SNE: an Energy-Proportional Digital Accelerator for Sparse Event-Based Convolutions

Alfio Di Mauro†, Arpan Suravi Prasad†, Zhikai Huang†, Matteo Spallanzani†, Francesco Conti‡, Luca Benini†‡
†Dept. of Information Technology and Electrical Engineering, ETH Zürich, Switzerland
‡Dept. of Electrical, Electronic and Information Engineering, University of Bologna, Italy

***Abstract*—Event-based sensors are drawing increasing attention due to their high temporal resolution, low power consumption, and low bandwidth. To efficiently extract semantically meaningful information from sparse data streams produced by such sensors, we present a 4.5TOP/s/W digital accelerator capable of performing 4-bits-quantized event-based convolutional neural networks (eCNN). Compared to standard convolutional engines, our accelerator performs a number of operations proportional to the number of events contained into the input data stream, ultimately achieving a high energy-to-information processing proportionality. On the IBM-DVS-Gesture dataset, we report 80uJ/inf to 261uJ/inf, respectively, when the input activity is 1.2% and 4.9%. Our accelerator consumes 0.221pJ/SOP, to the best of our knowledge it is the lowest energy/OP reported on a digital neuromorphic engine.**

***Index Terms*—Event-based computing, neuromorphic platform, edge-computing**

I. INTRODUCTION

In recent years, we experienced a profound transformation in how digital systems deployed around us operate. Edge-computing devices evolved toward lower power consumption, longer lifetime, heterogeneity, and higher computational capabilities. Similarly, sensors improved to efficiently convey information to processing engines while balancing low power consumption, responsiveness, and energy spent on data transfer.

Event-based sensors are an emerging class of devices that measure a physical quantity and transfer such information in a frame-less fashion. Event-based vision sensors (EVSs), which asynchronously measure the brightness changes in the field of view, belong to this category. Compared to a traditional frame-based imaging sensor, an EVS outputs a stream of events that encodes the brightness change's time, location, and polarity. EVSs typically feature higher temporal resolution (in the order of few µs) and a lower power consumption (ranging from $250\,\mu W$ to $2\,mW$); in low activity scenarios, their bandwidth can be as low as a few kB/s [1].

A key advantage introduced by event sensors is the proportionality between the primary sensor input and the number of output events generated by it [2]. To efficiently exploit the inherently sparse nature of such data streams, the energy to information proportionality needs to be preserved across the whole processing pipeline. CPU and GPU class devices can marginally profit from unstructured data sparsity, while dedicated deep neural networks (DNN) accelerators often rely on dedicated architectural features tailored to the specific type of data sparsity to achieve high energy efficiencies in such scenarios [3]. To overcome this limitation, a paradigm shift in how such sparse data are processed is needed.

Neuromorphic algorithms, i.e. algorithms inspired by how biological brains work, are promising candidates to solve the unstructured data processing problem. Among neuromorphic algorithms, Spiking Neural Networks (SNNs) represent the leading model-free algorithmic approach for EVSs data processing [4]. Like many artificial neural networks (ANNs), SNNs rely on elementary computational units, i.e. neurons, which are interconnected through synaptic weights to form computational networks [5]. A distinctive feature of SNN neurons is the presence of a *neuron internal state*, which evolves over the entire inference process. Recent advances in SNNs show that such a class of networks can achieve accuracy levels comparable to state-of-the-art (SoA) deep learning networks while significantly reducing the number of required computational operations [6], therefore making them a suitable candidate to achieve high energy-to-information processing proportionality.

In this work, we present a novel digital sparse neural engine (SNE) to efficiently accelerate SNN inference tasks at the extreme edge. Our accelerator exploits an explicit input event temporal and spatial location encoding, the `SNE` architecture is designed to improve input data and weight reuse, reducing the traffic towards the memory. `SNE` achieves a maximum performance of 51.2 GSOP/s, and an energy efficiency of 4.5TSOP/s/W. Ultimately, `SNE` shows 3.55X higher energy efficiency than SoA neuromorphic platform [7], approaching classical DNN accelerators energy efficiencies [8], while performing energy-proportional computations. As a proof of concept, we show that `SNE` consumes 0.221 pJ/SOP and achieves 92.8% accuracy on a classification task performed on the IBM DVS-Gesture data set.

