Synaptic Plasticity Dynamics for Deep Continuous Local Learning (DECOLLE)

Jacques Kaiser `jkaiser@fzi.de`,
FZI Research Center For Information Technology, Karlsruhe, Germany

Hesham Mostafa `hesham@ucsd.edu`,
Department of Bioengineering, University of California San Diego, La Jolla, USA

Emre Neftci `eneftci@uci.edu`,
Department of Cognitive Sciences, Department of Computer Science, University of California Irvine, Irvine, USA

Abstract

A growing body of work underlines striking similarities between biological neural networks and recurrent, binary neural networks. A relatively smaller body of work, however, discusses similarities between learning dynamics employed in deep Artificial Neural Network and synaptic plasticity in spiking neural networks. The challenge preventing this is largely caused by the discrepancy between the dynamical properties of synaptic plasticity and the requirements for gradient backpropagation. Learning algorithms that approximate gradient backpropagation using locally synthesized gradients can overcome this challenge. Here, we show that synthetic gradients enable the derivation of Deep Continuous Local Learning (DECOLLE) in spiking neural networks. DECOLLE is capable of learning deep spatio-temporal representations from spikes relying solely on local information. Synaptic plasticity rules are derived systematically from user-defined cost functions and neural dynamics by leveraging existing autodifferentiation methods of machine learning frameworks. We benchmark our approach on the MNIST and the event-based neuromorphic DvsGesture dataset, on which DECOLLE performs comparably to the state-of-the-art. DECOLLE networks provide continuously learning machines that are relevant to biology and supportive of event-based, low-power computer vision architectures matching the accuracies of conventional computers on tasks where temporal precision and speed are essential.

1 Introduction

Understanding how the plasticity dynamics in multilayer biological neural networks is organized for efficient data-driven learning is a long-standing question in computational neurosciences [27, 23, 5]. The generally unmatched success of deep learning algorithms in a wide variety of data-driven tasks prompts the question of whether the ingredients of their success are compatible with their biological counterparts, namely Spiking Neural Networks (SNNs). Biological neural networks distinguish themselves from Artificial Neural Networks (ANNs) by their continuous-time dynamics, the locality of their operations [2], and their spike(event)-based communication. Taking these properties into account in a neural network is challenging, as the spiking nature of the neurons' non-linearity makes it non-differentiable, the continuous-time dynamics raises a temporal credit assignment problem and the assumption of computations being local to the neuron disqualifies the use of Back-Propagation-Through-Time (BPTT).

Preprint. Under review.

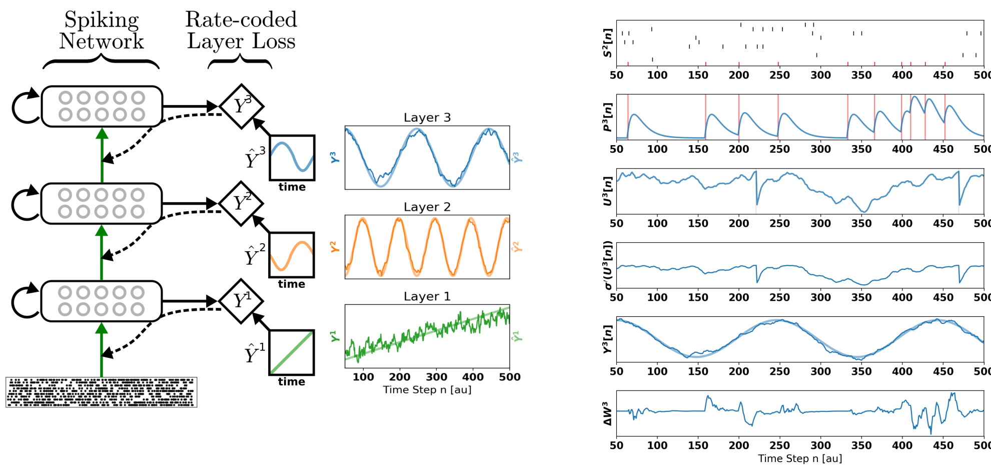

Figure 1: Deep Continuous Local Learning (DECOLLE). (Left) Each layer consists of spiking neurons with continuous dynamics. Each layer feeds into a local classifier through fixed, random connections (diamond-shaped, y). The classifier is trained to produce auxiliary targets $\hat{Y}$. Errors in the local classifiers are propagated through the random connections to train weights coming into the spiking layer, but no further (curvy, dashed line). To simplify the learning rule and enable linear scaling of the computations, the cost function is formulated using a rate code. The state of the spiking neurons (membrane potential, synaptic states, refractory state) is carried forward in time. Consequently, even in the absence of recurrent connections, the neurons are stateful in the sense of recurrent neural networks. (Right) Snapshot of the neural states illustrating the DECOLLE learning rule in the top layer. In this example, the network is trained to produce three time-varying pseudo-targets $\hat{Y}^1$, $\hat{Y}^2$, $\hat{Y}^3$.

