project for standardized FL research. We refer to Appendix C.2 for more information on the models, datasets, training objectives and implementation.

System heterogeneity In practice, FL has to cope with clients that have different compute capabilities. We follow Li et al. [9] and simulate this *system heterogeneity* by randomly reducing the number of epochs to E' for a random subset $\mathbb{S}'_t \subset \mathbb{S}_t$ of clients at each round t, where E' is sampled from $[1, \ldots, E]$ and $|\mathbb{S}'_t| = 0.9K$. Like Li et al., we also add the proximal term $\frac{\mu}{2}\|\boldsymbol{p}^k_{t+1} - \boldsymbol{p}_t\|^2$ to the local objective $F_k(\boldsymbol{p}^k_{t+1})$ to improve FL convergence under system heterogeneity.

Baselines We compare DAdaQuant against competing quantization-based algorithms, namely Federated QSGD, FedPAQ, GZip with fixed-point quantization (FxPQ + GZip), UVeQFed and FP8. Federated QSGD (see Section 2) is our most important baseline because it outperforms the other algorithms. FedPAQ only applies fixed-point quantization, which is equivalent to Federated QSGD without lossless compression. Similarly, FxPQ + GZip is equivalent to Federated QSGD with gzip for its lossless compression stages. UVeQFed [18] generalizes scalar quantization to vector quantization, followed by arithmetic coding. FP8 [20] is a floating-point quantizer that uses an 8-bit floating-point format designed for storing neural network gradients. We also evaluate all experiments without compression to establish an accuracy benchmark.

3.2. Results

We repeat the main experiments three times and report average results and their standard deviation. Table 1 shows the highest accuracy and total communication for each experiment. Figure 2 plots the maximum accuracy achieved for any given amount of communication.

Baselines Table 1 shows that the accuracy of most experiments lies within the margin of error of the uncompressed experiments. This reiterates the viability of quantization-based compression algorithms for communication reduction in FL. For all experiments, Federated QSGD achieves a significantly higher compression factor than the other baselines. The authors of FedPAQ and UVeQFed also compare their methods against QSGD and report them as superior. However, FedPAQ is

compared against "unfederated" QSGD that communicates gradients after each local training step and UVeQFed is compared against QSGD without its lossless compression stages.

Time- and client-adaptive quantization The purely time-adaptive and client-adaptive versions of DAdaQuant, DAdaQuant$_{\text{time}}$ and DAdaQuant$_{\text{clients}}$, outperform Federated QSGD and the other baselines in Table 1, achieving comparable accuracies while lowering communication costs. Unsurprisingly, the performance of DAdaQuant$_{\text{clients}}$ is correlated with the coefficient of variation $c_v = \frac{\sigma}{\mu}$ of the numbers of samples in the local datasets with mean μ and standard deviation σ: Synthetic ($c_v = 3.3$) has a significantly higher compression factor than Sent140 ($c_v = 0.3$) and FEMNIST ($c_v = 0.4$). Appendix C.3 shows similar results for Celeba ($c_v = 0.3$) and Shakespeare ($c_v = 1.7$)

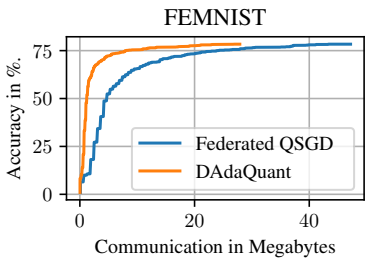


Figure 2: Communication-accuracy trade-off curves for training on FEMNIST with Federated QSGD and DAdaQuant. We plot the average highest accuracies for any given amount of communication. Appendix C.4 shows similar results for all datasets.

DAdaQuant DAdaQuant outperforms DAdaQuant$_{\text{time}}$ and DAdaQuant$_{\text{clients}}$ in communication while achieving similar accuracies. The compression factors of DAdaQuant are roughly multiplicative in those of DAdaQuant$_{\text{clients}}$ and DAdaQuant$_{\text{time}}$. This demonstrates that we can effectively combine time- and client-adaptive quantization for maximal communication savings.

Pareto optimality Figure 2 shows that DAdaQuant achieves a higher accuracy than the strongest baseline, Federated QSGD, for any fixed amount of communication. This means that DAdaQuant is Pareto optimal for the datasets we have explored.

4. Conclusion

We introduced DAdaQuant as a computationally efficient and robust algorithm to boost the performance of quantization-based FL compression algorithms. We showed intuitively and mathematically how DAdaQuant's dynamic adjustments of the quantization level across time and clients minimize client→server communication while maintaining convergence speed. Our experiments establish DAdaQuant as nearly universally superior over static quantizers, achieving state-of-the-art compression factors when applied to Federated QSGD. The communication savings of DAdaQuant effectively lower FL bandwidth usage, energy consumption and training time.

