Efficient Training on Edge Devices Using Online Quantization

Michael H. Ostertag[1], Sarah Al-Doweesh[2], and Tajana Rosing[1]

***Abstract*— Sensor-specific calibration functions offer superior performance over global models and single-step calibration procedures but require prohibitive levels of sampling in the input feature space. Sensor self-calibration by gathering training data through collaborative calibration or self-analyzing predictive results allows these sensors to gather sufficient information. Resource-constrained edge devices are then stuck between high communication costs for transmitting training data to a centralized server and high memory requirements for storing data locally. We propose online dataset quantization that maximizes the diversity of input features, maintaining a representative set of data from a larger stream of training data points. We test the effectiveness of online dataset quantization on two real-world datasets: air quality calibration and power prediction modeling. Online Dataset Quantization outperforms reservoir sampling and performs equally to offline methods.**

I. Introduction

Sensor nodes at the edge of the Internet of Things often require sensor-specific calibration functions that relate input features to a phenomenon of interest. For air quality sensing, the calibration function transforms input data from onboard sensors to target pollutant concentrations, and for application power prediction, internal performance metrics can be used to predict device power. Individual calibration requires substantial time and resources to collect data that adequately covers the feature space. The best solution is to empower sensors to gather their own training data.

For air quality sensors, collaborative calibration is a promising technique that allows edge devices to continually increase their knowledge of the surrounding world by pairing a sensor's internal sensor reading with a target value from a co-located reference sensor [1]–[3]. For prediction tasks where the true value will be known in the future, a sensor can collect relevant data at a current time point and associate it with the future target value once the event occurs.

To complicate matters, calibration functions can change over time to due inherent sensors characteristics (e.g. drift in air quality electrochemical sensors) or different usage scenarios (e.g. predicting power usage on new applications). As training data is continually gathered, resource-constrained edge devices are caught between incurring high computation, storage, and memory costs by performing training locally or spending large amounts of power to send data to a central node. Recent advancements in machine learning on edge devices are tackle these issues by intelligent subset selection for classification [4], [5] and dedicated machine learning coprocessors [6], [7]. While novel, these methods focus on classification tasks and do not extend well to regression tasks.

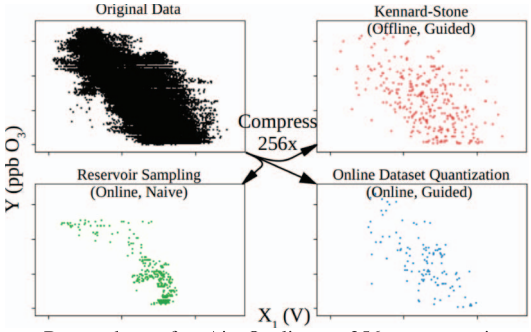

Fig. 1. Data subsets for Air Quality at 256x compression resulting from Reservoir Sampling (RS), Kennard-Stone (KS), and Online Dataset Quantization (ODQ).

The only works that the authors found pertaining to reduced data subset selection for regression tasks were reservoir sampling (RS) [8] or methods based on the Kennard-Stone sampling (KS) algorithm [9]–[12]. The RS algorithm performs online sampling of a data stream that results in a uniform random subset selection. The complexity of the algorithm is very low and the expected distribution as a result of RS will maintain the original data distribution. KS takes a different approach: the algorithm iteratively evaluates a distance metric on the dataset in an offline manner, selecting points for inclusion that are the farthest from the set of already selected points. The goal of the KS selection procedure is to select a subset that has a uniform distribution over the target feature.

Existing methods select the best representative data subset based on informativeness, representativeness, and diversity. Online methods use these concepts, but only for classification tasks. Offline methods based on KS have been developed for regression problems, but no solutions exist for performing subset selection online for regression tasks. A comparison of subset selection for the air quality dataset can be seen in Fig. 1.