II. RELATED WORKS

Over the last years, research and industry have proposed various deep learning engines to accelerate inference at the edge, achieving extreme energy efficiencies [9]. As algorithmic research proposed low-precision, highly-quantized networks, hardware platform evolution kept the pace by proposing new architectures capable of exploiting low memory footprints and performing low-precision operations [10], [11]. Recently, neuromorphic algorithms have been attracting increasing attention as a more energy-efficient alternative to conventional deep learning approaches [12], especially in those contexts where input feature maps are produced at a nonconstant rate and are also characterized by high unstructured sparsity [13].

Such algorithms run on neuromorphic platforms that can be divided into two main categories: Analog and mixed-signal, and digital SNN accelerators. Analog and mixed-signal implementation present several advantages over digital implementations, e.g. they typically achieve higher energy efficiencies and smaller neuron area footprint [14]. Also, they typically implement more complex neuron models. However, these designs are hard to scale, as their functionality is often technology-dependent, requiring laborious tuning to the technology node. Additionally, mixed-signal operations require many biases generated on-chip, often degrading the system-level energy efficiency. Contrarily, digital implementation typically features a less complex and more scalable neuron model [15]–[17], as well as fast integration in digital SoCs and technology porting.

Compared to the accelerators mentioned above, `SNE` explicitly encodes the temporal and spatial location of the events to reduce the temporary data memory footprint of highly sparse input and intermediate feature maps. `SNE` also maximizes input data and weight reuse, eventually reducing the traffic towards the memory. Compared to the existing neuromorphic platforms (table II), `SNE` performs synchronous parallel execution. This feature, coupled with the explicit event encoding, compresses long intervals of sparse input activity into dense computational phases performed at high frequency. Our accelerator improves the SoA in energy efficiency by 3.55X while achieving SoA accuracy of 92.8% on the IBM DVS-Gesture data set.

III. Architecture

A. Spiking Neural Network

A Spiking Neural Network (SNN) is an Artificial Neural Network (ANN) subcategory whose elementary unit mimics biological neurons [18]. Compared to ANNs, SNNs neurons, named as *spiking neurons*, feature an internal state variable, a *membrane potential*. A firing behavior characterizes spiking neurons, i.e. a spiking neuron generates an impulse on its output when the membrane potential value exceeds a threshold. In SNNs, input and output feature maps can be encoded as binary tensors; the presence of a non-zero value represents the spatial and temporal position of the spike produced by a neuron. Connections among spiking neurons are weighted by synaptic weights. In complex neuron models, synapses can also time-shift input spikes. Synaptic connections between successive layers of an SNN can follow a convolutional or fully-connected scheme.

B. SNE neuron model

Elementary SNN neurons can be modeled at a different biological plausibility level, ranging from highly approximated neuronal behavior to bio-plausible models [19]. In `SNE`, we implemented a leaky integrate and fire (LIF) neuron. We linearly approximated the exponential membrane potential decay to simplify the hardware design as an iterative linear decay. The elementary neuron membrane potential update is given by $V_{mem}[t+1] = -L + \sum_j W_{ij}S_i[t]$; in our implementation, L is a re-programmable leakage quantity that is subtracted at every time step. The firing rule is described by $S[t] = \Theta(V_{mem}[t] - V_{th})$, where Θ is the Heaviside step function, and V_{th} is a programmable firing threshold.

C. SNE execution model

The `SNE` accelerator data path has been optimized for event-driven computation. Compared to a standard CNN layer, in event-based convolutional neural network (eCNN) layers we find an additional time dimension. The event-based convolution is performed for each time step, and the state of each neuron resets at the beginning of a new inference. The input synaptic contributions are accumulated in the state variable across the entire inference process.

