Flow cytometry with event-based vision and spiking neuromorphic hardware

Steven Abreu
University of Groningen
Groningen, Netherlands
s.abreu@rug.nl

Muhammed Gouda
Ghent University - imec
Ghent, Belgium
MuhammedGoudaAhmed.Gouda@ugent.be

Alessio Lugnan
Ghent University - imec
Ghent, Belgium
Alessio.Lugnan@UGent.be

Peter Bienstman
Ghent University - imec
Ghent, Belgium
Peter.Bienstman@UGent.be

Abstract

Imaging flow cytometry systems play a critical role in the identification and characterization of large populations of cells or micro-particles. Such systems typically leverage deep artificial neural networks to classify samples. Here we show that an event-based camera and neuromorphic processor can be used in a flow cytometry setup to solve a binary particle classification task with less memory usage, and promising improvements in latency and energy scaling. To reduce the complexity of the spiking neural network, we combine the event-based camera with a free-space optical setup which acts as a non-linear high-dimensional feature map that is computed at the speed of light before the event-based camera receives the signal. We demonstrate, for the first time, a spiking neural network running on neuromorphic hardware for a fully event-based flow cytometry pipeline with 98.45% testing accuracy. Our best artificial neural network on frames of the same data reaches only 97.51%, establishing a neuromorphic advantage also in classification accuracy. We further show that our system will scale favorably to more complex classification tasks. We pave the way for real-time classification with throughput of up to 1,000 samples per second and open up new possibilities for online and on-chip learning in flow cytometry applications.

1. Introduction

Flow cytometry is a technology that enables the separation and analysis of different types of cells based on their physical and chemical properties, such as size, shape, and fluorescence. It involves passing cells through a flow cell in a fluid stream, where they are analyzed using laser-based detection systems. This technique has become an essential tool in various fields of biomedical research, including cancer diagnosis, immunology, microbiology, genetics, and has also been applied to the analysis of microparticles [8].

In traditional flow cytometry experiments, labeling cells with biomarkers can cause chemical interactions that alter the cells' characteristics, potentially affecting the accuracy of the experimental results. *Label-free* imaging flow cytometry provides an alternative by capturing thousands of images of cells without the need for labeling. This technique allows for the analysis of cells in their natural state, providing a more reliable assessment of cell properties. Label-free imaging flow cytometry is particularly useful in situations where cell labeling may not be possible, such as with rare or primary cells or in vivo samples. It is also valuable in cases where the effect of biomarker labeling on the cells must be avoided [6].

Two main challenges with current imaging flow cytometers can be identified [4]. The first challenge relates to memory as high-speed imaging flow cytometers generate a large volume of data, requiring a camera with a high frame rate. The second challenge is related to background noise, which affects the accuracy of pixel values in the frames captured by a CMOS camera, and spatial information required for machine learning models is often corrupted by this noise. To address this, an event-based camera was utilized which only respond to variations in intensities of light. Therefore, by including this type of dynamic sensors, the noise filtering becomes simpler and can be implemented directly in the vision sensor or on the neuromorphic processor. Using a neuromorphic processor leverages the data sparsity to further reduce the number of operations required to classify each particle, thus making the whole setup more energy efficient and faster than conventional solutions.

The goal of our flow cytometry setup is to accurately and quickly classify particles based on their physical and chemical properties. With an event-based camera and spiking neuromorphic chip, it is possible to achieve high accuracy, low latency and low power consumption, making it an attractive solution for real-time and energy-efficient flow cytometry. Using a coherent laser as a light source further allows us to leverage the complex nonlinear interaction between light and the particles, which we use as a nonlinear high-dimensional feature map. This allows us to build a high-performing spiking neural network with a simple architecture - further reducing energy and latency requirements and paving the way for online and on-chip learning.

