Optimal Quantization using Scaled Codebook

Yerlan Idelbayev[*]
University of California, Merced
yidelbayev@ucmerced.edu

Pavlo Molchanov
NVIDIA Corporation
pmolchanov@nvidia.com

Maying Shen
NVIDIA Corporation
mshen@nvidia.com

Hongxu Yin
NVIDIA Corporation
dannyy@nvidia.com

Miguel Á. Carreira-Perpiñán
University of California, Merced
mcarreira-perpinan@ucmerced.edu

Jose M. Alvarez
NVIDIA Corporation
josea@nvidia.com

Abstract

We study the problem of quantizing N sorted, scalar datapoints with a fixed codebook containing K entries that are allowed to be rescaled. The problem is defined as finding the optimal scaling factor α and the datapoint assignments into the α-scaled codebook to minimize the squared error between original and quantized points. Previously, the globally optimal algorithms for this problem were derived only for certain codebooks (binary and ternary) or under the assumption of certain distributions (Gaussian, Laplacian). By studying the properties of the optimal quantizer, we derive an $\mathcal{O}(NK \log K)$ algorithm that is guaranteed to find the optimal quantization parameters for any fixed codebook regardless of data distribution. We apply our algorithm to synthetic and real-world neural network quantization problems and demonstrate the effectiveness of our approach.

1. Introduction

The appealing efficacy of modern deep neural networks in a wide spectrum of tasks generally comes with a seemingly sharp increase in network complexity. The resulting computation burden hinges network deployment to real-world tasks that typically face stringent resources and constraints, e.g., mobile applications or autonomous driving. A myriad of approaches has thus been proposed to reduce network redundancy, quantization being the most popular and simple-to-use approach. The central question behind quantization is around weight simplification: how one could reassign a large number of high-precision weights from the network, to a far less diverse set of points specified by a codebook, hence drastically cut down on required compute. Despite remarkable insights into this problem, theoretical analysis lacks far behind, leaving no viable guarantees to optimality of any sort even in the substeps of this process.

In this work we consider a general formulation of the scaled codebook quantization problem: given a sorted[1] data vector $\mathbf{w}$ with elements $w_1 \leq w_2 \leq \cdots \leq w_N$ and a fixed codebook $\mathcal{C} = \{c_1, c_2, \dots, c_K\} \subset \mathbb{R}$ such that $c_1 < c_2 < \cdots < c_K$ we would like to learn the optimal rescaling of the codebook (by a scalar $\alpha > 0$) and the corresponding assignments of datapoints into the codebook entries so that the ℓ_2 quantization error (MSE) is minimized:

$$\begin{aligned} \min_{\alpha, \mathbf{z}_1, \dots, \mathbf{z}_N} \quad & \mathrm{Loss}(\alpha, \mathbf{Z}) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} (w_n - \alpha\, c_k)^2 \\ \text{s.t.} \quad & \mathbf{z}_n^T \mathbf{1} = 1, \quad \mathbf{z}_n \in \{0,1\}^K \end{aligned} \tag{1}$$

Here, $\mathbf{z}_n$ is a binary assignment vector defined for each datapoint w_n: if $z_{nk} = 1$ then the datapoint w_n is assigned to the codebook entry c_k.

The problems of this type arise in various settings of signal processing and have recently gained a significant interest in the field of neural network compression. For instance, if we set $\mathcal{C} = \{0, \pm 1, \pm 2 ... \pm 2^7\}$ we recover the symmetric variant of INT8 quantization scheme of Jacob et al. [1] which is particularly advantageous for efficient inference, and currently has become a standard part of deep-learning frameworks with many accompanying studies and implementations [2, 3]. With a codebook of $\mathcal{C} = \{-1, 0, 1\}$ we recover the scaled ternary quantization scheme, and if we set $\mathcal{C} = \{0, \pm 2^1, \pm 2^2, \dots, \pm 2^b\}$ we recover a scaled powers-of-two quantization scheme.

Previously, it was believed that the general formulation of scaled quantization problem (1) is hard to optimize [4] and requires exponential-time algorithm [5]. Therefore, most of the approaches in the literature used heuristic search or alternating optimization without any optimality guarantees. In this paper, we present an optimal algorithm that

[*]Work performed during a Summer Internship at NVIDIA.

[1]We do not include the cost of sorting into our analysis.

is guaranteed to find the solution of (1) in $\mathcal{O}(NK \log K)$ time using $\mathcal{O}(N)$ memory. While optimal algorithms are known for special types of codebooks (see sec. 2), to the best of our knowledge, this is the first result with a generic algorithm that *can handle any codebook* $\mathcal{C}$. Our approach is based on studying the properties of the optimal quantizer of (1) and leveraging possible shortcuts which we present in sections 3–4. The numerical experiments are presented in section 5.

2. Related work

The optimality properties of scalar MSE quantization with adaptive codebook (where entries of $\mathcal{C}$ are learned) have been studied by Lloyd [6] in the context of pulse-code modulation. Lloyd gave the closed-form solutions for quantizers of Gaussian and Laplacian distributions and introduced an alternating optimization algorithm.

Unlike globally optimal solutions, the alternating optimization (alt-opt) algorithms converge to a local minima of the problem where no further improvement is possible. Yet, it is a leading method in solving most of the special cases of (1). For instance, Hwang and Sung [4] used the alt-opt algorithm for a case of $\mathcal{C} = \{-1, 0, 1\}$, Anwar et al. [7] used it for the uniform integer codebooks, and Leng et al. [8] employed it for the powers-of-two codebooks.

For some specific cases of problem (1) the optimal algorithms have been derived: Rastegari et al. [9] gave a solution for scaled binary quantization where $\mathcal{C} = \{-1, 1\}$, Carreira-Perpiñán and Idelbayev [10] and Yin et al. [11] gave a solution for optimal scaled ternarization with $\mathcal{C} = \{-1, 0, 1\}$. However, these algorithms cannot be generalized for the arbitrary codebooks and it is unclear how to extend them. Our paper closes this gap.

2.1. Optimization of scaled INT8 quantization

The scaled INT8 quantization where datapoints are quantized into the codebook of $\mathcal{C} = \{0, \pm 1, \pm 2, \dots, \pm 2^7\}$ with appropriate rescaling parameter α has gained a significant interest in neural network compression field where the problem of (1) arises in a post-training quantization setting [3, 12, 13, 14, 15] or as a proximal step in the quantization-aware training setting [1, 4, 7, 10, 16].

