Neuromorphic Computing and Engineering

PAPER • **OPEN ACCESS**

D-SELD: Dataset-Scalable Exemplar LCA-Decoder

To cite this article: Sanaz Mahmoodi Takaghaj and Jack Sampson 2024 *Neuromorph. Comput. Eng.* **4** 044009

NEUROMORPHIC
Computing and Engineering

OPEN ACCESS

RECEIVED
7 July 2024

REVISED
24 November 2024

ACCEPTED FOR PUBLICATION
12 December 2024

PUBLISHED
30 December 2024

PAPER

D-SELD: Dataset-Scalable Exemplar LCA-Decoder

Sanaz Mahmoodi Takaghaj[1,2,*] **and Jack Sampson**[1]

[1] Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802, United States of America
[2] School of Engineering Design and Innovation, Pennsylvania State University, University Park, PA 16802, United States of America
[*] Author to whom any correspondence should be addressed.

E-mail: sxm788@psu.edu

Keywords: encoder-decoder, LCA, exemplar learning, spiking neural networks, sparse coding, neuromorphic computing

Abstract

Neuromorphic computing has recently gained significant attention as a promising approach for developing energy-efficient, massively parallel computing systems inspired by the spiking behavior of the human brain and natively mapping spiking neural networks (SNNs). Effective training algorithms for SNNs are imperative for increased adoption of neuromorphic platforms; however, SNN training continues to lag behind advances in other classes of ANN. In this paper, we reduce this gap by proposing an innovative encoder–decoder technique that leverages sparse coding and the locally competitive algorithm (LCA) to provide an algorithm specifically designed for neuromorphic platforms. Using our proposed Dataset-Scalable Exemplar LCA-Decoder we reduce the computational demands and memory requirements associated with training SNNs using error backpropagation methods on increasingly larger training sets. We offer a solution that can be scalably applied to datasets of any size. Our results show the highest reported top-1 test accuracy using SNNs on the ImageNet and CIFAR100 datasets, surpassing previous benchmarks. Specifically, we achieved a record top-1 accuracy of 80.75% on ImageNet (ILSVRC2012 validation set) and 79.32% on CIFAR100 using SNNs.

1. Introduction

Neuromorphic computing, inspired by the capabilities of the human brain, aims to revolutionize the field of computing by offering energy-efficient and massively parallel processing elements. This paradigm holds great promise in addressing the limitations of conventional computing systems, such as data movement, power consumption, and real-time processing of complex data, especially for artificial intelligence applications. Conventional processors, like CPUs and GPU/TPUs, excel at efficiently handling dense, structured data structures by maximizing the utilization of instruction/data pipelines. However, their efficiency diminishes when dealing with sparse, asynchronous event streams [1]. A variety of neuromorphic chips, including TrueNorth [2, 3], Loihi [4, 5], SpiNNaker [6–8], BrainScaleS [9, 10], Tianjic [11] DYNAP-SE [12], and in-memory compute (IMC) cores [13, 14], have been developed to offer low-latency and low-power consumption, suited to model spiking neural networks (SNNs) and handle sparse data stream from event-based sensors. Neuromorphic chips are characterized as interconnected many-core systems capable of instantiating vast populations of spiking neurons through highly parallelized and energy-efficient hardware architectures. They have the capability to perform inference based on pre-trained neural networks and process data in a brain-inspired manner—marked by energy-efficient operations conducted in parallel, and asynchronous communication. Another source of efficiency on these platforms stem from the proximity of synapses to where their weights are calculated and updated, reducing data movement. Neuromorphic chips support neurons that generate binary output values (spikes) and enable fast evaluation of the matrix vector multiplication (MVM) operations thorough the use of crossbar arrays and memristors [15, 16] which allow reliable long-term storage of multi-bit quantities as the conductance values and dot-product operations as an analog primitive via Kirchoff's law. MVM operations are fundamental to deep learning algorithms, playing a crucial role in their efficient training and deployment.

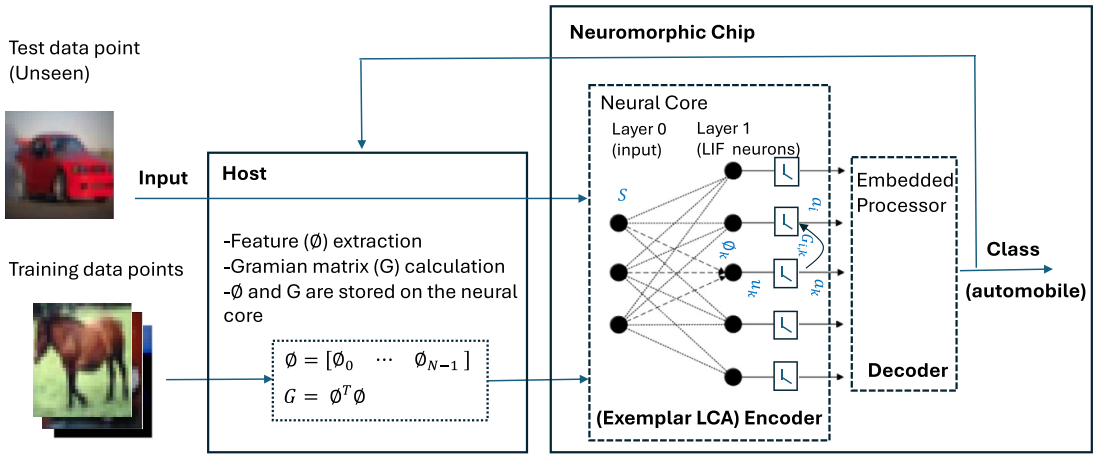


Figure 1. The D-SELD encoder–decoder architecture for image classification, illustrating the structure and flow of information between the encoder and decoder components.

Most recent strategies for training SNNs prominently leverage gradient calculations and error backpropagation [17–23]. Error backpropagation [24] is a widely-used training algorithm in deep learning that minimizes a loss function by adjusting the weights of a neural network through successive iterations, allowing complex neural networks to learn hierarchical representations and approximate intricate functions. Implementing error backpropagation for training SNNs on neuromorphic systems is highly resource-intensive, requiring significant computational power and memory usage across multiple time steps per neuron. On the other hand, training deep neural networks offline using GPU/TPU acceleration and converting them to SNNs could overlook the intrinsic neural dynamics and temporal coding inherent to SNNs, resulting in reduced accuracy and increased latency [25–30], necessitating a re-evaluation of SNN training strategies and algorithmic optimizations to fully harness the advantages of these sparse architectures.

Sparse coding is believed to be a key mechanism by which biological neural systems can efficiently process large amounts of complex sensory data [31–33]. It plays a significant rule in information processing within the primary visual cortex (V1). The neurons in V1 form an over-complete dictionary of signal elements such as texture, orientation, scale, etc and the spikes emitted in response to a given input stimulus are highly sparse [32]. A biologically plausible implementation of sparse coding is achieved through the use of leaky integrator-and-fire (LIF) neurons in locally competitive algorithms (LCA) [34]. LCA is a computational model and learning algorithm that recursively updates the activity of neurons to find a sparse representation of the input data. The competitive mechanism within LCA ensures that only a limited number of neurons become active at any given time, enabling sparsity and efficient coding of high-dimensional data. Some recent neuromorphic chips like 'Loihi' by Intel and 'TrueNorth' by IBM have implemented LCA for sparse coding and image reconstruction tasks. While LCA has been studied as a neuromorphic-implementation-friendly algorithm [2–4, 35–37], it has not been studied as a direct SNN classifier.

