Online Few-Shot Gesture Learning on a Neuromorphic Processor

Kenneth Stewart, *Graduate Student Member, IEEE*, Garrick Orchard, Sumit Bam Shrestha, and Emre Neftci, *Member, IEEE*

***Abstract*— We present the Surrogate-gradient Online Error-triggered Learning (SOEL) system for online few-shot learning on neuromorphic processors. The SOEL learning system uses a combination of transfer learning and principles of computational neuroscience and deep learning. We show that partially trained deep Spiking Neural Networks (SNNs) implemented on neuromorphic hardware can rapidly adapt online to new classes of data within a domain. SOEL updates trigger when an error occurs, enabling faster learning with fewer updates. Using gesture recognition as a case study, we show SOEL can be used for online few-shot learning of new classes of pre-recorded gesture data and rapid online learning of new gestures from data streamed live from a Dynamic Active-pixel Vision Sensor to an Intel Loihi neuromorphic research processor.**

***Index Terms*— Neuromorphic computing, spiking neural networks, on-chip learning, few-shot learning, online learning.**

I. Introduction

THE current generation of Artificial Neural Networks (ANNs) achieve state of the art performance in applications ranging from image classification and object recognition, to object tracking, signal processing, natural language processing, self driving cars, health care diagnostics, and many more [1]. ANNs mainly rely on the backpropagation of errors as the key to their learning prowess, but they require large amounts of data and memory for training. Training these networks relies on GPUs and thousands of iterations over the data sampled in an i.i.d. fashion. To achieve the high throughput necessary to quickly train the networks, GPUs rely on high volumes of data movement and tensor-based computations, both of which consume large amounts of energy [2], [3], making GPU less than ideal for implementation at scale in mobile systems.

Manuscript received July 9, 2020; revised September 22, 2020; accepted October 12, 2020. Date of publication October 19, 2020; date of current version December 11, 2020. This work was supported in part by the Intel Corporation (KS, EN), in part by the National Science Foundation under Grant 1652159 (EN), and in part by the Programmatic Grant A1687b0033 from the Singapore government's Research, Innovation and Enterprise 2020 Plan (Advanced Manufacturing and Engineering Domain) (SBS). This article was recommended by Guest Editor T. Serrano-Gotarredona. *(Corresponding author: Kenneth Stewart.)*

Kenneth Stewart is with the Department of Computer Science, University of California Irvine, Irvine, CA 92697-2625 USA (e-mail: kennetms@uci.edu).

Garrick Orchard is with the Intel Labs Intel Corporation, Santa Clara, CA 95054-1549 USA (e-mail: garrick.orchard@intel.com).

Sumit Bam Shrestha is with the Institute for Infocomm Research, A*STAR, Singapore 138632 (e-mail: sumit_bam@i2r.a-star.edu.sg).

Emre Neftci is with the Department of Cognitive Sciences, Department of Computer Science, University of California Irvine, Irvine, CA 92697-2625 USA (e-mail: eneftci@uci.edu).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JETCAS.2020.3032058

Neuromorphic computing platforms offer an energy-efficient alternative to perform training and inference in neural networks while being suitable for power-constrained applications in mobile systems [4]. Neuromorphic systems mimic the brain's event-driven dynamics, distributed architecture and massive parallelism to overcome the limitations of conventional von Neumann computing architectures [5]. Neuromorphic hardware equipped with synaptic plasticity capability can perform training and inference online, using local information [6], [7], making them particularly interesting for problems requiring fast adaptation to new data. Despite the many technical advances in neuromorphic learning hardware, the practical role of learning and synaptic plasticity in neuromorphic hardware has remained elusive. This is because gradient-based learning is notoriously slow, requiring many iterations to achieve an acceptable generalization. Furthermore, synaptic plasticity is inherently online and local, but learning online using streaming data breaks the i.i.d. assumptions required for convergence of the neural network.

In this paper, we take a realistic and practical approach to learning in neuromorphic hardware by combining the best of conventional and neuromorphic hardware: we pre-train networks on GPUs on a class of tasks and adapt the key layers on the neuromorphic hardware. The result is a system that moves the non-local and energy-intensive phases of learning to a cloud or mainframe, and deploys on the neuromorphic hardware the data sensitive portions of the learning. This approach is realized by using recent theories that combine machine learning theory with SNNs and few-shot/transfer learning. We demonstrate our approach on fast online learning of streaming, real-world visual patterns on a neuromorphic processor. The pre-training is carried out using a functional model of an Intel Loihi neuromorphic processor. During deployment, the model is then fine-tuned on the processor using local synaptic plasticity rules. The key contribution of this work is few-shot Surrogate-gradient Online Error-Triggered Learning (SOEL), a plasticity rule compatible with neuromorphic hardware derived from gradient-descent on SNNs and its efficient implementation using signals local to the neuromorphic cores.

A. Surrogate-gradient Online Error-Triggered Learning (SOEL)

While gradient Back-Propagation (BP) is the workhorse for training nearly all deep neural network architectures, it is generally incompatible with non Von Neumann computers, including brains and neuromorphic hardware. By identifying SNNs as a type of Recurrent Neural Network (RNN), recent

studies showed two reasons for this incompatibility [8]. Firstly, the spiking neuron has a non-differentiable activation function, which prevents the gradients from flowing across the network. Secondly, the computation of local errors requires the evaluation of a global loss function, which is a spatially and temporally non-local computation.

These problems are addressed using Surrogate Gradient (SG) learning [8]. SG methods define a differentiable surrogate network to calculate weight updates in a local fashion, and formulate the updates as three-factor synaptic plasticity rules. The SG method reveals from first principles the mathematical nature of the three factors, and a learning dynamic that is temporally continuous and compatible with synaptic plasticity. The three factor rules include a pre-synaptic factor, a post-synaptic factor and an external error signal. In comparison, Spike Time Dependent Plasticity (STDP), a common synaptic plasticity model in neuroscience, contains only two factors and lacks the external signal [9]. The third factor drastically improves learning by projecting task-specific errors to the neurons [10]. In short, the SG bridges the worlds of ANNs and SNNs without simplifying assumptions on the latter.

SG methods pave the road towards neuromorphic learning machines with performances similar to deep-learning [11], while being able to learn online, with input streams, spike timing and potentially using a fraction of the energy compared to conventional computers.

While temporal continuity is a plausible property in the brain, updating a large number of weights continuously is both energetically expensive and prone to dynamical instabilities. A recent development of SG learning in spiking neurons suggested that updating at every timestep is not necessary if weight updates are triggered by task errors [12]. In such *error-triggered* learning, weight updates are made only when an error threshold is crossed. Consequently, the number of updates can be drastically reduced with a small penalty in final accuracy.