In this article, we describe Deep Continuous Local Learning (DECOLLE), a SNN model with plasticity dynamics that solves the three problems above, and that performs at proficiencies comparable to that of small deep neural networks. DECOLLE uses layerwise local classifiers [17], which enables gradients to be computed locally. To tackle the temporal dynamics of the neurons, we use a recently established equivalence between SNNs and recurrent ANNs [18]. This equivalence defines a computational graph of the SNN, which can be implemented with standard machine learning frameworks. Contrary to BPTT, DECOLLE is formulated in a way that the information needed to compute the gradient is propagated forward, making the plasticity rule temporally local. Existing rules of this sort require dedicated state variables for every synapse, thus scaling at least quadratically with the number of neurons [26, 27]. In contrast, DECOLLE scales linearly with the number of neurons. This is achieved using a local rate-based cost function reminiscent of readout mechanisms used in liquid state machines [15], but where the readout is performed over a fixed random combination of the neuron outputs. Our approach can be viewed as a type of synthetic gradient, a technique used to decouple one or more layers from the rest of the network to prevent layerwise locking [11]. Although synthetic gradients usually involve an outer loop that is equivalent to a full Back-Propagation (BP) through the network, DECOLLE instead relies on the initialization of the local random classifier weights and forgoes the outer loop.

Conveniently, DECOLLE can leverage existing autodifferentiation tools of modern machine learning frameworks. Its linear scalability enables the training of hundreds of thousands of spiking neurons on a single GPU, and continual learning on very fine time scales. We demonstrate our approach on the classification of gestures, IBM DvsGesture dataset [1], recorded using an event-based neuromorphic sensor and report comparable performance to deep neural networks and even networks trained with BPTT.

2 Related Work

SuperSpike employs a surrogate gradient descent to train networks of Linear Integrate & Fire (LI&F) neurons on a spike train distance measure. Because the LI&F neuron output is non-differentiable, SuperSpike uses a surrogate network with differentiable activation functions to compute an approximate gradient. The authors show that this learning rule is equivalent to a forward-propagation of errors using eligibility traces, and is capable of efficient learning in hidden layers of feedforward multilayer networks. However, the approximations used in SuperSpike prevent efficient learning in deep layers. Furthermore, because the eligibility traces need to be computed for every trainable weight, the algorithm scales temporally and spatially as $O(N^2)$, where N is the number of neurons. While the complex biochemical processes at the synapse could account for the quadratic scaling, it prevents an efficient implementation in digital hardware. Like SuperSpike, DECOLLE uses surrogate gradients to perform weight updates, but as discussed below, the cost function is local and rate-based, such that only one eligibility trace per input neuron is required. This enables the algorithm to scale spatially as $O(N)$.

The local classifier in DECOLLE acts like an encoder-decoder layer in the flavor of linear readouts in reservoir type networks [8, 15, 23]. The key difference with DECOLLE is that the encoder weights are trained, whereas the decoder (readout) weights are random and fixed. This is the opposite of reservoir networks where the encoder is typically random and fixed and the decoder is trained. The training of the encoder weights allows the network to learn representations that are amenable as easier to classify inputs to subsequent layers.

Spiking neural networks can be viewed as a subclass of binary, recurrent ANNs. Spiking neurons are recurrent in the ANN sense even if all the connections are feed-forward because the neurons maintain a state that is propagated forward at every time step [18]. Binary neural networks, where both activations and/or weights are binary were studied in deep learning as a way to decrease model complexity during inference [6, 20].

Surrogate-gradient descent and forward propagation of the traces for computing gradients (as in DECOLLE and SuperSpike) is the flip-side of BPTT, where gradients are computed using past activities [18]. BPTT for SNNs was investigated in [4, 12, 21, 3, 10]. While these approaches provide an unbiased estimation of the gradients, DECOLLE can perform equally or better than these techniques using considerably less computational resources for training (*e.g.* networks sizes in [21] were limited by GPU memory). The computational and memory demands are high for BPTT because the entire sequence must be stored to compute the gradients exactly. Although the truncation of the sequences can mitigate this problem, it is not adequate when discretizing continuous-time networks, such as the SNN [18]. Furthermore, forward-propagation techniques such as DECOLLE can be formulated as local synaptic plasticity rules, and are thus amenable to implementation in dedicated, event-based (neuromorphic) hardware [7] and compatible with neurobiology.