In this paper, we present the 'Dataset-Scalable Exemplar LCA-Decoder' (***D-SELD***), a novel approach to LCA that approximates the input signal by expressing it as a linear combination of dictionary atoms representing features learned from a training dataset. We then utilize this sparsely encoded set of codes in the 'Decoder' for classification tasks. The features ϕ_k are extracted once and stored on neuromorphic platforms like Loihi, as shown in figure 1, which prioritize specialized operations and energy-efficient computation. This contrasts with GPUs or TPUs, which are optimized for high-throughput tasks. An alternative implementation of this architecture could leverage various specialized hardware solutions, such as FPGAs, memory crossbar arrays [15, 38], or IMC cores [13, 14], to efficiently execute the operations outlined in equations (3), (4) and (6). The decoder executes equation (9) or (10).

D-SELD is derived from fundamental principles, has high convergence rate and reduces the need for extensive gradient computations and memory utilization during training. Additionally, the sparse representation of stimuli minimizes the number of firing neurons needed to represent and encode a given input. This introduces sparsity to the network and reduces the computational and memory demands typically associated with training SNNs.

The D-SELD framework we present showcases promising classification results, excelling in both input reconstruction and the accuracy of classification tasks across well-established benchmark datasets. In summary, the contributions of this paper include:

- We introduce a dataset-size scalable training methodology for neuromorphic platforms, which mitigates the computational and memory demands inherent in training SNNs using error backpropagation.
- We achieved the highest reported top-1 accuracy of 80.75% on the ImageNet ILSVRC2012 validation dataset and 79.32% on CIFAR-100 using SNNs.
- Our training framework can accommodate the introduction of new training examples in time proportional to the number of new examples, not to the new resulting dataset size, easing post-deployment or task-adjustment of the network for datasets of any size.

2. Related work

As the field of neuromorphic computing continues to advance, novel training methods are being investigated to harness the potential of SNNs in various neuromorphic platforms [3, 4, 6, 8, 11, 13, 14]. Here, we will review several key approaches used in training SNNs that are scalable and applicable to larger datasets or have been successfully deployed on neuromorphic hardware, followed by an exploration of works on LCA and its deployment on neuromorphic platforms.

• **Transfer learning and ANN-to-SNN conversion:** transfer learning is a technique in machine learning that leverages knowledge from one domain to assist in learning another but related task [39–46]. Initially, a model is trained on a large dataset such as ImageNet [47] to extract general features and patterns relevant to its primary task. Subsequently, the pretrained model undergoes fine-tuning on a smaller dataset specific to the new task. For example, Sharma *et al* [44] reported that ResNet-50 achieved top-1 accuracy rates of 78.10% on CIFAR-10 and 59.82% on CIFAR-100. More recently, Bello *et al* [39] achieved a top-1 accuracy of 88.1% by fine-tuning a ResNet-152 model on the CIFAR-100 dataset. Transfer learning has also been applied to on-chip learning on edge devices with stringent energy and memory limitations [48–50]. For instance, [49] achieved an accuracy of 93.3% on CIFAR-10 [51] and 81.2% on CIFAR-100 using MobileNet [52]. Zhu *et al* [50] employed sparse backpropagation to fine-tune a ResNet-50 model on the CIFAR-10 dataset, achieving an accuracy of 96.2%.

In the context of SNNs, transfer learning typically involves training an ANN with ReLU neurons using supervised learning algorithms. Afterwards, an SNN with spiking neurons and a similar architecture to the source ANN model is created, initializing the SNN with weights learned from the pretrained ANN model [25, 26, 28, 29, 53]. This approach does not explicitly consider neuron dynamics during training, leading to reduced accuracy and increased latency compared to the original ANN. Moreover, fine-tuning requires a significant memory footprint during error backpropagation, making it impractical for the training computation to be performed on neuromorphic devices.

Incorporating learned feature maps into D-SELD introduces a novel transfer learning approach, paving the way for deploying larger models on neuromorphic platforms while maintaining high accuracy. In this work, we extract features from the penultimate layer of ubiquitous convolutional neural network (CNN) models to construct the LCA dictionary. However, this framework is not limited to CNNs; it can also utilize features extracted from other architectures, such as PointNets [54] or Vision Transformers (ViT) [55].

• **Direct SNN training:** several methods perform credit assignment in SNNs using error backpropagation [17, 20, 27, 30, 56–61]. Directly implementing backpropagation in SNNs is however challenging due to the discontinuous and non-differentiable nature of spiking neurons. To address this issue, surrogate gradients [27, 61, 62] are used to approximate the discontinuous derivative as a continuous function, enabling end-to-end backpropagation training of SNNs. The utilization of error backpropagation still demands massive computations and memory, making the training process resource-intensive. Despite its success in training SNNs, the implementation of error backpropagation for on-chip training has been delayed due to its substantial demands on computational and memory resources.

D-SELD eliminates the need for any gradient computations associated with training. Additionally, the sparse encoding of stimuli with LCA, minimizes the number of firing neurons needed to represent a given input. This introduces sparsity to the network which further reduces the computational demands.

•**Sparse coding and LCA:** sparse coding and applications of LCA have been extensively studied in the past [36, 63–66]. Several approaches such as pruning [67–69], regularization [70, 71], dropout [72], and probabilistic quantization [73] have been employed in ANNs to promote sparsity. While LCA shares similarities with regularization techniques through its utilization of ℓ_1-minimization, its distinct incorporation of spiking behavior sets it apart, making it well-suited for neuromorphic deployment. Recent implementations of sparse coding and LCA on neuromorphic platforms such as Loihi [4] and TrueNorth [2, 3] and others [35] serve to further validate this advantage.

•**Neuromorphic hardware deployment:** Cramer *et al* [74] successfully implemented SNN training using in-the-loop surrogate gradients on the analog neuromorphic platform BrainScaleS-2. Similiarly, Renner *et al* [75] demonstrated the successful implementation of SNN training through error backpropagation on the digital neuromorphic processor Loihi.

Fair *et al* [76] utilized the original LCA for sparse coding on IBM's TrueNorth neuromorphic chip, showing comparable accuracy to conventional CPUs, which highlights LCA's potential for energy-efficient sparse representation on neuromorphic platforms. Additionally, LCA has been deployed on the Loihi platform for image reconstruction and to solve LASSO optimization problems, achieving over three orders of magnitude improvement in energy-delay product compared to conventional CPU-based solvers [4]. Parpart *et al* [37] provide a comprehensive implementation and benchmarking of LCA on Loihi2 for image reconstruction.

3. D-SELD design

In this section, we begin by providing an overview of sparse coding and the LCA algorithm. Then, we introduce the D-SELD framework and derive the equations that govern its operations.

3.1. Sparse coding and LCA

In the sparse coding algorithm [34], the input signal S is represented as:

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

where ϕ is a dictionary of features ϕ_i and a is a vector of activation coefficients a_i. The term ε represents Gaussian noise. The dictionary ϕ is over-complete, meaning that the number of columns in ϕ (M) is greater than the number of rows N, and as a result, it is non-orthogonal. This non-orthogonality leads to an infinite number of possible combinations, and each combination is an equally valid representation of signal S. Prior work has demonstrated that increasing over-completeness leads to higher spatial frequency features and a more even distribution of orientations on natural images [77].

