# Optimizing LCA-based Sparse Dictionary Classification for Low-Power Pattern Recognition in Edge Devices

Lawrence Roman A. Quizon
Electrical and Electronics Engineering Institute
University of the Philippines-Diliman
Quezon City, Philippines

***Abstract*—Sparse representation can be used to perform feature extraction and classification on high-dimensional data and is a mechanism which biological neural sensory systems employ. The Locally Competitive Algorithm (LCA) is a biologically plausible sparse coding mechanism that, if combined with memristor crossbars for in-memory computing, can be effective with low enough power to be suitable for use in edge devices for classifying signals such as energy harvesting contexts and images. This paper explores the tradeoffs of different aspects of the algorithm such as the dictionary completeness, activation threshold, and number of iterations in order to increase LCA efficiency in memristor crossbars on the edge in anticipation of challenges such as the quantization, variance and energy constraints. An accuracy as high as 100% was achieved for classifying 5 different gestures from synthetic solar energy harvesting data using 2 iterations of the LCA and a 128x50 dictionary.**

## I. Introduction

Internet of Things (IoT) devices have a wide range of applications in areas ranging from wearable devices, infrastructure, and wireless sensing. IoT devices have stringent resource constraints due to their small size and portability requirements, limiting area available for batteries. To this end, energy harvesting IoT devices have also been developed. In order to further reduce the size and energy consumption, schemes in which the context around an IoT device can be inferred from its energy harvesting patterns, thus removing the need for sensors, have been developed. These schemes, however, usually involve machine learning or other signal processing which need the data to be transferred to a more powerful central server [1].

In the paradigm of edge computing, information processing takes place not only in powerful servers in "center nodes" but also in the "edges" of a network. By effectively reducing the volume of transmitted data by sending only postcomputation much less energy, bandwidth, and time is needed for the entire network to send information. This allows networks of edge nodes like wireless sensor networks designed for low-power operation tasks to save even more energy by avoiding energy expensive transmission over multiple hops and/or long distances.

Sparse Dictionary Classification was classification algorithm developed for face recognition which has now also found use in EH context sensing [2][3]. While there are a variety of algorithms that exist to obtain sparse representations, the "locally competitive algorithm" (LCA) is particularly promising for use in edge computing due to its compatibility with the memristor crossbar architecture, on which it has been implemented before [4]. As memristor crossbars have problems such like conductance variability and quantization, these problems are also examined for its effects on the accuracy of the scheme.

## II. Sparse Dictionary Classification using BPDN and the Locally Competitive Algorithm

Sparse representation involves forming a dictionary of features as a linear combination of the training feature vectors of the same class. Mathematically, the objective of sparse coding can be formulated as the following:

$$min_a(|x - Da^T|_2 + \lambda|a|_0)$$

where $x$ refers to the input feature vector, $a$ refers to the sparse representation, and $|\cdot|_2$ and $|\cdot|_0$ are the $L^2$ and $L^0$ norms, terms which prioritize reconstruction accuracy and sparseness respectively [3]. Aside from forming the dictionary just from the training feature vectors, further optimizations can be applied to the dictionary, known as sparse dictionary learning, where dictionary elements are refined according to some objective function.

Sparse dictionary classification can be done by appending multiple sparse dictionaries together. The coefficients of the linear combination of training vectors of some test vector of class K will then be nonzero for elements of the dictionary corresponding to class K, thus enabling classification. In this work, sparse dictionary classification was used to classify 5 different gestures that could be synthesized by the SolarGest program. The SolarGest program simulates the current vs time waveforms produced by solar panels when these gestures made above them [5]. SolarGest takes in parameters such as hand speed, hand size, and average start and end position in order to generate the waveform. To obtain a distribution of samples, these parameters were taken as a normal distribution based on the means and variances outlined in Table II. Furthermore, in order to turn the gesture waveforms into feature vectors, these current-time waveforms are subsampled to the same number of samples as each other, aligning them. In this paper, the feature vectors are vectors of length 128.