Decoupled Neural Interfaces (DNI) were proposed to mitigate layerwise locking in training deep neural networks [11]. Layerwise locking occurs when the computations in one layer are locked until the quantities necessary for the weight update become available. In DNI, this decoupling is achieved using a synthetic gradient, a neural network that estimates the gradients for a portion of the network. In an inner loop, the network parameters are trained using the synthetic gradients, and in an outer loop, the synthetic gradient network parameters are trained using a full BP step. The gradient computed using local errors in DECOLLE described below can be viewed as a type of synthetic gradient, which ignores the outer loop to avoid a full BP step. Although ignoring the outer loop limits DECOLLE's cross-layer feature adaptation, we find that the network performs strikingly well nevertheless.

This work builds on a combination of SuperSpike and local errors with the realization that a rate-based, instantaneous cost function combined with a differentiable spike-to-rate decoder can exploit temporal dynamics of the spiking neurons while considerably reducing the computational requirements compared to a temporally global loss.

3 Methods

3.1 Neuron and Synapse Model

The neuron model used for DECOLLE is a Linear Integrate & Fire (LI&F) neuron. The dynamics of a LI&F post-synaptic neuron i can be described in discrete-time by the following equations:

$$\begin{aligned} U_i^l[n] &= \sum_j \underbrace{W_{ij}^l P_j^l[n]}_{\text{PSP}} \underbrace{-\delta R_i^l[n]}_{\text{Reset}} + b_i^l, \qquad & P_j^l[n+1] &= \alpha P_j^l[n] + Q_j^l[n], \\ & & Q_j^l[n+1] &= \beta Q_j^l[n] + S_j^{l-1}[n], \\ S_i^l[n] &= \Theta(U_i^l[n]), & R_j^l[n+1] &= \gamma R_j^l[n] + S_j^{l-1}[n], \end{aligned} \tag{1}$$

where $U_i^l[n]$ is the membrane potential of neuron i at layer l at time step n. The constants $\alpha = \exp(-\frac{\Delta t}{\tau_{\text{mem}}})$, $\gamma = \exp(-\frac{\Delta t}{\tau_{\text{ref}}})$ and $\beta = \exp(-\frac{\Delta t}{\tau_{\text{syn}}})$ capture the decay dynamics of the membrane potential U, the refractory (resetting) state R and the synaptic state Q during a Δt timestep (the τ are respective times constants). States P and Q describe the traces of the membrane and the current-based synapse, respectively (also called eligibility traces). For each incoming spike, the trace Q undergoes a jump of height W_{ij} and otherwise decays exponentially with a time constant τ_{syn}. When weighted with W^l, the feedforward weights, $W_{ij}^l P_j^l[n]$ become the Post–Synaptic Potentials (PSPs) on neuron i, layer l caused by input neuron j, layer $l-1$ at time n. The step function Θ ensures that the neurons spike ($S_i^l[n] = 1$) when $U_i^l[n] = 0$. R is a refractory state that resets and inhibits the neuron after it has emitted a spike, and δ is the constant that controls its magnitude. The bias term b_i^l above can be interpreted as a constant current injection or trainable excitability of the neuron. Note that Eq. (1) is equivalent to a discrete-time version of the Spike Response Model (SRM)$_0$ with linear filters [9].

3.2 Deep Continuous Local Learning (DECOLLE)

Exact gradients cannot be computed in SNN due to their all-or-none behavior and locality [18]. Instead, parameters updates in DECOLLE are based off a differentiable but slightly different version of the task-performing network. This approach is called surrogate gradient-based learning [18, 27]. The surrogate network used to compute the gradients is differentiated as a special case of Real-Time Recurrent Learning [26], as follows. Assuming a global cost function $\mathcal{L}$, the gradients with respect to the weights are formulated as

$$\frac{\partial}{\partial W_{ij}^l}\mathcal{L} = \frac{\partial}{\partial S_i^l}\mathcal{L}\frac{\partial}{\partial U_i^l}S_i^l\frac{\partial}{\partial W_{ij}^l}U_i^l \tag{2}$$

We discuss below the calculation of each of the three terms above from right to left in the context of SNNs.