References

[1] Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. QSGD: Communication-efficient SGD via gradient quantization and encoding. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1709–1720. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6768-qsgd-communication-efficient-sgd-via-gradient-quantization-and-encoding.pdf.

[2] Mohammad Mohammadi Amiri, Deniz Gunduz, Sanjeev R. Kulkarni, and H. Vincent Poor. Federated learning with quantized global model updates. arXiv:2006.10672, 2020.

[3] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. Flower: A friendly Federated Learning research framework. arXiv:2007.14390, 2020.

[4] Nader Bouacida, Jiahui Hou, Hui Zang, and Xin Liu. Adaptive Federated Dropout: Improving communication efficiency and generalization for federated learning. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2021.

[5] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A benchmark for federated settings. arXiv:1812.01097, 2018.

[6] Yonggan Fu, Haoran You, Yang Zhao, Yue Wang, Chaojian Li, Kailash Gopalakrishnan, Zhangyang Wang, and Yingyan Lin. FracTrain: Fractionally squeezing bit savings both temporally and spatially for efficient DNN training. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12127–12139. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/8dc5983b8c4ef1d8fcd5f325f9a65511-Paper.pdf.

[7] Divyansh Jhunjhunwala, Advait Gadhikar, Gauri Joshi, and Yonina C Eldar. Adaptive quantization of model updates for communication-efficient federated learning. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3110–3114. IEEE, 2021.

[8] Yuang Jiang, Shiqiang Wang, Bong Jun Ko, Wei-Han Lee, and Leandros Tassiulas. Model pruning enables efficient Federated Learning on edge devices. arXiv:1909.12326, 2019.

[9] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. arXiv:1812.06127, 2018.

[10] Amirhossein Malekijoo, Mohammad Javad Fadaeieslam, Hanieh Malekijou, Morteza Homayounfar, Farshid Alizadeh-Shabdiz, and Reza Rawassizadeh. FEDZIP: A compression framework for communication-efficient Federated Learning. arXiv:2102.01593, 2021.

[11] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[12] Hung T Nguyen, Vikash Sehwag, Seyyedali Hosseinalipour, Christopher G Brinton, Mung Chiang, and H Vincent Poor. Fast-convergent federated learning. *IEEE Journal on Selected Areas in Communications*, 39(1):201–218, 2020.

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach,

H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[14] Xinchi Qiu, Titouan Parcolle, Daniel J Beutel, Taner Topa, Akhil Mathur, and Nicholas D Lane. A first look into the carbon footprint of federated learning. *arXiv preprint arXiv:2010.06537*, 2020.

[15] Amirhossein Reisizadeh, Aryan Mokhtari, Hamed Hassani, Ali Jadbabaie, and Ramtin Pedarsani. FedPAQ: A communication-efficient Federated Learning method with periodic averaging and quantization. arXiv:1909.13014, 2019.

[16] Daniel Rothchild, Ashwinee Panda, Enayat Ullah, Nikita Ivkin, Ion Stoica, Vladimir Braverman, Joseph Gonzalez, and Raman Arora. FetchSGD: Communication-efficient federated learning with sketching. In *International Conference on Machine Learning*, pages 8253–8265. PMLR, 2020.

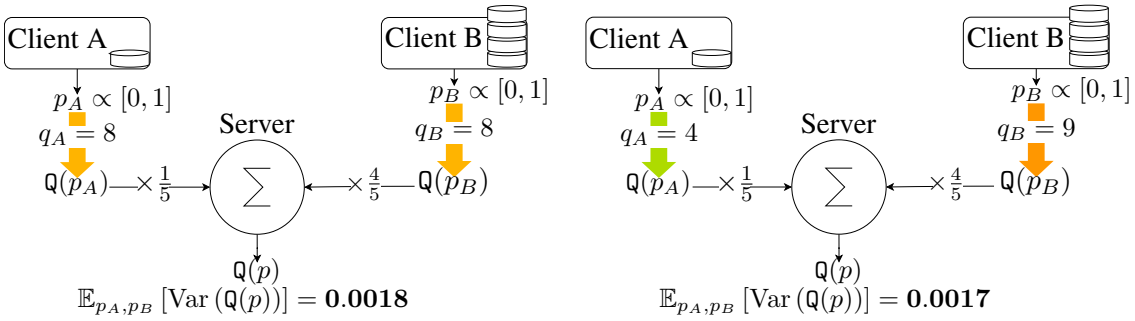
[17] Jianghao Shen, Yue Wang, Pengfei Xu, Yonggan Fu, Zhangyang Wang, and Yingyan Lin. Fractional skipping: Towards finer-grained dynamic CNN inference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5700–5708, 2020.