To exploit the input feature map sparsity, events are encoded explicitly and stored into the memory with the format reported in Fig.1; `SNE` is fed with individual events instead of input tensor tiles. Listing 1 reports the SNE input event processing pseudo-code. To better exploit spatial and temporal data reuse opportunities, as an outermost loop, we span the time dimension and process all events occurring at a certain time step. Then, `SNE` updates all the output neurons depending on the current input, without spanning the horizontal and vertical dimension of the input tensor in the innermost loop. This loop organization has been chosen because data-path instances are stateful. Therefore, output spike sequences related to each output neuron have to be produced entirely once the input is presented. Multiple input channels can be accumulated on the same output neuron, `SNE` can store up to 256 sets of weights in a filter buffer, and they can be independently selected on-the-fly by each `Cluster`, according to the addressing of the input event. An `SNE` event is defined by a 32bits value partitioned into the quadruple: $E_i := (OP_e, t, x, y)$. OP_e stands for event operation, and it can be of three different types:

- `RST_OP` is the event operation that resets the state variable of the neurons to zero.
- `UPDATE_OP` is the operation that updates the membrane potential of the neurons having the current input event in their receptive field.
- `FIRE_OP` is the operation that concludes the neuron state update phase and allows the neuron whose status variable value is above the threshold to produce an output event.

D. SNE architecture

The `SNE` architecture (Fig.2) is composed by a set of independent parallel processing engines called slices (`SLs`). Each `SL` is connected to a synaptic crossbar (`C-XBAR`), which also connects two autonomous direct memory access engines (`DMA`) used to transfer events from the memory to the `SLs` and vice versa. Output event streams produced by the `SLs` are

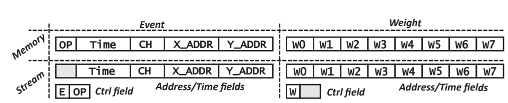


Fig. 1: SNE data format for event and weight.

```
# SW managed loops ---------------------------------
for k_o in range(0,C_o):                #output Ch events
 program_sne(W)                          #change weights
#---------------------------------------------------
  # SNE managed loops ------------------------------
  for t in range(0,T):                    #time dimension
   for evt_i in events_in[t]        #explicit evt repr
    k_i,e_x,e_y = get_address(evt_i)
    for i in range(0,H_o):              #output h neurons
     for j in range(0,W_o):             #output v neurons
      w_ij = weight(i,j,k_i,e_x,e_y,W)   #weight calc
      evt_o = neuron_dynamics(i,j,w_ij)
      events_out[t].append(evt_o)          #push evt out
  #-----------------------------------------------
```

Listing 1: SNE sparse eCNN layer execution.

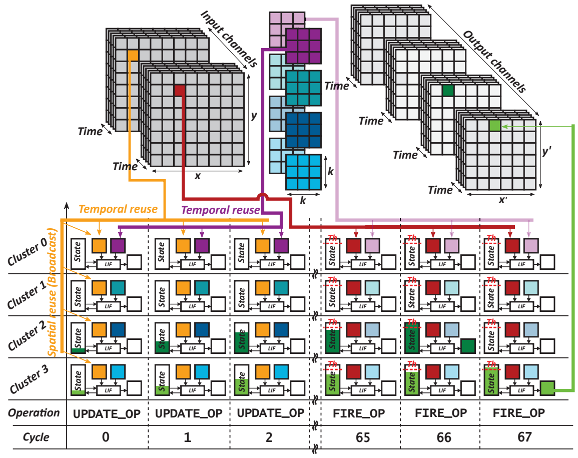


Fig. 3: SNE convolution operation execution.

joined in a single stream using a `collector`, which is also connected as a master to the `C-XBAR`. `SNE` can be integrated as a memory-mapped peripheral into a system on chips (SoC) and programmed through a register interface. The following subsections provide a more detailed description of each `SNE` top-level module.

1) c-xbar: The `C-XBAR` routes both streams of events and weights from the main memory to the slices or vice versa. The data format used for the internal event representation is described in Fig.1. Each `SL` is connected to the `C-XBAR` with communication protocol using a ready-valid (RV) handshake for flow control. The `C-XBAR` can operate in two distinct modes: *i)* single master to single slave port (point-to-point); this configuration is also used to both transfer events and load configuration parameters. *ii)* single master to multiple slave ports (broadcast); in this configuration, the `C-XBAR` can perform flow control and pause the transaction until all slave ports have received the event.