Contributions We build on previous work [4] for our experimental setup combining free-space optics with an event-based camera. We further extend on a previous work-in-progress [3] which demonstrated the promise of neuromorphic flow cytometry through GPU simulations. We improve the spiking neural network and its learning algorithm to make it more performant and hardware compatible. Finally, we implement the network on the Loihi 2, a spiking neuromorphic chip developed by Intel. To the best of our knowledge, this is the first time that an event-based camera is used with a spiking neuromorphic chip for a fully event-based flow cytometry system. Our dataset is open-source and available online [2] and code for our experiments is available at https://github.com/stevenabreu7/dvs_flow.

2. Background & Related Work

Instead of capturing frames of data at fixed intervals, event-based cameras only report changes in brightness, known as events, on a per-pixel basis. When the brightness of a given pixel changes, the camera generates an event that includes the pixel's coordinates, the time of the event, and the direction of the brightness change. Because events are generated asynchronously and in response to changes in the scene, event-based cameras are highly efficient at capturing motion and high-speed motion, while consuming very little power and bandwidth. This makes them suitable for applications such as flow cytometry [3, 5, 6, 12], where high temporal resolution and low latency are essential.

Neuromorphic chips with spiking neural networks are highly suitable for event-based cameras. Traditional digital computing architectures process data in a sequential, clock-driven manner, which is not well-suited for processing the highly sparse data generated by event-based cameras. In contrast, neuromorphic chips with spiking neural networks are designed to process data in a massively parallel, event-driven manner, which is better suited to the sparse nature of event-based camera data. By using a spiking neural network, the neuromorphic chip can efficiently process only relevant events, while ignoring the vast majority of the unchanging background data. This leads to a highly energy-efficient processing pipeline, which is essential for applications such as flow cytometry, where low power consumption is important. Overall, the combination of event-based cameras and spiking neural network neuromorphic chips holds great promise for achieving highly efficient and accurate event-based processing in a wide range of computer vision applications [1].

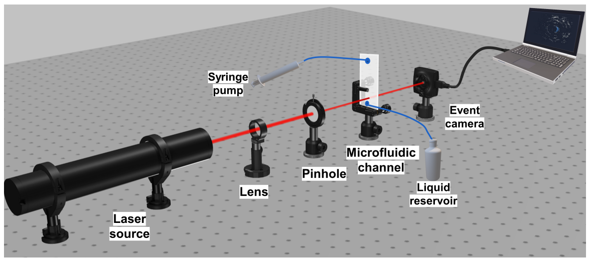

Figure 1. The free-space optical setup built for data collection. Light emitted by the 633 nm He-Ne laser is focused on a Polymethyl methacrylate microfluidic channel. Diffraction, scattering and interference of light due to moving particles cause changes in the intensity of light incident on the event sensor.

2.1. Related work

Recent work has shown that an event-based camera can achieve high throughput of $>$1,000 samples per second [5]. The operating principle of the event-based camera inherently filters out temporally redundant information, allowing for higher sample throughput at similar data rates. However, He *et al.* [5] use an artificial neural network (ANN) for which the event-based data is converted into frames, thus losing the data sparsity. In order to use conventional computer vision algorithms on event-based data, it is typically necessary to convert the data into frames. A more natural approach is to process event-based data using event-based processing schemes. We propose the use of spiking neural networks (SNNs) to process the imaging data in a fully event-based way. Zhang *et al.* [12] have outlined a proof of concept system for the curation of a neuromorphic imaging cytometry dataset, using an event-based camera for imaging cytometry and a spiking neural network for classification, but did not present any performance results.

3. Approach

3.1. Optical setup for flow cytometry

The experimental setup is shown in Fig. 1. We send artificial Polymethyl methacrylate microfluidic microbeads of different sizes to a narrow microfluidic channel (of width 200 μm). The intensity of the light detected by the event-camera stays approximately constant as long as no particle is moving across the field of view. This does not trigger any pixel to fire events. However, once particles enter the field of view, diffraction, scattering and interference of the light trigger events which we use later to train our spiking neural network. We use a Prophesee EVK-1-VGA camera in our setup.