One implementation of sparse coding that minimizes the energy function (equation (2)) is through the LCA. In LCA, activation coefficients a_i correspond to the outputs of LIF neurons, defined by equations (3)–(6). The primary objective of LCA is to minimize reconstruction error while maintaining an acceptable level of network sparsity. The reconstructed input $\hat{S}$ is expressed as $\hat{S} = \sum_{i=0}^{M-1} \phi_i a_i$, where ϕ_i are basis functions or dictionary atoms and, a_i represents the activation of the LIF neuron

$$E = \underbrace{\frac{1}{2}\left\|S - \widehat{S}\right\|_2^2}_{\text{Reconstruction}} + \underbrace{\lambda \sum_{i=0}^{M-1} |a_i|}_{\text{Sparsity}}. \tag{2}$$

The LIF neuron's membrane potential u_i, at each time step k, is subject to a driving excitatory input b_i and an inhibition matrix (Gramian) G:

$$\tau \dot{u}_i[k] + u_i[k] = b_i - \sum_{m \neq i}^{M-1} G_{i,m} a_m[k] \tag{3}$$

$$b_i = S\phi_i \tag{4}$$

$$G = \phi^{\mathrm{T}} \phi. \tag{5}$$

The Gramian matrix allows stronger neurons to prevent weaker neurons from becoming active, resulting in a sparse representation. Specifically, the inhibition signal from the active neuron m to any other neuron i is proportional to the activity level a_m of neuron m and to the inner product between the two neurons' receptive fields (proportional dictionary elements). And, finally, the thresholding function is:

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

Here, the threshold 'λ' refers to the level that the membrane potential must exceed for the neuron to become active. This threshold is the same sparsity penalty trade-off utilized in the energy function (equation (2)). After obtaining the neuron activations for an initial dictionary ϕ, these activations are utilized to update the dictionary using equation (7) where η is the dictionary learning rate:

$$\phi_i = \phi_i + \eta\left(S - \hat{S}\right) a_i. \tag{7}$$

3.2. D-SELD framework

To leverage the efficiency of LCA and the sparsity of spiking neurons for future deployment on neuromorphic hardware platforms, we adopt a novel strategy: constructing the dictionary ϕ directly from a collection of features extracted from the training dataset, thus bypassing the costly process of dictionary learning. In other words, each ϕ_i will represent a feature learned from a data point x_i in the training dataset. This can be expressed as:

$$\phi = [\phi_0, \phi_1, \ldots, \phi_{M-1}] \tag{8}$$

where $\phi_i = f(x_i)$, and $f(\cdot)$ denotes the function that extracts features from x_i. This eliminates the need for image reconstruction $\hat{S}$ and dictionary updates in equation (7), differing from original LCA, where stochastic gradient descent could be used to learn and update the dictionary for each batch of input data.

Next, we will map any unseen test data S_{test} to the resulting M-dimensional space of feature vectors (dictionary atoms ϕ_i) and find activation codes a_i that approximate the new input using equation (6).

3.2.1. Feature extraction

While constructing the dictionary ϕ directly from the training dataset suffices for image reconstruction tasks and classification of simpler datasets like MNIST [78], achieving higher classification accuracy on more complex datasets requires the use of more complex features from the training dataset. Prior works have demonstrated that features extracted from convolutional layers in CNNs exhibit rich characteristics [40, 41, 43]. Additionally, while these models are typically trained on large-scale image datasets such as ImageNet [47], the extracted features have proven effective across various other datasets. We investigated several off-the-shelf CNN models such as MobileNet [52], DenseNet [79], ResNet [80] and EfficientNet [81], renowned for their demonstrated efficacy on ImageNet dataset.

Figure 2(a) shows the trade-off between top-1 accuracy on ImageNet dataset and the reported computational workload across these models [82]. GFLOPS are estimates of the number of (Giga) floating point operations performed during a single full pass of a model. Among these models, EfficientNet achieves the highest accuracy, while MobileNet has the lowest number of GFLOPS. Despite variations in architecture, accuracy, and workload, the features extracted from these models can be uniformly utilized in D-SELD. The primary distinction lies in the sizes of their feature maps. Since the features are extracted once prior to deployment on the neuromorphic hardware, the size of feature maps provides a more accurate indication of the workload for D-SELD. Figure 2(b) illustrates the trade-off between workload and accuracy for D-SELD, using feature map sizes as a metric. Features extracted from the ResNet-152 architecture, with a feature size of 2048, achieved the highest accuracy on CIFAR-10 [51] and CIFAR-100 datasets, followed by features from EfficientNet with a feature size of 1280. With a feature map size of 512, VGG-16 achieved the lowest accuracy. Since all models utilized the same decoder, the superior performance of ResNet-152 in transfer learning could be attributed to its ability to extract more discriminative features.

3.2.2. Design considerations for decoding algorithms

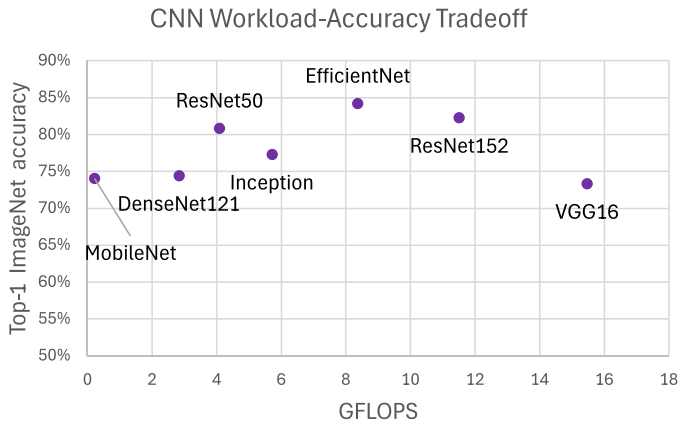
In this section, we develop a decoder aimed at decoding sparse codes a_i generated by Exemplar LCA and map them to K distinct object classes from the training dataset. Later, we will employ the same decoder to accurately predict the class of unseen test data points.

With an adequate number of training data points M, any data point will have a sparse coding representation as $a = [a_0, a_1, a_2, \ldots, a_{M-1}]$. In an ideal scenario, for any input S, the non-zero entries in vector a will exclusively correspond to the dictionary atoms ϕ_i belonging to a singular object class k where k ranges from r1 to K. However, the presence of modeling errors and noise may result in small non-zero entries that are associated with multiple object classes. We will design decoders that can resolve this issue:

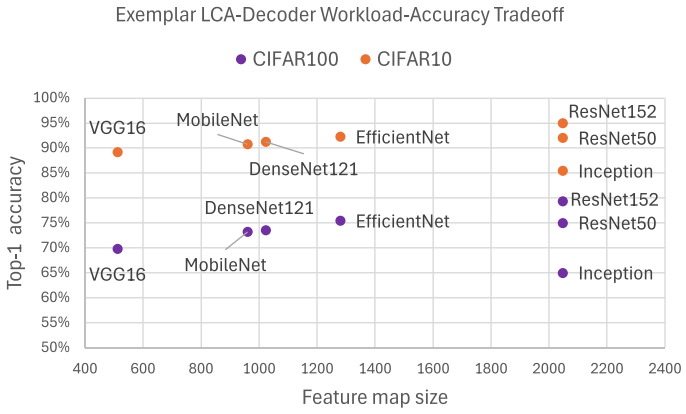
3.2.2.1. I. Maximum activation code

For a given unseen test input $S_{\text{Test}} = \sum_{i=0}^{M-1} \phi_i a_i + \epsilon$, we calculate sparse activation codes a_i and determine the class C_{Test} as the class associated with the maximum value in the sparse coding set of S_{Test}:

$$C_{\text{Test}} = \text{Class}\left[\underset{i}{\text{argmax}}\,[a_i]\right]. \tag{9}$$



(a) The numerical values are sourced from reference [83]



(b)

Figure 2. Trade-off between accuracy and workload across CNN models used as feature extractors in D-SELD.

