# Enabling Resource-Aware Mapping of Spiking Neural Networks via Spatial Decomposition

Adarsha Balaji, Shihao Song, Anup Das, Jeffrey Krichmar, Nikil Dutt,
James Shackleford, Nagarajan Kandasamy, and Francky Catthoor

***Abstract*—With growing model complexity, mapping Spiking Neural Network (SNN)-based applications to tile-based neuromorphic hardware is becoming increasingly challenging. This is because the synaptic storage resources on a tile, viz. a crossbar, can accommodate only a fixed number of pre-synaptic connections per post-synaptic neuron. For complex SNN models that have many pre-synaptic connections per neuron, some connections may need to be pruned after training to fit onto the tile resources, leading to a loss in model quality, e.g., accuracy. In this work, we propose a novel unrolling technique that decomposes a neuron function with many pre-synaptic connections into a sequence of homogeneous neural units, where each neural unit is a function computation node, with two pre-synaptic connections. This spatial decomposition technique significantly improves crossbar utilization and retains all pre-synaptic connections, resulting in no loss of the model quality derived from connection pruning. We integrate the proposed technique within an existing SNN mapping framework and evaluate it using machine learning applications on the DYNAP-SE state-of-the-art neuromorphic hardware. Our results demonstrate an average 60% lower crossbar requirement, 9x higher synapse utilization, 62% lower wasted energy on the hardware, and between 0.8% and 4.6% increase in model quality.**

***Index Terms*—Neuromorphic Computing, Spiking Neural Networks (SNNs), Machine Learning, Computation Graph.**

## I. Introduction

SPIKING Neural Networks (SNNs) are machine learning approaches designed using spike-based computation and bio-inspired learning algorithms [1]. SNNs are executed on tile-based neuromorphic hardware such as DYNAP-SE [2]. A tile consists of a crossbar. At each crosspoint of a crossbar there is a synaptic element, which can be implemented using Non-Volatile Memory (NVM) [3]. A neuron circuit takes as its input current and generates at its output a train of spike voltages. A spike is generated only if the current is higher than a threshold. The spike firing frequency increases with input current, saturating at a frequency determined by the refractory period of the neuron. A $n \times n$ crossbar in a tile can accommodate only $n$ pre-synaptic connections per post-synaptic neuron ($n = 128$ for the crossbars in DYNAP-SE). Fig. 1 reports the number of neurons with more than 128 pre-synaptic connections, i.e., fanins as a fraction of the total number of neurons in a few standard machine learning models, which are pruned iteratively to eliminate the near-zero weights without loss in accuracy [4] (see Fig. 2).

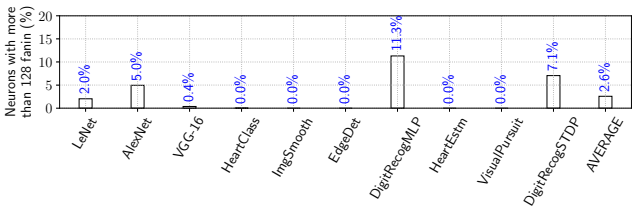


Fig. 1. Fraction of neurons with 128 or more pre-synaptic connections (fanin).

We observe that, even after model pruning, on average 2.6% of neurons in these models cannot be mapped to the crossbars in DYNAP-SE. There are two currently-used solutions to this problem – 1) implement larger crossbars, which increases the power consumption exponentially, and 2) remove synaptic connections, which reduces the model quality.

Instead, we propose an unrolling technique which decomposes each neuron having many pre-synaptic inputs or *fanin* into a sequence of homogeneous neural units, where each neural unit is a function computation node having a maximum fanin-of-two (or FIT). Our technique ensures information integrity as well as the quality of these unrolled machine learning models. Furthermore, the unrolling technique allows denser packing of the homogeneous neural units per crossbar, significantly improving the crossbar utilization. We evaluate our technique using standard machine learning applications and demonstrate an average 60% lower resource usage and 62% lower wasted energy. The proposed spatial decomposition technique retains all pre-synaptic connections for each neuron in an SNN, improving model quality between 0.8% and 4.6% compared to an existing SNN mapping approach.

