PointLCA-Net: Using Point Clouds for Energy Efficient Sparse Spatio-Temporal Signal Recognition in Neuromorphic Systems

Sanaz Mahmoodi Takaghaj

Department of Computer Science and Engineering, and
School of Engineering Design and Innovation
Penn State University
University Park, PA USA
sxm788@psu.edu

Abstract—In recent years, the field of neuromorphic systems has gained significant traction as the demand for AI applications in energy-constrained autonomous systems, such as unmanned aerial vehicles (UAVs) and robotics, continues to grow. The advent of Dynamic Vision Sensor (DVS) and other event-based sensors, which generate sparse spatio-temporal signals, has propelled the field of neuromorphic engineering. This advancement calls for the development of new machine learning models or the refinement of existing ones to effectively leverage neural dynamics and exploit spike-based communication among neurons. These sparse spatio-temporal signals can be treated as 3D point sets (point clouds), representing the space and time at which these events occur, and can be processed using PointNet architectures. However, applying deep learning techniques such as backpropagation and deploying these models on energy-constrained neuromorphic devices presents challenges in computational resources and memory, as well as real-time processing and power consumption. This work presents PointLCA-Net, which leverages the strengths of PointNets for extracting robust features from input point sets, while utilizing the efficiency of Exemplar LCA to encode these features. PointLCA-Net achieves a high accuracy of 93.41% on DVS128 while reducing energy consumption by approximately 92% compared to other spiking neural networks applied to point clouds, making the system more efficient and well-suited for energy-constrained applications.

Index Terms—Encoder-Decoder architecture, Exemplar LCA, Spatio-temporal data, PointNets

I. INTRODUCTION

Deep learning has significantly advanced the analysis of complex data structures, with architectures such as PointNet [1] and PointNet++ [2] revolutionizing the processing of three-dimensional (3D) point sets (point clouds). These architectures have led to substantial improvements in tasks such as 3D object classification and segmentation [1]–[3]. PointNet takes point clouds as input, applies input and feature transformations, and aggregates the point features using a max pooling function. The network outputs a set of classification scores for k classes, along with segmentation scores. PointNet++ is a hierarchical extension of PointNet that applies the architecture recursively to a nested partitioning of the input set and adaptively combines features from multiple scales to enhance the generalizability of PointNet to complex 3D shapes

and point cloud structures. Unlike PointNet, which processes points globally, PointNet++ processes points within local neighborhoods, performing additional clustering and multi-scale feature extraction.

Point clouds can represent discrete spatio-temporal signals by capturing both spatial and temporal information as discrete points in space-time. Wang et al. [4] proposed a method for constructing Space-time Event Clouds from Dynamic Vision Sensors (DVSs) and applied the PointNet and PointNet++ architectures to gesture recognition task on DVS128 [7]. Despite these advancements, applying such techniques on edge devices — where energy efficiency is crucial — remains challenging due to the limitations of conventional computing systems. These challenges include issues related to data movement, memory capacity, and power consumption. Conventional processors, such as CPUs and GPUs/TPUs, are optimized for handling dense, synchronized data structures through efficient instruction and data pipelines. However, their performance significantly deteriorates when dealing with sparse, asynchronous event streams [5]. This calls for a departure from traditional von Neumann computing systems, with neuromorphic computing serving as one potential solution.

Recently, neuromorphic systems – drawing inspiration from the human brain's capabilities and/or structures - have emerged as a transformative approach in computing systems, offering energy-efficient and massively parallel processing elements. A new wave of neuromorphic processors including TrueNorth [6], [7], Loihi [8], [9], SpiNNaker [10]–[12], BrainScaleS [13], [14], Tianjic [15], and DYNAP-SE [16] has been developed to deliver low-latency and low-power consumption, making them well-suited for modeling Spiking Neural Networks (SNNs) [17], [18] and processing sparse data streams from event-based sensors.