The solutions to the INT8 version of (1) available in the literature can be divided into the following categories: alternating optimization solutions [4, 5, 7, 16, 17], heuristics based on maximum values [1, 3] or percentiles [3], grid search search techniques [4, 7, 18, 19], and analytical solutions assuming a certain distribution on datapoints [12, 15, 20, 21, 22]. None of these approaches, except for the finely-spaced grid search, can guarantee a global optimum of INT8 quantization problem on arbitrary data. However, an exhaustive sweep through the entire search space for α-values is expensive; thus, some approximations are used: Hwang and Sung [4] first find locally optimal solution using alt-opt and then improve it by a limited grid search; Choukroun et al. [19] fix the number of points in the grid; Liu et al. [18] give a heuristic rule on how finely to space the grid.

2.2. Adaptive codebook for scalar quantization

If in (1) we allow the codebook entries to be learned as well, we recover the adaptive codebook quantization scheme which has been extensively studied in the literature. In fact, it becomes a $1d$ version of the k-means problem; however, unlike the higher dimensional cases, which are NP-hard, the $1d$ version admits an efficient optimal solution. Bruce [23] gave a $\mathcal{O}(NK^2)$ optimal algorithm using dynamic programming (DP) for the $1d$ k-means. Wu and Rokne [24] improved Bruce's DP algorithm to have a runtime of $\mathcal{O}(NK \log K)$ using divide-and-conquer approach, and Wu [25] further reduced the runtime to $\mathcal{O}(NK)$ using a matrix search technique.

3. Characterization of the optimum

3.1. Locally optimal scale

For fixed values of the assignments vectors $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_N\}$ we can solve for a locally optimal rescaling by minimizing the objective of (1) wrt α:

$$\min_{\alpha} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk}(w_n - \alpha c_k)^2 \Longleftrightarrow \min_{\alpha} \sum_{k=1}^{K} \sum_{w_n \in c_k} (w_n - \alpha c_k)^2$$

Here, the notation $w_n \in c_k$ means all the points w_n assigned to the codebook value c_k, i.e., those having $z_{nk} = 1$. With fixed $\mathbf{Z}$, the objective is convex wrt α, thus by setting the derivative to zero and solving for α we get $\mathbf{Z}$-induced optimal scaling factor:

$$\alpha^*(\mathbf{Z}) = \frac{\sum_{k=1}^{K} \sum_{w_n \in c_k} w_n c_k}{\sum_{k=1}^{K} \sum_{w_n \in c_k} c_k^2}. \tag{2}$$

We will be referring to eq. (2) as $\textsc{OptScale}(\mathbf{Z})$.

3.2. Locally optimal assignments

Given a fixed value of α, we can recover the locally optimal assignments by solving (1) wrt $\mathbf{z}_1, \dots, \mathbf{z}_N$:

$$\min_{\mathbf{z}_1, \dots, \mathbf{z}_N} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk}(w_n - \alpha\, c_k)^2 \quad \text{s.t.} \quad \mathbf{z}_n^T \mathbf{1} = 1, \;\; \mathbf{z}_n \in \{0, 1\}^K.$$

This separates over every $(w_n, \mathbf{z}_n)$-pair into N problems of:

$$\min_{\mathbf{z}_n} \sum_{k=1}^{K} z_{nk}(w_n - \alpha\, c_k)^2 \;\; \text{s.t.} \;\; \mathbf{z}_n^T \mathbf{1} = 1, \;\; \mathbf{z}_n \in \{0, 1\}^K,$$

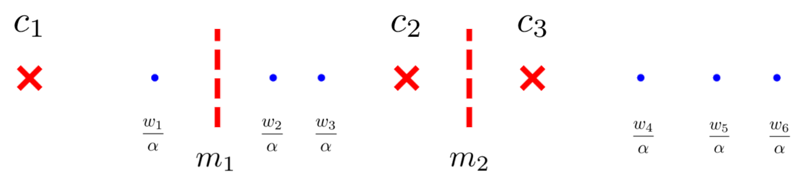

Figure 1. The optimal assignments of the $1/\alpha$-scaled datapoints (blue dots) into the codebook entries (red crosses) is given by relative ordering with respect to the midpoints between codebook entries (vertical dashed red lines). For example, with the given α value, the datapoint w_1 will be assigned to codebook entry c_1, w_2 and w_3 to c_2, and w_4, w_5, w_6 to c_3.

and has the solution of:

$$z^*_{nk}(\alpha) = \begin{cases} 1, \text{ if } c_k = \arg\min_{c_k \in \mathcal{C}} (w_n - \alpha c_k)^2 \\ 0, \text{ otherwise.} \end{cases} \tag{3}$$

We will be referring to eq. (3) as $\textsc{OptAssignment}(\alpha)$.

We can make the following characterization of α-induced optimal assignments given by (3). Consider an arbitrary point w_n that is optimally assigned to the codebook entry c_k, i.e., $z^*_{nk} = 1$. The c_k is chosen because it has the closest rescaled value αc_k to w_n among codebook values of c_{k-1}, c_k, c_{k+1}. We can ignore other values as codebook entries come pre-sorted. Thus, we have:

$$(w_n - \alpha c_k)^2 \leq (w_n - \alpha c_{k+1})^2$$
$$(w_n - \alpha c_k)^2 \leq (w_n - \alpha c_{k-1})^2.$$

Since $\alpha > 0$ and $c_{k-1} < c_k < c_{k+1}$, we solve this system of inequalities wrt w_n and conclude that

$$z^*_{nk} = 1 \iff \alpha\, m_{k-1} \leq w_n \leq \alpha\, m_k. \tag{4}$$

Here m_{k-1} and m_k are the midpoints between consecutive codebook entries defined as:

$$m_{k-1} = \frac{c_{k-1} + c_k}{2} \quad \text{and} \quad m_k = \frac{c_k + c_{k+1}}{2},$$

and for convenience we set $c_0 = -\infty$ and $c_{K+1} = +\infty$.

The midpoint characterization of (4) defines the following structure on the optimal α-induced assignments: the scaled midpoints $\alpha m_1, \ldots, \alpha m_{K-1}$ split the datapoints $w_1, \ldots, w_N$ into K disjoint groups such that all datapoints in a single group are assigned to the same codebook entry (see Fig. 1). Additionally, due to $\mathbf{w}$ being a sorted vector, the datapoints that are optimally assigned to the same codebook k form a continuous i-to-j chunk of $\mathbf{w}$ as $w_i, w_{i+1}, \ldots, w_{j-1}, w_j$ with $z_{ik} = \cdots = z_{jk} = 1$.