## II. Proposed Technique

Fig. 2 illustrates the integration of the proposed technique (shown in the two colored boxes) inside an existing design flow SpiNeMap [5] for mapping SNNs to the hardware.[1] This design flow incorporates 1) Artificial Neural Network (ANN) models written in PyTorch and Tensorflow, and 2) SNN models written in PyCARL. For ANN models, analog operations are first converted to spike-based operations using the `ANN2SNN` converter [18]. These models are first pruned to eliminate near-zero weights [4]. This is to keep the model size small for embedded platforms. The pruned models are then presented to the proposed unrolling technique for mapping to hardware.

Our proposed technique works in two steps – 1) the unrolling step, which decomposes the SNN to limit the number

A. Balaji, S. Song, A. Das, J. Shackleford, and N. Kandasamy are with the Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA, 19104 USA e-mail: {shihao.song,anup.das,jas64,nk78}@drexel.edu

J. Krichmar and N. Dutt are with the Department of Computer Science, University of California, Irvine, CA, USA.

F. Catthoor is with Imec, Belgium and KU Leuven, Belgium.

Manuscript received Month Day, Year; revised Month Day, Year.

[1] Our approach can be combined with other mapping approaches [6]–[17].

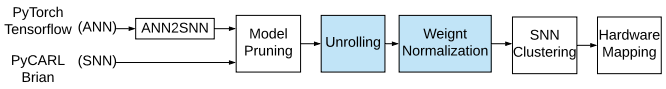

Fig. 2. Integration of the proposed approach in an existing design flow.

of pre-synaptic connections for each neuron, and 2) the weight normalization step, which ensures the quality of the decomposed model. The latter is then clustered and mapped to the neuromorphic hardware using SpiNeMap [5].

### A. Spatial Decomposition using Model Unrolling

We propose an unrolling approach, which decomposes a neuron function computation with many fanins into a sequence of homogeneous *neural units*, where each neural unit is a computation node with a maximum fanin-of-two (FIT). Fig. 3 illustrates the decomposition (❷) of the neuron function shown in ❶. Here, one $m$-input neuron function is decomposed into ($m-1$) two input neural units connected in sequence.

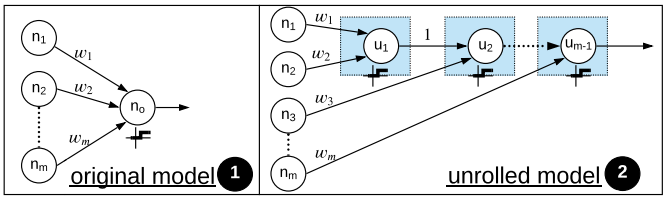

Fig. 3. Unrolling a neuron functionality.

The neuron function $y_o = f\left(\sum_{i=1}^{m} n_i \cdot w_i\right)$ is represented as

$$y_o = f(u_{m-1}),\ \text{where } u_i = \begin{cases} f(n_1 \cdot w_1 + n_2 \cdot w_2), & \text{for } i = 1 \\ f(u_{i-1} + n_{i+1} \cdot w_{i+1}), & \text{otherwise.} \end{cases} \tag{1}$$

where $f$ represents the neuron functionality of generating spike trains with a mean firing rate proportional to its input excitation, $n_1, n_2, \cdots, n_m$ are the $m$ pre-synaptic neurons of the post-synaptic neuron $n_o$, and $w_1, w_2, \cdots, w_m$ are the corresponding synaptic weights. The total number of FIT neural units generated from a neural network with $N$ neurons is $\mathcal{N} = \sum_{i=1}^{N}(m_i - 1)$, where $m_i$ is the fanin of neuron $n_i$.

### B. Weight Normalization

We apply weight normalization to optimize the synaptic weights $w_i$ of the unrolled model (Fig. 3❶) such that the firing rate of a neuron in this model is proportional to its input activation in the original model (Fig. 3❷). The weight normalization is performed for each decomposed neural unit and the weight normalization factor is applied to all its pre-synaptic weights. Using Fig 3, the weight updates are

$$w_i = \begin{cases} w_i / S^1_{\text{norm}}, & \text{for } i = 1, 2 \\ w_i / S^{i-1}_{\text{norm}}, & \text{otherwise.} \end{cases} \quad \text{and } w_{u_i, u_j} = 1/S^j_{\text{norm}} \tag{2}$$

The normalization factor is computed as the maximum activation on the corresponding synaptic weight in the original model using a batch from the training set, i.e.,

$$S^i_{\text{norm}} = \begin{cases} k \cdot \max\{a_1 + a_2\} \text{ for } i = 1 \\ k \cdot \max\{a_{i+1}\} \text{ otherwise.} \end{cases}, \tag{3}$$

where $a_i$ is the activation on the synaptic weight $w_i$ in the original model and the scaling factor $k$ is used to limit the mean firing rate of a neuron to lower energy consumption on the hardware [18]. The weight normalization overhead can be reduced by allowing non-uniform decomposition of neuron functions. This is part of our future exploration.