However, even the fewer error-triggered learning updates is incompatible with online learning as it can lead to catastrophic forgetting. Catastrophic forgetting occurs when the data generating process for training the neural network is non i.i.d. This problem can be generally solved by increasing the complexity of the neuron and synapses [13], [14], experience replays [15], or meta-learning and the related few-shot learning [16].

Here, we focus on the latter approach for the following reasons: Firstly, the ability to solve difficult recognition tasks using few samples is a key capability of the brain [17]. Few-shot learning is a subset of deep learning concerned with such fast adaptation in situations where prior knowledge of the task domain is available. Bringing such capability to neuromorphic hardware is a top priority for local learning and adaptation on mobile systems, such as the learning of human gestures for device control or adapting to a user's voice.

In our approach, few-shot learning consists of first pre-training a model on the class of problems of interest, and then making (presumably) few error-triggered updates to learn new but related tasks.

To achieve the error-triggered learning in neuromorphic hardware, we implement the SOEL algorithm, an extension of SG and error-triggered learning for rapid, few-shot learning. We demonstrate SOEL on the Intel Loihi Neuromorphic research chip, and capitalized on its specialized local plasticity processors to carry out the updates. SOEL is an extension of the Surrogate Gradient learning algorithm designed that fix issues of a previous implementation [18].

II. Related Work

Previous work has shown that the first layers of neural networks learn general features and learn increasingly task specific features the deeper within the network the layer is [19]. The general features learned by the first layers of a network can be transferred to other networks for task-specific training of later layers, referred to as transfer learning.

The first layers of a network are trained on one dataset to learn general features that can be transferred to a second network trained on a target task. Using the transferred features yields better generalization of the target task than without transferring the general lower layer features [19]. Transfer learning is useful for few-shot learning, *i.e.* when the target task only has a small amount of data available for training, but a similar task with a larger dataset is available. The goal of few-shot learning is to train a model to generalize from as few examples as possible [20]. Transfer learning can be used to assist the training of few-shot learning models allowing for greater generalization on a target domain from few examples for both ANNs and deep SNNs [18], [21], [22].

As discussed earlier, training deep SNNs is challenging due to a spatiotemporal credit assignment problem and non-differentiabilities. Previous work overcame the learning problem in multiple layers of SNNs with methods such as feedback alignment [23], [24], backpropagation-through-time (BPTT) [25]–[27], and spike-based backpropagation [28], [29]. The successful gradient-based training of deep SNNs usually approximate the spiking function's derivative using a surrogate activation function [8]. By being able to train deep SNNs, the ingredients of deep learning that make ANNs successful such as dropout, batch normalization, convolutions, pooling etc. can be applied to SNNs, in addition to SNNs being compatible with neuromorphic hardware. BPTT is shown to achieve state-of-the-art accuracy on certain target domains such as NMNIST and gesture recognition. However, BPTT is inherently offline as it propagates error through the unrolled network and therefore is not suited for applications where online adaptation is desired. Rather than backpropagating through the network, SOEL computes local errors between pre- and post-synaptic neurons by propagating gradients forward in time [8].

While previous work has shown increasing success in training SNNs using spike-based gradient descent on a variety of tasks, they are trained and tested offline just like ANNs and do not demonstrate online on-chip learning on neuromorphic hardware. [30] demonstrated rapid online on-chip learning using the Intel Loihi neuromorphic research chip but did not use spike-based gradient descent. To our knowledge this work is the first to demonstrate online, on-chip gradient-based learning on a neuromorphic processor. Using gestures as a

case study we show the success of rapid online learning using *SOEL* which can be used for applications that require online adaptation.

III. Background

A. Dynamic Vision Sensor

The datasets used in this work are obtained using neuromorphic sensors, namely the DVS and DAVIS cameras. Each pixel of Dynamic Vision Sensors (DVSs) quantize local relative intensity changes to generate spike events [31]. In our experiments we use data from a DVS 128, and a DAVIS 240C [32]. The IBM DvsGesture dataset used here for pre-training consists of recordings of 29 different individuals performing 10 different actions such as clapping and an unspecified gesture for a total of 11 classes. The actions are recorded using a DVS camera, an event-based neuromorphic sensor, under three different lighting conditions. The task is to classify an action sequence video. Samples from the first 23 subjects were used for training and the last 6 subjects were used for testing. The training set contains 1078 samples and the test set contains 264 samples. Each sample consists of the first 1.45 seconds of the gesture performed.

B. Neural Network Model

The neural network model follows leaky, Integrate & Fire (I&F) dynamics. The dynamics of the membrane potential U_i of a neuron i is governed by the following differential equations:

$$\begin{aligned} U_i(t) &= V_i(t) - U_{th} R_i(t) + b_i, \\ \tau_{mem} \frac{\mathrm{d}}{\mathrm{d}t} V_i(t) &= -V_i(t) + I_i(t), \\ \tau_{ref} \frac{\mathrm{d}}{\mathrm{d}t} R_i(t) &= -R_i(t) + S_i(t), \end{aligned} \tag{1}$$

with $S_i(t) = \sum_f \delta(t - t_i^f)$ representing the spike train of neuron i spiking at times t_i^f, where δ is the Dirac delta. A spike is emitted when the membrane potential reaches a threshold U_{th}.

The constant b_i represents the intrinsic excitability of the neuron. The reset mechanism is captured with the dynamics of R_i. The factors τ_{mem} and τ_{ref} are time constants of the membrane and reset dynamics, respectively. I_i denotes the total synaptic current of neuron i, expressed as:

$$\tau_{syn} \frac{\mathrm{d}}{\mathrm{d}t} I_i(t) = -I_i(t) + \sum_{j \in \mathrm{pre}} W_{ij} S_j(t), \tag{2}$$

where W_{ij} is the synaptic weights between pre-synaptic neuron j and post-synaptic neuron i. Because V_i and I_i are linear with respect to the weights W_{ij}, the dynamics of V_i can be rewritten as:

$$\begin{aligned} V_i(t) &= \sum_{j \in \mathrm{pre}} W_{ij} P_j(t), \\ \tau_{mem} \frac{\mathrm{d}}{\mathrm{d}t} P_j(t) &= -P_j(t) + Q_j(t), \\ \tau_{syn} \frac{\mathrm{d}}{\mathrm{d}t} Q_j(t) &= -Q_j(t) + S_j(t). \end{aligned} \tag{3}$$

The states P and Q describe the traces of the membrane and the current-based synapse, respectively. For each incoming spike, each trace undergoes a jump of height 1 and otherwise decays exponentially with a time constant τ_{mem} (for P) and τ_{syn} (for Q). Weighting the trace P_j with the synaptic weight W_{ij} results in the Post–Synaptic Potentials (PSPs) of neuron i caused by input neuron j.