We leverage the light-matter interaction between our coherent light source, a 633 nm He-Ne laser, and the particles passing through the microfluidic channel as a nonlinear high-dimensional expansion. We thus extract features for the classifier in the optical domain at the speed of light. This scales favorably to more complex classification tasks, such as classifying biological cells, at constant computation time for the optical feature extraction.

The fact that we obtain events only in the presence of particles significantly improves the memory efficiency of the overall system. The size of the dataset we get with our system is on the order of 1.25GB per minute of data recording, which presents a 10x reduction compared to work using a traditional frame-based camera [7]. Moreover, using a spiking neural network to process the data from the event-based camera will allow for a reduction in dynamic power consumption of the processor, as the spiking neural network only processes data when particles are passing through the camera's field of vision.

3.2. Data collection

We used two different classes of spherical microparticles (class A of diameter 16 µm and class B of diameter 20 µm) which we purchased from PolyAn GmbH based in Germany. We ran four separate experiments for each class of microparticles, where each experiment ran for $T_{exp} \approx 60s$ for a total of 480 seconds of data per class. The accumulation time for a single particle is $T_{acc} \approx 1ms$. Therefore, we have around 6,000 samples per experiment, or 24,000 samples in total per class. Our event-based camera has microsecond resolution with a sensor of size 640x480 with two channels for events of 'up' and 'down' polarity. Our dataset is open-source and available online [2].

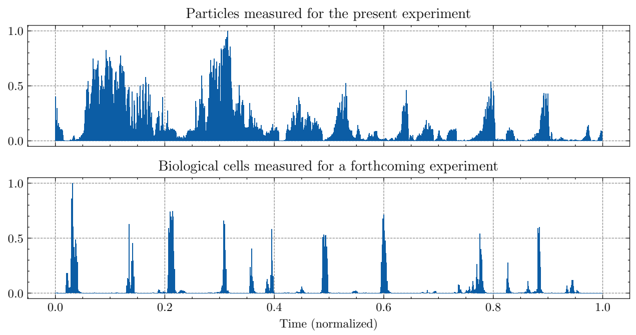

Figure 2. Normalized event count for recordings from the event-based camera. The x-axis was scaled differently to adjust for particle flow speed. *Top*: data recorded from artificial particles for the present experiment over 10ms. *Bottom*: data recorded from biological cells for a forthcoming experiment over 1s.

The data recorded is shown in Figure 2 (top). The background noise is easily removed by a low-pass filter, which we will describe as part of our data pre-processing pipeline. The particles are not passing through the flow cell at equal speed or equal time intervals, which can be seen by the peaks of varying width and distances from another in Figure 2 (top). This can be remedied, for example, by utilizing an automated microfluidic pump which guarantees a constant flow rate in our system. Using the same experimental setup with different particles, we obtained much cleaner measurements. In Figure 2 (bottom), we show data from measurements on biological cells. The different cells passing through the flow cell can be identified much more easily by pronounced peaks that are clearly separated in time.

3.2.1 Preprocessing

We implement two pre-processing steps to compress our data. First, we downsample the event camera's resolution by 20x to yield a 32x24 pixel grid with two channels. At this stage, duplicate events are not dropped. Second, we pass all samples through a layer of leaky integrate-and-fire (LIF) neurons. For every downsampled pixel, we have a LIF neuron with:

$$v_{xyp}(t) = \sigma v(t-1) + w n^{in}_{xyp}(t) \qquad (1)$$

$$s_{xyp}(t) = (v_{xyp}(t) > v_{thr}) \qquad (2)$$

where $x \in \{0, ..., 31\}$, $y \in \{0, ..., 23\}$, $p \in \{0, 1\}$ index the neurons, $n^{in}_{xyp}(t)$ is the number of spikes at time t at the corresponding index, and $\sigma = 0.9, w = 1.0, v_{thr} = 3.0$ are parameters of the neuron. We further add a refractory period $T_{refr} = 2$ in which all incoming spikes are ignored. After a spike $s_{xyp}(t^*)$ at time t^*, the membrane potential is reset to $v_{xyp}(t^* + 1) = v_{rest} = 0$. Time is discretized using a timestep of 1µs, as this is the temporal resolution of the event-based camera. This pre-processing pipeline leads to a $\approx 85\%$ compression of the dataset. It is important to note that our entire pre-processing pipeline is compatible with the neuromorphic chip we use in this experiment.