3.2.2.2. II. Maximum sum of activation codes

To better harness the relevance of neuron activations within each class, we sum the ℓ_1 norms of the activations for each object class c, and assign $S_{\text{Test}} = \sum_{i=0}^{M-1} \phi_i a_i + \epsilon$ to the class with the highest value:

$$C_{\text{Test}} = \text{Class}\left[\underset{c}{\text{argmax}}\left(\sum_i \left|a_i^{(c)}\right|\right)\right]. \tag{10}$$

This approach yields significantly improved results compared to the maximum activation decoder. However, to fully leverage the discriminative coding among various object classes, we also explored the application of a shallow neural network.

3.2.2.3. III. Shallow neural network

Here, we employ a shallow network (single-layer perceptron) to map sparse activation codes a_i to K class labels. Sparse activation codes, which represent the presence or absence of specific features or patterns, serve as inputs to the network. Through the process of training, the network learns to associate these sparse activations with corresponding class labels. By adjusting the weights within the network, it can discern patterns in the sparse activation space and more accurately predict the class labels of unseen test data. This approach incurs additional memory and computational costs compared to the previous two decoders, and should only be used if previous decoders fail. When extracted features capture rich content from the training dataset, the performance of maximum sum of activation codes closely approaches that of a shallow NN decoder (less than a 1% difference). Therefore, we include the results of this decoder in table 3 for review and omit them from the benchmarking table and efficiency discussions.

Figure 3. The reconstructed images of an automobile from the CIFAR-10 dataset. The 'Dataset' dictionary performs better in reconstructing the input image. Input image is sourced from unseen test dataset.

Table 1. Reconstructed image quality metrics.

	Random dictionary	Dataset dictionary
Mean squared error (MSE)	675.3	92.45
Peak signal-to-noise ratio (PSNR)	19.84 dB	28.47 dB
Structural similarity index (SSIM)	0.75	0.99
Number of iterations	40	6

4. Experimental results

In this section, we present the results obtained by applying our LCA-Decoder approach to image classification tasks on MNIST [78], CIFAR-10 [51], CIFAR-100 [51] and ImageNet [47] datasets. MNIST consists of a training dataset of 60 K images and a test dataset of 10 K images, each with images sized 28×28 pixels in grayscale. The CIFAR-10 and CIFAR-100 datasets consist of 50 K training images and 10 K test images, each with dimensions of 32×32 pixels in RGB format. The ImageNet dataset includes 1.28 million training images and 50 K validation images. We randomly split the training dataset and selected 50 K samples to construct the dictionary. We exclusively used the validation dataset for testing purposes, ensuring it was not used prior to testing. All datasets were obtained using Torchvision [82].

4.1. Image reconstruction

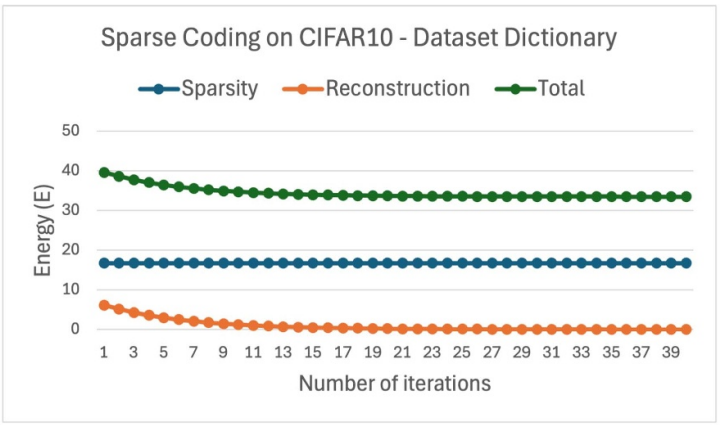
We initially conducted experiments to assess the effectiveness of exemplar encoding in image reconstruction as a proof of concept. We used unseen images from the MNIST and CIFAR-10 datasets. First, we utilized a random dictionary, where the dictionary atoms ϕ_i were initialized using random values drawn from a Gaussian distribution $N(0, 1)$. This choice aligns with the common practice of initializing weights to random values in ANNs. Additionally, as discussed in section 3.2, in the 'Dataset Dictionary', we directly selected dictionary atoms from the training dataset. The results depicted in figure 3 demonstrate that using the 'Dataset' dictionary surpasses the performance of a 'Random' dictionary by achieving higher reconstruction quality and faster convergence (see table 1). Figure 4 illustrates all components of the energy function E during image reconstructions and confirms the convergence of LCA in both cases.

4.2. Dictionary size and over-completeness

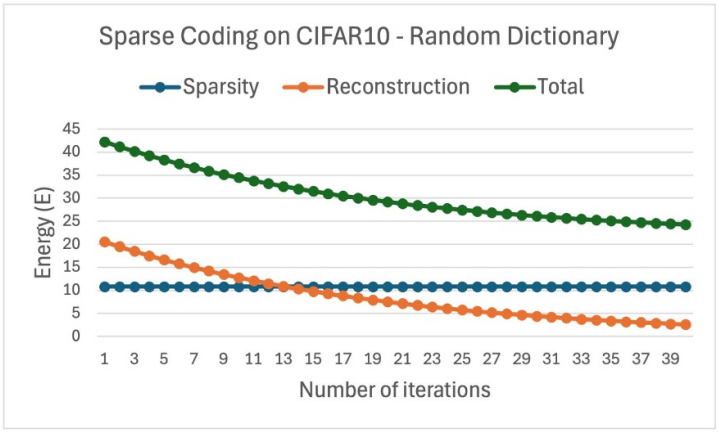
A dictionary ϕ is considered 'over-complete' when the number of dictionary atoms (neurons) is greater than the number of input dimensions (image pixels). Figure 5 shows that in classification tasks, increasing over-completeness is associated with enhanced top-1 accuracy, up to an upper bound defined by the training dataset size. As the size of the dictionary can be adjusted based on the size of the training dataset, this scalability enables the an LCA-Decoder approach to uniformly apply to datasets of any size. Moreover, new training data points or classes can be incrementally added to the dictionary online.

4.3. Classification results

Figure 6 shows the top-1 accuracy performance. All experiments were consistently conducted using the hyperparameters specified in table 2. In all cases, using the maximum sum of activation codes decoder resulted in higher accuracy compared to using the maximum activation codes decoder. Among the tested



(a) Dataset Dictionary



(b) Random Dictionary

Figure 4. Energy function E during image reconstructions (a and b).