Discrete Spike Response Model of the Neuron and Synapse Dynamics: In a digital system, the continuous dynamics above are simulated in discrete time, with time step Δt. The dynamical equations in Eq. (1) and Eq. (3) are expressed in discrete time as:

$$\begin{aligned} U_i[t] &= \sum_j W_{ij} P_j[t] - U_{th} R_i[t] + b_i, \\ S_i[t] &= \Theta(U_i[t]), \\ P_j[t + \Delta t] &= \alpha P_j[t] + (1 - \alpha) Q_j[t], \\ Q_j[t + \Delta t] &= \beta Q_j[t] + (1 - \beta) S_j[t] \end{aligned} \tag{4}$$

where the constants $\alpha = \exp(-\frac{\Delta t}{\tau_{\mathrm{mem}}})$ and $\beta = \exp(-\frac{\Delta t}{\tau_{\mathrm{syn}}})$ reflect the decay dynamics of the membrane potential U and the synaptic state I during a Δt timestep. $\Theta(U_i[t])$ is the unit step function, where $\Theta(U_i) = 0$ if $U_i < U_{th}$, otherwise 1. Note that Eq. (4) is equivalent to a discrete-time version of the Spike Response Model (SRM_0) with linear filters [33].

C. Gradient-Based Training of SNNs

A number of recent methods for training SNNs using gradient descent have recently emerged. The mathematical principle of gradient descent lies on incrementally updating the parameters in the direction opposite to the gradient of a loss function. As mentioned in the introduction, difficulties of training SNNs arise due to the spatiotemporal credit assignment problem and the non-differentiability of the activation function. The spatial credit assignment problem arises when the parameters of neurons with no direct target are trained. The temporal credit assignment arises organically due to the temporal dependencies (dynamics) of spiking neurons. For example, the effect of an input spike at a particular time affects the membrane potential of a spiking neuron in the future. The magnitude of the effect is determined by the *normalized post-synaptic response* of the synapse, *i.e.* the P traces in Eq. (4). As a consequence, during learning, the credit of the error at a given point of time must be assigned to the input synapse at some point in the past. One factor to the magnitude of this temporal credit assignment is proportional to the normalized post-synaptic response, reversed in time. An illustration of this temporal credit assignment policy is shown in Fig. 1.

Assuming a global cost function $\mathcal{L}(S^N)$ defined on the spikes S^N of the top layer and targets Y, the gradients with respect to the weights in layer l are:

$$\nabla_{W_{ij}} \mathcal{L}(S^N) = \frac{\partial \mathcal{L}(S^N)}{\partial S_i} \frac{\partial S_i}{\partial U_i} \frac{\partial U_i}{\partial W_{ij}}. \tag{5}$$

We discuss below the three factor above. For didactic reasons, we proceed first with the middle term, then the first term, then the third. The middle term is the derivative of the activation

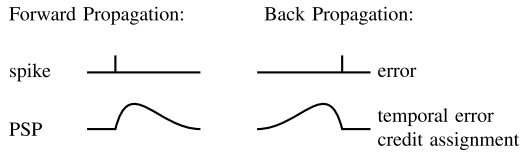


Fig. 1. Temporal credit assignment of an error at a point in time during SLAYER backpropagation.

function Θ of the spiking neuron which is non-differentiable. As discussed earlier, the SG approach consists in using a smooth surrogate function in place of the non-differentiable step function, such as the boxcar function [11], [23]. The first term on the right-hand side describes how the loss changes as the spiking states in the network, S_i, change. If the loss function is the mean-squared error and the network consists of only one layer, the first term becomes the task error $(Y_i - S_i^N)$. Computing $\frac{\partial \mathcal{L}(S^N)}{\partial S_i}$ for hidden layers is non-trivial and equivalent to solving a spatiotemporal credit assignment problem. Two methods exist to solve this problem: (1) it can be computed *offline* using gradient backpropagation on the time-unfolded graph (*i.e.* Back-Propagation-Through-Time (BPTT)), or *online* by using local loss functions [11]. This work uses a combination of offline and online SNN learning, namely SLAYER and SOEL for pre-training hidden layers and online three-factor rules for learning in output layers, respectively. In the following paragraphs, we provide further detail about these two learning methods.

1) Surrogate-gradient Online Error-Triggered Learning (SOEL) Online Training for Loihi: Online training on physical substrate requires all the information necessary for computing the gradient to be available at the synaptic plasticity processor. The first two terms of Eq. (5) discussed above are errors and postsynaptic states. The last term in Eq. (5) can be computed from Eq. (1–3) (or Eq. (4) in the discrete case). The derivative of the reset term introduces the full history of the spiking neuron, which cannot be computed locally in time. However, in low firing rate regimes, the error in omitting this term in the gradient calculation is small. By omitting the reset process, the third term becomes simply the trace P_j. Finally, we are left with the following three factors:

$$\nabla_{W_{ij}} \mathcal{L}(S) \;\; = -(Y_i - S_i)\sigma'(U_i) P_j . \tag{6}$$

Provided that pre-synaptic traces, membrane potential and errors are available at the synapse, learning can be performed locally as a synaptic plasticity rule. In computational neurosciences, rules of this type are referred to as three-factor rules [34]. Three-factor rules are consistent with biological synapse dynamics and constitute a normative theory of learning in the brain.

Eq. (6) prescribes updates at every timestep. While this is consistent with biological dynamics, it is not efficient in hardware. Updates can instead be made when the error $(Y_i - S_i^N)$ crosses a threshold, thus forming a binary "error event" [12]. This is reminiscent of STDP, where updates are triggered when pre-synaptic or post-synaptic neurons events occur [9]. Here, updates are instead triggered by error events. Error-triggered learning allows the conditional activation of the plasticity operations, which can drastically reduce the footprint of online learning. Recent work showed that the number of updates can be reduced by 20 fold for a small loss in accuracy [12]. Using a piecewise SG function, $B_i = \sigma'(U_i)$ becomes a box function where $B_i \in \{0, 1\}$. Then, the SOEL rule can be written in the following compact, three-factor form:

$$\nabla_{W_{ij}} \mathcal{L}(S^N) \propto -E_i B_i P_j . \tag{7}$$

where E_i is a integer error event for neuron i.