3.3. Optimal solution

The globally optimal solution $(\alpha^*, \mathbf{Z}^*)$ of (1) cannot be locally improved by using the $\textsc{OptScale}$ or $\textsc{OptAssignment}$, thus, the solution must satisfy the following fixed-point conditions:

$$\alpha^* = \textsc{OptScale}(\mathbf{Z}^*) \text{ and } \mathbf{Z}^* = \textsc{OptAssignment}(\alpha^*). \tag{5}$$

Additionally, the pair $(\alpha^*, \mathbf{Z}^*)$ must attain the smallest loss among all other fixed points (local solutions) satisfying (5). An important question we need to ask is how many such fixed points exist? We have the following result:

Lemma 3.1. *The number of fixed points satisfying (5) is at most $NK + 1$.*

Proof. We can bound the total number of fixed points satisfying (5) by bounding the total number of distinct locally optimal assignments given by $\textsc{OptAssignment}$. Say, we are given $\mathbf{Z} = \textsc{OptAssignment}(\beta)$ for some scaling factor $\beta > 0$. By the midpoint characterization (4) we know that for all n, k such that $z_{nk} = 1$ we have:

$$\beta m_{k-1} \leq w_n \leq \beta m_k.$$

Therefore, a particular $z_{nk} = 1$ will remain unchanged as long as β satisfies:

$$\frac{w_n}{m_k} \leq \beta \leq \frac{w_n}{m_{k-1}}.$$

By intersecting it over every $z_{nk} = 1$ we define a region

$$\mathcal{R}(\beta) = \bigcap_{n,k:\, z_{nk}=1} \left(\frac{w_n}{m_k}, \frac{w_n}{m_{k-1}} \right), \tag{6}$$

such that for any scalar $\alpha \in \mathcal{R}(\beta)$ the α-induced optimal assignments are the same as the $\textsc{OptAssignment}(\beta)$:

$$\forall \alpha \in \mathcal{R}(\beta) : \textsc{OptAssignment}(\alpha) \equiv \textsc{OptAssignment}(\beta)$$

Since the optimal assignments for $\alpha \in \mathcal{R}(\beta)$ remain the same, there can be at most one locally optimal α computed by $\textsc{OptScale}$ in this region $\mathcal{R}(\beta)$, which means each region $\mathcal{R}(\beta)$ contains at most one fixed point. By inspecting (6), we observe that none of the points w_n/m_k can be contained within any $\mathcal{R}(\beta)$, and can only be on the endpoints.

Algorithm 1 Required pre-computation routines to calculate the sums in $\mathcal{O}(1)$ time.

```
function PRECOMPUTE($\mathbf{w}$)
    $(s_1, \dots, s_n) \leftarrow (0, \dots, 0)$
    $(s^{\text{sq}}_1, \dots, s^{\text{sq}}_n) \leftarrow (0, \dots, 0)$
    $s_1, s^{\text{sq}}_1 \leftarrow w_1, w^2_1$
    for $n = 2 \dots N$ do
        $s_n \leftarrow s_{n-1} + w_n$
        $s^{\text{sq}}_n \leftarrow s^{\text{sq}}_{n-1} + w^2_n$

function QUICKSUM($i, j$)
    return $s_j - s_i$

function QUICKSUMSQ($i, j$)
    return $s^{\text{sq}}_j - s^{\text{sq}}_i$
```

Algorithm 2 The $\mathcal{O}(NK^2 \log N)$ implementation of our optimal MSE quantization algorithm that learns α to quantize the datapoints $\mathbf{w}$ using α-scaled codebook $\mathcal{C}$. The quick summation routines (QUICKSUM, QUICKSUMSQ) and necessary precomputations are detailed in Alg. 1.

```
function LOSS($\alpha, \mathbf{Z}^c$)
    $L \leftarrow 0$
    for $k = 1 \dots K$ do
        $(i, j) \leftarrow \mathbf{z}^c_k$
        $L \leftarrow L + \text{QUICKSUMSQ}(i, j) + (\alpha c_k)^2 \times (j - i)$
        $L \leftarrow L - 2c_k \times \text{QUICKSUM}(i, j)$
    return $L$
function OPTSCALE($\mathbf{Z}^c$)
    enum, denom $\leftarrow 0, 0$
    for $k = 1 \dots K$ do
        $i, j \leftarrow \mathbf{z}^c_k$
        enum $\leftarrow$ enum $+ c_k \times \text{QUICKSUM}(i, j)$
        denom $\leftarrow$ denom $+ \text{QUICKSUMSQ}(i, j)$
    return enum/denom
function OPTASSIGNMENT($\alpha$)
    $\mathbf{Z}^c = (\mathbf{z}^c, \dots, \mathbf{z}^c_K) \leftarrow (\mathbf{0}, \dots, \mathbf{0})$
    start $\leftarrow$ BINARYSEARCH($\mathbf{w}, \alpha\, m_k$)
    for $k = 1 \dots K - 1$ do
        end $\leftarrow$ BINARYSEARCH($\mathbf{w}, \alpha\, m_{k+1}$)
        $\mathbf{z}^c = (\text{start} + 1, \text{end})$
        start $\leftarrow$ end
    return $\mathbf{Z}^c$
function OPTQUANT($\mathbf{w}, \mathcal{C}$)
    $\mathbf{Z}^c_{\text{min}} = (\mathbf{z}^c_1, \dots, \mathbf{z}^c_K) \leftarrow (\mathbf{0}, \dots, \mathbf{0})$
    $L_{\text{min}} \leftarrow \infty$, $\alpha_{\text{min}} \leftarrow \infty$
    for $k = 1 \dots K - 1$ do
        $m_k = \frac{c_k + c_{k+1}}{2}$
    for $n = 1 \dots N$ do
        for $k = 1 \dots K - 1$ do
            $\alpha = \frac{w_n}{m_k}$
            $\mathbf{Z}^c \leftarrow \text{OPTASSIGNMENT}(\alpha)$
            $\alpha \leftarrow \text{OPTSCALE}(\mathbf{Z}^c)$
            $L \leftarrow \text{LOSS}(\alpha, \mathbf{Z}^c)$
            if $L \leq L_{\text{min}}$ then
                $L_{\text{min}} \leftarrow L$, $\mathbf{Z}^c_{\text{min}} \leftarrow \mathbf{Z}^c$, $\alpha_{\text{min}} \leftarrow \alpha$
    return $\alpha_{\text{min}}, \mathbf{Z}_{\text{min}}$
```

Therefore, the regions of the form $\mathcal{R}(\beta)$ (with constant α-induced assignments) are given by partitioning of the real line by the points w_n/m_k. Since there are $N \times K$ such points, the total number of regions will be $NK + 1$. □

We have discovered that values of α are partitioned into regions of form (6) where each region contains at most one fixed point. Since each region can be characterized by its left endpoint, let us number them as $\mathcal{R}_{nk}$ where the left endpoint of $\mathcal{R}_{nk}$ is the point w_n/m_k; there is one special region $\mathcal{R}_0$ with the left endpoint of 0. Our globally optimal algorithm will be based on finding optimal solution of every region $\mathcal{R}_{nk}$, enumerating over them using efficient data structures, and selecting the best one.