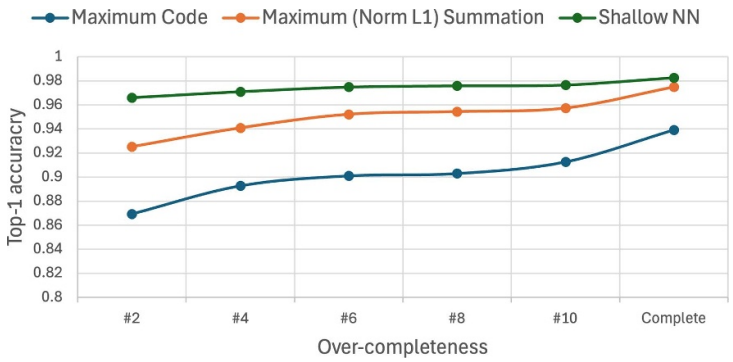
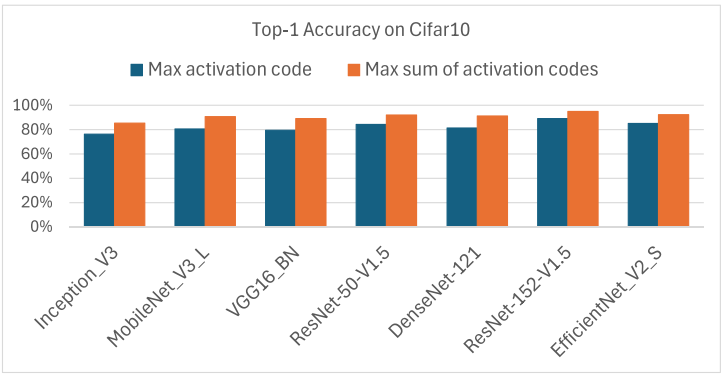


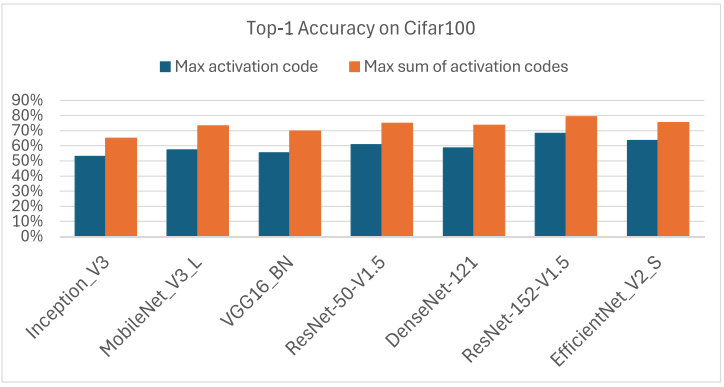
Figure 5. Top-1 Accuracy comparison across three decoders with increasing dictionary over-completeness on MNIST dataset (image size: $28 * 28$). Here, #2 represents dictionary size of $28 * 28 * 2$ and 'Complete' denotes the utilization of all training data points.

CNN feature extractors, ResNet-152-V2 [83] consistently achieved the highest accuracy. Further hyperparameter optimization and fine-tuning threshold and/or leakage are required to achieve accuracy comparable to or higher than that reported at $K = 100$. For instance, setting a threshold of 0.3 and τ to 100 achieves 74.26% accuracy on CIFAR100 using ResNet152.

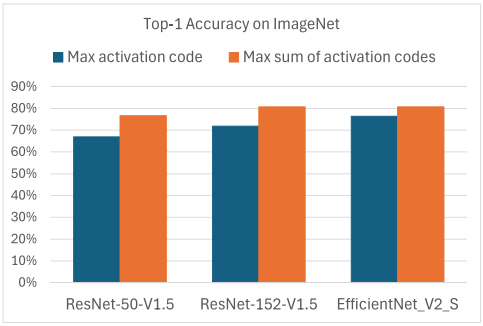
Table 3 shows the highest top-1 accuracy scores achieved for tested datasets using all decoders described in section 3.2.2. For MNIST, data points were used directly without applying any feature extractor. For the CIFAR-10, CIFAR-100 and ImageNet datasets, we achieved the highest accuracy using features extracted through ResNet-152-V2 architecture. And among the tested decoders, shallow NN decoder achieved slightly



(a) CIFAR-10



(b) CIFAR-100



(c) ImageNet

Figure 6. Comparison of top-1 accuracy across three datasets using maximum activation code (blue) and maximum sum of activation codes (orange), with features extracted from various CNN models.

Table 2. Hyperparameters.

Symbols	Description	Value
λ	Threshold	2
M	Dictionary size	50 K
τ	Leakage	100
k	Number of time steps	100

higher accuracy on MNIST, CIFAR-10 and CIFAR-100 datasets, while the maximum sum of sparse codes for each class resulted in the highest accuracy for ImageNet.

4.4. Sparsity

Sparsity in D-SELD arises due to firing neurons transmitting inhibition signals to non-firing neurons, impeding their ability to reach the activation threshold. To assess D-SELD's sparsity, we examine the proportion of neurons within the encoding layer that fire during the processing of input data. Figure 7 illustrates the average number of firing neurons upon applying the Exemplar LCA-Decoder on CIFAR-100

Table 3. Comparison of Top-1 accuracy scores. Bold numbers indicate the maximum accuracy achieved for each dataset.

			Decoding Methods		
			D-SELD		D-SELD+BP
	Feature extractor	Feature map size	$\max_i \{a_i\}$	$\max_k \sum_i \left\lvert a_i^{(k)} \right\rvert$	Shallow NN
MNIST	none	$28 * 28 * 2$	93.93%	97.48%	**98.25%**
CIFAR-10	ResNet-152-V2	2048	89.16%	94.98%	**95.19%**
CIFAR-100	ResNet-152-V2	2048	68.29%	79.32%	**79.33%**
ImageNet	ResNet-152-V2	2048	71.83%	**80.75%**	21.60%
ImageNet	EfficientNet-V2-S	1280	76.40%	**80.68%**	21.49%

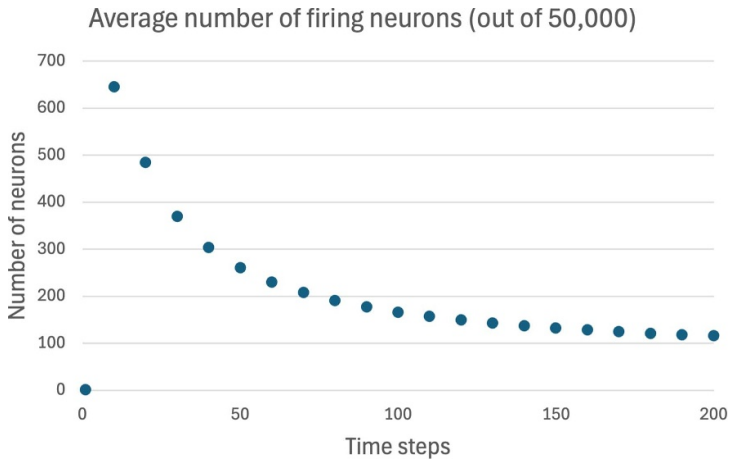


Figure 7. Average firing neurons count for Dataset-Scalable Exemplar LCA-Decoder on CIFAR-100.

with 10 000 test samples. We noted that only a minuscule (less than 1% with an average value of 0.4%) proportion of the 50 000 neurons were found to fire at each time step. Since each firing neuron only fires one spike at each time step, this plot also shows the number of spikes occurring in each time step. While this number reaches a maximum at early time steps, it declines exponentially over subsequent time steps.