Contributions. In this work, we propose Online Dataset Quantization (ODQ), a method to quantize and compress a continually growing dataset that enables long-term calibration dataset collection on resource-constrained edge devices. ODQ evaluates the distance between training samples using a local approximation of the gradient. The resulting reduced dataset can be used for training a calibration model with error minimized over the entire range of the input feature space for non-linear relationships between input features and target

We gratefully acknowledge support from NSF CNS-1446912, NSF NRI CNS-1830399, and the King Abdulaziz City for Science and Technology

[1]The authors are with the Computer Science Engineering Department, University of California, San Diego, La Jolla, CA 92093, USA {mosterta, tajana}@ucsd.edu

[2]The author is with the King Abdulaziz City for Science and Technology, Ar Raid Riyadh 12354, Saudi Arabia

values. We also propose a reduced-memory implementation of KS and an online linear implementation of ODQ (OLDQ) that has a reduced storage overhead. We compare these methods to the naive implementation of RS on two real-world data sets for air quality calibration and device power prediction using a multi-layer perceptron model.

II. Problem Definition

Consider the problem where we want to train a supervised machine learning algorithm for a regression task on a memory-constrained edge device with dataset D that is growing over time. Each entry in the dataset consists of N_x input features $X = [x_1, ..., x_{N_x}] \in \mathbb{R}^{N_x}$ and a single regression target $Y \in \mathbb{R}$. The input features X and the output feature Y are related as $X = f^{-1}(Y) + n$ where n is additive noise. The aggregation of all N entries forms the dataset $D = \{(X_i, Y_i)\}_{i=1}^{N}$. Our goal is to determine a reduced subset of the data $\widetilde{D} = \{(\widetilde{X}_j, \widetilde{Y}_j)\}_{j=1}^{M}$ where $M < N$ that will produce the best calibration model $f(\cdot)$ that predicts the target value Y for a given set of input features, i.e. $f(X) = \hat{Y}$. In general, $\widetilde{X}$ and $\widetilde{Y}$ can be points from the original dataset D or can be artificial prototypes drawn from the same feature and target space as X and Y, but for this work, we are selecting a subset of points from D.

Our approach for maximizing the utility J of a calibration dataset includes notions of representativeness, diversity, and informativeness. Traditionally, representativeness is a metric that encourages a distribution to closely match an original distribution, but our goal is to create a representative dataset that can build a robust calibration model. Since training data commonly has imbalanced distributions with large amounts of redundant information, random sampling results in saving redundant information in dense regions and losing valuable information in tail regions as seen in Fig. 1.

We seek to save a reduced subset of points such that the probability distribution of the reduced dataset $p(\widetilde{Y})$ is uniform. A uniform distribution over an arbitrary, finite range $[a, b]$ represents the maximum level of uncertainty about the incoming value. When optimizing over a dataset with a uniform distribution, the resulting model will attempt to reduce error evenly across the range, which is optimal when developing a calibration model with the goal of minimizing error across a measurement range. The original distributions $P(Y)$ is unlikely to be uniform, so the question becomes how to sample from D such that $\widetilde{D}$ is approximately uniform. For a single random variable and monotonic region, the desired distribution is proportional to dy/dx as shown below:

$$P\big(X \le x\big) = P\big(f^{-1}(Y) \le x\big) = P\big(Y \le f(x)\big)$$
$$= \int_{-\infty}^{f(x)} p(y)dy = \frac{1}{b-a}\big(f(x) - a\big) \quad (1)$$
$$p(x) = \frac{d}{dx}P\big(X \le x\big) \propto \left|\frac{dy}{dx}\right| \quad (2)$$

The intuition is that when small changes in X result in large changes in Y, more points should be captured in the region to better capture the transition. When changes in X result in no change in Y, fewer points should be saved. Using a Euclidean distance metric weighted by the local partial derivative, the concept is captured succinctly with the following cost function:

$$J = \min_{i \neq j} \; d_i(x_i, x_j) = \min_{i \neq j} \; x_i^T \text{diag}(\nabla f_t|_{x_i}) x_j \quad (3)$$

where $\nabla f_t = [\delta f_t/\delta x_1, \delta f_t/\delta x_2, ..., \delta f_t/\delta x_{N_k}]$ is the local approximations of the gradient.