4. Optimal algorithm

We construct the algorithm that checks every possible fixed point of (1) belonging to α-regions with constant optimal assignments (regions $\mathcal{R}_0, \mathcal{R}_{11}, \dots, \mathcal{R}_{NK}$) and keeps track of the $(\alpha, \mathbf{Z})$-pair corresponding to the minimal encountered loss value. The fixed-point candidates are generated by calculating OPTASSIGNMENT for every region's left endpoint (0 or w_n/m_k). We present the pseudocode in Algorithm 2.

Implementation details and runtime The naive implementation of OPTASSIGNMENT requires $\mathcal{O}(NK)$ operations as we need to find a closest codebook entry for every datapoint w_n. Since the codebook entries are sorted, we can use a binary search to find closest codebook entry in $\mathcal{O}(\log K)$ time, reducing the total runtime to $\mathcal{O}(N \log K)$. Better yet, due to the midpoint characterization (4) we know that each codebook c_k will be optimally assigned to a continuous block of datapoints $w_i, w_{i+1}, \dots, w_{j-1}, w_j$. Therefore, we do not need to assign every datapoint, but only need to find the indexes of i and j using binary search in $\mathcal{O}(K \log N)$. Most importantly, this observation also allows us to store the assignment matrix $\mathbf{Z}$ *much more compactly* as $\mathbf{Z}^c$: instead of storing the assignment vectors $\mathbf{z}_n$ per datapoint w_n we simply store the (i, j) pair for every codebook as $\mathbf{z}^c_k = (i, j)$.

The implementation of OPTSCALE requires the sum over the datapoints that are assigned to each codebook entry c_k. Thus a straightforward implementation will require $\mathcal{O}(N+K)$ operations. However, if we pre-compute the cumulative running sum over the sorted datapoints (w_n-s) using an additional storage of $\mathcal{O}(N)$, we can compute the sums in $\mathcal{O}(1)$ (see Alg. 1). With such implementation, the OPTSCALE computation becomes an $\mathcal{O}(K)$ operation.

Similarly, the evaluation of the LOSS given by (1) can be computed in $\mathcal{O}(K)$ using our pre-computed running sums as well. To show it, let us first expand the loss of (1):

$$\textsc{Loss}(\alpha, \mathbf{Z}) = \sum_{k=1}^{K} \sum_{n=1}^{N} z_{nk}(w_n^2 - 2\alpha w_n c_k + (\alpha c_k)^2).$$

Recalling that under the optimal assignments the points with $z_{nk} = 1$ (for a fixed k) correspond to a continuous block of weights $w_i, \dots, w_{j-1}$, the inner loop (over n) of the above equation can be written as:

$$\sum_{n=i}^{j-1}(w_n^2 - 2\alpha c_k w_n + (\alpha c_k)^2) = \textsc{QuickSumSq}(i,j) - \\ - 2c_k \times \textsc{QuickSum}(i,j) + (\alpha c_k)^2 \times (j-i)$$

Here, sum over w_n^2 is computed using $\textsc{QuickSumSq}(i,j)$, and sum over w_n is computed using $\textsc{QuickSum}(i,j)$.

Finally, the main algorithm OPTQUANT iterates over the steps OPTASSIGNMENT, OPTSCALE, and LOSS using two loops over N and K, which results in an algorithm with the complexity of $\mathcal{O}(NK^2 \log N)$.

4.1. Improving the runtime

We can drive the complexity of the Alg. 2 to be $\mathcal{O}(NK \log K)$ if we change the order of the evaluations over the α-regions $\mathcal{R}_0, \mathcal{R}_{11}, \dots, \mathcal{R}_{NK}$ given by the corollary of Lemma 3.1.

Consider the calculation of locally optimal assignments for two neighboring α-regions $\mathcal{R}_a = (a, b)$ and $\mathcal{R}_b = (b, c)$. What is the difference between the optimal assignments for $\alpha_1 \in \mathcal{R}_a$ and the optimal assignments for $\alpha_2 \in \mathcal{R}_b$? Assuming datapoints w_n are distinct, only a single datapoint will change its assignment. Indeed, it is the datapoint w_n which defines the boundary between $\mathcal{R}_a$ and $\mathcal{R}_b$ with $b = w_n/c_k$ for some n and k. Hence, the $\textsc{OptAssignment}(\alpha_2)$ differs only by a single z_{nk} value from the previously computed $\textsc{OptAssignment}(\alpha_1)$, making the complexity of computing the OPTASSIGNMENT an $\mathcal{O}(1)$ operation. This region evaluation ordering makes the computation of OPTSCALE and LOSS an $\mathcal{O}(1)$ operation as well.

To make this evaluation strategy possible, we need to extract the region endpoints $\{w_n/m_k : \forall n, k\}$ in the sorted order, for which we will use a min-heap data structure [26, ch. 6.1]. We will use region endpoints (w_n/m_k) as keys in the heap, and we will store the corresponding n, k values as satellite along the key. Implemented straightforwardly, we

Algorithm 3 The improvement to $\mathcal{O}(NK \log N)$ of our scaled codebook quantization algorithm comes from using the min-heap data structure.

```
function OPTQUANT($\mathbf{w}, \mathcal{C}$)
    $L_{\min} \leftarrow \infty$
    endpoints ← empty MINHEAP
    for $k = 1 \dots K-1$ do
        $m_k = \frac{c_k + c_{k+1}}{2}$
    for $k = 1 \dots K$ do
        INSERT(endpoints, $(w_1/m_K, 1, K)$)
    while endpoints.size $> 0$ do
        $\alpha, n, k \leftarrow$ EXTRACTMIN(endpoints)
        $\mathbf{Z} \leftarrow$ OPTASSIGNMENT($\alpha$)
        $\alpha \leftarrow$ OPTSCALE($\mathbf{Z}^c$)
        $L \leftarrow$ LOSS($\alpha, \mathbf{Z}^c$)
        if $L < L_{\min}$ then
            $L_{\min} \leftarrow L$, $\mathbf{Z}^c_{\min} \leftarrow \mathbf{Z}^c$, $\alpha_{\min} \leftarrow \alpha$
        if $n < N$ then
            INSERT(endpoints, $(w_{n+1}/m_k, n+1, k)$)
    return $\alpha_{\min}, \mathbf{Z}_{\min}$
```