Calculation of $\frac{\partial}{\partial W_{ij}^l}U_i^l$: The refractory term R introduces a recurrent dependency that is costly to compute. Here, we ignored the refractory term R in the gradient to simplify the rule. R prevents the neuron from reaching high firing rates. To enforce a low firing rate, we use an activity-dependent regularizer strong enough that the contribution of the refractory term to the gradient becomes negligible [27]. The feedforward connectivity ensures that P_j^l is independent of W_{ij}^l. With these assumptions $\frac{\partial}{\partial W_{ij}^l}U_i^l[n] = P_j^l[n]$.

Calculation of $\frac{\partial}{\partial U_i^l}S_i^l$: This factor is problematic because $S_i^l = \Theta(U_i^l)$, and Θ is not differentiable. A well-established method to sidestep this problem is to assume a smooth surrogate function for the purposes of optimization, *e.g.* $\frac{\partial}{\partial U_i^l}S_i^l \cong \sigma'(U_i)$ where σ' is the derivative of a sigmoid function [18].

Calculation of $\frac{\partial}{\partial S_i^l}\mathcal{L}$: This factor captures the backpropagated errors, *i.e.* the credit assignment. It generally involves non-local terms, including the activity of other neurons, their errors, and their history. While an increasing body of work is showing that approximations to the backpropagated errors are possible, for example in feedback alignment [14], how to maintain their history efficiently remains a challenging problem. SuperSpike [27] deals with it by explicitly computing this history at the synapse. In the exact form, this results in $N+1$ nested convolutions for a network of N layers, which is computationally inefficient. Although the number of convolutions can be reduced to 2 using a straight-through estimator, this approach does not scale well when two or more layers are used, and the 2 nested convolutions lead to a quadratic scaling in the number of state variables (eligibility traces) stored and updated at every time step.

The approach we follow instead is to enforce locality by using local gradients, or equivalently, local errors. One difficulty in defining a local error signal at a neuron in a deep layer is that the cost function is almost always defined using the network output at the top layer. Thus, using local information only, a neuron in a deep layer cannot infer how a change in its activity will affect the top-layer cost. To address this conundrum, [17] attaches random local classifiers to deep layers and defines auxiliary cost functions using their output. These auxiliary cost functions provide a task-relevant source of error for neurons in deep layers. Surprisingly, training deep layers using auxiliary local errors that minimize the cost at the local classifiers still allows the network as a whole to reach a small top-layer cost. That is because minimizing the local classifiers' cost puts pressure on deep layers to learn useful task-relevant features that will allow the random local classifiers to solve the task. Moreover, each layer builds on the features of the previous layer to learn even better features for its local random classifier. Thus, even though no error information propagates downwards through the layer stack, the layers indirectly learn useful hierarchical features that end up minimizing the cost at the top layer.

The DECOLLE rule combines SuperSpike with deep local learning described above to solve the temporal and spatial credit assignment problem in continuous (spiking) neural networks. To achieve this, we organize N layers of spiking neurons, and train each layer to predict a pseudo-target using a random local classifier $Y_i^l = \sum_j G_{ij}^l \sigma(U_j^l)$, where G_{ij}^l are fixed, random matrices (one for each layer l). The global loss function is the sum of the layerwise loss functions, *i.e.* $\mathcal{L} = \sum_{l=1}^N L^l$.

DECOLLE Rule: Finally the rule weight governing the weight parameter updates becomes:

$$-\Delta W_{ij}^l \propto \frac{\partial}{\partial W_{ij}^l}\mathcal{L} = (\frac{\partial}{\partial S_i^l} L^l)\sigma'(U_i^l[n])P_j^l[n] \tag{3}$$

We note that the gradient of the loss, L^l, is only used to update W^l and does not backpropagate further through the network (see orange arrows (Fig. 2)). In the special case of the Mean Square Error (MSE) loss for layer l, $L^l[n] = \frac{1}{2}\sum_i \left(Y_i^l - \hat{Y}_i^l\right)^2$ where $\hat{Y}_i^l$ is the pseudo-target for layer l, and $\frac{\partial}{\partial S_i}^l L^l = error_i[n]$. where $error_i[n] = Y_i^l - \hat{Y}_i^l$. The variables P and U required for computing the weight updates are local and readily available from the forward dynamics. With the error computed locally, DECOLLE does not need to store any additional intermediate variables, *i.e.* space requirement for the parameter update computation scales linearly with the number of neurons. Only one trace P and Q per input is maintained, as opposed to each input-output pair in [27]. The computational cost of the weight update is the same as the Widrow-Hoff rule (one product per connection, see Eq. (3)). This makes DECOLLE significantly cheaper to implement compared to BPTT for training SNN, *e.g.* SLAYER [21] which scales spatially as $O(NT)$, where T is the number of timesteps.