SNNs represent a compelling paradigm for modeling temporal dynamics, providing advantages in processing sequential data through discrete spike events that emulate biological neural activity. SNNs have the potential to excel at representing and processing temporal sequences, making them particularly well-suited for tasks involving dynamic patterns and time-
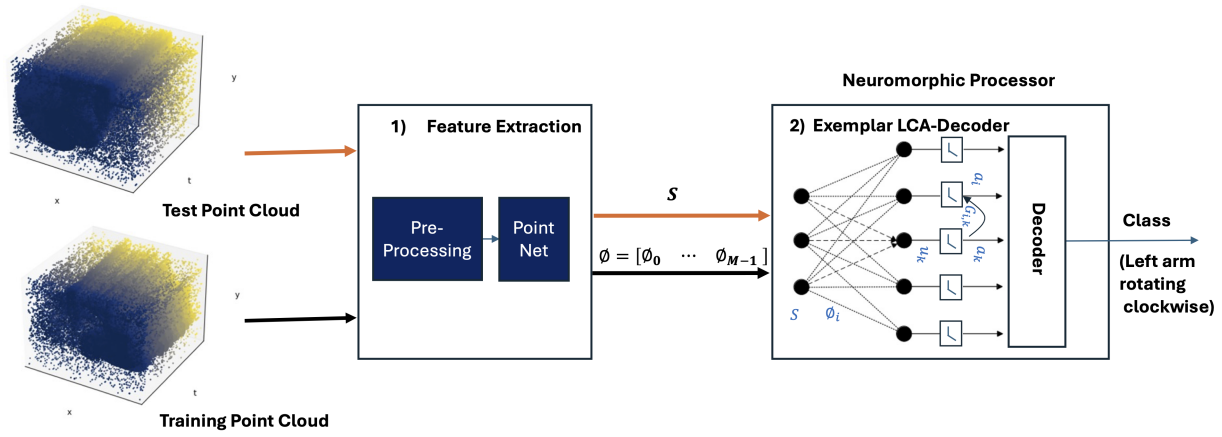
Fig. 1. PointLCA-Net Architecture: Features (ϕ_i) are extracted from the training data and stored in the synaptic weights of the Exemplar LCA-Decoder. The orange arrows represent the inference process, which follows the completion of training (feature extraction).

varying signals.

Neuromorphic systems can efficiently execute Vector-Matrix Multiplication (VMM) operations, also known as Multiply-and-Accumulate (MAC) operations, by leveraging a crossbar array of memory elements [42], which are fundamental to deep learning algorithms. Non-volatile memory elements, such as PCM [19]–[21] and RRAM [22]–[25], have been proposed for integration into synapse and neuron circuits [26]. While most efforts to apply in-memory computing to SNNs have focused on unsupervised learning with local learning rules, such as Spike-Timing Dependent Plasticity (STDP) [27], which adjusts synaptic weights based on the relative timing between inputs and outputs, these approaches encounter significant challenges when applied to complex datasets and tasks. STDP-based learning rules have limited scalability and are not well-suited for more complex tasks, such as recognizing spatio-temporal signals. As a result, there is a clear need for new learning rules that can handle dynamic data and address these complex challenges.

A particularly interesting model in neuromorphic computing is the Exemplar LCA-Decoder [28], [29], which builds upon the Locally Competitive Algorithm (LCA) [30], [31]. The Exemplar LCA-Decoder is a computational model and learning algorithm that iteratively updates neuron activity to achieve a sparse representation of input data. As a single-layer encoder-decoder architecture, it first encodes the input data features by identifying a sparse representation through neuron activity and then uses this sparse representation for classification tasks during the decoding phase. A key component of this algorithm is the construction of a comprehensive dictionary of input data features.

This work builds upon the foundational principles of the Exemplar LCA-Decoder and introduces a novel algorithm, PointLCA-Net, to address the challenges of recognizing spatio-temporal signals on neuromorphic platforms. We propose a two-stage approach in which spatio-temporal data, such as that from DVS cameras, is initially preprocessed and

converted into point clouds. In the first stage, a pre-trained PointNet generates robust feature representations from this 3D point cloud data. These features, which capture both geometric and semantic information, are then input into the Exemplar LCA-Decoder architecture in the second stage for encoding and decoding. These features are extracted once and stored on non-volatile memory elements, allowing for in-memory computation on neuromorphic platforms that prioritize specialized operations and energy-efficient processing. The Locally Competitive Algorithm (LCA) [30] is used as the encoder to ensure that only a limited number of neurons are active at any given time, enabling sparsity and efficient coding of data. We evaluated PointLCA-Net on three datasets: NMNIST [32], DVS128 [7], and Spiking Heidelberg Digits (SHD) [33], demonstrating how it can be uniformly applied to each of these datasets in a way that is deployable on neuromorphic platforms. We achieved high recognition accuracy across all these datasets while maintaining low computational overhead and energy efficiency, making the approach well-suited for deployment on neuromorphic systems.

The contributions of this work include the following:

- A neuromorphic and energy-efficient adaptation of the PointNet and PointNet++ models.
- First application of the event cloud approach to neuromorphic datasets NMNIST and SHD.
- A proof of concept for an algorithm that can be uniformly applied to various spatio-temporal datasets for energy-efficient signal recognition on neuromorphic hardware.

II. RELATED WORK

Only a modest number of recent works have studied how to apply SNNs to point clouds, primarily due to the prohibitive computational overhead involved in training.