2) streamer: `DMAs` autonomously transfer events and weights from the main memory to the `SNE` internal buffers and vice versa. `SNE` input events can be stored linearly into the external memory. Therefore, `DMAs` implement a simple 1D data movement scheme; they also operate the conversion between the event memory format and event stream format shown in Fig.1. The `DMA` contains a 16-words First-In-First-Out (FIFO) event memory to absorb memory latency cycles (e.g., due to access contention).

3) collector: The `collector` allows packing the output event streams from each `SL` into a single time synchronized stream and sending it to the `C-XBAR`. Since the activity of the `SLs` is sparse, a single `DMA` can provide significantly more bandwidth than required on a single `SL` output port. Therefore, the collector arbitrates between the `SLs` output ports and multiplexes them into a single event stream towards the memory.

4) slices and clusters: The number of `SLs` in `SNE` is parametric; the architecture of a single `SL` is shown in Fig.2. Each `SL` instantiates 16 parallel computational units, called `Clusters`. Each `Cluster` data-path is designed to compute a neuron state update in a single clock cycle, using a single data-path: the implementation of multiple neurons is achieved by time-domain multiplexing (TDM), storing the neuron states in a local latch-based buffer. Each `Cluster` implements 64 TDM neurons using 4 bits for synaptic weights and 8 bits for the internal state. Before dispatching input events to the `Clusters`, the `SL` needs to decode the event operation to perform. Units that do not have to update their internal state (i.e., membrane potential) are clock-gated to reduce power consumption. In the case of a `RST_OP`, all the `Clusters` are activated, and the membrane potential resets for all the neurons of the `SL`. All `Clusters` of an `SL` receive the same input event. We implemented an address filtering mechanism to selectively redirect input events to specific neurons. Execution on all `Clusters` happens synchronously and is orchestrated by a module called `Sequencer`. The `Sequencer` provides the address of the current TDM neuron update. When the slice is executing a `SPIKE_OP`, all the 64 TDM neurons of each `Cluster` can potentially produce an output event. To avoid stalling the TDM neurons update, each `Cluster` is connected

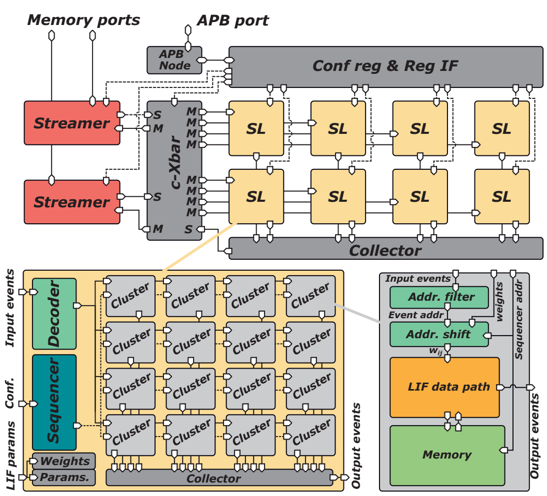


Fig. 2: SNE architecture block diagram.

to an output event FIFO, and all FIFOs are connected to a `collector` module. All computed output neurons across the `Clusters` have the same relative position. The absolute spatial mapping of the output neurons is achieved by shifting each address with respect to the `Cluster` base address. To achieve high throughput, the following measures were put in place at the `Cluster` level: *i)* the LIF neuron dynamic data path is combinational. *ii)* To overcome the high memory bandwidth that characterizes SNN inference, the neuron states of each `Cluster` are stored locally into two dedicated state memories. During the neuron state update, values are fetched alternatively from each memory and stored the cycle after in a double buffering fashion, practically achieving one state update per cycle. *iii)* a time-of-last-update (TLU) is stored per `Cluster`, the next neuron state is computed based on the current timestep value and TLU, skipping the state update in the absence of input activity between two successive timesteps.