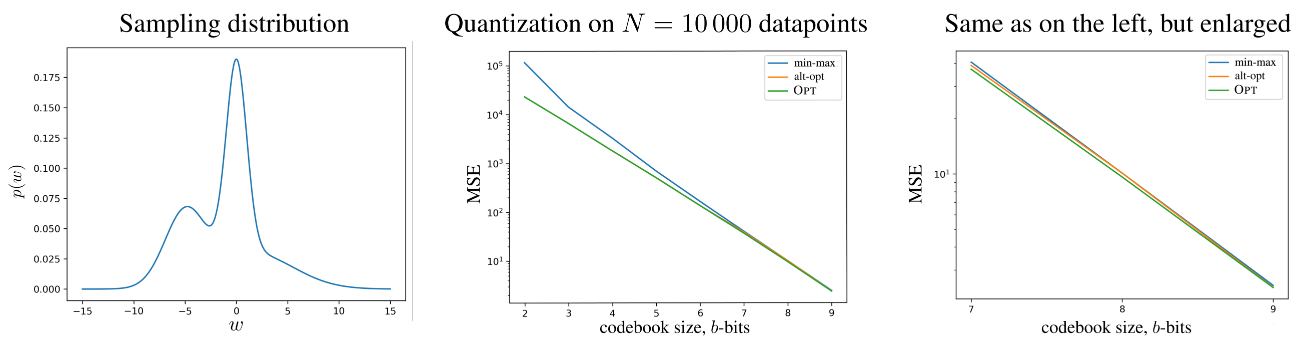

Figure 2. Quantization of $N = 10\,000$ datapoints sampled from the Gaussian mixture distribution. The probability density function of the distribution is plotted on the left. For this particular distribution our algorithm achieves optimal MSE error which outperforms other baselines (min-max and alt-opt). The gap between our optimal MSE and other methods is more visible when enlarged (on the right).

would maintain a heap over all NK endpoints. However, we can get away with a heap containing only K items. For a midpoint $m_{k'}$ the order of extraction of the $m_{k'}$ related region boundaries from the min-heap is the following:

$$\frac{w_1}{m_{k'}} \le \frac{w_2}{m_{k'}} \le \dots \le \frac{w_{N-1}}{m_{k'}} \le \frac{w_N}{m_{k'}}.$$

Therefore, there is no need to maintain all of the $m_{k'}$'s endpoints in the heap: once $w_n/m_{k'}$ is extracted, we simply add $w_{n+1}/m_{k'}$ to the heap and still maintain the extraction order, which makes each region's endpoint extraction an $\mathcal{O}(\log K)$ operation. Overall, we perform NK evaluations of OPTASSIGNMENT, OPTSCALE, and LOSS, where cost for an evaluation includes extracting an endpoint from the min-heap: this yields a final runtime of $\mathcal{O}(NK \log K)$. See Alg. 3 for full details.

5. Experiments

In this section, we present numerical experiments to demonstrate the benefits of our approach. To this end, we conduct two experiments. First, we focus on quantizing synthetic data and then, we focus on real-work applications and apply the algorithm to quantizing the weights and activations of a neural network in a post-training quantization process.

We implemented the $\mathcal{O}(NK \log K)$ version (Alg. 3) of our quantization algorithm in Python using the NumPy library [27]. We run the experiments on quantizing synthetic data (sec. 5.1) and on quantizing the data coming from real-world applications as part of the post-training quantization of the weights and the activations of the neural-networks (sec. 5.2).