TABLE I
SOLARGEST PARAMETER MEANS AND VARIANCES

| Parameter | Mean | StDev |
|---|---|---|
| Hand Diameter | $9.72cm$ | $1.14cm$ |
| Hand Pos Low | $2cm$ | $0.1cm$ |
| Hand Pos High | $10cm$ | $1cm$ |
| Gesture Speed | $20cm/s$ | $1cm/s$ |
| Hand Height (for horizontal gestures) | $5cm$ | $1cm$ |
| Solar Cell Radius | $2cm$ | |
| Light Intensity | $200lux$ | |
| Solar Cell Current Density | $7mA/cm^2$ | |

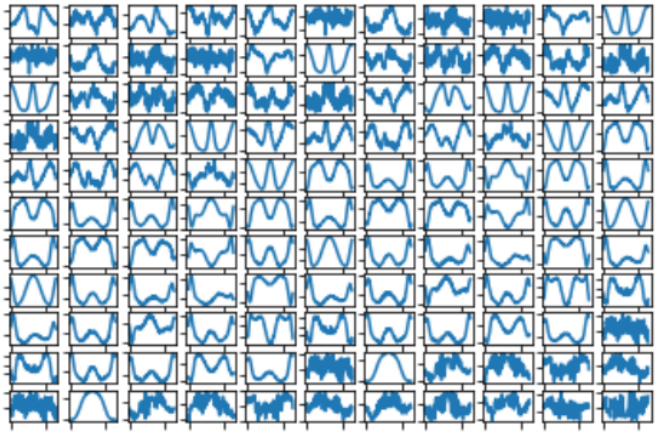

Fig. 1. Unquantized K=25 Concatenated Dictionary

A training set of 100 feature vectors each of 5 gestures was used to learn the dictionaries. Dictionaries have a total of $K * Nclasses$ elements, where $N$ is the number of classes and $K$ is the number of elements per class. Each element is a feature vector, and thus have the length of a feature vector. A total of 4 dictionaries with $K = 150, 50, 25, 10$ were learned using BPDN, along with an unlearned dictionary containing the test vectors and a completely random dictionary. A separate test set of 500 samples of each class was also obtained from the program.

The sparse codes of each testing sample were obtained using the LCA. In the LCA, dictionary elements are modelled as leaky neurons with membrane potentials. The membrane potential $u$ of an LCA neuron is described by the differential equation:

$$\frac{du}{dt} = \frac{1}{\tau}(-u + x^T D - a(D^T D - I_n))$$

$$a = \begin{cases} u & if u > \lambda \\ 0 & otherwise \end{cases}$$

Governed by the parameters $\tau$ and $\lambda$, the change in the potential $u$ is proportional by $1/\tau$ to three terms: its size