III. Online Dataset Quantization

Our goal is to create a reduced-size representative dataset $\widetilde{D}$ of M samples where $M < N$ that maximizes the usefulness of the reduced dataset for regression tasks where utility is measured by interpoint distances weighted by each feature's sensitivity to Y. While N will continue to grow throughout the lifetime of the device, the size of the reduced dataset remains fixed, representing the limited, fixed memory reserved for a training dataset in edge devices.

As shown in Fig. 2, initially, all incoming points c_t are added until the memory is full. Then, using local gradient as calculated from the hyperplane that minimizes a least squares equation with the K nearest neighbors, the weighted distance function d_t is calculated at point (x_t, y_t). If the addition of the new point c_t would improve the cost J, then the worst data point is replaced and local neighborhoods are recalculated as needed. In the following subsections, we describe how to determine a Local Neighborhood and how the utility function J is evaluated.

A. Local Neighborhood Determination

Determining the local neighborhood of a data point is a compute intensive task. In our algorithm flow in Algorithm 1, local neighborhoods are calculated for each incoming point, and if it is incorporated into the dataset, local neighborhoods are re-calculated for each point whose neighborhood contained the excised point. Selecting the appropriate neighborhood for the best estimate of the local gradient is a difficult problem. The local gradient calculation depends on the selection of the local neighborhood, but conversely, the local neighborhood selection depends on the local gradient.

We select a neighborhood by iteratively calculating the distance between the data point under evaluation c_t and all other data points of a set that decreases with each iteration. Initially, the neighborhood consideration set is reduced by 50% per iteration until its size is $2K$ then it is decremented by 1 element. The worst case number of iterations is $2K - 1 + \lfloor \log_2(N - 2K) \rfloor$, but during our experimentation, the algorithm converged much faster.

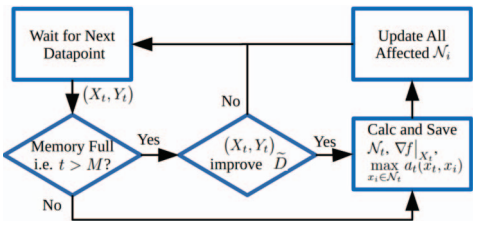

Fig. 2. Algorithm overview for Online Dataset Quantization.

Algorithm 1 Local Neighborhood Calculation

Input: x_t, K, $\widetilde{D}_t$, ∇f_i, $\mathcal{N}_i \; \forall \; i = \{1, ..., M\}$
Output: ∇f_t, $\mathcal{N}_t$

$\hat{D} \leftarrow D_t$
if $t \leq K$ **then**
  $\mathcal{N}_t \leftarrow \widetilde{D}_t$
  Calculate local gradient ∇f_t
  return $\nabla f_t, \mathcal{N}_t$
end if
$\mathcal{N}_t \leftarrow \arg\min_{x_i \in \hat{D}} d_i(x_i, x_t)$ s.t. $|\mathcal{N}_t| = K$
while $|\hat{D}| > K$ **do**
  Calculate local gradient ∇f_t
  $\mathcal{N}_{t,new} \leftarrow \arg\min_{x_i \in \hat{D}} d_t(x_t, x_i)$ s.t. $|\mathcal{N}_t| = K$
  if $\mathcal{N}_{t,new} = \mathcal{N}_t$ **then**
    break
  end if
  if $|\hat{D}| \geq 3K$ **then**
    $\hat{D}^- \leftarrow \arg\max_{x_i \in \hat{D}} d_t(x_t, x_i)$ s.t. $|\hat{D}^-| = \lfloor |\hat{D}|/2 \rfloor$
  else
    $\hat{D}^- \leftarrow \arg\max_{x_i \in \hat{D}} d_t(x_t, x_i)$ s.t. $|\hat{D}^-| = 1$
  end if
  $\hat{D} \leftarrow \hat{D} \setminus \hat{D}^-$
  $\mathcal{N}_t \leftarrow \mathcal{N}_{t,new}$
end while
return $\nabla f_t, \mathcal{N}_t$