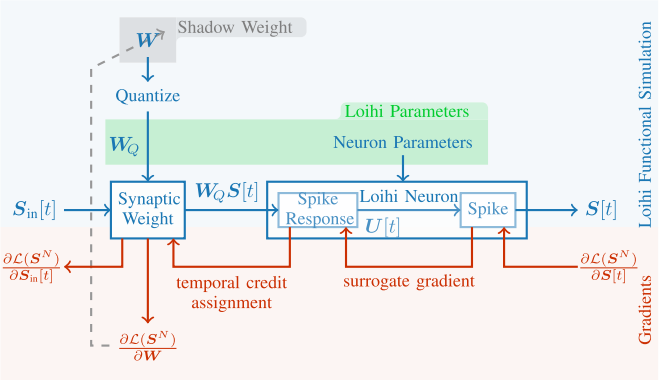


Fig. 2. Computational blocks for offline pre-training of SNN for Loihi using SLAYER. The SNN is modeled with a functional Loihi simulator and the normalized post-synaptic response is used for temporal credit assignment. Since Loihi uses integer weights full precision shadow weights are quantized and used during the forward inference phase.

2) SLAYER Offline Training for Loihi: SLAYER is a gradient computation method for training deep SNNs directly in the spiking domain [25]. It treats the inputs and outputs of the SNN as temporal signals and backpropagates the error at the output layer accordingly. There are two basic guiding principles in SLAYER: Temporal error credit assignment, and the surrogate gradient. Temporal credit assignment is done by unfolding the temporal dynamics in time and backpropagating through the unfolded graph. Further, it is typical for the normalized post-synaptic response to decay to practically zero after some time. Therefore, it is sufficient in practice to apply temporal error credit assignment only up to a finite point in history. SLAYER uses a proxy function as an approximation of spike function derivative, similar to the surrogate gradient learning described in Section III-C. These principles form the essential link in the computational graph used to calculate the gradients of the weights of the SNN and train it using standard deep learning optimization methods.

SLAYER PyTorch[1] also supports training an SNN with the CUBA leaky integrate and fire neuron model compatible with the Loihi chip. For one-to-one mapping of the trained network in Loihi hardware, the SNN is modeled with a functional Loihi simulator and the normalized post-synaptic response is used for temporal error credit assignment during backpropagation. In addition, since Loihi only supports integer weights, a strategy of full precision shadow weights [35], [36] is used, which are quantized during the forward inference phase only. The computational blocks for SLAYER-Loihi training are shown in Fig. 2.

[1] https://github.com/bamsumit/slayerPytorch

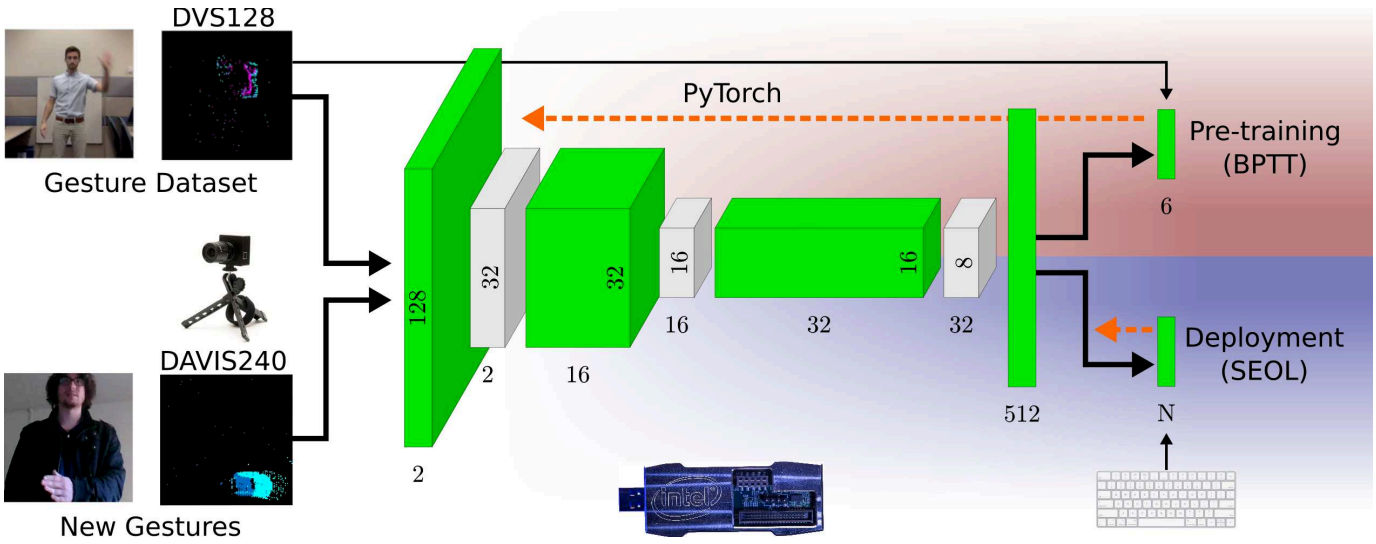


Fig. 3. Experimental setup. During a pre-training phase, the Loihi compatible convolutional network is trained on a computer using an event-based gestures dataset, the functional simulator, and SLAYER/BPTT. In this work, the pre-training dataset consisted of the IBM DVS Gestures dataset recorded using a DVS128 camera. The entire network along with quantized parameters of the functional simulation are then transferred on to the Loihi cores. During deployment, new gestures recorded using a DAVIS are streamed to an Intel Kapoho Bay. Few-shot learning is performed on the final layer using on-chip SOEL. The deployed network, including inference and training dynamics are performed on the Loihi chips. Dashed orange arrows indicate the extent of the spatial credit assignment, and thus which layers are trained in each of the two phases.

D. Intel Loihi

The Intel Loihi is a neuromorphic processor that integrates a wide range of features such as hierarchical connectivity, dendritic compartments, synaptic delays, and programmable synaptic learning rules [7]. Each Loihi chip is composed of a many core mesh comprising of 128 neuromorphic cores with each core implementing 1024 primitive spiking neural units, three embedded x86 processor cores, with an asynchronous network-on-chip (NoC) for between core communication. Loihi offers a variety of local information for programmable synaptic learning processes such as spike traces with configurable time constants that can have different time constants.

1) Plasticity Processor: Synaptic weights can be updated via a learning rule expressed as a finite-difference equation with respect to a synaptic state variable that follows a sum-of-products form as follows [7]:

$$W_{ij}[t+1] = W_{ij}[t] + \sum_k C_k \prod_l F_{kl}[t], \tag{8}$$

where W_{ij} is the synaptic weight variable defined for the destination-source neuron pair being updated; C_k is a scaling constant; and $F_{kl}[t]$ may be programmed to represent various state variables, including pre-synaptic spikes or traces, post-synaptic spikes or traces, where traces are represented as first-order linear filters. The weights are stochastically rounded according to the programmed weight precision. Traces are stochastically rounded to 7-bits of precision.