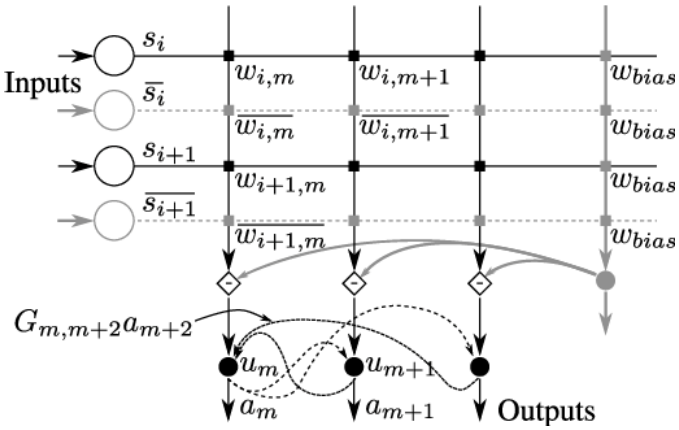


Fig. 2. Locally Competitive Algorithm Schematic

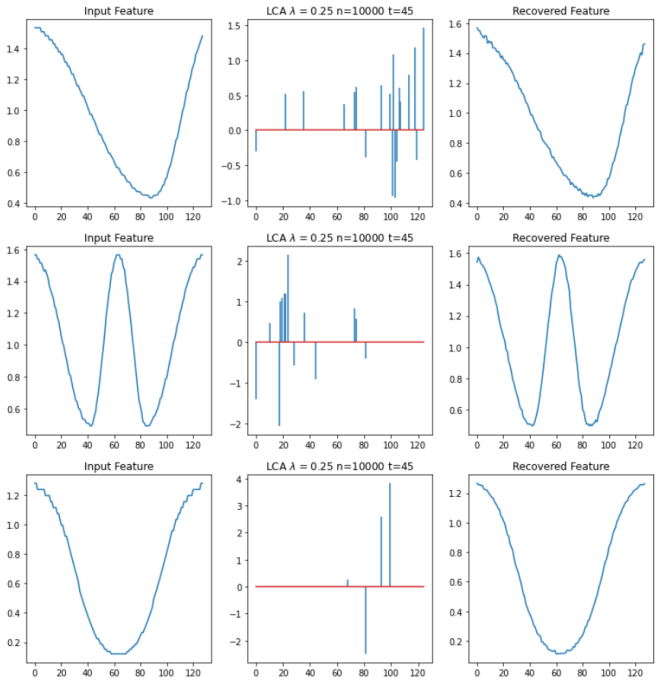


Fig. 3. Sample Gesture Sparse Coding using $K = 25, Q = \infty$ Dictionary. From the top: Left gesture, LeftRight gesture, DownUp gesture.

$(-u)$, its similarity to the input $(x^T D)$ and inhibition from similar neurons $(a(D^T D - I_n))$ where $D^T D - I_n$ is the neuron similarity matrix. If a neuron's potential $u$ is above the threshold $\lambda$, it is deemed active, and it starts contributing to the inhibition term.

## III. RESULTS

### A. Reconstruction Accuracy

Shown in Figure 4 is a sample feature reconstruction of various gestures. The LCA fails to converge to a sparse representation using either the untrained dictionary containing the only the training feature vectors or the random dictionary.

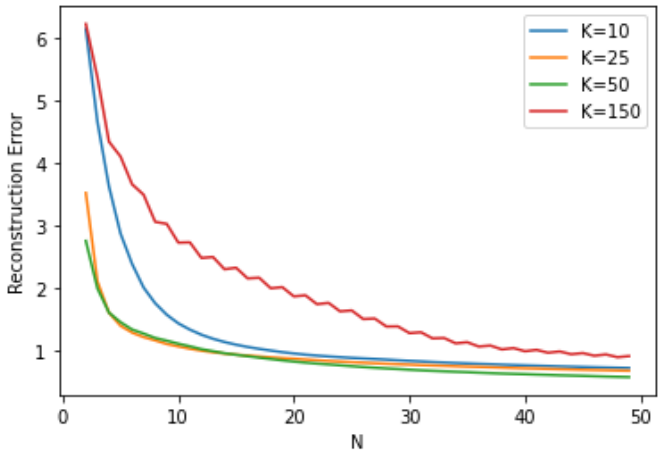


Fig. 4. Average reconstruction error $|x - Da|_2$ for each dictionary vs $n$ ($Q = \infty$). $\lambda, \tau = 0.25, 40$.

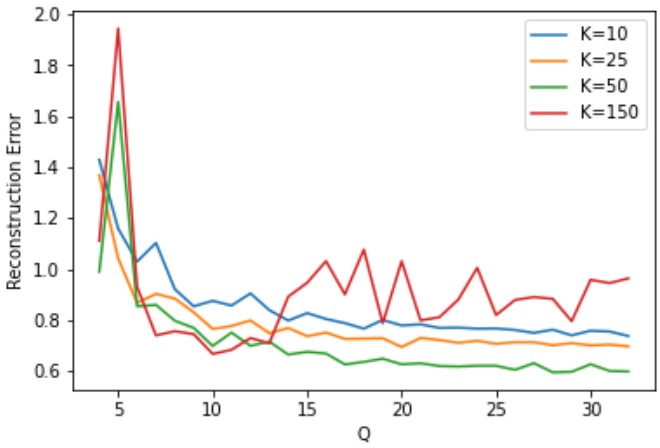


Fig. 5. Average reconstruction error $|x - Da|_2$ vs number of quantization levels $Q$ vs $K$ at $n = 45$. $\lambda, \tau = 0.25, 40$.