### C. Motivating Example

Fig. 4 provides a motivating example demonstrating 1) how the proposed technique results in a difference in the SNN mapping to hardware when compared to SpiNeMap and 2) the improvements in model quality.

This example illustrates the mapping of three neuron functions $y_1, y_2$ and $y_3$, shown in ❶ to a neuromorphic hardware consisting of $4 \times 4$ crossbars. SpiNeMap and other similar techniques will use three crossbars to implement the three functions, as shown in ❷. Since a crossbar can accommodate only a limited number of pre-synaptic connections per output neuron (4 in this example), the component $x_6 \cdot w_g$ of neuron function $y_2$ cannot be mapped to the crossbar. This may result in a degradation of the model quality.

In the proposed technique, neuron functions $y_2$, with fanin of 5 and $y_3$, with fanin of 4 are decomposed to generate homogeneous neural computation units $u_1, u_2, \cdots, u_7$ as shown in ❸. The neuron function $y_1$ is not decomposed because the proposed technique unrolls only those neuron functions that have fanin greater than 2. During packing of these units to a crossbar with a limited number of input ports, some of the decomposed units may need to be combined to generate larger units. In this example, the new computations are represented using functions $f_a, f_b, f_c$ & $f_d$, where $f_a$ and $f_c$ can be implemented on the first crossbar alongside $y_1$, while $f_b(y_2)$ and $f_d(y_3)$ are implemented on a second cluster (❹). Finally, the two generated clusters are shown in ❺.

We make the following three key observations. First, the proposed technique only uses two crossbars to implement the three neuron functionalities $y_1, y_2$ and $y_3$, one less than that used by SpiNeMap. This reduces hardware requirement and improves energy consumption. Second, it does not eliminate any component of the neuron functionality, which improves the model quality. Third, it increases the crossbar utilization.

## III. Results and Discussion

### A. Evaluation Methodology

We evaluated 10 machine learning applications, which are reported in Table I along with their baseline accuracy.

TABLE I
Applications used to evaluate our approach.

| Class | Applications | Synapses | Neurons | Topology | Accuracy |
|---|---|---|---|---|---|
| CNN | LeNet | 282,936 | 20,602 | CNN | 85.1% |
| | AlexNet | 38,730,222 | 230,443 | CNN | 90.7% |
| | VGG16 | 99,080,704 | 554,059 | CNN | 69.8 % |
| | HeartClass [19] | 1,049,249 | 153,730 | CNN | 63.7% |
| MLP | DigitRecogMLP | 79,400 | 884 | FeedForward (784, 100, 10) | 91.6% |
| | EdgeDet | 114,057 | 6,120 | FeedForward (4096, 1024, 1024, 1024) | 100% |
| | ImgSmooth | 9,025 | 4,096 | FeedForward (4096, 1024) | 100% |
| RNN | HeartEstm [20] | 66,406 | 166 | Recurrent Reservoir | 100% |
| | VisualPursuit [21] | 163,880 | 205 | Recurrent Reservoir | 47.3% |
| | DigitRecogSTDP [22] | 11,442 | 567 | Recurrent Reservoir | 83.6% |

We evaluated these applications on the DYNAP-SE neuromorphic hardware [2] with $128 \times 128$ crossbars.

### B. Hardware Requirement

Figure 5 compares the number of crossbars required by SpiNeMap and the proposed technique for each of the evaluated applications. Crossbar numbers are normalized to SpiNeMap. We make the following *four* observations.