4.5. Workload efficiency analysis

Here, we use floating-point operations (FLOPs) as a metric to analyze computational efficiency. Since D-SELD eliminates the need for dictionary learning, the calculations reduce to equations (3), (6) and (10). Computing b_i requires $N * M$ multiplications and $(N-1) * M$ addition, where N is the feature map size and M is the length of the dictionary (also the number of neurons). The inhibition signal involves $M^2 - M$ multiplications and $M^2 - 2M$ additions. The leakage term adds M multiplication operations, and $3M$ additions/subtractions combine these terms and update the neurons' membrane potentials. Additionally, the Gramian matrix G requires $\frac{M(M+1)N}{2}$ multiplications and $\frac{M(M+1)(N-1)}{2}$ additions. This is considered the training cost, computed once per task (dataset), and excluded from the inference cost. As a result, the total floating point operations for inference over a total time step K is:

$$\text{FLOPs} = K\left(\frac{(2N-1)M}{K} + 2M^2 + M\right). \tag{11}$$

Note that b_i is computed once per input image and remains constant across iterations. Next we factor in the observed sparsity from section 4.4 by redefining M associated with firing neurons as $\hat{M}$

$$\text{FLOPs} = K\left(\frac{(2N-1)M}{K} + 2M\hat{M} + M\right). \tag{12}$$

Table 4 displays the estimated FLOPs for the exemplary LCA-Decoder. According to the findings in section 4.4, we assumed that, on average, 0.4% of neurons ($\hat{M}$) spike at each time step for each model. 'Training (TFLOPs)' denotes the TeraFLOPs required to compute the Gramian matrix, while 'Inference (GFLOPs)' represents the GigaFLOPs for inference operations. We conducted two scenarios with 100 and r10 time steps K. Reducing the number of time steps from 100 to r10 resulted in an average reduction of 84% in

Table 4. Estimated workload analysis of D-SELD. Bold numbers indicate the lowest FLOPs.

Feature Extractor	Training (TFLOPs)	K	N	M	$\hat{M}$	Inference (GFLOPs)
Inception	5.12	100	2048	50 K	200	2.21
		10	2048	50 K	200	0.41
ResNet-50 &	5.12	100	2048	50 K	200	2.21
ResNet-152		10	2048	50 K	200	0.41
EfficientNet	3.2	100	1280	50 K	200	2.13
		10	1280	50 K	200	0.33
DenseNet	2.56	100	1024	50 K	200	2.11
		10	1024	50 K	200	0.3
MobileNet	2.4	100	960	50 K	200	2.1
		10	960	50 K	200	0.29
VGG-16	**1.28**	100	512	50 K	200	2.06
		10	512	50 K	200	**0.25**

Table 5. D-SELD Execution Time: LCA encoding stands to benefit from deployment on neuromorphic chips.

Processing device	Batch size	Feature extraction time (ms)	LCA encoding time (ms)	Decoding time (ms)	Total time (ms)
NVIDIA A100	64	0.9	2.52	0.1	3.52
Apple M1 Pro	1	138	4689.7	2	4829.7

Inference GFLOPs. VGG-16 has the lowest computation workload among the tested models, as anticipated from figure 2(b).

We conducted two experiments on the NVIDIA A100 and Apple MacBook, averaging execution times over r5 runs per experiment. Table 5 displays the measured execution times for individual components of D-SELD. In this experiment, features were extracted from the ResNet-152 architecture and used for classifying the CIFAR-100 dataset. We executed batches of 64 on the A100 GPU; however, the reported numbers correspond to a single test data. As evident, the computational bottleneck resides in the LCA encoding phase, which stands to gain significant advantages from hardware specialization in neuromorphic chips.

4.6. Comparison to state-of-the-art

In comparing our results to the state-of-the-art, several critical considerations were addressed to ensure a fair assesment. Firstly, we examined SNNs with neurons that exhibit a (leaky) integrate-and-fire mechanism, considering their suitability for deployment on neuromorphic platforms.

Secondly, we propose a solution that holds a distinct advantage: it requires no explicit training when utilizing 'maximum activation code' and 'maximum sum of activation codes' decoders. Deploying D-SELD on Neuromorphic chips entails programming synaptic weights on neural cores using values derived from the dictionary. The only requirement is the construction of the dictionary using extracted feature maps, a process that can be executed off-chip.

Thirdly, it is important to note that the models we utilized for feature extraction were the original models trained on ImageNet, and we have specified the exact versions of these models used. To maintain the simplicity of D-SELD, we opted not to fine-tune these architectures for specific targeted datasets. The only pre-processing steps applied to the data included resizing, cropping, rescaling, and normalizing. However, should new or improved models become available in the future, their extracted features can still be integrated into D-SELD to potentially enhance its performance further.

Lastly, in our benchmarking (see table 6), we specifically selected studies that used SNNs and achieved the highest reported accuracy on all the datasets we tested. Our results indicate that we achieved the highest reported top-1 accuracy of 80.75% using SNNs on ImageNet and 79.32% on CIFAR100. For CIFAR10, we attained the second-highest reported accuracy of 94.98%, with the added benefit of not employing backpropagation and enabling direct deployment on neuromorphic platforms.

Our benchmarking analysis included results from deploying a CNN model on IBM TrueNorth for CIFAR-10 and CIFAR-100 datasets [84]. However, that study did not include any test results on ImageNet, so direct comparison on that dataset is not performed.

Table 6. Benchmarking results. Bold numbers indicate the maximum accuracy achieved for each dataset.

Dataset	Method	Architecture	Accuracy
CIFAR-10	**IM-Loss** [57]	ResNet-19	**95.49%**
	Dspike [20]	ResNet-18	94.25%
	OTTT [59]	VGG	93.73%
	Inference on TrueNorth [84]	CNN	89.32%
	D-SELD	D-SELD (ResNet-50-V2)	92.00%
		D-SELD (Resnet-152-V2)	94.98%
CIFAR-100	Dspike [20]	ResNet-18	74.24%
	OTTT [59]	VGG	71.05%
	IM-Loss [57]	VGG-16	70.18%
	Inference on TrueNorth [84]	CNN	65.48%
	D-SELD	D-SELD (ResNet-50-V2)	74.98%
		D-SELD (EfficientNet-V2-S)	75.41%
		D-SELD (ResNet-152-V2)	**79.32%**
ImageNet	RMP-SNN [26]	VGG-16	73.09%
	S-ResNet [19]	ResNet-50	72.75%
	Dspike [20]	VGG-16	71.24%
	D-SELD	D-SELD (ResNet-50-V2)	76.64%
		D-SELD (EfficientNet-V2-S)	80.68%
		D-SELD (ResNet-152-V2)	**80.75%**

5. Conclusion

In this study, we introduce the D-SELD approach designed for effective deployment of SNNs on neuromorphic platforms. Our method achieves the highest test accuracy on ImageNet and CIFAR100 datasets, surpassing previous benchmarks for SNN performance on these datasets, while reducing computational demands associated with conventional error backpropagation methods for SNN training. Moreover, the D-SELD approach offers model scalability, enabling network adjustment to various dataset sizes and incorporation of new data points. This scalability is particularly advantageous in dynamic environments where continuous updates to the training set are necessary without incurring prohibitive training costs. Finally, computational analysis using FLOPs characterizes the training/inference costs associated with our method, demonstrating compelling trade-offs between workload, latency, and performance compared to alternative SNN approaches.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