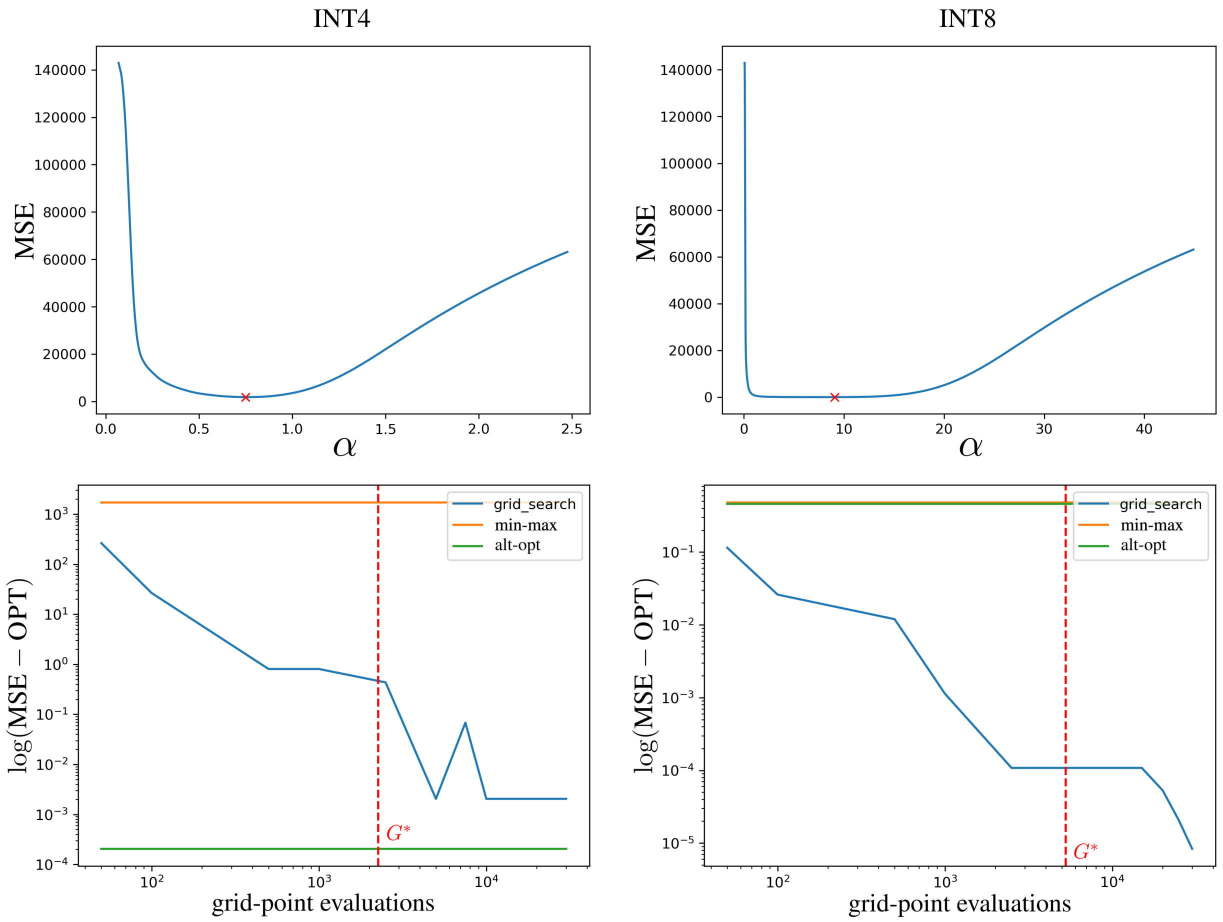

Figure 3. *Top*: the MSE loss as a function of α for every possible locally optimal assignments when quantizing 10K datapoints sampled from the synthetic distribution (Fig. 2, left); red cross corresponds to the found minimum value. *Bottom*: Results of the grid search (similar to Choukroun et al. [19]) as a function of the resolutions of the grid (number of grid-point evaluations). We plot the difference between achieved MSE and OPT MSE in log scale. For comparison, we give the results of quantization using min-max and alternating optimization. The horizontal dashed red line marked with G^* is the number of grid-point evaluation for which runtime of grid-search has the same complexity as running our optimal algorithm.

5.1. Synthetic experiments

Unlike other scaled codebook quantization approaches (sec. 2), our algorithm does not require a specific data distribution on datapoints to produce the optimal MSE quantization. To demonstrate this, we run our algorithm on the datapoints sampled from a mixture of three Gaussians with the mix-in proportions $\boldsymbol{\pi} = [0.3,\, 0.3,\, 0.4]$, means $\boldsymbol{\mu} = [-5,\, 1.5,\, 0]$ and standard deviations $\boldsymbol{\sigma} = [2,\, 4,\, 1]$ (Fig. 5.1, left). We sample $N = 10\,000$ datapoints from this distribution and quantize using the $K = 2^b - 1$ (b-bit) codebook of $\mathcal{C} = \{0, \pm1, \pm2, \ldots, \pm2^{b-1}\}$. By varying the value of b we obtain MSE error vs codebook size curve, which we plot in Fig. 2.

To put our algorithm in perspective, we additionally plot in Fig. 2 the quantization results using the following baselines: the min-max quantization and the alternating optimization algorithm. The min-max quantization is the widely used approach [1, 3, 13] where the α scale is obtained by assuming the uniform distribution on the datapoints $\mathbf{w}$. The alternating optimization (alt-opt) is the iterative approach similar to k-means: given an initial value of α we alternate between OptAssignment and OptScale procedures. As we can see, our optimal algorithm achieves the smallest quantization error when compared to the baselines and the min-max quantization has the worst MSE error among all tested methods. The quantization error of alt-opt algorithm is close to our optimal MSE, yet there is a considerable gap which can be observed on the enlarged plot (see Fig. 2, right). Note that since alt-opt's performance is initialization dependent, it occasionally can find a pretty good or even optimal solution, however, there is no guarantees or rules of thumb that allows selecting the proper initialization.

Comparison to grid search methods We separately compare our algorithm to the grid search approaches from the literature where α-scales are evaluated over the grid points. For this purpose, we reimplemented the grid search of Choukroun et al. [19] and run it to quantize the $N = 10\,000$ datapoints sampled from our synthetic distribution (Fig. 2, left). We use b-bit symmetric integer codebook $\mathcal{C} = \{0, \pm1, \pm2, \ldots, \pm2^{b-1}\}$ with $b = 4$ (INT4) and $b = 8$ (INT8). The quality of the grid-search solution can be controlled by varying the number of grid points G: more points we evaluate over, the better is the found solution. The paper of Choukroun et al. [19] does not provide runtime for their algorithm; in our implementation it has the asymptotic runtime of $\mathcal{O}(GK \log N)$.

We give the results of quantization on the bottom of Fig. 3. Since our algorithm achieves the optimal MSE error (which we denote as OPT), we plot the log difference between grid-search-found MSE and our OPT over the number of the grid points G. We additionally plot the log gap between OPT error and the quantization results using alternating optimization and min-max approach. While the grid search might be faster to finish for some values of G, at certain value of grid-point evaluations G^* it will require the same amount of computation as our optimal algorithm. We empirically compute this G^* and highlight it on our plots using a red dashed line. As we expected, the quality of the grid-search optimization is indeed dependent on G. However, even when we spend more runtime on grid search

INT8 quantization results, top-1 accuracy

		Min-max weight with activation calibration						our		
Model	FP32	Max	Entropy	99.9%	99.99%	99.999%	99.9999%	Q	Q+B+S	Q+B+Sv2
MobileNet-v1	71.88	69.84	70.99	70.76	70.89	70.67	70.44	70.78	**71.08**	71.12
MobileNet-v2	71.88	69.41	70.21	70.70	**71.02**	70.47	69.78	70.79	70.72	70.59
ResNet18	69.74	69.42	69.58	68.36	69.55	69.62	69.58	69.61	69.55	**69.63**
ResNet50	76.16	75.87	76.06	75.39	76.04	76.09	75.99	76.00	**76.10**	76.00

INT4 quantization results, top-1 accuracy

		Min-max weight with activation calibration						our		
Model	FP32	Max	Entropy	99.9%	99.99%	99.999%	99.9999%	Q	Q+B+S	Q+B+Sv2
MobileNet-v1	71.88	0.09	0.25	0.19	0.14	0.12	0.1	0.18	**6.38**	4.47
MobileNet-v2	71.88	0.11	0.47	0.38	0.13	0.11	0.08	3.49	**5.46**	4.24
ResNet18	69.74	0.84	42.81	41.07	31.30	19.19	7.15	44.74	57.85	**58.05**
ResNet50	76.16	0.21	52.44	53.45	37.03	7.40	0.93	53.81	63.74	**63.73**

Table 1. Comparison of top-1 accuracies on ImageNet2012 validation set when using the various post-training quantization recipes using the code of Wu et al. [3] and when quantizing using our optimal MSE quantization. Here, Q stands for simple weight and activation quantization, and Q+B+S is quantization with bias and scale correction, and Q+B+Sv2 is quantization with bias and scale correction with per-channel/per-neuron granularity.