In contrast, a BPDN sparse coding algorithm can successfully create a decent sparse representation using these [6]. For dictionaries trained with the BPDN, the LCA can settle into a good sparse representation for all $K = 10, 25, 50, 150$ where $K$ is the number of dictionary elements per class (albeit with a bit of difficulty with the $K = 150$ dictionary).

The LCA can be thought of as a sort of refinement algorithm for the obtained sparse code, where more iterations lead to sparser outputs and less reconstruction error, as shown in Figure 4. In general, it seems that a lower $K$ will lead to more reconstruction error for the same number of iterations. However, overincreasing $K$ will introduce a lot of unnecessary neurons to the code whose feature space (the loose set of feature vectors that will activate it strongly) will overlap with each other, introducing a lot of oscillation and generally worse reconstruction.

When the dictionaries are quantized, the lower the quantization number the lower the reconstruction accuracy gets for a specific number of iterations. This reconstruction accuracy does not get better over more iterations, resulting in a sort of steady-state error, as seen in Figure 5. The quantization can sometimes randomly be better for the reconstruction, and $K = 150$ having more elements is subject to this effect the most while the others can be observed to get less random fluctionations as the dictionaries get smaller.

### B. Classification Accuracy

Using as few iterations for the LCA as possible is ideal to lower the energy consumption of the computation in a memristor crossbar. Very few iterations (as low as 2) can be used to get as high as 100% classification accuracy on the SolarGest test set. At that point, however, the scheme more resembles a simple transformation matrix (the dictionary) with a single refinement step (the one LCA step).

Intuitively, it should be desirable to decrease the parameter $\tau$ as it's inversely propotional to the speed at which LCA "learns" with respect to each iteration. However, decreasing $\tau$ too much can lead to suboptimal reconstruction accuracy. The parameter $\lambda$ contributes to the sparsity of the final code since it decides when a neuron can start inhibiting others.

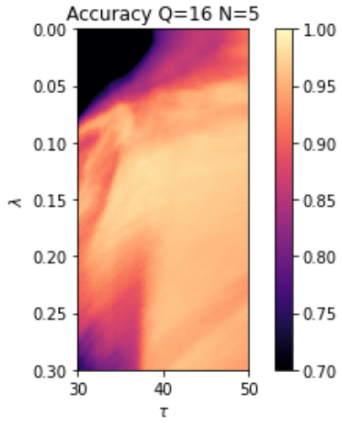


Fig. 6. $\tau - \lambda$ colormap of classification accuracies for $K = 150$,$Q = 16, n = 5$. Includes a traversal map of the gradient ascent algorithm.

A low $\tau$ can lead to lower classification accuracies, as can be seen in the colormap in Figure 6. It can also be garnered from the same figure that there are optimal combinations for $\tau$and $\lambda$. It can be observed that high classification accuracies do not directly correspond to higher reconstruction accuracies, which makes sense because the only thing important for the classification is the maximum value of the sparse representation $max(a)$ while the entire sparse representation $a$ is important for feature vector reconstruction. As the number of iterations is changed, so is the optimal $\tau - \lambda$ combination, as can be observed in Figure 7. The graphs look somewhat smooth, however, the surface starts to get a bit rough with lots of local maxima as plots with more resolution are taken.

In order to traverse these approximately smooth-looking maps, a stochastic gradient ascent algorithm that seeks to maximize classification accuracy over the 2d plane of $\tau$ and $\lambda$ was used. The derivatives are linearly approximated with a set

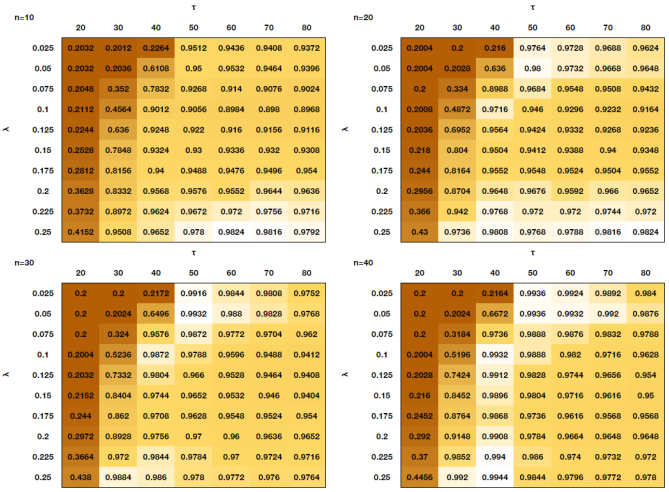


Fig. 7. Classification Accuracies on $K, Q = 150, 32$ for different $n$, showing the movement of the optimal $\tau - \lambda$ point as $n$ changes.