IV. Methods

We present a system for online learning of gestures from DVS data using only a few shots. Our workflow consist of a pre-training phase, followed by a deployment phase. The pre-training phase uses SLAYER and its functional Loihi simulator to train a Loihi compatible convolutional network on a GPU. For our targeted human gesture recognition application, we use the event-based IBM DVS Gestures dataset to pre-train this network. The trained network and quantized parameters are then transferred to the Loihi cores. During deployment on Loihi, few-shot learning of new gestures is performed on the top layer on-chip with SOEL. The system is shown in figure 3 and its components are detailed in the following subsections.

A. Dataset and Gesture Sampling

Visual input to the model was recorded with either a DVS 128 in the case of the IBM DVSGesture[2] dataset [37] or with a DAVIS 240C [32] in the case of real-world gestures. The network was pre-trained data recorded with a DVS 128, which has a smaller resolution compared to the more recent DAVIS 240C. During experiments involving input live-streamed from a DAVIS 240C to an Intel Kapoho Bay, data was scaled down to the same dimensions as the DVS 128 before being input into the network. Data was taken by one subject under three different lighting conditions, natural light from the sun, incandescent light, and fluorescent light which is shown in Fig. 5.

B. Neural Network Model and Offline Pre-Training Using SLAYER for Loihi

We trained a spiking CNN using SLAYER for Loihi (c.f. Section III-C.2) on the DVS Gesture dataset [37]. It has eleven output gestures, out of which six (the even classes) were used for offline training using SLAYER. The input is a 128×128 spatial event with two polarities (ON and OFF). The spiking

[2]The DVS Gesture dataset is used under a Creative Commons Attribution 4.0 license.

TABLE I
NETWORK ARCHITECTURE

Layer	Kernel	Output	Training Method
input		128×128×2	DVS128/DAVIS240C (Sensor)
1	4a	32×32×2	SLAYER (BPTT)
2	16c5z	32×32×16	
3	2a	16×16×16	
4	32c3z	16×16×32	
5	2a	8×8×32	
6	-	512	
output	-	N	SOEL

Notation: `Ya` represents `YxY` sum pooling, `XcYz` represents `X` convolution filters (`YxY`) with zero padding. N is the number of classes, which is task-dependent.

CNN architecture shown in table I consisted of 7 layers. The input spikes are OR'ed in 1 ms time bins and then fed to the network. N refers to the number of output classes, which depend on the experimental conditions.

The threshold for all the neurons were set to 80×2^6 and the current decay and voltage decay were set to 1024 (time constant of 32ms) and 128 (time constant of 4ms) respectively. The weights of the network were trained to be in the set $\{-256, -254, \cdots, 254\}$ i.e. 8 bit signed weights with step of 2.

The network was pre-trained for 2000 epochs. For better generalization performance, the input was augmented during training: x-y jitter of up to 8 pixels, rotation jitter of up to 10°, and random sampling of 1450ms spike sequence. The Nadam [38] optimizer was used with a learning rate of 0.003 and default $\beta = (0.9, 0.999)$. The network without the final fully connected output layer (layers 1–6), is the feature extraction network which is subsequently used in our on-chip learning experiments described below.

C. Online Few-Shot Learning Using Surrogate-gradient Online Error-Triggered Learning (SOEL) Plasticity for Loihi

SOEL requires the pre-synaptic trace P to be a second-order linear filter. Second-order kernels can be implemented as a subtraction of two first-order kernels [39]. This subtraction is enabled by the sum-of-products formulation of the plasticity rule (8). The error, err_i, is computed with the post-synaptic neuron using the following:

$$err_i[t] = Y - \bar{S}_i[t] \tag{9}$$

where Y is the target, $\bar{S}_i = \sum_{t-T}^{T} S_i[t]$ is defined here as the number of post-synaptic spikes by neuron i in the previous T timesteps. T is a constant number of timesteps that is a fraction of the total presentation time of the sample. This number determines the rate at which errors are computed. Using a spike-count instead of spike states is an approximation because the update will be subsequently made using the states in the final timestep, *i.e.* $P_j[t]$. However, for T smaller than the neuron and synaptic time constants, $P_j[t]$ will not vary much during this time window, and the approximation will remain close to the exact case.

Since the post-synaptic trace is not necessary for the SG rule, SOEL writes the error on the same register used for the post-synaptic trace. This enables the error value to be available in the plasticity processor for learning. On the chip, post-traces can only be positive but errors can be both positive and negative. This problem is solved by offsetting the weight updates with a constant term C.

Algorithm 1 SOEL

Result: Error-triggered Synaptic Plasticity
$\theta = 0$;
if *neuron i is learning* **then**
  $\bar{S}_i \leftarrow \bar{S}_i + S_i$;
  $err_i \leftarrow Y - \bar{S}_i$;
  if $err_i > \theta$ *or* $err_i < -\theta$ **then**
    $E_i \leftarrow C + err_i$
    $W_{ij} \leftarrow W_{ij} - \eta(E_i - C)P_j$;
    increase θ by constant;
    $\bar{S}_i \leftarrow 0$;
  else
    decrease θ by constant;
  end
end

$$E_i[t] = \begin{cases} C + err_i[t], & \text{if } err_i > \theta \text{ or } < -\theta \\ C, & \text{otherwise} \end{cases} \tag{10}$$

where θ is an error threshold.

Intuitively, SOEL can be interpreted as follows. If err is higher than θ, meaning the neuron is spiking at too high a frequency, then there is a positive error and the weight of the synapse will be penalized. Conversely if err is below $-\theta$ then the weight of the synapse weight will be increased. The term σ' (the "second factor" in Eq. (6)) cannot be implemented directly on the Loihi because the membrane state is not available at the plasticity processor. Since only the final layer N is trained, setting this term to 1 regardless of the membrane value only has the effect of continued learning even after the neuron output saturates in either direction. This strategy, referred to straight-through estimator in the machine learning field, has the disadvantage of yielding biased estimated of the gradients, but the advantage of faster learning since every error leads to an update. Since our goal is to perform fast, one-shot learning, SOEL implemented here uses a straight-through estimator. The full learning rule can then be expressed as:

$$W_{ij} = W_{ij} + \eta(E_i - C)P, \tag{11}$$

where W_{ij} is the synapse from pre-synaptic neuron j to post-synaptic neuron i, η is the learning rate, E_i is the error, and P_j is the pre-synaptic trace. The learning rule can be implemented in Intel Loihi as:

$$\begin{aligned} X_j^1[t+1] &= \alpha^1 X_j^1[t] + S_j^1, \\ X_j^2[t+1] &= \alpha^2 X_j^2[t] + S_j^2, \\ Y_i[t] &= E_i[t], \\ \Delta W_{ij} &= \eta(X_j^2[t] - X_j^1[t])(Y_i[t] - C). \end{aligned} \tag{12}$$