B. Evaluating Dataset Cost

The first M incoming points $c_t = (x_t, y_t)$ are directly added to the dataset $\widetilde{D}$. Once filled, each successive data point c_t needs to be evaluated to determine if its addition will improve the overall utility J as calculated in Eq. 3. If $\min_{x_i \in \widetilde{D}} d_t(x_t, x_i) < J_{t-1}$ and $\min_{x_i \in \widetilde{D}} d_i(x_i, x_t) < J_{t-1}$, then the point with the previous minimum distance is removed from $\widetilde{D}$ and c_t is added to the dataset.

In many cases, the incoming point c_t will be near another point c_τ, resulting in a small distance, but c_t may result in an improvement to utility J over c_τ. To evaluate this scenario, the distance metric is calculated using c_t on the reduced dataset without c_τ. If c_t improves J, then c_t replaces c_τ.

C. Metainformation

Each entry c_i into the dataset $\widetilde{D}$ requires a certain amount of space for storage of the input features and target values. While the gradient ∇f_i and neighborhood $\mathcal{N}_i$ can theoretically be recalculated for each new point, the computation cost is prohibitively expensive. Instead, ∇f_i, $\mathcal{N}_i$, and $\min_i x_i^T \text{diag}(\nabla f_t|_{xi}) x_j$ are saved for each data point, resulting in storage reserved for metainformation. The memory required for every data point is $m_{entry} = p_x|X| + p_y|Y| + m_{header}$, where p_x and p_y are the number of bits of precision for X and Y, respectively, and m_{header} is metainformation pertaining to the sample.

Since $|X| = |\nabla f_i|$ and $|Y| \leq K$, the memory required for the metainformation will always be equal or greater than memory required for the data itself, assuming the precision of the data and metainformation is the same. The uncalibrated device S_0 has a limited amount of storage for a training dataset m_{max}, which allows the first N_{max} entries to be saved directly where $N_{max} = \lfloor m_{max}/m_{entry} \rfloor$.

IV. Comparison Algorithms

A. Reservoir Sampling

Uniform random sampling is a common comparison metric for subset selection algorithms. The online version of random sampling is termed reservoir sampling (RS), which can be performed efficiently for large datasets and only requires a random integer number generator [8]. The result is a reduced subset $\widetilde{D}$ that has the same distribution as the original data D. The algorithm saves the first M incoming data points. Then, for each subsequent data point c_t, a random integer is drawn n_r between 1 and t. If $n_r \leq M$, then the entry at index n_r is replaced with c_t.

B. Reduced-Memory Kennard-Stone

Kennard-Stone algorithm (KS) [9]–[12] is an offline algorithm that operates on the entire dataset, selecting a subset that is uniformly distributed across the feature space. Interpoint distances are calculated once using static weights, stored in a large matrix in memory of scale $\mathcal{O}(M^2)$, and evaluated for selecting the optimal subset. While acceptable for smaller datasets, modern datasets can have 100,000 entries with > 10 features, requiring > 93 GB of storage for 16-bit values, prohibitive on edge devices.

We propose a modern implementation of KS that exchanges a smaller memory footprint for increased computation. Instead of pre-computing the distance between points and storing in large matrix, interpoint distance is calculated, compared to the current max-min distance, and either kept or discarded in a single vector U, which tracks the minimum distance of the i-th element to any data point in $\widetilde{D}$. At each iteration, the data point with the max-min distance is added to $\widetilde{D}$ and distances are recalculated with the minimum interpoint distances being saved in U. The new algorithm has an increase in distance calculations ($(T-1)^2/2 + TM$ vs $(T-1)^2/2$) but memory requirements that scale $\mathcal{O}(M)$ instead of $\mathcal{O}(M^2)$.