Spiking PointNet [34] is the first spiking version of PointNet, which leverages spiking neurons. The model is trained using a single time step, with membrane potential perturbation introduced to improve its performance. It provides some speedup and storage savings during the training phase and, in

certain experiments, surpasses its ANN counterpart, suggesting the potential of SNNs for point cloud tasks. However, training with a single time step limits the model's ability to capture complex temporal dynamics, restricting its generalizability to new datasets. Additionally, the inherent complexity of training SNNs in this manner may become prohibitive when applied to more complex datasets beyond the ModelNet10 and ModelNet40 datasets, which are the only ones tested in the Spiking PointNet work. Unlike the structured 3D object data in the ModelNet datasets, DVS128 consists of unstructured spatio-temporal event data, which presents a different set of challenges. This method has not been evaluated on the DVS128 or other datasets considered in our work.

SpikePoint [35] uses the PointNet++ architecture and converts point coordinates into spikes via rate coding. It employs surrogate backpropagation for training the network and has primarily been tested on event camera action recognition datasets. The use of surrogate gradients in this approach involves computing the derivative of a surrogate function, which can be computationally expensive, particularly for complex architectures such as PointNet++. The accuracy and convergence of training with surrogate gradients is also sensitive to the choice of surrogate function's parameters, potentially leading to additional computational costs for tuning and optimization.

III. POINTLCA-NET

Fig. 1 illustrates the architecture of PointLCA-Net. The training dataset is first converted into point clouds consisting of (x, y, t) point sets, pre-processed, and then fed into PointNet for feature extraction. These features are used to construct a dictionary, which is utilized by the Exemplar LCA-Decoder [28] to encode and then decode unseen test point clouds.

A. Feature Extraction

Fig. 2 illustrates the feature extraction process using a pre-trained PointNet [1], which processes point clouds by first aligning the input data points into a canonical space using an input transformation network (Input Transform). Each data point is then independently processed through a shared Multi-Layer Perceptron MLP(64, 64), feature transformation network (Feature Transform) and MLP(64, 128, 1024) to extract point-wise features. These features are aggregated using a symmetric Max Pooling function, to produce a permutation-invariant global feature vector ϕ_i. The resulting 1024-dimensional feature vector captures comprehensive geometric and semantic information from the point cloud. These feature vectors are then used to construct the dictionary ϕ (see Fig. 1), which is later used by the Exemplar LCA-Decoder algorithm (see Eq. 5)).

B. Overview of the Exemplar LCA-Decoder

The Exemplar LCA-Decoder algorithm [28] combines the sparse coding algorithm [30] with the LCA algorithm [31] to encode the input signal S as the activations a_i of the Leaky Integrate and Fire (LIF) neurons where ϕ denotes a dictionary

of features (ϕ_i). The term ε accounts for Gaussian noise in the reconstruction:

$$S = \sum_{i=0}^{M-1} \phi_i a_i + \varepsilon \tag{1}$$

The membrane potential of the LIF neuron at time step k, denoted as $u_i[k]$, is influenced by a driving excitatory input b_i and an inhibition matrix (Gramian) G. The Gramian matrix enables stronger neurons to suppress weaker ones, thereby promoting a sparse coding.

$$\tau \dot{u}_i[k] + u_i[k] = b_i - \sum_{m \neq i}^{M-1} G_{i,m} a_m[k] \tag{2}$$

$$b_i = S\phi_i \tag{3}$$

$$G = \phi^T \phi \tag{4}$$

$$\phi = [\phi_0, \phi_1, \ldots, \phi_{M-1}] \tag{5}$$

In PointLCA-Net, each ϕ_i represents a feature learned from a training point cloud. The thresholding function is:

$$\begin{aligned} a_i[k] &= T_\lambda(u_i[k]) \\ &= \begin{cases} u_i[k] - \lambda \operatorname{sign}(u_i[k]), & |u_i[k]| \geq \lambda \\ 0, & |u_i[k]| < \lambda \end{cases} \end{aligned} \tag{6}$$

Here, the threshold λ represents the level the membrane potential must exceed for the neuron to spike. Next, a decoder is designed to decode the sparse codes a_i and map them to K distinct classes from the training dataset. This same decoder is then used to generate classifications for unseen test point clouds.