(with $G \geq G^*$) there is still a measurable gap in MSE error. Interestingly, for INT4 codebook the alternating optimization achieves a better MSE and is the closest to our OPT result, yet, for the INT8 case the alt-opt solution is no better than min-max solution. Overall, none of the baselines is able to give optimal (or close to optimal) MSE solution that runs in reasonable time.

5.2. Compression of neural networks

Our quantization algorithm can be used as a drop replacement for many neural network quantization setups. Some of such approaches would use the quantization of the form (1) once at the end of the training (post-training), while others will apply it on the fly (e.g., Learning-Compression algorithm [10, 28, 29]). In this section, we explore the post-training quantization of the neural networks where both weights and the activations are quantized using b-bit integer codebooks of $\mathcal{C} = \{0, \pm 1, \pm 2, \dots, \pm 2^{b-1}\}$. We use a single codebook per layer, and apply quantization to all fully connected and convolutional layers. To quantize the activations, we use the standard calibration approach [3, 13] where we collected the activation values computed during the forward pass of a small batch (512) of train images. We do not perform any finetuning after the quantization. We evaluate models trained on ImageNet using the PyTorch [30]; the pre-trained models were obtained from the official PyTorch repository through torchvision. The full precision top-1 accuracies of the used models are given in FP32 column in Table 1.

Our results can be significantly improved when combined with bias and scale correction. Indeed, the quantized weights and activations coming from the optimization of the MSE loss (eq. 1) will not be optimal wrt the model loss, unless the $\|\mathbf{W} - \mathbf{Q}\| \approx 0$. Here, we denoted by $\mathbf{Q}$ the optimally compressed version of $\mathbf{W}$. A simple, yet surprisingly effective way to improve the performance is to correct for the effects of quantization by altering the layer's bias [14, 15] and weight scales [31]. If we denote the input to the quantized layer as $\mathbf{x}$, and its output as $\mathbf{y}$, the bias and scale correction can be formulated as the problem over $(\mathbf{x}, \mathbf{y})$-pairs on the calibration dataset (usually sampled from the train data) as

$$\min_{s,\mathbf{b}} \textstyle\sum_{\mathbf{x},\mathbf{y}} \|\mathbf{y} - s\mathbf{Q}\mathbf{x} - \mathbf{b}\|^2,$$

where $\mathbf{b}$ is the layer's new bias and s is the scale correction. The solution of this optimization problem can be computed in closed form during the calibration stage with:

$$s = \frac{\sum_{\mathbf{y},\mathbf{z}} (\mathbf{y}^T\mathbf{z} - \mathbf{z}^T\bar{\mathbf{y}})}{\sum_{\mathbf{z}} (\mathbf{z}^T\mathbf{z} - \mathbf{z}^T\bar{\mathbf{z}})},$$
$$\mathbf{b} = \bar{\mathbf{y}} - \alpha\bar{\mathbf{z}}.$$

Here, we used $\mathbf{z}$ as a shorthand for $\mathbf{Q}\mathbf{x}$, i.e. $\mathbf{z} \equiv \mathbf{Q}\mathbf{x}$, and $\bar{\mathbf{y}}$ and $\bar{\mathbf{z}}$ are the sample averages over all $\mathbf{y}$s and $\mathbf{z}$s, respectively. The scale s can be introduced for the entire layer, or per neuron/channel (if the layer is followed by batch normalization layer).

In Table 1 we report post-training quantization results using our algorithm which include: regular quantization (Q), quantization and bias/scale correction per layer (Q+B+S), and quantization with bias/scale correction per channel or neuron (Q+B+Sv2). We compare our results to the state-of-the-art recipes available in the literature. In particular, we compare to the recipe of Wu et al. [3] which uses the min-max approach for the weight quantization, and give various heuristics for the activation quantization. These strategies include: a) min-max quantization, b) optimizing the scale α to minimize the entropy between quantized and the original activation distribution, and c) choosing the α-scale based on the percentiles (so that α-scaled codebook covers $x\%$ of the data). As we show in Table 1, quantizing both weights and activations using our single MSE optimal algorithm performs on par or better comparing to the heuristic calibrations. For example, our INT8 quantization results outperform most of the heuristic calibration strategies, and when combined with scale/bias correction we get the smallest performance drop when compared to the performance of the full precision network. The advantage of the optimal post-training quantization gets even more apparent when we quantize using an INT4 codebook: for example we can achieve an accuracy of 63.75% on ResNet50 by simply replacing the calibration strategies with our optimal MSE quantizer.

6. Conclusion

We have presented the derivation of an efficient and globally optimal solution for the general formulation of scaled fixed codebook quantization problem of (1), which runs in $\mathcal{O}(NK \log K)$ time. The formulation we are solving includes many important quantization problems as a particular case (e.g., INT8, binarization, etc). To achieve the globally optimal minimum squared error solution, we do not assume any specific data distribution nor do we perform a heuristic search. We empirically demonstrated that the proposed algorithm could be used as a drop replacement in many quantization and compression settings where problem (1) is being solved. For a neural network compression setting, our algorithm can be used as an initial quantization point or as a solution of proximal step problems like in [8, 10, 28, 29, 32]. The quantization problems of the same type occur in novel number representation formats developed by Intel [33], and Xilinx [34] for which our algorithm is optimal too. The derived algorithm is of interest for the signal compression field in general, and might be used in other compression problems: e.g., in compressed image/video storage formats like JPEG or H.264.

References

[1] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. of the 2018 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'18)*, Salt Lake City, UT, June 18–22 2018, pp. 2704–2713.

[2] Marat Dukhan, Yiming Wu, and Hao Lu, "QNNPACK: Open source library for optimized mobile deep learning," https://engineering.fb.com/ml-applications/qnnpack/, Accessed: 2020-Jun-8.

[3] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius, "Integer quantization for deep learning inference: Principles and empirical evaluation," arXiv:2004.09602, Apr. 20 2020.

[4] Kyuyeon Hwang and Wonyong Sung, "Fixed-point feedforward deep neural network design using weights $+1$, 0, and -1," in *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, Belfast, UK, Oct. 20–22 2014, pp. 1–6.

[5] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks," in *Proc. 15th European Conf. Computer Vision (ECCV'18)*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, Eds., Munich, Germany, Sept. 8–14 2018, pp. 365–382.

[6] Stuart P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Information Theory*, vol. 28, no. 2, pp. 129–137, Mar. 1982.

[7] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung, "Fixed point optimization of deep convolutional neural networks for object recognition," in *Proc. of the IEEE Int. Conf. Acoustics, Speech and Sig. Proc. (ICASSP'15)*, Brisbane, Australia, Apr. 19–24 2015, pp. 1131–1135.