[18] Nir Shlezinger, Mingzhe Chen, Yonina C Eldar, H Vincent Poor, and Shuguang Cui. UVeQFed: Universal vector quantization for federated learning. *IEEE Transactions on Signal Processing*, 69:500–514, 2020.

[19] Speedtest. Speedtest global index. https://www.speedtest.net/global-index. Accessed: 2021-05-12.

[20] Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7675–7684. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7994-training-deep-neural-networks-with-8-bit-floating-point-numbers.pdf.

Appendix A. Supplementary illustrations



(a) Static quantization. (b) Client-adaptive quantization.

Figure 3: Static quantization vs. client-adaptive quantization when accumulating parameters p_A and p_B. (a): Static quantization uses the same quantization level for p_A and p_B. (b) Client-adaptive quantization uses a slightly higher quantization level for p_B because p_B is weighted more heavily. This allows q_A to use a significantly lower quantization level while keeping the quantization error measure $\mathrm{E}_{p_A,p_B}[\mathrm{Var}(\mathtt{Q}(p))]$ roughly constant. Since communication is approximately proportional to $q_A + q_B$, client-adaptive quantization communicates less data.

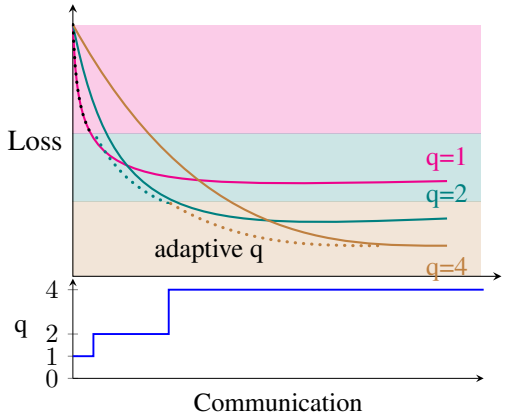


Figure 4: Time-adaptive quantization. A small quantization level (q) decreases the loss with less communication than a large q, but converges to a higher loss. This motivates an adaptive quantization strategy that uses a small q as long as it is beneficial and then switches over to a large q. We generalize this idea into an algorithm that monotonically increases q based on the training loss.

Appendix B. Pseudocode of DAdaQuant

Algorithm 1: The DAdaQuant algorithm. The uncolored lines list the standard FL training routine described in Section 1. Adding the colored lines creates DAdaQuant. — quantization, — client-adaptive quantization, — time-adaptive quantization.

Function `RunServer()`
  Initialize $w_i = \frac{|D_i|}{\sum_j |D_j|}$ for all $i \in [1, \ldots, N]$;
  for $t = 0, \ldots, T-1$ **do**
    Choose $\mathbb{S}_t \subset \mathbb{C}$ with $|\mathbb{S}_t| = K$, including each $c_k \in \mathbb{C}$ with uniform probability;
    $q_t \longleftarrow \begin{cases} q_{\min} & t = 0 \\ 2q_{t-1} & t > 0 \text{ and } \hat{\hat{G}}_{t-1} \geq \hat{\hat{G}}_{t-\phi} \text{ and } t > \phi \text{ and } q_t \leq q_{\max} \text{ and } q_{t-1} = q_{t-\phi} \\ q_{t-1} & \text{else} \end{cases}$;
    for $c_k \in \mathbb{S}_t$ **do** in parallel
      $q_t^k \longleftarrow \sqrt{\sum_{j=1}^K w_j^{2/3} / \sum_{j=1}^K \frac{w_j^2}{q^2}}$;
      $Send(c_k, \boldsymbol{p}_t, q_t^k)$;
      $Receive(c_k, \boldsymbol{p}_{t+1}^k, \hat{G}_t^k)$;
    end
    $\boldsymbol{p}_{t+1} \longleftarrow \sum_{k \in \mathbb{S}_t} w_k \boldsymbol{p}_{t+1}^k$;
    $\hat{G}_t \longleftarrow \sum_{k \in \mathbb{S}_t} w_k \hat{G}_t^k$;
    $\hat{\hat{G}}_t \longleftarrow \begin{cases} \hat{G}_0 & t = 0 \\ \psi \hat{\hat{G}}_{t-1} + (1-\psi)\hat{G}_t & \text{else} \end{cases}$;
  end
end
Function `RunClient(`c_k`)`
  $Receive(\text{Server}, \boldsymbol{p}_t, q_t^k)$;
  $\hat{G}_t^k \longleftarrow F_k(\boldsymbol{p}_t)$;
  $\boldsymbol{p}_{t+1}^k \longleftarrow F_k(\boldsymbol{p}_{t+1}^k)$ trained with SGD for E epochs with learning rate η;
  $Send(\text{Server}, \mathtt{Q}_{q_t^k}(\boldsymbol{p}_{t+1}^k), \hat{G}_t^k)$;
end