The locality of DECOLLE makes it compatible with known biological constraints. In fact, Eq. (3) consists of three types of factors, one modulatory ($error_i[n]$), one post-synaptic ($\sigma'(U_i^l[n])$) and pre-synaptic ($P_j^l[n]$). These types of rules are often termed three-factor rules, which have been shown to be consistent with biology [19] while being compatible with a wide number of unsupervised, supervised and reinforcement learning paradigms [25].

Computational Graph and Implementation using Automatic Differentiation: Perhaps one of the strongest points of DECOLLE is its out-of-the-box compatibility with automatic differentiation tools provided by modern machine learning frameworks. As shown in the computational graph of the SNN (Fig. 2), it can be implemented similarly to a standard recurrent neural network. The rule is temporally BP-free since the information necessary for computing the gradient (P, Q and U) are automatically propagated forward for the neuron dynamics. This enables gradients to be computed and applied within each time step n, improving sample-efficiency. Automatic differentiation thus computes the spatial gradients, locally, for each layer. This enables the integration of DECOLLE in machine learning frameworks *without* BPTT. With the integration in machine learning frameworks, one can build large convolutional neural networks, as well as leverage any type of layer, operation,

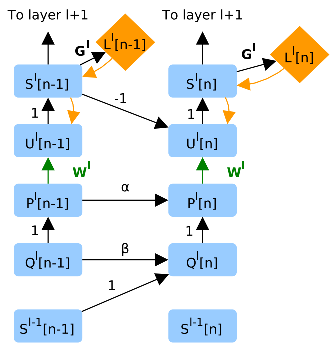

Figure 2: The unfolded computational graph of a feedforward SNN, *e.g.* Eq. (1). Time flows to the right and only temporal dependencies between timestep $n-1$ and n are shown here. Green edges indicate variables trained in the presented version of DECOLLE. Orange edges indicate the flow of the gradients. Note that this graph is similar to that of a simple Recurrent Neural Networks. To avoid clutter, the graph is presented for $\gamma = 0$ (i.e. the node for R has been omitted).

optimizer and cost function provided by the software. We leverage this integration in all our experiments under the Experiments section.

From a technological point of view, SNNs are interesting when the spike rate is low (typically 4%) as dedicated neuromorphic hardware can directly exploit this sparsity to reduce computations by the same factor [16, 7]. To ensure reasonable firing rates and prevent sustained firing, we use two regularizers. One keeps U below to the firing threshold on average, and one activity regularizer enforces a minimum firing rate in each layer. The final loss function is:

$$\mathcal{L}_g = \sum_l L^l + \lambda_1 \langle ReLU(U_i^l + .01) \rangle_i + \lambda_2 ReLU(.1 - \langle U_i^l \rangle_i) \tag{4}$$

where $\langle \cdot \rangle_i$ denotes averaging over index i and λ_1, λ_2 are hyperparameters. We use a symmetric σ centered on Θ, *i.e.* such that $\sigma(0) = .5$. In all our experiments, weight updates are made for each time step of the simulation. We use on Adamax optimizer with parameters $\beta_1 = 0$, $\beta_2 = 95$ and learning rate 10^{-9}, and a smooth L1 loss. Biases were used for all layers and trained in all DECOLLE layers. The weights G^l used for the local classifiers were initialized uniformly. The neural and synaptic time constants τ_{mem} and τ_{syn} were drawn randomly from a uniform distribution between $5-35$ms and $5-10$ms, respectively and $\tau_{ref} = 2.86$ms Pytorch code and a tutorial are publicly available on Github[1]. DECOLLE is simulated using mini-batches to leverage the GPU's parallelism.