[8] Cong Leng, Hao Li, Shenghuo Zhu, and Rong Jin, "Extremely low bit neural network: Squeeze the last bit out with ADMM," in *Proc. of the 32nd AAAI Conference on Artificial Intelligence (AAAI 2018)*, New Orleans, LA, Feb. 2–7 2018, pp. 3466–3473.

[9] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi, "XNOR-net: ImageNet classification using binary convolutional neural networks," in *Proc. 14th European Conf. Computer Vision (ECCV'16)*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, Eds., Amsterdam, The Netherlands, Oct. 11–14 2016, pp. 525–542.

[10] Miguel Á. Carreira-Perpiñán and Yerlan Idelbayev, "Model compression as constrained optimization, with application to neural nets. Part II: Quantization," arXiv:1707.04319, July 13 2017.

[11] Penghang Yin, Shuai Zhang, Yingyong Qi, and Jack Xin, "Quantization and training of low bit-width convolutional neural networks for object detection," arXiv:1612.06052, Aug. 17 2017.

[12] Sreekanth Annapureddy Darryl Lin, Sachin Talathi, "Fixed point quantization of deep convolutional networks," in *Proc. of the 33rd Int. Conf. Machine Learning (ICML 2016)*, Maria-Florina Balcan and Kilian Q. Weinberger, Eds., New York, NY, June 19–24 2016, pp. 2849–2858.

[13] Raghuraman Krishnamoorth, "Quantizing deep convolutional networks for efficient inference: A whitepaper," arXiv:1806.08342, June 21 2018.

[14] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling, "Data-free quantization through weight equalization and bias correction," in *Proc. 17th Int. Conf. Computer Vision (ICCV'19)*, Seoul, Korea, Oct. 27 – Nov. 2 2019, pp. 1325–1334.

[15] Ron Banner, Yury Nahshan, and Daniel Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," In Wallach et al. [35], pp. 7950–7958.

[16] Raziel Alvarez, Rohit Prabhavalkar, and Anton Bakhtin, "On the efficient representation and execution of deep acoustic models," in *Proc. of Interspeech'16*, San Francisco, CA, Sept. 8–12 2016, pp. 2746–2750.

[17] Sungho Shin, Yoonho Boo, and Wonyong Sung, "Fixed-point optimization of deep neural networks with adaptive step size retraining," in *Proc. of the IEEE Int. Conf. Acoustics, Speech and Sig. Proc. (ICASSP'17)*, New Orleans, LA, Mar. 5–9 2016, pp. 1203–1207.

[18] Xingchao Liu, Mao Ye, Dengyong Zhou, and Qiang Liu, "Post-training quantization with multiple points: Mixed precision without mixed precision," arXiv:2002.09049, Feb. 20 2020.

[19] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev, "Low-bit quantization of neural networks for efficient inference," in *ICCV Workshop on Compact and Efficient Feature Representation and Learning in Computer Vision*, 2019.

[20] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos, "Deep learning with low precision by half-wave Gaussian quantization," in *Proc. of the 2017 IEEE Computer Society Conf. Computer Vision and Pattern Recognition (CVPR'17)*, Honolulu, HI, July 21–26 2017, pp. 5918–5926.

[21] Jungwook Choi, Swagath Venkataramani, Vijayalakshmi Srinivasan, Kailash Gopalakrishnan, Zhuo Wang, and Pierce Chuang, "Accurate and efficient 2-bit quantized neural networks," in *Proc. of the 2nd Conf. Systems and Machine Learning (SysML 2019)*, Stanford, CA, Mar. 31 – Apr. 2 2019.

[22] Jun Fang, Ali Shafiee, Hamzah Abdel-Aziz, David Thorsley, Georgios Georgiadis, and Joseph Hassoun, "Post-training piecewise linear quantization for deep neural networks," arXiv:2002.00104, Jan. 31 2020.

[23] James D Bruce, "Optimum quantization," Tech. Rep. 429, Massachussetts Institute of Technology, 1965.

[24] Xiaolin Wu and John Rokne, "An $\mathcal{O}(KN \log N)$ algorithm for optimum K-level quantization on histograms of n points," in *Proc. 17th ACM Annual Computer Science Conference*, Louisville, KY, Feb. 21–23 1989, pp. 339–343.

[25] Xiaolin Wu, "Optimal quantization by matrix searching," *J. Algorithms*, vol. 12, no. 4, pp. 663–673, Dec. 1991.

[26] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, *Introduction to Algorithms*, MIT Press, Cambridge, MA, third edition, 2009.

[27] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 16 2020.

[28] Miguel Á. Carreira-Perpiñán, "Model compression as constrained optimization, with application to neural nets. Part I: General framework," arXiv:1707.01209, July 5 2017.

[29] Yerlan Idelbayev and Miguel Á. Carreira-Perpiñán, "More general and effective model compression via an additive combination of compressions," Submitted, 2020.

[30] 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 Wallach et al. [35], pp. 8026–8037.

[31] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Chris De Sa, and Zhiru Zhang, "Improving neural network quantization without retraining using outlier channel splitting," in *Proc. of the 36th Int. Conf. Machine Learning (ICML 2019)*, Kamalika Chaudhuri and Ruslan Salakhutdinov, Eds., Long Beach, CA, June 9–15 2019, pp. 7543–7552.

[32] Yerlan Idelbayev and Miguel Á. Carreira-Perpiñán, "A flexible, extensible software framework for model compression based on the LC algorithm," arXiv:2005.07786, May 15 2020.

[33] Urs Koster, Tristan J. Webb, Xin Wang, Marcel Nassar, Arjun K. Bansal, William H. Constable, Oğuz H. Elibol, Scott Gray, Stewart Hall, Luke Hornof, Amir Khosrowshahi, Carey Kloss, Ruby J. Pai, and Naveen Rao, "Flexpoint: An adaptive numerical format for efficient training of deep neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. 2017, vol. 30, pp. 1742–1752, MIT Press, Cambridge, MA.

[34] Sean O. Settle, Paolo D'Alberto Manasa Bollavaram, Elliott Delaye, Oscar Fernandez, Nicholas Fraser, Aaron Ng, Ashish Sirasao, and Michael Wu, "Quantizing convolutional neural networks for low-power high-throughput inference engines," arXiv:1805.07941, May 21 2018.

[35] H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, Eds., *Advances in Neural Information Processing Systems (NEURIPS)*, vol. 32. MIT Press, Cambridge, MA, 2019.