Appendix C. Additional simulation details and experiments

C.1. Hyperparameters

With the exception of CelebA, all our datasets and models are also used by Li et al.. We therefore adopt most of the hyperparameters from Li et al. and use LEAF's hyperparameters for CelebA [5]. For all experiments, we sample 10 clients each round. We train Synthetic, FEMNIST and CelebA for 500 rounds each. We train Sent140 for 1000 rounds due to slow convergence and Shakespeare for 50 rounds due to rapid convergence. We use batch size 10, learning rates 0.01, 0.003, 0.3, 0.8, 0.1 and μs (proximal term coefficients, see Section 3.1) 1, 1, 1, 0.001, 0 for Synthetic, FEMNIST,

Sent140, Shakespeare, CelebA respectively. We randomly split the local datasets into 80% training set and 20% test set.

To select the quantization level q for static quantization with Federated QSGD, FedPAQ and FxPQ + GZip, we run a gridsearch over $q = 1, 2, 4, 8, \ldots$ and choose for each dataset the lowest q for which Federated QSGD exceeds uncompressed training in accuracy. We set UVeQFed's "coding rate" hyperparameter $R = 4$, which is the lowest value for which UVeQFed achieves negligible accuracy differences compared to uncompressed training. We set the remaining hyperparameters of UVeQFed to the optimal values reported by its authors. Appendix C.5 shows further experiments that compare against UVeQFed with R chosen to maximize its compression factor.

For DAdaQuant's time-adaptive quantization, we set ψ to 0.9, ϕ to $1/10^{th}$ of the number of rounds and q_{max} to the quantization level q for each experiment. For Synthetic and FEMNIST, we set q_{min} to 1. We find that Sent140, Shakespeare and CelebA require a high quantization level to achieve top accuracies and/or converge in few rounds. This prevents time-adaptive quantization from increasing the quantization level quickly enough, resulting in prolonged low-precision training that hurts model performance. To counter this effect, we set q_{min} to $q_{\text{max}}/2$. This effectively results in binary time-adaptive quantization with an initial low-precision phase with $q = q_{\text{max}}/2$, followed by a high-precision phase with $q = q_{\text{max}}$.

For our experiments in Appendix C.6, AdaQuantFL requires a hyperparameter s that determines the initial quantization level. We set s to 2, the optimal value reported by the authors of AdaQuantFL. The remaining hyperparameters are identical to those used for the Synthetic dataset experiments in Table 1.

C.2. Models, datasets, trainining objectives and implementation details

Here, we give detailed information on the models, datasets, training objectives and implementation that we use for our experiments. We set the five following FL tasks:

- Multinomial logistic regression (MLR) on a synthetic dataset called Synthetic that contains vectors in $\mathbb{R}^{60}$ with a label of one out of 10 classes. We use the synthetic dataset generator in Li et al. [9] to generate synthetic datasets. The generator samples Synthetic's local datasets and labels from MLR models with randomly initialized parameters. For this purpose, parameters α and β control different kinds of data heterogeneity. α controls the variation in the local models from which the local dataset labels are generated. β controls the variation in the local dataset samples. We set $\alpha = 1$ and $\beta = 1$ to simulate an FL setting with both kinds of data heterogeneity. This makes Synthetic a useful testbed for FL.
- Character classification into 62 classes of handwritten characters from the FEMNIST dataset using a CNN. FEMNIST groups samples from the same author into the same local dataset.
- Smile detection in facial images from the CelebA dataset using a CNN. CelebA groups samples of the same person into the same local dataset. We note that LEAF's CNN for CelebA uses BatchNorm layers. We replace them with LayerNorm layers because they are more amenable to quantization. This change does not affect the final accuracy.
- Binary sentiment analysis of tweets from the Sent140 dataset using an LSTM. Sent140 groups tweets from the same user into the same local dataset. The majority of local datasets in the raw Sent140 dataset only have a single sample. This impedes FL convergence. Therefore, we filter Sent140 to clients with at least 10 samples (i.e. one complete batch). Caldas et al. [5], Li et al. [9] similarly filter Sent140 for their FL experiments.

- Next character prediction on text snippets from the Shakespeare dataset of Shakespeare's collected plays using an LSTM. Shakespeare groups lines from the same character into the same local dataset.