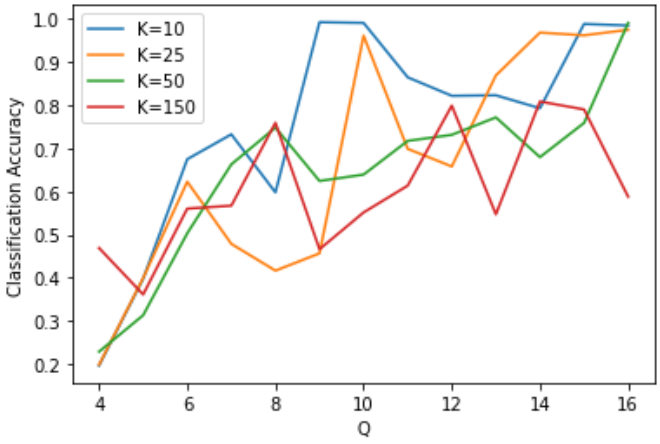


Fig. 8. Line graph of classification accuracies for $K = 150, 50, 25, 10$ at differing $Q$, $n = 2$. Note how there is a specific $Q$ for some $K$ at which the accuracy starts plummeting. Gradient Ascent was used to obtain these. It should be noted that this graph's shape remains approximately the same for any tau and threshold combination that give good results. $K = 150$ underperforms at $n = 2$, but as $n$ increases the others start underperforming and $K = 150$ starts getting better.

stepsize for calculation and steps are made proportional to the approximated derivatives with a learning rate. This algorithm tends to quickly find some local maximum, but the function gets very rough as you zoom in, and so it is very unlikely to encounter a global maxima. For all configurations of $K, n$, and $Q$, good starting points for the algorithm tend to be around $\tau \in (20, 30), \lambda \in (0.1, 0.3)$ at least for $n$ under 100.

A lower $K$ and $Q$ surprisingly results in better classification accuracies, getting as much as $100\%$ classification accuracy on this synthetic SolarGest test set. Therefore, this may suggest that concatenating dictionaries as undercomplete (as low $K$) as possible might be ideal for classification using LCA. An interesting behavior arises from reducing the dimensionality $K$ of the dictionary (dictionaries are matrices with length of the feature vector and width $K * Nclasses$) . It can be seen from Figure 8 that some combinations of $Q$ and $K$ lead to poorer results than for lower $Q$. This may be a somewhat random contribution where some quantizations are coincidentally better. In particular, it can be noted that $K = 10$ and 25 plummet after $Q = 10$ and 9 respectively. After $Q = 8$, all the dictionaries work suboptimally.

A higher $K$ also benefits from a higher number of iterations as seen in Figure 9. This can likely be due to the fact that a high $K$ allows for more coincidentally activated elements of the wrong class, while more quantized (lower $Q$) dictionaries are more susceptible to a sort of steady state error as the number of iterations increase.

Interestingly as well, $K = 150$ is shown to perform better for odd $n$. This may be due to oscillation in the LCA output, where the input $x$ tends to activate a specific set of neurons $a_x \subset a$, but these neurons are extremely similar to each other and thus tend to beat each other down through inhibition in the very next cycle. This is removed by decreasing $K$, since forcing the BPDN dictionary learning algorithm to work with less elements forces it to make the elements more dissimilar.