5) mapping: The `SNE` can be used in two modes. If the neurons of an SNN can be mapped entirely on the `SNE` available `Clusters` along the spatial dimensions, each `SL` can be used to implement a different layer of the network, and the synaptic connections between neurons of consecutive layers are achieved through the `C-XBAR`. In this mode, events from the `collector` can be redirected to any `SL`, output events are produced simultaneously to the input event processing, and all the layers of the network can execute in parallel. Alternatively, if the network needs to allocate more neurons than available in the `SNE`, intermediate feature maps (output events) must be stored in the external memory. In this case, the `SNE` can be used in a time-multiplexed way to execute only a tile of the network. In this operating mode, synaptic connections are implemented by both the `C-XBAR` and the `DMAs` through the external memory. Fig.3 shows the execution pipeline of operations to compute an eCNN layer. An input event is fetched and made available to all `Clusters`. Then, the `SNE` updates all the neurons of each `Cluster` that are sensitive to the current input event, this operation is performed in 48 clock cycles. The state of each output neuron is held across multiple input event processing, and as soon as a firing operation is received, all the neurons having the membrane potential above the threshold fire an output event.

IV. Experimental results

This section provides post-synthesis estimates for the `SNE` as a standalone engine. We synthesized the accelerator with *Synopsys Design Compiler 2020.09*, in *GlobalFoundries* 22nm FDX process. Specifically, we used 8T, 20, 24, 28, L, and SL voltage threshold cells, SSG corner, 0.72V nominal supply voltage, -40C, 400MHz target clock frequency. Power consumption estimates have been performed at target 400MHz clock frequency, TT corner, 0.8V supply voltage, 25C, by using *Synopsys Prime-Power 2019.12*.

A. SNE energy efficiency benchmark

In this subsection, we evaluate how the performance, power consumption, area, and energy efficiency scale when the `SNE` number of `SLs` is configured to 1, 2, 4, and 8. We report

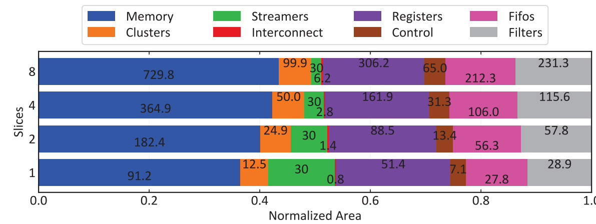
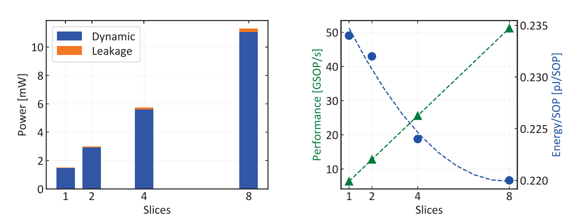


Fig. 4: SNE area breakdown for a different number of Slices. Values on the plot report the absolute area in kGE

the results for the `SNE` configured with 16 `Cluster` per slice and 64 TDM neurons per `Cluster`.

1) Area exploration: Fig.4 reports the area estimation in gate equivalent (kGE) for each configuration. This number is obtained as the total area estimate in μm^2 provided by the synthesis tool, divided by the area of an ND2X1 gate (8T library). Most of the area is occupied by latch-based memories holding the neuron state. As the number of `SLs` increase, the `SLs` and `C-XBAR` area scales proportionally. `DMA` area remain constant. Area exploration shows that the fixed cost of the `DMAs` is progressively absorbed by the data path area increase.

2) Power analysis: We estimated the power consumption using the simulated value change dump (VCD) activity of the post-synthesis netlist. The benchmark used for power consumption estimation is a sample eCNN layer where input events cause a neuron state update on all the `SLs` and all `Clusters` of each `SL`. Input events are distributed across 100 time steps, and the layer is generating 5% output event activity, which is comparable with the average network activity observed for the IBM-DVS Gesture data set and reported in section IV-B. The VCD file used to extract the switching activities has been generated with Questasim-10.6b, while the power consumption has been estimated with Synopsys PrimePower-2019.12. Fig.5a reports the power consumption for the different `SNE` configurations. Dynamic power significantly dominates the total power consumption. Notice that the power consumption reported for this experiment is a worst-case estimate, as all computational units of the `SNE` are



(a) Power consumption at average network firing activity of 5%)

(b) Performance and energy per operation versus Number of Slices.

Fig. 5: `SNE` energy and power consumption.