The Maximum Activation Code (Eq.7) and Maximum Sum of Activation Codes (Eq.8) were used as decoders for classification. In contrast to a shallow (single layer) neural network decoder, which requires additional training and error back-propagation, the Maximum Activation Code and Maximum Sum of Activation Codes offer computational advantages and greater efficiency.

$$C_{Test} = Class[\underset{i}{argmax}[a_i]], \tag{7}$$

$$C_{Test} = Class[\underset{c}{argmax}(\sum_i \left| a_i^{(c)} \right|)] \tag{8}$$

IV. EXPERIMENT SETUP

We evaluated PointLCA-Net using PyTorch [36] on the following datasets: NMNIST, DVS128 and Spiking Heidelberg Digits (SHD). Events in these discrete spatio-temporal datasets were converted into point clouds, with each event represented by its spatial coordinates (x, y) and temporal component (t). We then concatenate these coordinates into a tensor of shape (x, y, t) for processing.
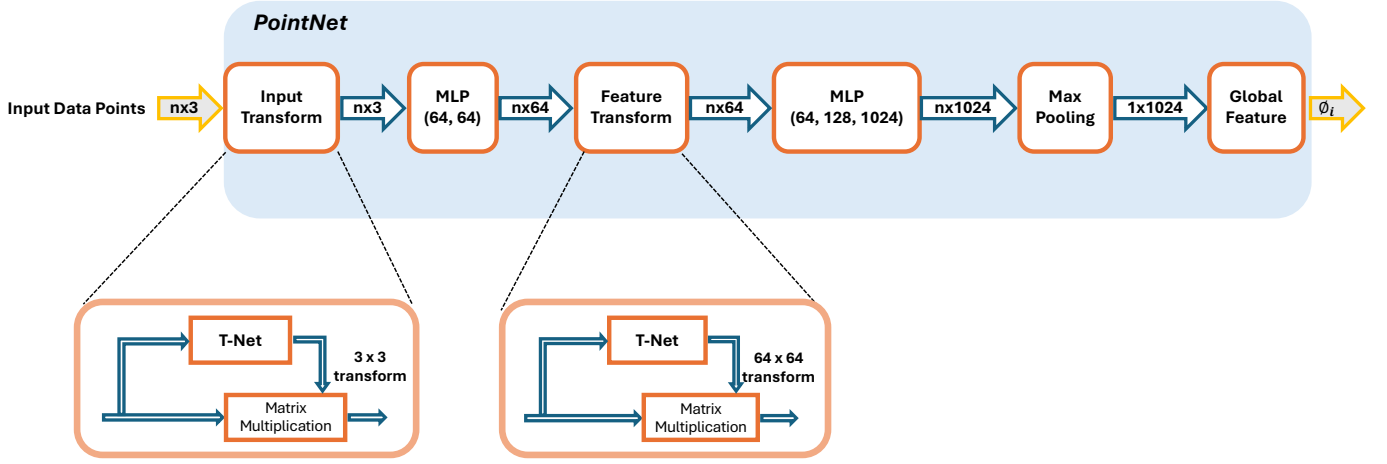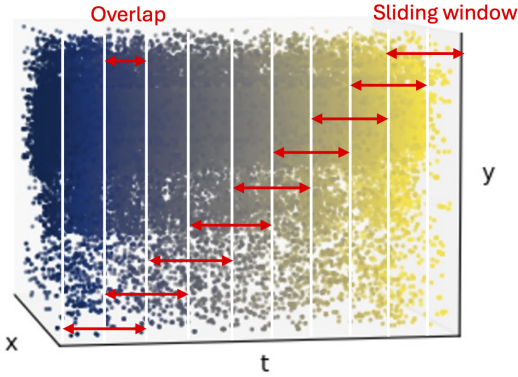
Fig. 2. Feature Extraction using PointNets.

Fig. 3. Data Pre-processing

and 288 samples for testing. These samples are further pre-processed, resulting in 28,606 training samples and 7,408 test samples. In our experiments, we included all 11 classes, including the random gesture class.

D. Dataset Pre-Processing

The preprocessing phase begins by evaluating the duration of the recordings. To account for the temporal variability in the dataset, we segment the data into sliding windows, with both the sizes and number of these windows determined based on the total duration of the recordings (see Fig. 3). The data within each window is then shuffled and down-sampled to select a fixed number of events (128, 256, 512, or 1024). These sampled events are further concatenated, shuffled, and down-sampled, and normalized to create a final subset of 1024 events per data, which is subsequently input into the PointNets for feature extraction.

E. Workload and Energy Efficiency Analysis

This section presents a thorough evaluation of the workload and energy efficiency of PointLCA-Net. We begin by estimating the number of Floating-Point Operations (FLOPs) to assess workload efficiency, followed by an analysis of its energy consumption. These estimations exclude the FLOPs required for feature extraction by PointNet and PointNet++.

Let N represent the global feature size, M the dictionary size (which also corresponds to the number of neurons), and K the total number of time steps. The number of active neurons, denoted as $\hat{M}$, represents the average number of firing neurons for which $a_i \neq 0$. The training and inference FLOPs are then calculated using the approach outlined in [28] and [29]:

$$\underset{(Training)}{FLOPs} = \frac{M(M+1)(2N-1)}{2} \qquad (9)$$

$$\underset{(Inferenec)}{FLOPs} = K(\frac{(2N-1)M}{K} + 2M\hat{M} + M) \qquad (10)$$

A. NMNIST

The NMNIST [32] dataset is a spiking variant of the MNIST dataset, generated by presenting images to a neuromorphic vision camera equipped with the ATIS (Asynchronous Time-based Image Sensor). This dataset comprises 60,000 training images, each consisting of 300 time samples, along with 10,000 test images used for accuracy evaluation.