3.2.2 Filtering low-activity samples

As described in Section 3.2, we have $\approx$ 24,000 samples per class. However, due to the slight inhomogenous distribution of particles in the solution, in some samples no particles are passing through the visual field, and they contain only background noise. To train our system only on meaningful data, we remove samples with less than 1,000 events which is our empirically determined threshold for a particle being observed. In this way, we filter out $\approx$ 80% of the samples. Four samples from the filtered and preprocessed dataset are visualized in Figure 3, showing both a frame reconstruction and the event count over time.

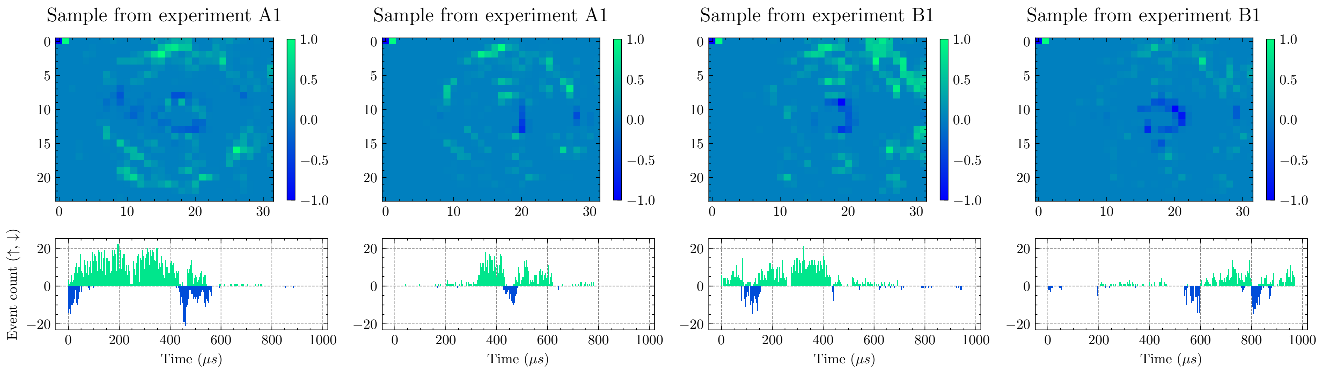

Figure 3. Visualization of four samples from the dataset. The left two plots show particles of class A, the right two plots show particles of class B. *Top*: image reconstruction from the event-based data using, where the polarities are collapsed by subtracting the number of negative events from the number of positive events for every pixel, accumulated over $1ms$ and then normalized. Two pixels in the top-left corner are fixed to extreme values. *Bottom*: number of positive and negative events over time for the same sample.

3.3. Classification task

The goal of our system is to classify particles as belonging to either class A, or class B. For a fair assessment of our system, we report its performance on data from novel experiments, in order to avoid overfitting to specific measuring conditions. As we have data from four experiments, we train our system on four different train-test splits. Each data split is using a different experiment as testing data, and the remaining three experiments as training data.

3.4. Spiking neural networks

In the following, we describe the model of our spiking neural networks, the SLAYER method for training a feed-forward spiking neural network, and how a trained network is tranferred to the Loihi 2 chip for inference.