Data set	SNN (SLAYER-SRM)	eCNN (SNE-LIF-4b)	Inf. energy [uJ/inf]	Inf. rate [inf/s]
NMNIST	97.81%	97.88%	43 - 142	261 - 79.5
IBM DVS Gest.	92.42%	92.80%	80 - 261	141 - 43

TABLE I: eCNN classification accuracy, energy per inference and inference rate

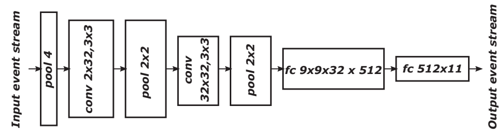


Fig. 6: SNN network topology used for the accuracy benchmark.

updating the internal state of their neurons.

3) Performance and energy efficiency benchmark: Fig.5b shows the accelerator performance reported as synaptic operations per second (SOP/s). `SNE` performance scales proportionally to the number of slices, as they operate independently, and the output bandwidth does not represent a bottleneck. Note that in the case where more `SLs` are added to the `SNE`, or when more activity is expected on the output of each `SL`, the `SNE` can be configured with a higher number of `DMAs` to sustain the `SLs` output bandwidth. Fig.5b also reports the energy per synaptic operation (SOP). This value has been calculated by dividing the energy consumed in a single cycle by the number of neuron updates performed in parallel. We remark that `SNE` takes 48 clock cycles to consume an input event and update all membrane potentials serially. Therefore, true energy-to-information processing proportionality is ensured by design, i.e. the more events are present in a given input event stream, the more time `SNE` spends to consume such an event stream. We obtained the lowest energy/SOP when `SNE` is configured for 8 `SLs`, consuming a constant energy of 0.221pJ/SOP. In this configuration, the ratio between the power consumed by the engines and other parts of the system is maximized, and most of the energy is spent on computation.

B. Accuracy benchmark

We conducted two experiments to evaluate the accuracy of eCNNs deployed on the `SNE`. We trained the same network, whose topology is reported in figure 6, on two event-based data sets. The networks have been trained with a supervised approach. Specifically, we used back-propagation-based training in the SLAYER [23] framework. As the `SNE` implements a quantized variant of the LIF dynamics, where a linear decay has approximated the exponential decay, we implemented our `SNE` neuron model and replaced the default *SLAYER* spike response model (SRM) [24]. We trained the same network with SRM neurons as a baseline comparison for our experiments.

In the first experiment, we trained the network on the NMNIST data set[1], and we evaluated its accuracy. In the second experiment, we followed the same approach to train and evaluate the network accuracy on a more complex task, the IBM-DVS-Gesture data set[2]. We used 65%, 10%, and 25% of samples for training, validation, and test set on the IBM-DVS Gesture data set, respectively. On the NMNIST data set, we divided the samples into 75%, 10%, and 15% of samples for training, validation, and test set, respectively. On both data sets, `SNE` eCNNs slightly improved the classification accuracy; accuracy results for both data sets are reported in Table I.

[1] https://www.garrickorchard.com/datasets/n-mnist
[2] https://www.research.ibm.com/dvsgesture/

As a further investigation, we estimated the maximum activity of each eCNN layer, obtaining that a sample extracted by the IBM DVS-Gesture data set generated a firing activity between 1.2% and 4.9%, on average, across the entire network. As `SNE` consumes an input event in 48 clock cycles, we used each layer activity to estimate then best case and worst case inference time, when the accelerator is clocked at 400MHz. In this operating point, an input event is consumed in 120 ns, based on this value and the network activity, the inference is performed in a best and worst case time interval of 7.1ms and 23.12ms, respectively. Similarly, we estimated that in this operating condition the `SNE` can perform a new inference at a rate comprised between 141inf/s and 43inf/s, consuming a total inference energy between 80 µJ/inf and 261 µJ/inf, respectively. Table I reports the `SNE` inference performance and inference energy results for both data sets.