*First*, on average, the proposed technique requires 60% fewer crossbars than SpiNeMap to execute these applications. This

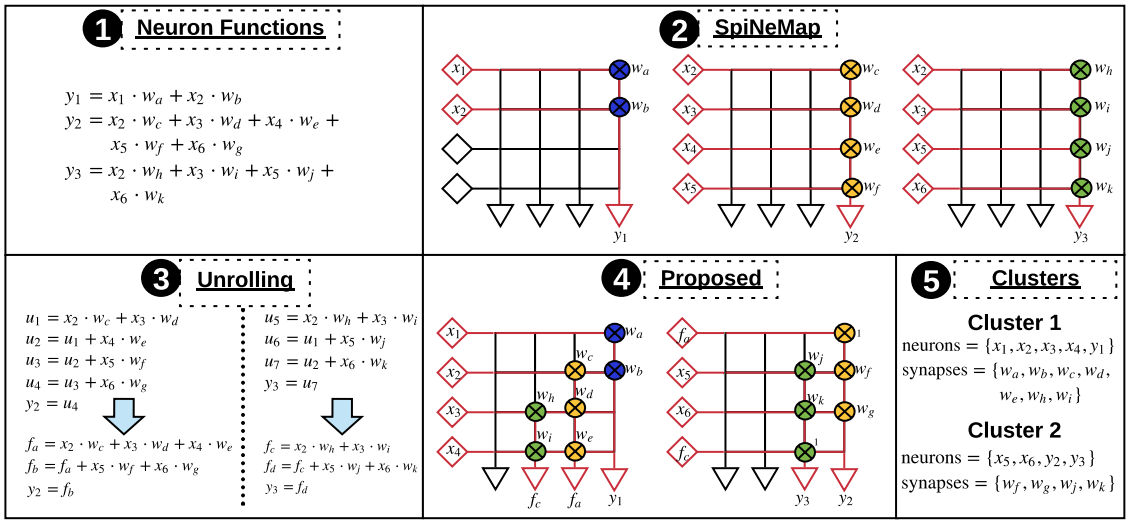


Fig. 4. Demonstration of SNN clustering using SpiNeMap [5] (❷) and the proposed approach (❹).

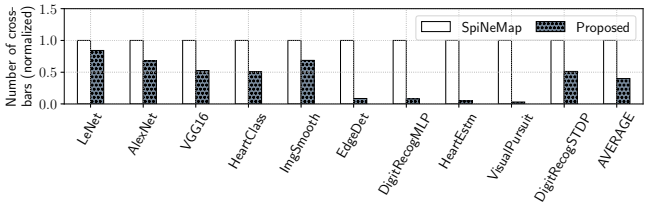


Fig. 5. Crossbars needed for the evaluated applications.

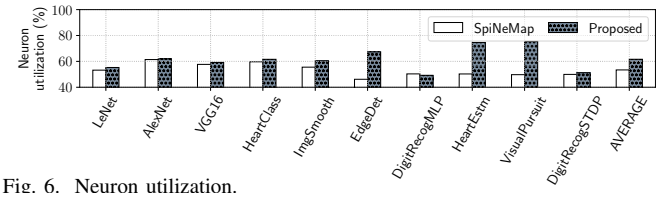


Fig. 6. Neuron utilization.

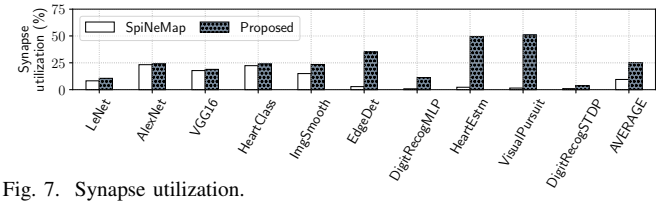


Fig. 7. Synapse utilization.