Table 2 provides statistics of our models and datasets.

We implement the models with PyTorch [13] and use Flower [3] to simulate the FL server and clients.

Dataset	**Model**	**Weights**	**Clients**	**Samples**	**Samples per client**			
					mean	**min**	**max**	**stddev**
Synthetic	MLR	610	30	9,600	320.0	45	5,953	1051.6
FEMNIST	2-layer CNN	6.6×10^6	3,500	785,582	224.1	19	584	87.8
CelebA	4-layer CNN	6.3×10^5	9,343	200,288	21.4	5	35	7.6
Sent140	2-layer LSTM	1.1×10^6	21,876	430,707	51.1	10	549	17.1
Shakespeare	2-layer LSTM	1.3×10^5	1,129	4,226,158	3743	3	66,903	6212

Table 2: Statistics of the models and datasets used for evaluation. MLR stands for "Multinomial Logistic Regression".

C.3. Complete table of test accuracies and total training communication

	Synthetic		**FEMNIST**		**Sent140**	
Uncompressed	78.3 ± 0.3	12.2 MB	77.7 ± 0.4	132.1 GB	69.7 ± 0.5	43.9 GB
Federated QSGD	-0.1 ± 0.1	$17\times$	$+0.7 \pm 0.5$	$2809\times$	-0.0 ± 0.5	$90\times$
FP8	$\mathbf{+0.1 \pm 0.4}$	$4.0\times$ ($0.23\ll$)	-0.1 ± 0.4	$4.0\times$ ($0.00\ll$)	-0.2 ± 0.5	$4.0\times$ ($0.04\ll$)
FedPAQ (FxPQ)	-0.1 ± 0.1	$6.4\times$ ($0.37\ll$)	$+0.7 \pm 0.5$	$11\times$ ($0.00\ll$)	-0.0 ± 0.5	$4.0\times$ ($0.04\ll$)
FxPQ + GZip	-0.1 ± 0.1	$14\times$ ($0.82\ll$)	$+0.6 \pm 0.2$	$1557\times$ ($0.55\ll$)	-0.0 ± 0.6	$71\times$ ($0.79\ll$)
UVeQFed	-0.5 ± 0.2	$0.6\times$ ($0.03\ll$)	-2.8 ± 0.5	$12\times$ ($0.00\ll$)	$+0.0 \pm 0.2$	$15\times$ ($0.16\ll$)
DAdaQuant	-0.2 ± 0.4	$\mathbf{48\times}$ ($\mathbf{2.81\ll}$)	$+0.7 \pm 0.1$	$\mathbf{4772\times}$ ($\mathbf{1.70\ll}$)	-0.1 ± 0.4	$\mathbf{108\times}$ ($\mathbf{1.19\ll}$)
DAdaQuant$_{\text{time}}$	-0.1 ± 0.5	$37\times$ ($2.16\ll$)	$\mathbf{+0.8 \pm 0.2}$	$4518\times$ ($1.61\ll$)	-0.1 ± 0.6	$93\times$ ($1.03\ll$)
DAdaQuant$_{\text{clients}}$	$+0.0 \pm 0.3$	$26\times$ ($1.51\ll$)	$+0.7 \pm 0.4$	$3017\times$ ($1.07\ll$)	$\mathbf{+0.1 \pm 0.6}$	$105\times$ ($1.16\ll$)

	Shakespeare		**Celeba**	
Uncompressed	$\mathbf{49.9 \pm 0.3}$	267.0 MB	90.4 ± 0.0	12.6 GB
Federated QSGD	-0.5 ± 0.6	$9.5\times$	-0.1 ± 0.1	$648\times$
FP8	-0.2 ± 0.4	$4.0\times$ ($0.42\ll$)	$\mathbf{+0.0 \pm 0.1}$	$4.0\times$ ($0.01\ll$)
FedPAQ (FxPQ)	-0.5 ± 0.6	$3.2\times$ ($0.34\ll$)	-0.1 ± 0.1	$6.4\times$ ($0.01\ll$)
FxPQ + GZip	-0.5 ± 0.6	$9.3\times$ ($0.97\ll$)	-0.1 ± 0.2	$494\times$ ($0.76\ll$)
UVeQFed	-0.0 ± 0.4	$7.9\times$ ($0.83\ll$)	-0.4 ± 0.3	$31\times$ ($0.05\ll$)
DAdaQuant	-0.6 ± 0.5	$\mathbf{21\times}$ ($\mathbf{2.21\ll}$)	-0.1 ± 0.1	$\mathbf{775\times}$ ($\mathbf{1.20\ll}$)
DAdaQuant$_{\text{time}}$	-0.5 ± 0.5	$12\times$ ($1.29\ll$)	-0.1 ± 0.2	$716\times$ ($1.10\ll$)
DAdaQuant$_{\text{clients}}$	-0.4 ± 0.5	$16\times$ ($1.67\ll$)	-0.1 ± 0.0	$700\times$ ($1.08\ll$)