C. Online Linear Dataset Quantization

The Online Linear Dataset Quantization (OLDQ) algorithm follows the same algorithmic pattern as ODQ, but instead of a non-linear approximation of the slope for the input features, OLDQ uses the Mahalanobis distance and incorporates both X and Y with a diagonal covariance matrix that is calculated offline based on prior knowledge. By saving a sensitivity per feature instead of per sample, metainformation required for OLDQ will be approximately 50% less than that required for ODQ. The drawback is that OLDQ uses a linear approximation of the generative function f, assuming that the input features have uniform sensitivity across their entire range.

V. Experiments

We compare ODQ to the methods in Sec. IV using mean-squared-error on regression tasks for air quality and power prediction datasets. For each dataset, we selected a subset of data and associated metadata that was compressed by ratio of 16, 128, 256, 512, and 1024. At each compression ratio, we generated 3 unique splits of the data and randomized the order of training data. All of the neural networks were trained using Keras and Tensorflow backend with 5 repeated experiments at each compression level.

A. Datasets

The air quality dataset consists of approximately 86,000 measurements for each of 9 devices that were co-located at 3 different EPA reference stations [13]. The data was split into 80% train and 20% test with a further split of the training set into 80% training and 20% validation. The power prediction dataset consists of 67,017 training and 1241 testing data points and predicts the power a device would consume for a given, profiled application using 36 input features of performance counters from devices running example workloads [14]. The split was generated using a set of training and target applications where the application used for the test sets was not included in any training.

B. Results

For the air quality dataset, ODQ and OLDQ outperformed RS once compression rates were above 256x. The data was collected by stationary sensors, so data redundabct was expected. The air quality sensor sensitivities to cross-contaminants and ambient conditions can be approximated linearly, which explains the comparable performance between the linear (OLDQ) and non-linear (ODQ) methods. For the device power dataset, OLDQ and KS maintained relatively constant accuracy across the compression ratios. Since reservoir sampling selects a random subset of points, resulting data subsets were inconsistent with which usage patterns were captured, resulting in a high level of variance in dataset quality. When the RS subset overlaps with the target distribution, then the results showed low error, but otherwise, the model had poor performance. In the 64x to 264x region, ODQ outperforms all other methods, likely due to its ability adjust the subset selection methods to account for nonlinearities in the dataset.

While ODQ and RS appear similar in Fig. 3 (right), we can analyze the error across the range of values of Y to determine the source of the error (left). At a 16x compression ratio, all of the results are similar due to the large number of available data points for training, but as the compression ratio increases, ODQ, OLDQ, and KS error levels remain low while RS error levels increase in regions of low probability.

Due to ODQ requiring more metainformation, at very high compression ratios, ODQ does not have a sufficient number of samples to cover the input space, resulting in a decrease in accuracy for both the air quality and power datasets. The amount of required metainformation stored with each point for ODQ is substantially higher than OLDQ, but could potentially be reduced using code books that contain gradient information. This is a direction of future research.

VI. Conclusions

We proposed online dataset quantization (ODQ) that allows a sensor to maintain a representative subset of training data for regression tasks. ODQ was tested on two common regression problems on the edge: sensor calibration and power prediction, outperforming existing online methods (RS) and performing comparably to offline methods (KSS). The largest drawback to ODQ is the storage required for metainformation, which reduces the number of samples that ODQ can save compared to other methods. Despite the reduced number of saved points, comparable test accuracies are achieved when training neural networks for regression.