reduction is because the unrolling technique allows to densely pack the fanin synapses of different neurons into the same crossbar, reducing the overall crossbar count. This reduces the hardware energy consumption. *Second*, the number of crossbars required in the proposed technique is 10x lower than SpiNeMap for EdgeDet, even though EdgeDet has no neurons with more than 128 fanin synapses. This is because many neurons in EdgeDet have fanin close to the maximum limit that a crossbar in DYNAP-SE can accommodate. Therefore, these high-fanin neurons cannot be packed in the same crossbar by SpiNeMap, needing a separate crossbar for each of these neurons. The proposed technique, on the other hand, unrolls a high-fanin neuron to create a structure with a maximum-fanin-of-two. This results in a denser packing of crossbars and a correspondingly lower crossbar requirement. *Third*, for CNN-based applications (LeNet, AlexNet, VGG16, and HeartClass), the proposed technique requires 37% fewer crossbars than SpiNeMap. We observe that the reduction over SpiNeMap is higher for networks with more layers; the reduction is 16% for LeNet, compared to 48% for VGG16. This is because, with more layers, the proposed technique has more freedom to improve crossbar utilization (see Sec. III-C). *Finally*, for RNN-based applications (HeartEstm, VisualPursuit, and DigitRecogSTDP), the proposed approach requires 80% fewer crossbars than SpiNeMap. This is due to the cyclic nature of connections between neurons in these applications. SpiNeMap cannot optimally pack neurons in cyclic connections to the crossbars. The proposed technique decomposes a cyclic connection to pack its neurons densely into crossbars, reducing the crossbar requirement.

### C. Crossbar Utilization

Figures 6 and 7 compare respectively, the neuron and synapse utilization of SpiNeMap and the proposed technique on DYNAP-SE for each evaluated application. We observe that the average neuron utilization of SpiNeMap is 53% and of the proposed technique is 62% for these applications. The average synapse utilization of SpiNeMap is only 9% and of the proposed technique is 25% for these applications. The reason for the high crossbar utilization is because of the proposed spatial decomposition technique, which allows to densely pack fanin and fanout synapses from different neurons in the same crossbar, improving utilization.

### D. Wasted Energy

Figure 8 reports the energy wasted for each evaluated application on DYNAP-SE using the proposed technique, normalized to SpiNeMap. The wasted energy incorporates the neurons and synapses in each crossbar that are not utilized during the execution of these applications. We observe that the energy wasted using the proposed technique is on average 62% lower than SpiNeMap. This significant reduction is due to 1) the reduction in the use of crossbars and 2) the increase in the utilization of neurons and synapses in each crossbar.

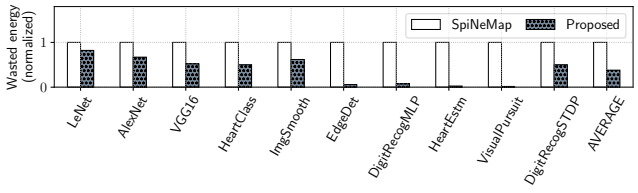


Fig. 8. Wasted energy on DYNAP-SE.

### E. Model Quality

Figure 9 plots the model quality i.e., the accuracy of each evaluated application on DYNAP-SE using SpiNeMap and the proposed technique. We make *two* key observations.

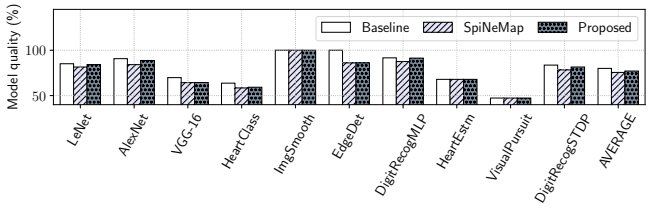


Fig. 9. Model quality for the evaluated applications.

*First*, the accuracy loss for some applications such as ImgSmooth, EdgeDet, HeartEstm, and VisualPursuit are the same for SpiNeMap and the proposed technique. This is because no neuron in these applications has more than 128 fanin synapses (see Fig. 1) and therefore, no pre-synaptic connections are eliminated by SpiNeMap. So, the quality of the two techniques are the same. However, for these applications, the proposed technique is significantly better than SpiNeMap in terms of crossbar usage, their utilization, and the wasted energy. Second, for all other applications that do not initially fit on the available crossbar space, the quality of the proposed technique is better than SpiNeMap by average 3% (between 0.8% and 4.6%). For these applications, the proposed technique is better in terms of all the evaluated metrics.