Table 3: Top-1 test accuracies and total communication of all baselines, DAdaQuant, DAdaQuant$_{\text{time}}$ and DAdaQuant$_{\text{clients}}$. Entry $x \pm y$ $p\times$ ($q\ll$) denotes an accuracy difference of x% w.r.t. the uncompressed accuracy with a standard deviation of y%, a compression factor of p w.r.t. the uncompressed communication and a compression factor of q w.r.t. Federated QSGD.

C.4. Complete communication-accuracy trade-off curves

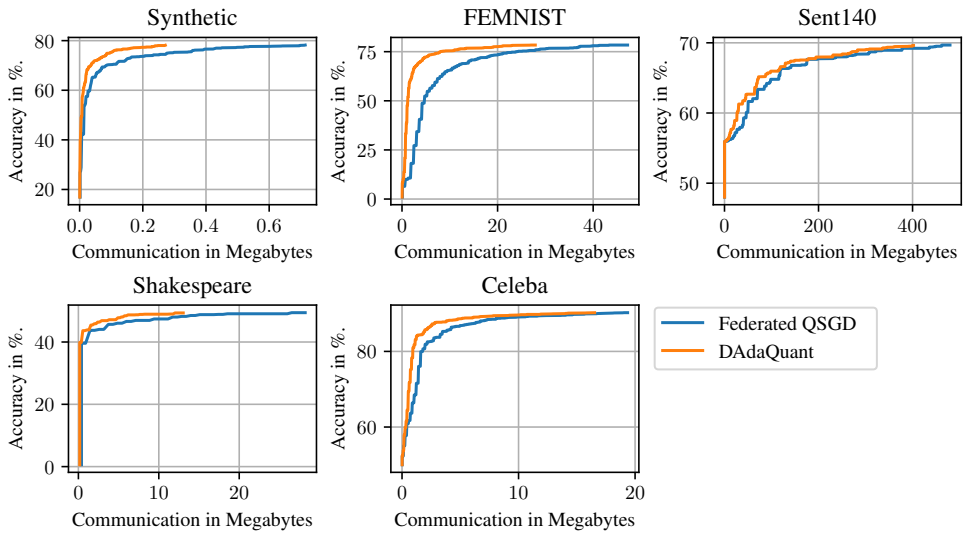


Figure 5: Communication-accuracy trade-off curves for QSGD and DAdaQuant. We plot the average highest accuracies achieved up to any given amount of communication.

C.5. Additional UVeQFed experiments

To demonstrate that the choice of UVeQFed's "coding rate" hyperparameter R does not affect our findings on the superior compression factors of DAdaQuant, we re-evaluate UVeQFed with $R = 1$, which maximizes UVeQFed's compression factor. Our results in Table 4 show that with the exception of Shakespeare, DAdaQuant still achieves considerably higher compression factors than UVeQFed.

	Synthetic	**FEMNIST**	**Sent140**	**Shakespeare**	**Celeba**
Uncompressed	12.2 MB	132.1 GB	43.9 GB	267.0 MB	12.6 GB
QSGD	17×	2809×	90×	9.5×	648×
UVeQFed (R=4)	0.6× (0.03 ≪)	12× (0.00 ≪)	15× (0.16 ≪)	7.9× (0.83 ≪)	31× (0.05 ≪)
UVeQFed (R=1)	13× (0.77 ≪)	34× (0.01 ≪)	41× (0.45 ≪)	**21× (2.22 ≪)**	93× (0.14 ≪)
DAdaQuant	**48× (2.81 ≪)**	**4772× (1.70 ≪)**	**108× (1.19 ≪)**	21× (2.21 ≪)	**775× (1.20 ≪)**

Table 4: Comparison of the compression factors of DAdaQuant, UVeQFed with $R = 4$ (default value used for our experiments in Table 1) and UVeQFed with $R = 1$ (maximizes UVeQFed's compression factor). Entry $p\times$ $(q \ll)$ denotes a compression factor of p w.r.t. the uncompressed communication and a compression factor of q w.r.t. Federated QSGD.