B. Spiking Heidelberg Digits (SHD)

The SHD [33] dataset comprises spoken digits from zero to nine in both English and German. Audio waveforms within the dataset are transformed into spike trains using an artificial cochlea model. The dataset includes 8,156 training samples and 2,264 test samples, each with a varying temporal duration.

C. DVS128

The DVS128 [7] dataset, captured using a Dynamic Vision Sensor (DVS) camera, includes recordings of 11 distinct hand gestures, with approximately 20,000 to 40,000 events per gesture class. It contains 1,176 samples allocated for training

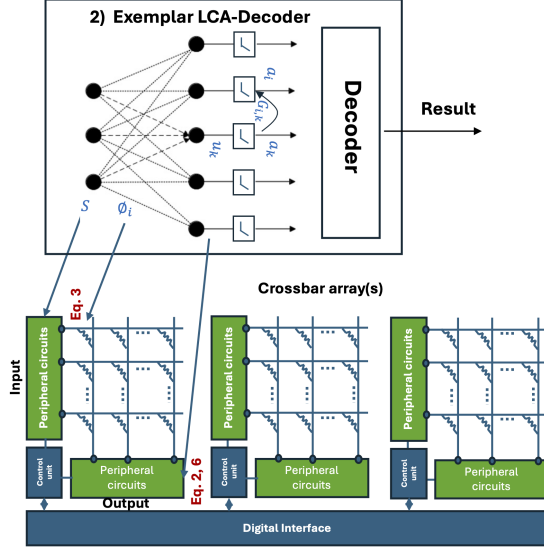| Dataset	Training (TFLOPs)	K	N	M	$\hat{M}$	Inference (GFLOPs)
NMNIST	3.7	100	1024	60K	240	3
		10	1024	60K	240	0.4
DVS128	0.84	100	1024	28.6K	114	0.7
		10	1024	28.6K	114	0.12
SHD	0.07	100	1024	8.156K	33	0.07
		10	1024	8.156K	33	0.02

Fig. 4. PointLCA-Net Hardware Deployment

Table I presents the estimated training and inference FLOPs for PointLCA-Net. "TFLOPs" refers to the Tera FLOPs required for training, while "GFLOPs" indicates the Giga FLOPs for inference operations. According to [28], up to 0.4% of neurons spike at each time step, and we have verified that this estimate is accurate. The spiking sparsity introduced by LCA encoding resulted in an average reduction of 99.54% in computational effort during inference. Furthermore, we explored two scenarios with 100 and 10 time steps (K). Reducing the number of time steps from 100 to 10 led to an average decrease of 80% in inference FLOPs.

F. Neuromorphic Hardware Mapping

The neuromorphic-friendly architecture of the Exemplar LCA-Decoder supports the mapping of PointLCA-Net to low-energy neuromorphic platforms, enabling the use of energy-efficient in-memory computing. Using RRAM crossbar arrays, Yao et al. [37] demonstrated an energy efficiency of 11 Tera FLOPs per watt for MAC operations, which corresponds to an energy consumption of approximately 9.09×10^{-14} joules per floating-point operation. Beyond improving workload efficiency through sparse coding, PointLCA-Net can also benefit from recent advancements in in-memory computing [26] and memristive crossbar arrays to further enhance energy efficiency. Mahmoodi et al. [29] proposed a hardware implementation of the Exemplar LCA-Decoder algorithm in ViT-LCA.

Similarly, Fig.4 depicts the hardware mapping of PointLCA-Net. In this architecture, the values of ϕ_i are represented as the conductance of memristors in each column, which are used to compute the neuron excitatory inputs, as described in Eq.3. Input data is preprocessed in the input peripheral circuits, while neuron dynamics and thresholding (defined in Eq.2 and Eq.6) are handled in the output peripheral circuits of the neural processor.

V. EVALUATION

Table II presents the energy estimations and test accuracy for all three datasets tested. All experiments were conducted using the hyperparameters specified in Table III, selected to achieve an accuracy near 100% when tested on the training data. The results in Table II indicate that the "Maximum Sum of Activations" decoder outperforms the "Maximum Activation" decoder, delivering significantly better performance across all datasets. Additionally, using PointNet++ for feature extraction yielded better results than PointNet. The energy estimation was calculated based on the estimated GFLOPs during inference (Table I) and the expected 9.09×10^{-14} joules energy consumption per floating point operation when the operations are mapped onto the RRAM memory crossbar arrays. Wang et al. [4] reported an accuracy of 90.20% on the DVS128 dataset with 10 classes when trained directly on PointNet, and 94.10% when trained using PointNet++. Their work was the first to apply PointNet and PointNet++ to spatio-temporal data; however, they did not address hardware deployment nor use a spiking approach, leaving the energy efficiency of the approach unknown.