ORCID iD

Sanaz Mahmoodi Takaghaj https://orcid.org/0000-0002-9523-690X

References

[1] Dave S, Baghdadi R, Nowatzki T, Avancha S, Shrivastava A and Li B 2021 Hardware acceleration of sparse and irregular tensor computations of ML models: a survey and insights *Proc. IEEE* **109** 1706–52
[2] Merolla P A *et al* 2014 A million spiking-neuron integrated circuit with a scalable communication network and interface *Science* **345** 668–73
[3] Amir A *et al* 2017 A low power, fully event-based gesture recognition system *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 7243–52
[4] Davies M *et al* 2018 Loihi: a neuromorphic manycore processor with on-chip learning *IEEE Micro* **38** 82–99
[5] Davies M, Wild A, Orchard G, Sandamirskaya Y, Guerra G A F, Joshi P, Plank P and Risbud S R 2021 Advancing neuromorphic computing with loihi: a survey of results and outlook *Proc. IEEE* **109** 911–34
[6] Furber S B, Galluppi F, Temple S and Plana L A 2014 The spinnaker project *Proc. IEEE* **102** 652–65
[7] Furber S 2016 Large-scale neuromorphic computing systems *J. Neural Eng.* **13** 051001
[8] Höppner S *et al* 2021 The spinnaker 2 processing element architecture for hybrid digital neuromorphic computing (arXiv:2103.08392)
[9] Schmitt S *et al* 2017 Neuromorphic hardware in the loop: training a deep spiking network on the brainscales wafer-scale system *2017 Int. Joint Conf. on Neural Networks (IJCNN)* (IEEE) pp 2227–34
[10] Pehle C, Billaudelle S, Cramer B, Kaiser J, Schreiber K, Stradmann Y, Weis J, Leibfried A, Müller E and Schemmel J 2022 The brainscales-2 accelerated neuromorphic system with hybrid plasticity *Front. Neurosci.* **16** 795876
[11] Pei J *et al* 2019 Towards artificial general intelligence with hybrid tianjic chip architecture *Nature* **572** 106–11

[12] Richter O, Wu C, Whatley A M, Köstinger G, Nielsen C, Qiao N and Indiveri G 2024 DYNAP-SE2: a scalable multi-core dynamic neuromorphic asynchronous spiking neural network processor *Neuromorph. Comput. Eng.* **4** 014003
[13] Khaddam-Aljameh R *et al* 2022 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* **57** 1027–38
[14] Le Gallo M *et al* 2023 A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference *Nat. Electron.* **6** 680–93
[15] Rao M *et al* 2023 Thousands of conductance levels in memristors integrated on CMOS *Nature* **615** 823–9
[16] Strukov D B, Snider G S, Stewart D R and Williams R S 2008 The missing memristor found *Nature* **453** 80–83
[17] Bohte S M, Kok J N and La Poutré J A 2000 Spikeprop: backpropagation for networks of spiking neurons *Esann* vol 48 pp 419–24
[18] Eshraghian J K, Ward M, Neftci E, Wang X, Lenz G, Dwivedi G, Bennamoun M, Seok Jeong D and Lu W D 2021 Training spiking neural networks using lessons from deep learning (arXiv:2109.12894)
[19] Hu Y, Tang H and Pan G 2023 Spiking deep residual networks *IEEE Trans. Neural Netw. Learn. Syst.* **34** 5200–5
[20] Li Y *et al* 2021 Differentiable spike: rethinking gradient-descent for training spiking neural networks *Advances in Neural Information Processing Systems* vol 34 pp 23426–39
[21] Rathi N, Chakraborty I, Kosta A, Sengupta A, Ankit A, Panda P and Roy K 2023 Exploring neuromorphic computing based on spiking neural networks: algorithms to hardware *ACM Comput. Surv.* **55** 1–49
[22] Werbos P J 1990 Backpropagation through time: what it does and how to do it *Proc. IEEE* **78** 1550–60
[23] Wu Y, Deng L, Li G, Zhu J and Shi L 2018 Enabling spike-based backpropagation for training deep neural network architectures *Proc. 35th Int. Conf. on Machine Learning (ICML 2018)* vol 80 pp 5143–52
[24] Hecht-Nielsen R 1992 Theory of the backpropagation neural network *Neural Networks for Perception* (Academic Press) pp 65–93
[25] Cao Y, Chen Y and Khosla D 2015 Spiking deep convolutional neural networks for energy-efficient object recognition *Int. J. Comput. Vis.* **113** 54–66
[26] Han B, Srinivasan G and Roy K 2020 Rmp-snn: residual membrane potential neuron for enabling deeper high-accuracy and low-latency spiking neural network *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition* pp 13558–67
[27] Lee J H, Delbruck T and Pfeiffer M 2016 Training deep spiking neural networks using backpropagation *Front. Neurosci.* **10** 508
[28] Rathi N and Roy K 2021 Diet-snn: a low-latency spiking neural network with direct input encoding and leakage and threshold optimization *IEEE Trans. Neural Netw. Learn. Syst.* **34** 3174–82
[29] Rueckauer B, Lungu I-A, Hu Y, Pfeiffer M and Liu S-C 2017 Conversion of continuous-valued deep networks to efficient event-driven networks for image classification *Front. Neurosci.* **11** 294078
[30] Shrestha S B and Orchard G 2018 Slayer: spike layer error reassignment in time *Advances in Neural Information Processing Systems* p 31
[31] Olshausen B A and Field D J 1996 Emergence of simple-cell receptive field properties by learning a sparse code for natural images *Nature* **381** 607–9
[32] Olshausen B A and Field D J 1997 Sparse coding with an overcomplete basis set: a strategy employed by v1? *Vis. Res.* **37** 3311–25
[33] Wright J, Ma Y, Mairal J, Sapiro G, Huang T S and Yan S 2010 Sparse representation for computer vision and pattern recognition *Proc. IEEE* **98** 1031–44
[34] Rozell C J, Johnson D H, Baraniuk R G and Olshausen B A 2008 Sparse coding via thresholding and local competition in neural circuits *Neural Comput.* **20** 2526–63
[35] Sheridan P M, Cai F, Du C, Ma W, Zhang Z and Lu W D 2017 Sparse coding with memristor networks *Nat. Nanotechnol.* **12** 784–9
[36] Teti M, Kenyon G, Migliori B and Moore J 2022 Lcanets: lateral competition improves robustness against corruption and attack *Int. Conf. on Machine Learning* (PMLR) pp 21232–52
[37] Parpart G, Risbud S, Kenyon G and Watkins Y 2023 Implementing and benchmarking the locally competitive algorithm on the loihi 2 neuromorphic processor *Proc. 2023 Int. Conf. on Neuromorphic Systems* pp 1–6
[38] Sebastian A, Le Gallo M, Khaddam-Aljameh R and Eleftheriou E 2020 Memory devices and applications for in-memory computing *Nat. Nanotechnol.* **15** 529–44
[39] Bello I, Fedus W, Du X, Dogus Cubuk E, Srinivas A, Lin T-Y, Shlens J and Zoph B 2021 Revisiting resnets: improved training and scaling strategies *Advances in Neural Information Processing Systems* vol 34 pp 22614–27
[40] Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E and Darrell T 2014 Decaf: a deep convolutional activation feature for generic visual recognition *Int. Conf. on Machine Learning* (PMLR) pp 647–55
[41] Girshick R, Donahue J, Darrell T and Malik J 2014 Rich feature hierarchies for accurate object detection and semantic segmentation *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 580–7
[42] Kornblith S, Shlens J and Le Q V 2019 Do better imagenet models transfer better? *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition* pp 2661–71
[43] Sharif Razavian A, Azizpour H, Sullivan J and Carlsson S 2014 Cnn features off-the-shelf: an astounding baseline for recognition *Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshops* pp 806–13
[44] Sharma N, Jain V and Mishra A 2018 An analysis of convolutional neural networks for image classification *Proc. Comput. Sci.* **132** 377–84
[45] Torrey L and Shavlik J 2010 Transfer learning *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods and Techniques* (IGI global) pp 242–64
[46] Weiss K, Khoshgoftaar T M and Wang D 2016 A survey of transfer learning *J. Big Data* **3** 1–40
[47] Deng J, Dong W, Socher R, Li Li-J, Li K and Fei-Fei Li 2009 Imagenet: a large-scale hierarchical image database *2009 IEEE Conf. on Computer Vision and Pattern Recognition* (IEEE) pp 248–55
[48] Cai H, Gan C, Zhu L and Han S 2020 Tinytl: reduce memory, not parameters for efficient on-device learning *Advances in Neural Information Processing Systems* vol 33 pp 11285–97
[49] Whatmough P N, Zhou C, Hansen P, Kolala Venkataramanaiah S, Seo J-Sun and Mattina M 2019 Fixynn: efficient hardware for mobile computer vision via transfer learning (arXiv:1902.11128)
[50] Zhu L, Hu L, Lin J, Chen W-M, Wang W-C, Gan C and Han S 2023 Pockengine: sparse and efficient fine-tuning in a pocket *Proc. 56th Annual IEEE/ACM Int. Symp. on Microarchitecture* pp 1381–94
[51] Krizhevsky A and Hinton G 2009 CIFAR-10 (canadian institute for advanced research) (available at: www.cs.toronto.edu/kriz/cifar.html)
[52] Howard A *et al* 2019 Searching for mobilenetv3 *Proc. IEEE/CVF Int. Conf. on Computer Vision*
[53] Yan Z, Zhou J and Wong W-F 2021 Near lossless transfer learning for spiking neural networks *Proc. AAAI Conf. on Artificial Intelligence* vol 35 pp 10577–84