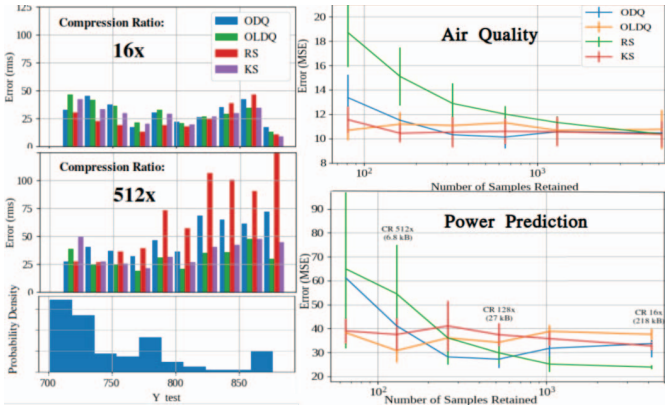

Fig. 3. (L) Test error (MSE) distribution across Y values for the Power Approximation dataset. Results from (TR) Air Quality and (BR) Power Approximation over compression ratios from 16x to 1024x.

References

[1] Y. Xiang, L. S. Bai, R. Pledrahita, R. P. Dick, Q. Lv, M. Hannigan, and L. Shang, "Collaborative calibration and sensor placement for mobile sensor networks," in *2012 ACM/IEEE 11th Int Conf on Information Processing in Sensor Networks (IPSN)*. IEEE, 2012, pp. 73–83.

[2] A. Arfire, A. Marjovi, and A. Martinoli, "Model-based rendezvous calibration of mobile sensor networks for monitoring air quality," in *2015 IEEE SENSORS*. IEEE, 2015, pp. 1–4.

[3] F. Kizel, Y. Etzion, R. Shafran-Nathan, I. Levy, B. Fishbain, A. Bartonova, and D. M. Broday, "Node-to-node field calibration of wireless distributed air pollution sensor network," *Environmental Pollution*, vol. 233, pp. 900–909, 2018.

[4] K. Wei, R. Iyer, and J. Bilmes, "Submodularity in data subset selection and active learning," in *International Conference on Machine Learning*, 2015, pp. 1954–1963.

[5] B. Du, Z. Wang, L. Zhang, L. Zhang, W. Liu, J. Shen, and D. Tao, "Exploring representativeness and informativeness for active learning," *IEEE transactions on cybernetics*, vol. 47, no. 1, pp. 14–26, 2017.

[6] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *2017 ACM/IEEE 44th Annual Int Symposium on Comp Arch (ISCA)*. IEEE, 2017, pp. 1–12.

[7] M. Imani, R. Garcia, S. Gupta, and T. Rosing, "Hardware-software co-design to accelerate neural network applications," *ACM Journal on Emerging Tech in Computing Sys (JETC)*, vol. 15, no. 2, p. 21, 2019.

[8] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985.

[9] R. W. Kennard and L. A. Stone, "Computer aided design of experiments," *Technometrics*, vol. 11, no. 1, pp. 137–148, 1969.

[10] R. K. H. Galvao, M. C. U. Araujo, G. E. Jose, M. J. C. Pontes, E. C. Silva, and T. C. B. Saldanha, "A method for calibration and validation subset partitioning," *Talanta*, vol. 67, no. 4, pp. 736–740, 2005.

[11] W. Gani and M. Limam, "A kernel distance-based representative subset selection method," *Journal of Statistical Computation and Simulation*, vol. 86, no. 1, pp. 135–148, 2016.

[12] T. Gao, L. Hu, Z. Jia, T. Xia, C. Fang, H. Li, L. Hu, Y. Lu, and H. Li, "Spxye: an improved method for partitioning training and validation sets," *Cluster Computing*, pp. 1–10, 2018.

[13] S. Vikram, A. Collier-Oxandale, M. H. Ostertag, M. Menarini, C. Chermak *et al.*, "Evaluating and improving the reliability of gas-phase sensor system calibrations across new locations for ambient measurements and personal exposure monitoring," *Atmospheric Measurement Techniques*, vol. 12, no. 8, pp. 4211–4239, 2019.

[14] Y. Kim, P. Mercati, A. More, E. Shriver, and T. Rosing, "P4: Phase-based power/performance prediction of heterogeneous systems via neural networks," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 683–690.