Furthermore, SpikePoint [35], a spiking variant of PointNet++, has been tested on the DVS128 dataset. SpikePoint uses a surrogate-based backpropagation method, achieving an accuracy of 98.6% with a GFLOPs value of 0.9, compared to 0.7 for PointLCA-Net. According to the authors, their study used 26,796 training samples and 6,959 test samples, with the test set comprising "20% randomly selected from the total samples". In contrast, the DVS128 dataset [7] used in this study contains 28,606 training samples and 7,408 test samples, with the test data kept entirely separate from the training data. Given the differences in data partitioning and sample size, directly comparing accuracy may not be fully representative. Additionally, PointLCA-Net is significantly more energy-efficient, with a total energy consumption of 0.065 mJ—approximately 24.23 times lower than that of SpikePoint, which has a reported dynamic energy consumption of 0.82 mJ and a static energy consumption of 0.756 mJ. All energy calculations are based on theoretical estimations and calculations.

VI. DISCUSSION

This study serves as a proof of concept and has significant potential as an algorithm that can be uniformly applied to various spatio-temporal data beyond DVS128, including video and audio. Our focus was on leveraging PointNet and PointNet++ in PointLCA-Net to recognize spatio-temporal signals in an

TABLE II
TOP-1 TEST ACCURACY SCORES FOR POINTLCA-NET.

Dataset	Method	GFLOPs	Energy	Feature Extractor	Accuracy	
					Decoding Methods	
					$\max_i \{a_i\}$	$\max_k \sum_i \left|a_i^{(k)}\right|$
NMNIST	**PointLCA-Net**	3	0.27 mJ	PointNet	95.49%	98.01%
				PointNet++	94.74%	**98.78%**
SHD	**PointLCA-Net**	0.07	6.4 μJ	PointNet	59.85%	65.11%
				PointNet++	66.30%	**78.46%**
DVS128	**PointLCA-Net**	0.7	0.065 mJ	PointNet	82.67%	90.02%
				PointNet++	86.69%	**93.41%**
DVS128	PointNet [4][a]	-	-	-	90.20%[c]	
	PointNet++ [4][a]	0.872[b]	4.01 mJ (Dynamic) and 1.936 mJ (Static)[b]	-	94.10%	
DVS128	SpikePoint [35][a]	0.9	0.82 mJ (Dynamic) and 0.756 mJ (Static)	-	98.1%[d]	

[a]These methods haven't been tested on NMNIST or SHD.

[b] The values are taken from the results reported in [35].

[c] This accuracy was achieved using 10 classes, excluding the random gestures class.

[d]This accuracy was achieved using a different data partitioning and sample size than ours.

TABLE III
HYPERPARAMETERS

Symbols	Description	Value
λ	Threshold	0.2
τ	Leakage	1000
k	Number of time steps	100
Data Pre-Processing	Sliding window	0.5 sec
	Overlap	0.25 sec
	Number of event points	1024

energy-efficient manner that could be mapped onto current neuromorphic hardware, rather than solely enhancing their accuracy. PointLCA-Net currently has limitations in accuracy compared to state-of-the-art methods, and further improvements could be made by exploring alternative architectures for feature extraction, such as Point Transformer [38].

PointLCA-Net does not require any additional training of the PointNet and PointNet++ models used for feature extraction. Once the features are extracted from the training data, a dictionary is constructed and used in the Exemplar LCA-Decoder.

Finally, in this study, we utilized the entire set of training data points to construct the dictionary. While larger dictionary lengths can improve accuracy, they may also create a computational burden for larger datasets. To enhance computational efficiency, one potential approach is to reduce the dictionary length M through Principal Component Analysis (PCA), Support Vector Machines (SVM), or by selecting discriminative features, thereby decreasing the computational overhead. For instance, a 10% reduction in dictionary length could, on average, reduce inference FLOPs by 17% and training FLOPs by 20%. As a result, this opens up new avenues for enhancing computational efficiency and model performance of PointLCA-Net in future studies.