C. Comparison with the state of the art

In Table II we compare the `SNE` to state-of-the-art neuromorphic engines implementing comparable neuron models and accelerating similar neural network topologies. To have a fair comparison with other engines reported in Table II, we remark that `SNE` does not provide online learning capabilities. Compared to other architectures, `SNE` shows both the lowest energy per operation 0.221pJ/SOP and the highest energy efficiency 4.54TSOP/s/W, while reaching SoA accuracy on event-based data sets; 92.8% on IBM DVS-Gesture. This result narrows the gap between neuromorphic platforms and classical DNN accelerators [8]. The energy efficiency reported on `SNE` improves by 3.55X the energy efficiency reported by Pei et al. [7]. Note that area-wise, both designs are implemented in a comparably scaled technology node. Additionally, assuming the same 400MHz target frequency and extrapolating our results to the 0.9V operating condition, `SNE` would still achieve 4.03TOP/s/W and consume 0.248pJ/SOP.

V. Conclusion

In this paper, we presented a configurable digital engine for brain-inspired event-based convolutional neural networks (eCNN). Our accelerator exploits the unstructured sparsity of data produced by event-based sensors by performing a number of operations proportional to the input stream activity. Our engine consumes explicitly spatial and temporal-encoded input events to achieve high energy efficiency, selectively updating the internal output neurons states. We demonstrated that `SNE` can reach SoA 98.2% classification accuracy on the IBM DVS-Gesture data set while performing up to 141inf/s. `SNE` achieves SoA energy efficiency of 4.54TSOP/s, comparable to classical DNN inference engines. Ultimately,

Name	Tech.	Neuron model	Learning	Type	Neuron number	Neuron area [um2]	Perf. [GOP/s]	Eff. [TOP/s/W]	Energy/SOP [pJ]	Freq. [MHz]	Power [mW]	bits	V
SNE (this work)	**Digital 22nm**	**LIF**	**offline**	**Conv SNN**	**8192**	**19.9**	**51.2**	**4.54**	**0.221**	**400**	**11.29**	**4**	**0.8**
Tianjic [7]	Digital 28nm	-	-	Hybrid	40000	361	649	1.28	6.18	300	950	8	0.9
Dynapsel [20]	Analog 28nm	-	online STDP	-	256	150390	-	0.6	2	-	-	4	1
ODIN [21]	Digital 28nm	Bio Plaus.	-	-	256	335.9	0.038	0.079	12.7	75	0.477	-	0.55
TrueNorth [16]	Digital 28nm	EXP LIF	online	SNN	1e6	389	58	0.046	27	Asynch	65	1	0.75
SPOON [17]	Digital 28nm	-	DRTP	Conv SNN	-	-	-	-	1700	150	-	8	0.6
Loihi [15]	Digital 14nm	LIF+	online STDP	SNN	131072	396.7	-	-	23	Asynch	-	1-64	-
SpiNNaker 2 [22]	Digital 22nm	Prog.	-	DNN/SNN	-	-	-	3.26	1700	200	-	var.	0.5

TABLE II: State of the sne comparison.

SNE consumes 0.221pJ/SOP, which is the lowest energy per operation reported on a neuromorphic platform to the best of our knowledge.

VI. Acknowledgement

This work has been supported in part by the *Ampere* project funded by the EU Horizon 2020 research and innovation programme under grant agreement No. 871669. The authors would like to thank *Armasuisse Science and Technology* for funding this research, and *IniVation* for kindly lending us a DVS camera.

References

[1] C. Li, L. Longinotti, F. Corradi, and T. Delbruck, "A 132 by 104 10um-pixel 250uw 1kefps dynamic vision sensor with pixel-parallel noise and spatial redundancy suppression," in *2019 Symposium on VLSI Circuits*, 2019, pp. C216–C217.

[2] A. Di Mauro, M. Scherer, J. F. Mas, B. Bougenot, M. Magno, and L. Benini, "Flydvs: An event-driven wireless ultra-low power visual sensor node," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, Feb 2021, pp. 1851–1854.

[3] 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," *Proceedings of the IEEE*, pp. 1–47, 2021.

[4] N. Rathi, A. Agrawal, C. Lee, A. K. Kosta, and K. Roy, "Exploring spike-based learning for neuromorphic computing: Prospects and perspectives," in *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, Feb 2021, pp. 902–907.

[5] P. Panda, S. A. Aketi, and K. Roy, "Toward Scalable, Efficient, and Accurate Deep Spiking Neural Networks With Backward Residual Connections, Stochastic Softmax, and Hybridization," *Frontiers in Neuroscience*, 2020.