Here, X^2 and X^1 are pre-synaptic trace variables available in the Loihi whose subtraction in the third equation yields

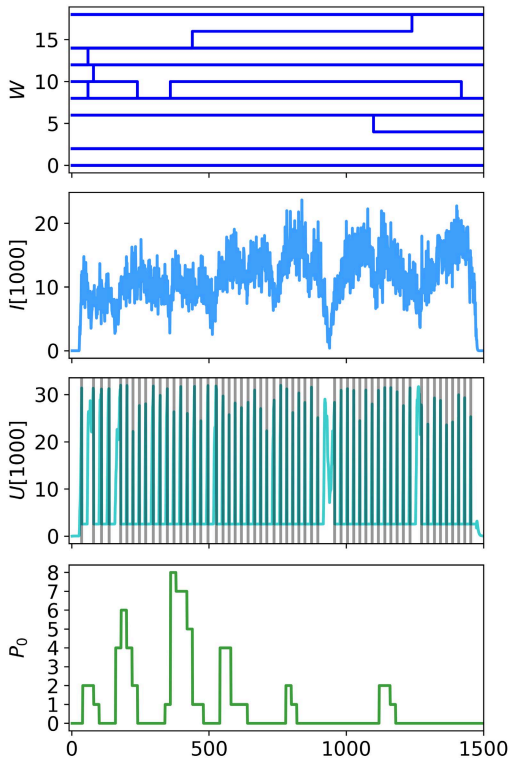


Fig. 4. Dynamics of one learning neuron when learning a new gesture. Only a subset of the synaptic weights W are shown. The weights only change when P_0 is non-zero during a learning epoch. The current I, and membrane potential U of the learning neuron are shown over the duration of the sample. Spikes are shown as grey vertical lines overlaying the membrane potential plot.

the second order kernel equivalent to P_j in Eq. (3).

$$P_j[t] \propto (X_j^2[t] - X_j^1[t]). \tag{13}$$

A Loihi Lakemont core computes the spike count $\bar{S}$ and evaluates err_i at regular intervals T. If the error exceeds the threshold θ, the post-synaptic trace value in the plasticity processor, Y, is written with the error E_i and a plasticity operation is initiated. As in Eq. (10), C is a constant bias term to account for negative error because traces cannot be negative.

D. System Specifications for Measurement [3]

SNN offline pre-training was performed with Ubuntu 16.04.6, SLAYER PyTorch commit id 598fc44, and PyTorch 1.4.0. The machine consists of an Intel Xeon E5-2630 CPU with 128GB RAM and Nvidia GeForce RTX 2080Ti GPU.

Loihi time and energy measurements were made using Ubuntu 16.04.6 with Nx SDK 0.95 and a Nahuku 32 board running on the Intel Neuromorphic Research Community (INRC) cloud. The machine consists of an Intel Xeon E5-2650 CPU with 4GB RAM.

Live gesture learning used a Kapoho Bay Loihi system connected to an IniVation DAVIS240C sensor. The host machine was an Intel Core i7-7700HQ CPU with 16GB of RAM running Ubuntu 16.04.6 and Nx SDK 0.95.

[3]All performance results are based on testing as of June 2020 and may not reflect all publicly available security updates. No product can be absolutely secure.

TABLE II
6+5-WAY FEW-SHOT CLASSIFICATION ON THE DVSGESTURE DATASET

Dataset	Learning Method	Shots	Train	Test
DVSGesture	SOEL	**1**	**96±4%**	**64.7±4.6%**
		5	**88±5.2%**	**65.1±5.1%**
		20	**87.7±2.3%**	**80.2±4.3%**
	SGD [18]	1	40%	40%
		5	60%	43.3%
		20	73.5%	56.2%
	SLAYER	1	78.5±2.6%	83.5±2.32%
		5	95.9±1%	83.5±2.9%
		20	99.7±.3%	91.2±1.9%

V. EXPERIMENTS

We used SOEL to train and test the last layer of the neural network pre-trained with SLAYER on 6 of the 11 gestures from the DVSGesture dataset, training the last layer with only a few-shots of the remaining 5 gestures of the dataset for a few-shot 6+5 way gesture classification task. The 6 refers to the 6 gestures the network is pre-trained to classify using SLAYER, and the 5 refers to the 5 new gestures we are training the last layer of models on using only a few-shots. "Train" refers to classification accuracy on training samples using saved weights with plasticity disabled. "Test" is the classification accuracy on the samples held out of the training procedure. Models were trained on one, five, or twenty shots of data and then tested on 100 held out samples. The results are obtained from performing a 5-fold cross-validation. A gesture is considered correctly classified if the desired neurons spike frequency is highest during the presentation time. We compare the accuracy of the model using SOEL to two other models, one whose last layer is trained using vanilla SGD used in [18] and another whose last layer is trained using SLAYER. Each model was trained on samples from the DVSGesture test dataset, and tested on samples from the test dataset not seen during training. Table II shows the accuracy comparisons of the different models trained on the few-shot 6+5-way gesture classification task. The results show the SOEL trained network is overall better than the vanilla SGD method from [18], achieving on average significantly better results at test time after seeing only one shot of training data, and better generalization. However while SOEL does better at training time on 1 shot experiments than the pure SLAYER network, SLAYER is better at generalizing than SOEL. This could be due to the SOEL model tending to over-fit on samples presented and the straight-through estimator. For similar reasons, we speculate that the accuracy of the SOEL model is more variable than SLAYER. When overfitting, samples that deviate too much from those samples will be more likely to be classified incorrectly, but all experiments show SOEL to be significantly better than our previous implementation [18]. We also compare the time taken and energy consumption needed for SOEL and [18] to train each gesture shown in table III. The results indicate that SOEL uses more energy but takes less time to train and achieves higher accuracy than [18]. Because

TABLE III
COMPARISON OF TIME AND ENERGY TAKEN FOR LEARNING ONE GESTURE

Measurement	SOEL	SGD [18]	% Diff
Learning Time (s)	.037	.31	-87.71%
Learning Energy (mJ)	167.58	37.04	358.46%
Learning Power (mW)	6.18	11.48	-46.17%
Total Time (s)	1.04	1.21	-14.23%
Total Energy (mJ)	481.98	323.2	49.13%
Total Power (mW)	511.65	391.07	30.83%