4 Experiments

4.1 Regression with Poisson Spike Trains

To illustrate the inner workings of DECOLLE, we first demonstrate DECOLLE in a regression task. A three-layer fully connected network consisting of 512 neurons each is stimulated with a frozen 500ms Poisson spike train. The network is trained with pseudo-targets $\hat{Y}^1$, a ramp function; $\hat{Y}^2$, a high-frequency sinusoidal function and $\hat{Y}^3$, a low-frequency sinusoidal function for each layer, respectively. (Fig. 1) illustrates the states of the neuron. For illustration purposes, the recording of the neural states was made in the absence of parameter updates (*i.e.* the learning rate is 0). The refractory mechanism clearly resets the membrane potential after the neuron spikes ($U[n]$). As discussed in the methods we use regularization on the membrane potential to keep the neurons from sustaining high firing rates and an activity regularizer to maintain a minimum firing rate. Updates to the weight are made at each time step and can be non-zero when the derivative of the activation function $\sigma'(U)$ and P are non-zero. The magnitude and direction of the update are determined by the error. Note that, in effect, the error is randomized as a consequence of the random local readout. The network learned to use the input spike times to reliably produce the targets.

4.2 Poisson MNIST

We validate our model and our method on the MNIST dataset. For DECOLLE, each digit is converted into a 500ms Poisson spike train, where the mean firing rates vary from 0 to 50Hz depending on the

[1]https://github.com/nmi-lab/dcll

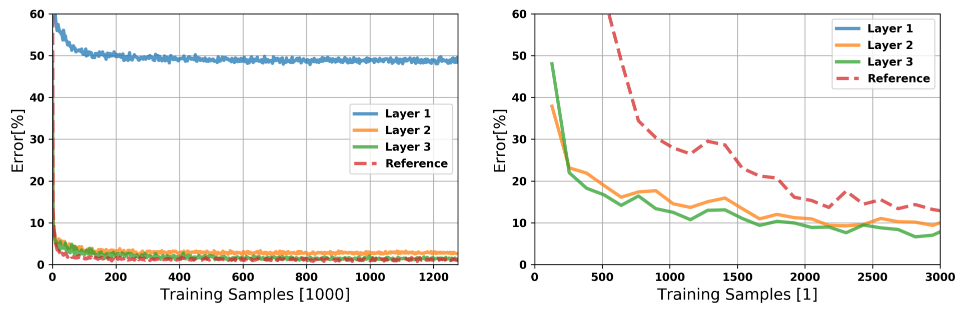

Figure 3: Classification results on the MNIST dataset for the three layers of our network against a reference analog network. (Left) Accuracy for the whole training procedure. (Right) accuracy for the first 3000 training samples.

pixel intensity. Gradient updates are performed at every simulation step after a 50ms burn-in (450 weight updates per mini-batch). Each mini-batch contains 64 samples. The test set consists of 1024 unseen samples converted into spike trains and presented for 1s to the network. We rely on a simple network architecture consisting of three convolutional layers of 16, 24 and 32 channels respectively with 7×7 kernels interleaved with max-pooling layers. For each local readout, Dropout was used to prevent over-reliance on one particular neuron. In total, the network has 3528 tunable weights and 72 biases and was trained using Adamax and on a smooth L1 loss. DECOLLE is compared to a conventional neural network with the same architecture, trained in the same regime by performing 500 BP steps for each batch. The pseudo-targets used for the local classifiers are class labels. In order to match the number of trainable parameters with the conventional convolutional network, we used one additional fully connected layer without spiking neuron dynamics.

The results on the MNIST dataset are shown in Figure 3. DECOLLE's final error samples are 1.23% for the third layer. A conventional network with the same architecture trained on standard MNIST achieves 0.91% error. However, the conventional network trained using the same regimen as DECOLLE achieves 2.0% error. DECOLLE's error is below 10% after the first 2000 samples. This is better than the reference network, and early convergence could be due to the regularizing effect of the noise added to the samples when converting them to spike trains.

4.3 DvsGesture

We test DECOLLE at the more challenging task of learning gestures recorded using a Dynamical Vision Sensor (DVS) [13]. Amir *et al.* recorded the DvsGesture dataset using a DVS, which comprises 1342 instances of a set of 11 hand and arm gestures, collected from 29 subjects under 3 different lighting conditions [1]. Unlike standard imagers, the DVS records streams of events that signal the temporal intensity changes at each of its 128×128 pixels. The unique features of each gesture are embedded in the stream of events. The event streams were downsized to 32×32 (events from 4 neighboring pixels were summed together as a common stream) and binned in frames of 1ms, the effective time step of the GPU-based simulation (Fig. 4). During training, random 500ms-long sequences were selected in mini-batches of 72 samples and presented to the network. Testing sequences were 1800ms-long, each starting from the beginning of each recording (288 testing sequences). Note that the shortest recording in the test set is 1800ms, and this duration selected to simplify and speed up the classification evaluation. The classification is obtained by counting spikes at the output starting from a burn-in period of 50ms and selecting as output class the neuron that spiked the most. Test results from the DECOLLE network are reported with the dropout layer kept active, as this provided better results. Contrary to [1], we did not use stochastic decay and the neural network structure is a three layer convolutional neural network, loosely adapted from [22]. We did not observe significant improvement by adding more convolutional layers. Most likely due to the limited depth, we find that 7×7 kernels provide much better results than the 3×3 or 5×5 kernels commonly used in image recognition. The optimal hyperparameters were found by a combination of manual and grid search. As with the MNIST case, use used the Adamx optimizer and an L1 Loss. The learning rate was divided by 5 every 1000 steps.