VII. CONCLUSION

In this work, we explored the integration of Exemplar LCA with PointNets for the first time and evaluated its performance on classification tasks using the NMNIST, DVS128, and SHD datasets. We utilized features extracted through PointNet and PointNet++ within an Exemplar LCA encoder-decoder framework, which eliminated the need for training a dictionary as done in the original LCA. Additionally, we demonstrated how this approach can be mapped to memristor crossbar arrays for efficient in-memory computing on edge devices. This study highlights the potential for further research, including the investigation of alternative architectures such as Point Transformer for even higher performance. Given the energy efficiency potential of PointLCA-Net and its adaptability to various datasets, it presents an opportunity for enhancement in spatio-temporal signal recognition.

VIII. ACKNOWLEDGEMENT

REFERENCES

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 652–660, 2017.

[2] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[3] X. Ma, C. Qin, H. You, H. Ran, and Y. Fu, "Rethinking network design and local geometry in point cloud: A simple residual MLP framework," *arXiv preprint arXiv:2202.07123*, 2022.

[4] Q. Wang, Y. Zhang, J. Yuan, and Y. Lu, "Space-time event clouds for gesture recognition: From RGB cameras to event cameras," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 1826–1835.

[5] S. Dave, R. Baghdadi, T. Nowatzki, S. Avancha, A. Shrivastava, and B. Li, "Hardware acceleration of sparse and irregular tensor computations of ML models: A survey and insights," *Proc. IEEE*, vol. 109, no. 10, pp. 1706–1752, 2021.

[6] Merolla, P. A., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197), 668–673.

[7] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, and others, "A low power, fully event-based gesture recognition system," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2017, pp. 7243–7252.

[8] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, and others, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[9] Davies, M., Wild, A., Orchard, G., Sandamirskaya, Y., Guerra, G. A. F., Joshi, P., Plank, P., & Risbud, S. R. (2021). Advancing neuromorphic computing with Loihi: A survey of results and outlook. *Proceedings of the IEEE*, 109(5), 911–934. IEEE.

[10] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proc. IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[11] Furber, S. (2016). Large-scale neuromorphic computing systems. *Journal of Neural Engineering*, 13(5), 051001. IOP Publishing.

[12] S. Höppner *et al.*, "The SpiNNaker 2 processing element architecture for hybrid digital neuromorphic computing," *arXiv preprint arXiv:2103.08392*, 2021.

[13] Schmitt, S., Klähn, J., Bellec, G., Grübl, A., Guettler, M., Hartel, A., Hartmann, S., Husmann, D., Husmann, K., Jeltsch, S., et al. (2017). Neuromorphic hardware in the loop: Training a deep spiking network on the BrainScaleS wafer-scale system. In *2017 International Joint Conference on Neural Networks (IJCNN)* (pp. 2227–2234). IEEE.

[14] Pehle, C., Billaudelle, S., Cramer, B., Kaiser, J., Schreiber, K., Stradmann, Y., Weis, J., Leibfried, A., Müller, E., & Schemmel, J. (2022). The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity. *Frontiers in Neuroscience*, 16, 795876. Frontiers Media SA.

[15] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, and others, "Towards artificial general intelligence with hybrid Tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, 2019.

[16] Richter, O., Wu, C., Whatley, A. M., Köstinger, G., Nielsen, C., Qiao, N., & Indiveri, G. (2024). DYNAP-SE2: A scalable multi-core dynamic neuromorphic asynchronous spiking neural network processor. *Neuromorphic Computing and Engineering*, 4(1), 014003. IOP Publishing.

[17] Ghosh-Dastidar, S., & Adeli, H. (2009). Spiking neural networks. *International Journal of Neural Systems*, 19(04), 295–308. World Scientific.

[18] Grüning, A., & Bohte, S. M. (2014). Spiking neural networks: Principles and challenges. In *ESANN*.

[19] S. Kim *et al.*, "NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous in-situ learning," in *Proc. Int. Electron Devices Meeting (IEDM)*, 2015, pp. 17–1.

[20] Kuzum, Duygu, Rakesh GD Jeyasingh, Byoungil Lee, and H-S Philip Wong. "Nanoelectronic Programmable Synapses Based on Phase Change Materials for Brain-Inspired Computing." *Nano Letters*, vol. 12, no. 5, 2012, pp. 2179-2186. ACS Publications.

[21] T. Tuma, M. Le Gallo, A. Sebastian, and E. Eleftheriou, "Detecting correlations using phase-change neurons and synapses," *IEEE Electron Device Lett.*, vol. 37, no. 10, pp. 1238–1241, Oct. 2016.

[22] E. Covi *et al.*, "Analog memristive synapse in spiking networks implementing unsupervised learning," *Front. Neurosci.*, vol. 10, p. 482, 2016.

[23] A. Pantazi, S. Woźniak, T. Tuma, and E. Eleftheriou, "All-memristive neuromorphic computing with level-tuned neurons," *Nanotechnology*, vol. 27, no. 35, p. 355205, 2016, IOP Publishing.