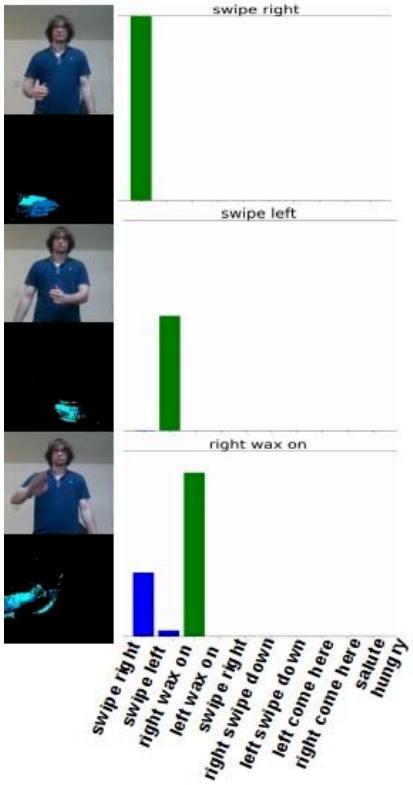


Fig. 5. Rapid online learning of gestures using data streamed from a DAVIS240C to an Intel Kapoho Bay. The upper part of the figure shows a person performing a gesture in front of a DAVIS240C, and the corresponding DAVIS240C output events shown in blue. The histogram shows the spiking frequency of each neurons response to the presented gesture after learning. After only a single one second presentation of each gesture the network can correctly classify the gestures it trained on.

the Intel Kapoho Bay does not support energy probing, energy and time measurements were taken with an Intel Nahuku board consisting of 32 Loihi chips.

A. Real-World Gesture Learning

In addition to the few-shot 6+5 way classification we also tested SOEL in a real-world gesture learning and recognition setting. To demonstrate rapid online gesture learning in a real-world setting we streamed gesture data in real time from a DAVIS 240C sensor connected to an Intel Kapoho Bay. For the experiment we tested one subject in a single lighting condition where the subject was under fluorescent light. The neural network model on the Kapoho Bay was pre-trained on all 11 gestures of the DVSGesture dataset using SLAYER, but the last layer is reset, made plastic, and trained using SOEL. The task was to train the network to classify 10 predetermined gestures outside of the DVSGesture dataset using as few shots as possible. Figure 5 shows an example of the learning and inference of the gestures. After being shown a gesture for a one second presentation, the network is able to classify other samples of the gesture. Additionally, training other gestures does not interfere with the networks ability to classify previously learned gestures. However, performance can degrade if the learned gestures spatially overlap because unique gestures within the same space may be seen as the same gesture.

The results of which some are shown in figure 5 demonstrate the capability of the SOEL learning rule to perform rapid few-shot learning on a neuromorphic processor from real-world data. A link to a video showing a live demonstration of the rapid learning of 10 new gestures is added as supplemental information.

VI. DISCUSSION AND FUTURE WORK

We presented SOEL, a new surrogate gradient based learning algorithm for few-shot online learning on an Intel Loihi neuromorphic processor using gesture recognition as a case study. To accomplish this we first pre-trained an Intel Loihi compatible SNN on a GPU using the current state-of-the-art SLAYER method, and then deployed the network on an Intel Kapoho Bay and retrained the last layer on few-shots of data using SOEL. We found that like ANNs, using a pre-trained network for transfer learning with SNNs significantly boosts few-shot learning accuracy. While we have achieved real-time online gesture learning using SOEL, there are limitations to our method. Currently, SOEL only supports training the last layer of the network. Being a local learning rule, SOEL only has information from pre- and post-synaptic neurons within its layer. Therefore training other layers will incur the spatial credit assignment because the neurons will not have a direct target to train on outside of the last layer. Consequently, if the signal is not separable in the penultimate layer then the last layer cannot learn. This can be potentially solved using layer wise local loss functions [11] and is beyond the scope of this article. Another limitation stems from the approximations made with the SOEL algorithm. First, the algorithm assumes that states do not change across the time window in which the error is calculated. This is beneficial to speed up training and can be adjusted to match the error dynamics. Second, due to limitations of the plasticity processor, the second term of the three factor rule cannot be implemented exactly and is instead ignored (set to one). These two approximations are likely to reduce the accuracy of the final result. The few-shot learning experiments using SOEL on gesture data with the Intel Loihi neuromorphic processor are slightly worse compared to training the last layer using GPU SLAYER. This discrepancy is expected since SOEL yields biased estimates of the gradients. The bias in the estimates is caused mainly by the straight-through estimator, and the approximate spike count loss which is computing using the neural states of the last time step. Furthermore, the discretization of neural and

synaptic states, and limited range of effective learning rates further widen the gap between GPU simulations and Loihi simulations. However, in the regime of interest, *e.g.* between one shot and five shots, the discrepancy remains acceptable. Furthermore, they are a major improvement from our previous work. Unlike vanilla SGD, which learns at every timestep, SOEL only learns when there is sufficient error to trigger learning. This error-triggered learning helps prevent weight saturation and catastrophic forgetting leading to increased accuracy. However the increased accuracy comes with an increase in power consumption when compared to vanilla SGD. We speculate that the power consumption for gesture recognition using SLAYER with a GPU is at least an order of magnitude higher than using SOEL with the Intel Loihi. Additionally we also showed SOEL is capable of few-shot learning from real world data. These experiments also showed SOEL was able to adapt to the differences of data taken from both a DAVIS 240 and a DVS 128 and was able to learn using data from both.

ACKNOWLEDGMENT

The preliminary experiments of this research were conducted at the Telluride Neuromorphic Cognition Engineering workshop, years 2018 and 2019 (all authors).

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[2] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, and S. Matsuoka, "Statistical power modeling of GPU kernels using performance counters," in *Proc. Int. Conf. Green Comput.*, Aug. 2010, pp. 115–122.

[3] G. Kestor, R. Gioiosa, D. J. Kerbyson, and A. Hoisie, "Quantifying the energy cost of data movement in scientific applications," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Sep. 2013, pp. 56–65.

[4] T. Hwu, J. Krichmar, and X. Zou, "A complete neuromorphic solution to outdoor navigation and path planning," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.

[5] G. Indiveri *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers Neurosci.*, vol. 5, pp. 1–23, May 2011.

[6] E. Chicca, F. Stefanini, C. Bartolozzi, and G. Indiveri, "Neuromorphic electronic circuits for building autonomous cognitive systems," *Proc. IEEE*, vol. 102, no. 9, pp. 1367–1388, Sep. 2014.