Model	**Error**	**Training Iterations**
DECOLLE (This Work)	$5.82 \pm 0.33\%$	Online .32M
SLAYER [21]	6.36 ± 0.49 %	Offline .27M
C3D [24]	$5.46 \pm 1.06\%$	Offline .32M
IBM EEDN (with averaging) [1]	8.23% (5.51%)	Offline 64M

Table 1: Classification error at the DvsGesture task.

Layer Type	#	Data Type	Dimensions
DVS	2	AEDAT 3.1	128×128
Downsample (Sum)	2	Integer	32×32
7×7 Conv	64	Integer	30×30
2×2 MaxPool	64	Integer	15×15
Spiking Non-linearity			
Dropout(p=.5)		Float	
Dense	11	Float	11
7×7 Conv	128	Integer	13×13
Spiking Non-linearity			
Dropout(p=.5)		Float	
Dense	11	Float	11
7×7 Conv	128	Integer	11×11
2×2 MaxPool	128	Integer	5×5
Spiking Non-linearity			
Dropout(p=.5)		Float	
Dense	11	Float	11

Table 2: DECOLLE Neural network used for the DvsGesture dataset. Note that dense layers are used for the local classifiers only and were not fed to the subsequent convolutional layers. AEDAT 3.1 is a data format used for event-based data. The spiking non-linearity was always applied after the pooling layers. Dropout layers were left active during testing.

We compared with C3D, a convolutional network commonly used for spatiotemporal classification in videos [24]. The network was similar to [24] except that is was adapted for 32x32 frames and using half of the features per layer (see SI for network layers). We note that the C3D network is deeper and wider than the DECOLLE network. We found that 16x32x32 frames, where each of the 16 representing 32ms slices of the DVS data performed best.

Overall, DECOLLE's performance is comparable or better than other published SNN implementations that use BP for training ((Tab. 1), (Fig. 4)) and close the much larger C3D network. DECOLLE reached the reported accuracies after a much smaller number of iterations and network size compared to the IBM EEDN case [1].

5 Conclusion

Understanding and deriving neural and synaptic plasticity rules that can enable hidden weights to learn is an ongoing quest in neuroscience and neuromorphic engineering. From a machine learning perspective, locality and differentiability are key issues of the spiking neuron model operations. While the latter problem is now being tackled with surrogate gradient approaches, how to achieve this in deep networks in a scalable and local fashion is still an open question.

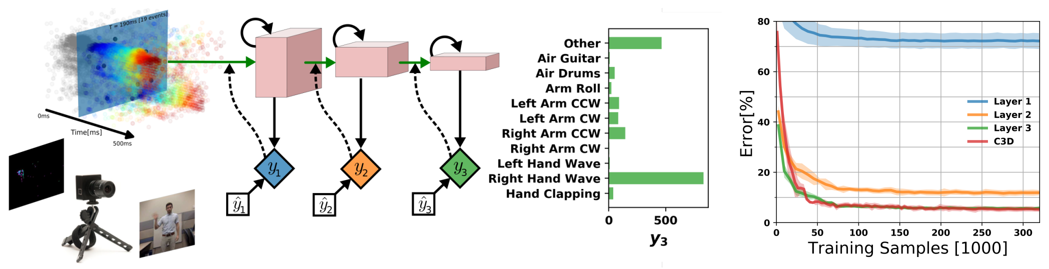

Figure 4: (Left) DECOLLE setup for DvsGesture recognition. Learning was performed on the dataset provided with [1] and consists of 11 gestures. The network consisted of three convolutional layers with max-pooling. A local classifier is attached to every layer and followed by dropout for regularization. DECOLLE is fed with 1ms integer frames. (Right) Classification Error for the DvsGesture task during learning for all local errors associated with the convolutional layers. Shading shows 1 standard deviation across 5 runs.