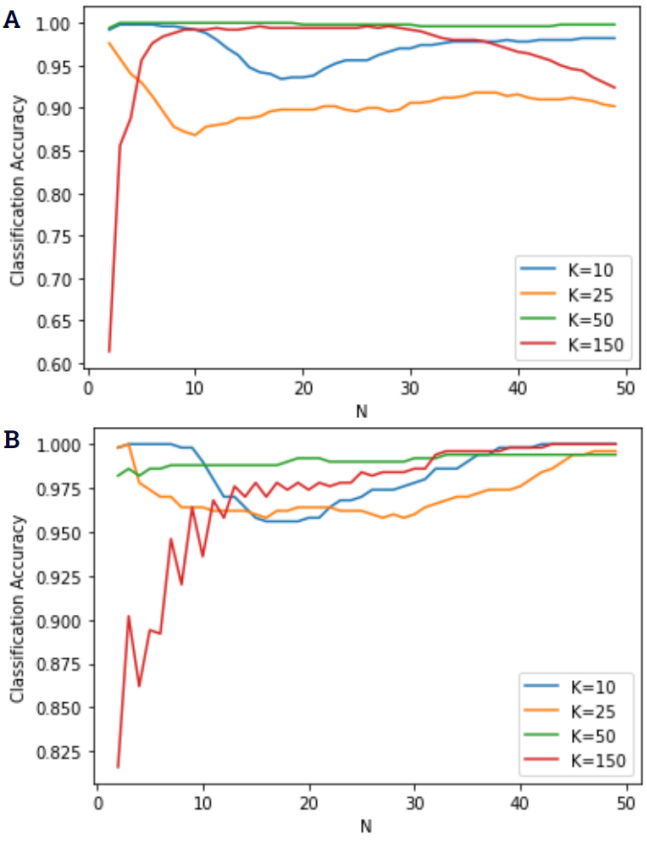


Fig. 9. Line graph of classification accuracy vs number of iterations for $Q = 16$ (a) and $Q = \infty$ (b)

**Algorithm 1** Stochastic Gradient Ascent

```
For each epoch until maxEpochs
  Obtain subsampled test set of N batches
  For each batch
     Test accuracy
     If accuracy < last accuracy
        Backtrack operating point
        Step forward by derivative approximation step
     Approximate the slope of the accuracy function s
     Update the operating point proportional to s
Test the accuracy of the winning operating point
Output the winning point and its accuracy
```

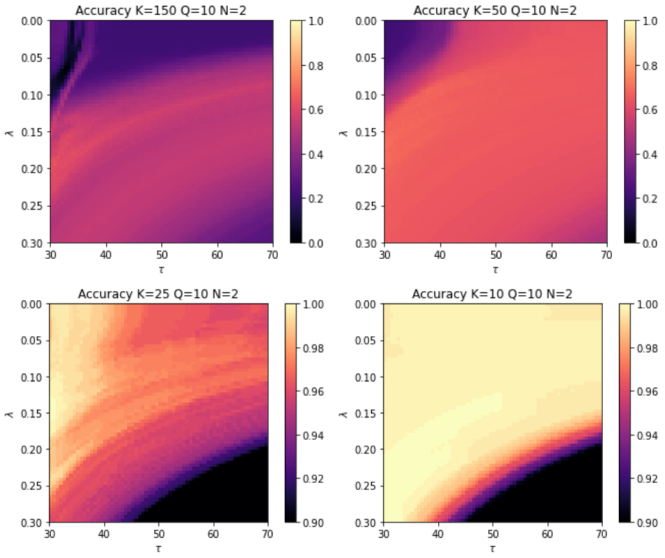


Fig. 10. $\tau - \lambda$ colormaps of classification accuracies for $K = 150, 50, 25, 10$ at $Q = 10$, $n = 2$