## IV. Discussion

We note that, the decomposition technique proposed here negatively impacts accuracy and energy in following two aspects. First, by decomposing a large neuron function, the proposed technique maps some of the decomposed synaptic connections on the shared interconnect of a neuromorphic hardware, negatively impacting spike latency. This can lower the model accuracy. However, the SpiNeMap framework, which integrates the decomposition technique minimizes such impact by intelligent cluster mapping and placement on the hardware [5]. Furthermore, by not eliminating any synaptic connections, the proposed technique, in fact, improves accuracy compared to SpiNeMap (see Sec. III-E). Second, with additional synapses mapped to the shared interconnect, the energy consumption on the interconnect increases. However, we find this increase in energy is much lower than the energy savings obtained by reducing the crossbar usage (see Sec. III-D).

## V. Conclusion

We present a technique to map SNN-based applications to crossbar-based neuromorphic hardware. The proposed technique involves unrolling of neurons, which decomposes a complex neuron functionality into a sequence of homogeneous neural units, where each neural unit is a fanin-of-two (FIT) neuron. The unrolling technique significantly improves crossbar utilization and ensures information integrity, resulting in no loss of model quality derived from connection pruning. We integrate this unrolling technique inside an existing SNN mapping framework and evaluate it using machine learning applications for a state-of-the-art neuromorphic hardware. Our results demonstrate an average 60% lower crossbar requirement, 9x higher synapse utilization, 62% lower wasted energy, and 3% increase in model quality compared to an existing SNN mapping approach. In the future, we will explore the trade-offs involved in non-uniform tree decomposition of neural function.

## Acknowledgment

This work is supported by the National Science Foundation Award CCF-1937419 (RTML: Small: Design of System Software to Facilitate Real-Time Neuromorphic Computing).

## References

[1] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Networks*, 1997.
[2] S. Moradi *et al.*, "A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)," *TBCAS*, 2017.
[3] A. Mallik *et al.*, "Design-technology co-optimization for OxRRAM-based synaptic processing unit," in *VLSIT*, 2017.
[4] S. Han *et al.*, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv*, 2015.
[5] A. Balaji *et al.*, "Mapping spiking neural networks to neuromorphic hardware," *TVLSI*, 2020.
[6] S. Song *et al.*, "Compiling spiking neural networks to neuromorphic hardware," in *LCTES*, 2020.
[7] A. Das *et al.*, "Mapping of local and global synapses on spiking neuromorphic hardware," in *DATE*, 2018.
[8] A. Balaji *et al.*, "A framework for the analysis of throughput-constraints of snns on neuromorphic hardware," in *ISVLSI*, 2019.
[9] A. Das *et al.*, "Dataflow-based mapping of spiking neural networks on neuromorphic hardware," in *GLSVLSI*, 2018.
[10] S. Song *et al.*, "Improving dependability of neuromorphic computing with non-volatile memory," in *EDCC*, 2020.
[11] A. Balaji *et al.*, "A framework to explore workload-specific performance and lifetime trade-offs in neuromorphic computing," *CAL*, 2019.
[12] S. Song *et al.*, "A case for lifetime reliability-aware neuromorphic computing," in *MWSCAS*, 2020.
[13] T. Titirsha *et al.*, "Thermal-aware compilation of spiking neural networks to neuromorphic hardware," in *LCPC*, 2020.
[14] T. Titirsha *et al.*, "Reliability-performance trade-offs in neuromorphic computing," in *CUT*, 2020.
[15] Y. Ji *et al.*, "Bridge the gap between neural networks and neuromorphic hardware with a neural network compiler," in *ASPLOS*, 2018.
[16] A. Balaji *et al.*, "Run-time mapping of spiking neural networks to neuromorphic hardware," *JSPS*, 2020.
[17] A. Balaji *et al.*, "PyCARL: A PyNN interface for hardware-software co-simulation of spiking neural network," in *IJCNN*, 2020.
[18] A. Balaji *et al.*, "Power-accuracy trade-offs for heartbeat classification on neural networks hardware," *JOLPE*, 2018.
[19] A. Das *et al.*, "Heartbeat classification in wearables using multi-layer perceptron and time-frequency joint distribution of ECG," in *CHASE*, 2018.
[20] A. Das *et al.*, "Unsupervised heart-rate estimation in wearables with Liquid states and a probabilistic readout," *Neural Networks*, 2018.
[21] H. J. Kashyap *et al.*, "A recurrent neural network based model of predictive smooth pursuit eye movement in primates," in *IJCNN*, 2018.
[22] P. U. Diehl *et al.*, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Front. in Comp. Neuroscience*, 2015.