C.6. AdaQuantFL vs. DAdaQuant

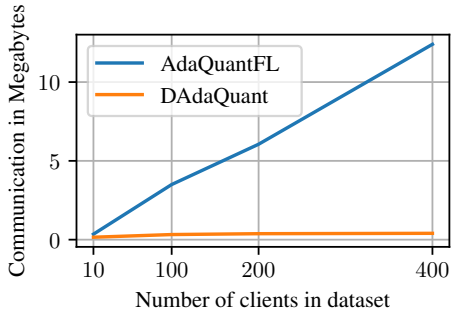


Figure 6: Comparison of AdaQuantFL and DAdaQuant. We plot the total communication required to train an MLR model on synthetic datasets with 10, 100, 200 and 400 clients. AdaQuantFL's communication increases linearly with the number of clients because it trains the model on all clients at each round to compute the global training loss. This makes AdaQuantFL prohibitively expensive for datasets with thousands of clients such as Celeba and Sent140. In contrast, DAdaQuant's communication does not change with the number of clients.

Appendix D. Proofs

Lemma 2 *For arbitrary $t > 0$ and parameter $p_i \in [-t, t]$, let $s_i = \frac{t}{q_i}$, $b_i = \operatorname{rem}(p_i, s_i)$ and $u_i = s_i - b_i$. Then,* $\operatorname{Var}\left(Q_{q_i}(p_i)\right) = u_i b_i$.

Proof

$$
\begin{aligned}
& \operatorname{Var}\left(\mathtt{Q}_{q_i}(p_i)\right) \\
= \; & \mathrm{E}\left[\left(\mathtt{Q}_{q_i}(p_i) - \mathrm{E}\left[\mathtt{Q}_{q_i}(p_i)\right]\right)^2\right] \\
= \; & \mathrm{E}\left[\left(\mathtt{Q}_{q_i}(p_i) - p_i\right)^2\right] && \mathtt{Q}_{q_i}(p_i) \text{ is an unbiased estimator of } p_i \\
= \; & \frac{b_i}{s_i} u_i^2 + \frac{u_i}{s_i} b_i^2 && \text{cf. fig. 7} \\
= \; & \frac{u_i b_i}{s_i}\left(u_i + b_i\right) \\
= \; & u_i b_i
\end{aligned}
$$

■

Lemma 3 *Let $\mathtt{Q}$ be a fixed-point quantizer. Assume that parameters $p_1 \dots p_K$ are sampled from $\mathcal{U}[-t, t]$ for arbitrary $t > 0$. Then, $\mathbb{E}_{p_1 \dots p_K}[\operatorname{Var}(e_p^{q_1 \dots q_K})] = \frac{t^2}{6} \sum_{i=1}^{K} \frac{w_i^2}{q_i^2}$.*

Proof

$\mathbb{E}_{p_1 \dots p_K}[\operatorname{Var}(e_p)]$

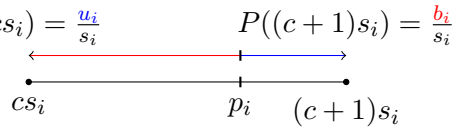


Figure 7: Illustration of the Bernoulli random variable $\mathtt{Q}_{q_i}(p_i)$ in Lemma 2. s_i is the length of the quantization interval. p_i is rounded up to $(c+1)s_i$ with a probability proportional to its distance from cs_i.

$$= \frac{1}{2t}\int_{-t}^{t}\frac{1}{2t}\int_{-t}^{t}\dots\frac{1}{2t}\int_{-t}^{t}\mathrm{Var}\left(\sum_{i=1}^{K}w_i\mathtt{Q}_{q_i}(p_i)-p\right)\,dp_1dp_2\dots dp_K$$

$$= \frac{1}{t}\int_{0}^{t}\frac{1}{t}\int_{0}^{t}\dots\frac{1}{t}\int_{0}^{t}\mathrm{Var}\left(\sum_{i=1}^{K}w_i\mathtt{Q}_{q_i}(p_i)-p\right)\,dp_1dp_2\dots dp_K \qquad \text{symmetry of } \mathtt{Q}_{q_i}(p_i) \text{ w.r.t. negation}$$

$$= \frac{1}{t^n}\int_{0}^{t}\int_{0}^{t}\dots\int_{0}^{t}\sum_{i=1}^{K}w_i^2\mathrm{Var}\left(\mathtt{Q}_{q_i}(p_i)\right)\,dp_1dp_2\dots dp_K \qquad \text{mutual independence of } \mathtt{Q}_{q_i}(p_i)\ \forall i$$

$$= \frac{1}{t^n}\sum_{i=1}^{K}\int_{0}^{t}\int_{0}^{t}\dots\int_{0}^{t}w_i^2\mathrm{Var}\left(\mathtt{Q}_{q_i}(p_i)\right)\,dp_1dp_2\dots dp_K \qquad \text{exchangeability of finite sums and integrals}$$