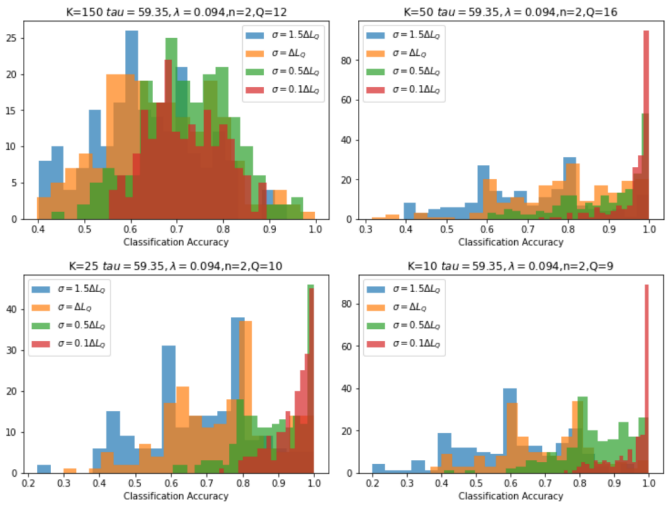


Fig. 11. Histograms of each dictionary on the lowest quantization level $Q$ where they achieve great accuracy results as seen in Figure 7 for $n = 2$. $K, Q = (150, 12), (50, 8), (25, 10), (10, 9)$, sweeping the standard deviation $\sigma$.

### C. Effects of Conductance Variation

The standard deviation $\sigma$ of the conductance of a memristor crossbar tends to be around the difference of two conductance levels, henceforth noted as $\Delta L_Q$. This analysis assumes that each quantization level varies as much as each other, and that the averages are uniformly distributed between the minimum and the maximum. Shown in Figure 11 are the histograms of each dictionary on the lowest quantization level $Q$ where they achieve great accuracy results, since a low $Q$ is ideal for memristor fabrication and write methods.

Standard deviations as much as $\Delta L_Q$ can render the scheme unusable, and it is visible that even a $0.1\Delta L_Q$ variation can affect the accuracy to as low as $80\%$, which needs to be noted when choosing the memristor to implement the scheme with. Fortunately, the scheme is shown to work to very low $Q$, which increases memristor design space for less variation.

## IV. Conclusion

This paper showed that Sparse Coding with LCA can potentially be used and optimized as a machine learning algorithm suitable for use in edge devices such as energy harvesting wireless sensor nodes if memristors crossbar hardware is used for the algorithm. As much as $100\%$ classification accuracy was able to be obtained with this generated SolarGest dataset.

A smaller dictionary size $K$ tends to give better classification accuracy in general. Larger dictionaries tend to perform better with more iterations $n$. Smaller dictionaries require less iterations to work well, and can also perform well up to less quantization levels $Q$.

Quantizing the dictionary using less levels $Q$ can give higher classification accuracies, but this is in a somewhat rough fashion but the trend is visible. On the other hand, lower $Q$ also leads to worse reconstruction accuracy and steady state reconstruction error that remains as the number of iterations $n$ increases.

When conductance variations are introduced, the accuracies

can vary significantly. To deal with this, the use of memristors less sensitive to process variations or the use of compound memristor nodes may be needed- the latter needing much more complicated write circuit design.

Using a quantization-aware dictionary learning algorithm like QK-SVD may allow the use of memristors with even less conductance levels $Q$ which may allow for more robustness and less variance for the same write precision for the memristors by allowing access to quantization levels below 8 [reference]. Further testing with other classification tasks such as face recognition and use on kinetic energy harvesting data is needed to confirm the effectiveness of the scheme, considering the apparent triviality of classifying the dataset generated from SolarGest.

## REFERENCES


[1] D. Ma *et al.*, "Sensing, computing, and communications for energy harvesting iots: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1222–1250, 2019.

[2] W. Xu *et al.*, "Keh-gait: Using kinetic energy harvesting for gait-based user authentication systems," *IEEE Transactions on Mobile Computing*, vol. 18, no. 1, pp. 139–152, 2018.

[3] S. J. Wright, R. D. Nowak, and M. A. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Transactions on signal processing*, vol. 57, no. 7, pp. 2479–2493, 2009.

[4] P. M. Sheridan *et al.*, "Sparse coding with memristor networks," *Nature nanotechnology*, vol. 12, no. 8, pp. 784–789, 2017.

[5] D. Ma *et al.*, "Solargest: Ubiquitous and battery-free gesture recognition using solar cells," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–15.

[6] B. Wohlberg, "Sporco: A python package for standard and convolutional sparse representations," in *Proceedings of the 15th Python in Science Conference, Austin, TX, USA*, 2017, pp. 1–8.