[54] Qi C R, Su H, Mo K and Guibas L J 2017 Pointnet: deep learning on point sets for 3D classification and segmentation *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 652–60
[55] Dosovitskiy A *et al* 2020 An image is worth 16x16 words: transformers for image recognition at scale *Int. Conf. on Learning Representations (ICLR)*
[56] Esser S K, Appuswamy R, Merolla P, Arthur J V and Modha D S 2015 Backpropagation for energy-efficient neuromorphic computing *Advances in Neural Information Processing Systems* p 28
[57] Guo Y, Chen Y, Zhang L, Liu X, Wang Y, Huang X and Ma Z 2022 Im-loss: information maximization loss for spiking neural networks *Advances in Neural Information Processing Systems* vol 35 pp 156–66
[58] Perez-Nieves N and Goodman D F M 2021 Sparse spiking gradient descent *Advances in Neural Information Processing Systems* vol 34 pp 11795–808
[59] Xiao M, Meng Q, Zhang Z, He Di and Lin Z 2022 Online training through time for spiking neural networks *Advances in Neural Information Processing Systems* vol 35 pp 20717–30
[60] Zenke F and Ganguli S 2018 Superspike: supervised learning in multilayer spiking neural networks *Neural Comput.* **30** 1514–41
[61] Zenke F and Vogels T P 2021 The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks *Neural Comput.* **33** 899–925
[62] Neftci E O, Mostafa H and Zenke F 2019 Surrogate gradient learning in spiking neural networks: bringing the power of gradient-based optimization to spiking neural networks *IEEE Signal Process. Mag.* **36** 51–63
[63] Lee H, Battle A, Raina R and Ng A 2006 Efficient sparse coding algorithms *Advances in Neural Information Processing Systems* p 19
[64] Mairal J *et al* 2014 Sparse modeling for image and vision processing *Found. Trends Comput. Graph. Vis.* **8** 85–283
[65] Rigamonti R, Brown M A and Lepetit V 2011 Are sparse representations really relevant for image classification? *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (IEEE) pp 1545–52
[66] Zhang Z, Xu Y, Yang J, Li X and Zhang D 2015 A survey of sparse representation: algorithms and applications *IEEE Access* **3** 490–530
[67] Frankle J and Carbin M 2018 The lottery ticket hypothesis: training pruned neural networks *CoRR* (arXiv:1803.03635)
[68] He Y, Lin J, Liu Z, Wang H, Li Li-J and Han S 2018 Amc: automl for model compression and acceleration on mobile devices *Proc. Eur. Conf. on Computer Vision (ECCV)*
[69] Sietsma and Dow 1988 Neural net pruning-why and how *IEEE 1988 Int. Conf. on Neural Networks* vol 1 pp 325–33
[70] Girosi F, Jones M and Poggio T 1995 Regularization theory and neural networks architectures *Neural Comput.* **7** 219–69
[71] Wright J, Yang A Y, Ganesh A, Sastry S S and Ma Y 2008 Robust face recognition via sparse representation *IEEE Trans. Pattern Anal. Mach. Intell.* **31** 210–27
[72] Srivastava N, Hinton G, Krizhevsky A, Sutskever I and Salakhutdinov R 2014 Dropout: a simple way to prevent neural networks from overfitting *J. Mach. Learn. Res.* **15** 1929–58
[73] Guo Y 2018 A survey on methods and theories of quantized neural networks (arXiv:1808.04752)
[74] Cramer B *et al* 2022 Surrogate gradients for analog neuromorphic computing *Proc. Natl Acad. Sci.* **119** e2109194119
[75] Renner A, Sheldon F, Zlotnik A, Tao L and Sornborger A 2024 The backpropagation algorithm implemented on spiking neuromorphic hardware *Nat. Commun.* **15** 9691
[76] Fair K L, Mendat D R, Andreou A G, Rozell C J, Romberg J and Anderson D V 2019 Sparse coding using the locally competitive algorithm on the truenorth neurosynaptic system *Front. Neurosci.* **13** 754
[77] Paiton D M 2019 Analysis and applications of the locally competitive algorithm *PhD Thesis* University of California Berkeley
[78] LeCun Y, Cortes C and Burges C J 1998 The MNIST database of handwritten digits (available at: http://yann.lecun.com/exdb/mnist/)
[79] Huang G *et al* 2017 Densely connected convolutional networks *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*
[80] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* pp 770–8
[81] Tan M and Le Q 2021 Efficientnetv2: smaller models and faster training *Int. Conf. on Machine Learning* (PMLR) pp 10096–106
[82] PyTorch 2017 Torchvision: datasets and transforms for computer vision (available at: https://pytorch.org/vision/stable/index.html) (Accessed Jan 2024)
[83] NVIDIA 2022 Resnet-50-v1.5 for pytorch (available at: https://catalog.ngc.nvidia.com/orgs/nvidia/resources/resnet_50_v1_5_for_pytorch) (Accessed 1 June 2024)
[84] Esser S K *et al* 2016 Convolutional networks for fast, energy-efficient neuromorphic computing *Proc. Natl Acad. Sci. USA* **113** 11441–6