$$= \frac{1}{t^n}\sum_{i=1}^{K}t^{n-1}\int_{0}^{t}w_i^2\mathrm{Var}\left(\mathtt{Q}_{q_i}(p_i)\right)\,dp_i$$

$$= \frac{1}{t}\sum_{i=1}^{K}w_i^2\int_{0}^{t}\mathrm{Var}\left(\mathtt{Q}_{q_i}(p_i)\right)\,dp_i$$

$$= \frac{1}{t}\sum_{i=1}^{K}w_i^2\int_{0}^{t}u_ib_i\,dp_i \qquad \text{Lemma 2}$$

$$= \frac{1}{t}\sum_{i=1}^{K}w_i^2q_i\int_{0}^{s_i}u_ib_i\,dp_i \qquad s_i\text{-periodicity of } u_i \text{ and } b_i$$

$$= \frac{1}{t}\sum_{i=1}^{K}w_i^2q_i\int_{0}^{s_i}(s_i-p_i)\,p_i\,dp_i$$

$$= \frac{1}{6t}\sum_{i=1}^{K}w_i^2q_is_i^3$$

$$= \frac{t^2}{6}\sum_{i=1}^{K}\frac{w_i^2}{q_i^2}$$

■

Lemma 4 *Let Q be a fixed-point quantizer. Assume that parameters $p_1 \dots p_K$ are sampled from $\mathcal{U}[-t,t]$ for arbitrary $t > 0$. Then, $\min_{q_1 \dots q_K} \mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^{q_1 \dots q_K})]$ subject to $Q = \sum_{i=1}^K q_i$ is minimized by $q_i = Q \frac{w_i^{2/3}}{\sum_{k=1}^K w_k^{2/3}}$.*

Proof Define

$$
\begin{aligned}
f(\boldsymbol{q}) &= \mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^{q_1 \dots q_K})] \\
g(\boldsymbol{q}) &= \left(\sum_{i=1}^n q_i \right) \\
\mathcal{L}(\boldsymbol{q}) &= f(\boldsymbol{q}) - \lambda g(\boldsymbol{q}) \text{ (Lagrangian)}
\end{aligned}
$$

Any (local) minimum $\hat{\boldsymbol{q}}$ satisfies

$$
\begin{aligned}
& \nabla \mathcal{L}(\hat{\boldsymbol{q}}) = \mathbf{0} \\
\iff & \nabla \frac{t^2}{6} \sum_{i=1}^K \frac{w_i^2}{q_i^2} - \lambda \nabla \sum_{i=1}^K q_i = 0 \wedge \sum_{i=1}^K q_i = Q && \text{Lemma 3} \\
\iff & \forall i = 1 \dots n.\ \frac{t^2}{-3} \frac{w_i^2}{q_i^3} = \lambda \wedge \sum_{i=1}^K q_i = Q \\
\iff & \forall i = 1 \dots n.\ q_i = \sqrt[3]{\frac{t^2}{-3\lambda} w_i^2} \wedge \sum_{i=1}^K q_i = Q \\
\implies & \forall i = 1 \dots n.\ q_i = Q \frac{w_i^{2/3}}{\sum_{j=1}^K w_j^{2/3}}
\end{aligned}
$$

■

D.1. Proof of Theorem 1

Proof Using Lemma 4, it is straightforward to show that for any V, $\min_{q_1 \dots q_K} \sum_{i=1}^K q_i$ subject to $\mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^{q_1 \dots q_K})] = V$ is minimized by $q_i = C w_i^{2/3}$ for the unique $C \in \mathbb{R}_{>0}$ that satisfies $\mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^{q_1 \dots q_K})] = V$.

Then, taking $V = \mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^q)]$ and $C = \sqrt{\frac{a}{b}}$ (see Theorem 1), we do indeed get

$$
\begin{aligned}
& \mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^{q_1 \dots q_K})] \\
= & \frac{t^2}{6} \sum_{i=1}^K \frac{w_i^2}{\left(C w_i^{2/3}\right)^2} && \text{Lemma 3} \\
= & \frac{1}{C^2} \frac{t^2}{6} \sum_{i=1}^K {w_i}^{2/3}
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{\sum_{j=1}^{K} \frac{w_j^2}{q^2}}{\sum_{j=1}^{K} w_j^{2/3}} \frac{t^2}{6} \sum_{i=1}^{K} {w_i}^{2/3} \\
&= \frac{t^2}{6} \sum_{j=1}^{K} \frac{w_j^2}{q^2} \\
&= \mathbb{E}_{p_1 \dots p_K}[\mathrm{Var}(e_p^q)] && \text{Lemma 3}
\end{aligned}
$$

■