[7] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[8] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 51–63, Nov. 2019.

[9] G.-Q. Bi and M.-M. Poo, "Synaptic modifications in cultured hippocampal neurons: Dependence on spike timing, synaptic strength, and postsynaptic cell type," *J. Neurosci.*, vol. 18, no. 24, pp. 10464–10472, Dec. 1998.

[10] P. Baldi, P. Sadowski, and Z. Lu, "Learning in the machine: The symmetries of the deep learning channel," *Neural Netw.*, vol. 95, pp. 110–133, Nov. 2017.

[11] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (DECOLLE)," *Frontiers Neurosci.*, vol. 14, p. 424, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2020.00424, doi: 10.3389/fnins.2020.00424.

[12] M. Payvand, M. E. Fouda, F. Kurdahi, A. Eltawil, and E. O. Neftci, "Error-triggered three-factor learning dynamics for crossbar arrays," in *Proc. 2nd IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Aug. 2020, pp. 218–222. [Online]. Available: http://arxiv.org/pdf/1910.06152

[13] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," 2017, *arXiv:1703.04200*. [Online]. Available: http://arxiv.org/abs/1703.04200

[14] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 114, no. 13, pp. 3521–3526, 2017.

[15] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[16] M. Andrychowicz *et al.*, "Learning to learn by gradient descent by gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3981–3989.

[17] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, Dec. 2015.

[18] K. Stewart, G. Orchard, S. B. Shrestha, and E. Neftci, "On-chip few-shot learning with surrogate gradient descent on a neuromorphic processor," in *Proc. 2nd IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Sep. 2020, pp. 223–227. [Online]. Available: http://arxiv.org/pdf/1910.04972

[19] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.

[20] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," 2016, *arXiv:1606.04080*. [Online]. Available: http://arxiv.org/abs/1606.04080

[21] S. Qiao, C. Liu, W. Shen, and A. Yuille, "Few-shot image recognition by predicting parameters from activations," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7229–7238.

[22] T. Scott, K. Ridgeway, and M. C. Mozer, "Adapted deep embeddings: A synthesis of methods for k-shot inductive transfer learning," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2018, pp. 76–85. [Online]. Available: http://papers.nips.cc/paper/7293-adapted-deep-embeddings-a-synthesis-of-methods-for-k-shot-inductive-transfer-learning.pdf

[23] E. O. Neftci, C. Augustine, S. Paul, and G. Detorakis, "Event-driven random back-propagation: Enabling neuromorphic deep learning machines," *Frontiers Neurosci.*, vol. 11, p. 324, Jun. 2017.

[24] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Commun.*, vol. 7, no. 1, Dec. 2016, Art. no. 13276.

[25] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 1412–1421.

[26] B. Yin, F. Corradi, and S. M. Bohté, "Effective and efficient computation with multiple-timescale spiking recurrent neural networks," 2020, *arXiv:2005.11633*. [Online]. Available: https://arxiv.org/abs/2005.11633

[27] D. Huh and T. J. Sejnowski, "Gradient descent for spiking neural networks," 2017, *arXiv:1706.04698*. [Online]. Available: http://arxiv.org/abs/1706.04698

[28] J. C. Thiele, O. Bichler, and A. Dupret, "SpikeGrad: An ANN-equivalent computation model for implementing backpropagation with spikes," 2019, *arXiv:1906.00851*. [Online]. Available: http://arxiv.org/abs/1906.00851

[29] C. Lee, S. S. Sarwar, P. Panda, G. Srinivasan, and K. Roy, "Enabling spike-based backpropagation for training deep neural network architectures," *Frontiers Neurosci.*, vol. 14, p. 119, Feb. 2020. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/32180697

[30] N. Imam and T. A. Cleland, "Rapid online learning and robust recall in a neuromorphic olfactory circuit," *Nature Mach. Intell.*, vol. 2, no. 3, pp. 181–191, Mar. 2020, doi: 10.1038/s42256-020-0159-4.

[31] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128×128 120 dB 15 us latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb. 2008.

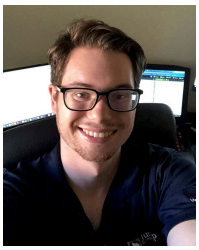
[32] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240×180 130 db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE J. Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct. 2014.

[33] W. Gerstner and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.

[34] W. Gerstner, M. Lehmann, V. Liakoni, D. Corneil, and J. Brea, "Eligibility traces and plasticity on behavioral time scales: Experimental support of NeoHebbian three-factor learning rules," *Frontiers Neural Circuits*, vol. 12, p. 53, Jul. 2018.

[35] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 6869–6898, 2017.

[36] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.
[37] A. Amir *et al.*, "A low power, fully event-based gesture recognition system," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7243–7252.
[38] T. Dozat, "Incorporating Nesterov momentum into adam," in *Proc. ICLR Workshop*, 2016, pp. 1–4.
[39] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge, U.K.: Cambridge Univ. Press, 2014.



Kenneth Stewart (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree with the University of California Irvine. His current research focuses on developing learning algorithms for neuromorphic hardware and their application to areas such as computer vision and robotics. His research interests include neuromorphic computing, online learning, robotics, artificial intelligence, and applications thereof.

Garrick Orchard received the Ph.D. degree from Johns Hopkins University in 2012. He joined the newly formed Singapore Institute for Neurotechnology (SINAPSE), National University of Singapore, as a Research Scientist. In 2015, he was awarded the Temasek Research Fellowship from the Singapore Ministry of Defence. In 2019, he joined Intel's Neuromorphic Computing Laboratory as a Senior Researcher, focusing on sensing and perception.

Sumit Bam Shrestha received the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, under SINGA Scholarship. He is currently a Research Scientist with the Institute for Infocomm Research (I2R), Agency of Science Technology and Research (A*STAR), Singapore, where he co-leads the Algorithm Development for Neuromorphic Computing Programme. His research is mainly focused on deep spiking neural networks, including neuromorphic computing, neuromorphic vision, neural networks, and machine learning.

Emre Neftci (Member, IEEE) received the M.Sc. degree in physics from Ecole Polytechnique Federale de Lausanne, Switzerland, and the Ph.D. degree from the Institute of Neuroinformatics, University of Zurich and ETH Zurich, in 2010. He is currently an Assistant Professor with the Department of Cognitive Sciences and Computer Science, University of California, Irvine, CA, USA. His current research explores the bridges between neuroscience and machine learning, with a focus on the theoretical and computational modeling of learning algorithms that are best suited to neuromorphic hardware and non-von Neumann computing architectures.