We presented a novel synaptic plasticity rule, DECOLLE, derived from a surrogate gradient approach with linear scaling in the number of neurons. The rule draws on recent work in surrogate gradient descent in spiking neurons and local learning with layerwise classifiers. The linear scalability is obtained through a (instantaneous) rate-based cost function on the local classifier. The simplicity of the DECOLLE rule equation makes it amenable for direct exploitation of existing machine learning software libraries. Thanks to the surrogate gradient approach, the updates computed through automatic differentiation are equal to the DECOLLE update.

A direct consequence of the local classifiers is the lack of cross-layer adaptation of the layers. To tackle this problem, one could use meta-learning to adapt the random matrix in the classifier. In effect, the meta-learning loop would act as the outer loop in the synthetic gradients approach [11]. The notion that a "layer" of neurons specialized in solving certain problems and sensory modalities is natural in computational neurosciences and can open multiple investigation avenues for understanding learning and plasticity in the brain.

DECOLLE is a departure from standard SNNs trained with Hebbian spike-timing-dependent plasticity, as it uses a 'normative" learning rule that is partially derived from first principles. Models of this type can make use of standard processors where it makes the most sense (i.e. readout, cost functions etc.) and neuromorphic dedicated hardware for the rest. Because it leverages the best of both worlds, DECOLLE is poised to make SNNs take off in event-based computer vision.

Acknowledgments

EN was supported by the Intel Corporation, the National Science Foundation under grant 1640081, and by the Korean Institute of Science and Technology. JK was supported by a fellowship within the FITweltweit programme of the German Academic Exchange Service (DAAD). HM was supported by the Swiss National Fund. JK, HM, EN wrote the paper, conceived the experiments. JK and EN ran the experiments and analyzed the data.

References

[1] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7243–7252, 2017.

[2] Pierre Baldi, Peter Sadowski, and Zhiqin Lu. Learning in the machine: The symmetries of the deep learning channel. *Neural Networks*, 95:110–133, 2017.

[3] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. *arXiv preprint arXiv:1803.09574*, 2018.

[4] Sander M Bohte, Joost N Kok, and Johannes A La Poutré. Spikeprop: backpropagation for networks of spiking neurons. In *ESANN*, pages 419–424, 2000.

[5] C. Clopath, L. Büsing, E. Vasilaki, and W. Gerstner. Connectivity reflects coding: a model of voltage-based stdp with homeostasis. *Nature Neuroscience*, 13(3):344–352, 2010.

[6] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

[7] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, P. Joshi, A. Lines, A. Wild, and H. Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, PP(99):1–1, 2018.

[8] C. Eliasmith and C.H. Anderson. *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT Press, 2004.

[9] W. Gerstner and W. Kistler. *Spiking Neuron Models. Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[10] Dongsung Huh and Terrence J Sejnowski. Gradient descent for spiking neural networks. *arXiv preprint arXiv:1706.04698*, 2017.

[11] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*, 2016.

[12] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10, 2016.

[13] P. Lichtsteiner, C. Posch, and T. Delbruck. An 128x128 120dB 15μs-latency temporal contrast vision sensor. *IEEE J. Solid State Circuits*, 43(2):566–576, 2008.

[14] Timothy P Lillicrap, Daniel Cownden, Douglas B Tweed, and Colin J Akerman. Random feedback weights support learning in deep neural networks. *arXiv preprint arXiv:1411.0247*, 2014.

[15] W. Maass, T. Natschläger, and H. Markram. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2531–2560, 2002.

[16] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[17] Hesham Mostafa, Vishwajith Ramesh, and Gert Cauwenberghs. Deep supervised learning using local errors. *arXiv preprint arXiv:1711.06756*, 2017.

[18] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks. *arXiv preprint arXiv:1901.09948*, 2019.

[19] Jean-Pascal Pfister, Taro Toyoizumi, David Barber, and Wulfram Gerstner. Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural computation*, 18(6):1318–1348, 2006.

[20] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[21] Sumit Bam Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. *arXiv preprint arXiv:1810.08646*, 2018.

[22] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[23] David Sussillo and Larry F Abbott. Generating coherent patterns of activity from chaotic neural networks. *Neuron*, 63(4):544–557, 2009.

[24] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[25] Robert Urbanczik and Walter Senn. Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3):521–528, 2014.

[26] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[27] Friedemann Zenke and Surya Ganguli. Superspike: Supervised learning in multi-layer spiking neural networks. *arXiv preprint arXiv:1705.11146*, 2017.