[6] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going Deeper in Spiking Neural Networks: VGG and Residual Architectures," *Frontiers in Neuroscience*, 2019.

[7] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He, F. Chen, N. Deng, S. Wu, Y. Wang, Y. Wu, Z. Yang, C. Ma, G. Li, W. Han, H. Li, H. Wu, R. Zhao, Y. Xie, and L. Shi, "Towards artificial general intelligence with hybrid Tianjic chip architecture," *Nature*, 2019.

[8] D. C. Daly, L. C. Fujino, and K. C. Smith, "Through the looking glass-2020 edition: Trends in solid-state circuits from isscc," *IEEE Solid-State Circuits Magazine*, vol. 12, no. 1, pp. 8–24, 2020.

[9] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, "Survey and benchmarking of machine learning accelerators," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, 2019, pp. 1–9.

[10] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," *Pattern Recognition*, vol. 105, p. 107281, 2020.

[11] A. D. Mauro, F. Conti, P. D. Schiavone, D. Rossi, and L. Benini, "Always-on 674uw@4gop/s error resilient binary neural networks with aggressive sram voltage scaling on a 22-nm iot end-node," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 11, pp. 3905–3918, 2020.

[12] M. Bouvier, A. Valentian, T. Mesquida, F. Rummens, M. Reyboz, E. Vianello, and E. Beigne, "Spiking neural networks hardware implementations and challenges: A survey," *J. Emerg. Technol. Comput. Syst.*, vol. 15, no. 2, Apr. 2019.

[13] A. R. Young, M. Dean, J. S. Plank, and G. S. Rose, "A Review of spiking neuromorphic hardware communication systems," *IEEE Access*, 2019.

[14] A. Rubino, C. Livanelioglu, N. Qiao, M. Payvand, and G. Indiveri, "Ultra-low-power fdsoi neural circuits for extreme-edge neuromorphic intelligence," 2020.

[15] M. Davies, N. Srinivasa, T. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[16] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha, "Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[17] C. Frenkel, J.-D. Legat, and D. Bol, "A 28-nm convolutional neuromorphic processor enabling online learning with spike-based retinas," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[18] C. Farabet, R. Paz, J. Pérez-Carrasco, C. Zamarreño-Ramos, A. Linares-Barranco, Y. LeCun, E. Culurciello, T. Serrano-Gotarredona, and B. Linares-Barranco, "Comparison between frame-constrained fix-pixel-value and frame-free spiking-dynamic-pixel convNets for visual processing," *Frontiers in Neuroscience*, 2012.

[19] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[20] C. S. Thakur, J. L. Molin, G. Cauwenberghs, G. Indiveri, K. Kumar, N. Qiao, J. Schemmel, R. Wang, E. Chicca, J. Olson Hasler, J.-s. Seo, S. Yu, Y. Cao, A. van Schaik, and R. Etienne-Cummings, "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," *Frontiers in Neuroscience*, vol. 12, p. 891, 2018.

[21] C. Frenkel, M. Lefebvre, J. D. Legat, and D. Bol, "A 0.086-mm^2 12.7-pj/sop 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm cmos," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 13, no. 1, pp. 145–158, 2019.

[22] S. Höppner, Y. Yan, A. Dixius, S. Scholze, J. Partzsch, M. Stolba, F. Kelber, B. Vogginger, F. Neumärker, G. Ellguth, S. Hartmann, S. Schiefer, T. Hocker, D. Walter, G. Liu, J. D. Garside, S. B. Furber, and C. Mayr, "The spinnaker 2 processing element architecture for hybrid digital neuromorphic computing," *CoRR*, vol. abs/2103.08392, 2021. [Online]. Available: https://arxiv.org/abs/2103.08392

[23] S. B. Shrestha and G. Orchard, "SLAYER: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 1419–1428.

[24] W. Gerstner, "Chapter 12 a framework for spiking neuron models: The spike response model," in *Neuro-Informatics and Neural Modelling*, ser. Handbook of Biological Physics, F. Moss and S. Gielen, Eds. North-Holland, 2001, vol. 4, pp. 469 – 516.