[24] M. Rao, H. Tang, J. Wu, W. Song, M. Zhang, W. Yin, Y. Zhuo, F. Kiani, B. Chen, X. Jiang, and others, "Thousands of conductance levels in memristors integrated on CMOS," *Nature*, vol. 615, no. 7954, pp. 823–829, 2023.

[25] A. Serb *et al.*, "Unsupervised learning in probabilistic neural networks with multi-state metal-oxide memristive synapses," *Nature Communications*, vol. 7, p. 12611, 2016.

[26] Sebastian, Abu, Le Gallo, Manuel, Khaddam-Aljameh, Riduan, and Eleftheriou, Evangelos. "Memory devices and applications for in-memory computing." *Nature Nanotechnology* 15, no. 7 (2020): 529–544. Nature Publishing Group UK London.

[27] Indiveri, Giacomo, and Liu, Shih-Chii. "Memory and information processing in neuromorphic systems." *Proceedings of the IEEE* 103, no. 8 (2015): 1379–1397. IEEE.

[28] Mahmoodi Takaghaj, Sanaz, and Jack Sampson."D-SELD: Dataset-Scalable Exemplar LCA-Decoder." *Neuromorphic Computing and Engineering*, 2024.

[29] Mahmoodi Takaghaj, Sanaz. "ViT-LCA: A Neuromorphic Approach for Vision Transformers." *arXiv preprint arXiv:2411.00140*, 2024.

[30] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen, "Sparse coding via thresholding and local competition in neural circuits," *Neural Computation*, vol. 20, no. 10, pp. 2526–2563, 2008.

[31] C. Rozell, D. Johnson, R. Baraniuk, and B. Olshausen, "Locally Competitive Algorithms for Sparse Approximation," in *Proc. 2007 IEEE Int. Conf. Image Processing*, vol. 4, pp. IV-169–IV-172, 2007, doi: 10.1109/ICIP.2007.4379981.

[32] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades," *Frontiers in Neuroscience*, vol. 9, p. 437, 2015."

[33] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The Heidelberg Spiking Data Sets for the Systematic Evaluation of Spiking Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 7, pp. 2744–2757, 2020.

[34] D. Ren, Z. Ma, Y. Chen, W. Peng, X. Liu, Y. Zhang, and Y. Guo, "Spiking pointnet: Spiking neural networks for point clouds," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[35] H. Ren, Y. Zhou, X. Lin, Y. Huang, H. Fu, J. Song, and B. Cheng, "SpikePoint: An Efficient Point-based Spiking Neural Network for Event Cameras Action Recognition," in *Proc. of the Twelfth International Conference on Learning Representations*, 2024.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, and others, "PyTorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[37] Yao, Peng, Wu, Huaqiang, Gao, Bin, Tang, Jianshi, Zhang, Qingtian, Zhang, Wenqiang, Yang, J. Joshua, and Qian, He. *Fully hardware-implemented memristor convolutional neural network*. Nature, vol. 577, no. 7792, pp. 641–646, 2020. Nature Publishing Group UK London.

[38] H. Zhao, L. Jiang, J. Jia, P. H. S. Torr, and V. Koltun, "Point Transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16259–16268, 2021.

[39] Chen, Can, Luca Zanotti Fragonara, and Antonios Tsourdos. "Go Wider: An Efficient Neural Network for Point Cloud Analysis via Group Convolutions." *Applied Sciences*, vol. 10, no. 7, 2020, p. 2391. MDPI.

[40] R. Khaddam-Aljameh, M. Stanisavljevic, J. F. Mas, G. Karunaratne, M. Brändli, F. Liu, A. Singh, S. M. Müller, U. Egger, A. Petropoulos, and others, "HERMES-core—A 1.59-TOPS/mm² PCM on 14-nm CMOS in-memory compute core using 300-ps/LSB linearized CCO-based ADCs," *IEEE J. Solid-State Circuits*, vol. 57, no. 4, pp. 1027–1038, 2022.

[41] M. Le Gallo, R. Khaddam-Aljameh, M. Stanisavljevic, A. Vasilopoulos, B. Kersting, M. Dazzi, G. Karunaratne, M. Brändli, A. Singh, S. M. Mueller, and others, "A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference," *Nature Electronics*, vol. 6, no. 9, pp. 680–693, 2023.

[42] Aguirre, Fernando, Sebastian, Abu, Le Gallo, Manuel, Song, Wenhao, Wang, Tong, Yang, J. Joshua, Lu, Wei, Chang, Meng-Fan, Ielmini, Daniele, Yang, Yuchao, et al. *Hardware implementation of memristor-based artificial neural networks*. Nature Communications, vol. 15, no. 1, p. 1974, 2024.