3.4.1 Neuron model

We choose a current-based leaky integrate-and-fire neuron model (CUBA), which is compatible with Loihi 2. The neuron dynamics are modeled in discrete time as:

$$u(n) = (1 - \tau_u)u(n-1) + a_{in} \tag{3}$$

$$v'(n) = (1 - \tau_v)v(n-1) + u(n) + b \tag{4}$$

$$s_{out}(n) = v'(n) > v_{thr} \tag{5}$$

$$v(n) = v'(n) * (1 - s_{out}(n)) \tag{6}$$

where $u(n)$ is the current flowing into the neuron, $v(n)$ is its membrane potential, τ_u and τ_v are time constants, b is a bias, v_{thr} is the spiking threshold, and s_{out} is the neuron's output spike train. The input current $a_{in}(n) = ws_{in}(n)$ is injected through the synapse with a trainable synaptic weight w.

3.4.2 Training

To train feed-forward spiking neural networks, we use spike error layer re-assigment in time (SLAYER [9]). Given uncoming spike trains $s_i(t) = \sum_f \delta(t - t_i^f)$, where $\{t_i^f\}$ are the spike times for input i, a neuron is modeled as:

$$u(t) = \sum_i w_i(\epsilon * s_i)(t) \tag{7}$$

$$s(t) = f_s(u(t)) \tag{8}$$

where w_i is a synaptic weight, $*$ is the convolution operator, ϵ is the spike response kernel, $s(t)$ is the neuron's output spike train and f_s is the spike function.

For the backward pass, SLAYER unrolls the computational graph of the SNN in time, and applies backpropagation through time (BPTT) to the spiking feed-forward neural network. However, as the spike function is non-differentiable, BPTT cannot be implemented directly. To solve this, SLAYER views the neuron probabilistically to approximate the spike function derivative using the spike escape rate of the probabilistic neuron model. This is similar to the concept of a surrogate gradient, which is used in other methods that approximate BPTT for SNNs.

Our output layer contains two neurons, one for each class that the particle may belong to. We encode the prediction of the spiking neural network into the spike rate of the output neurons. The loss function for a single sample is given by:

$$\hat{\mathbf{r}}_i = \begin{cases} r_{true} \text{ if } i \text{ is the correct class} \\ r_{false} \text{ if } i \text{ is the incorrect class} \end{cases} \tag{9}$$

$$L_{rate} = \sum_i (\mathbf{r}_i - \hat{\mathbf{r}}_i) \tag{10}$$

where $\mathbf{r}$ is the vector of output spike rates of the network and $\hat{\mathbf{r}}$ is the true vector of output spike rates, as determined by the class of the current sample and the hyperparameters r_{true} and r_{false} which indicate the desired spike rate for the correct class and the incorrect class, respectively. We set $r_{true} = 0.3$ and $r_{false} = 0.02$ for all experiments, except where noted otherwise.

A key advantage to SLAYER over other methods is that it also provides a temporal credit assignment policy, while many other methods neglect the effect of earlier spike inputs [9]. Moreover, SLAYER can optimize both the synaptic weights and the axonal delays of a network, thereby enabling the network to learn richer spatiotemporal patterns. This presents a major advantage for flow cytometry, as learning is facilitated not only in the spatial domain (as with conventional frame-based cameras), but also in the temporal domain.

The spike response model that SLAYER uses allows for an efficient convolution-based implementation of the forward pass. We use the PyTorch-based lava-dl[1] library to train the network using SLAYER.

3.4.3 Inference on chip

In order to run our trained SNN on the Loihi 2, we use a limited-precision model that is compatible with the chip. All parameters and variables are integer values of fixed precision according to the constraints of the hardware. To train for the fixed precision constraints on weights and delays of Loihi 2, we train the network with the quantization constraints where the constrained network is used in the forward propagation phase and full precision shadow variables are used in the backward pass.

4. Experiments

We proceed by introducing different frame-based classification systems, using linear models and artificial neural networks, as a baseline, and different spiking neural networks for use in an event-based neuromorphic classification pipeline.

4.1. Linear classifier on frames

As a first step, we train a linear classifier on our dataset. The performance of this linear model then serves as a baseline against which we will compare our spiking neural network. We aggregate the events of each sample of $T_{acc} = 1ms$ into a frame with two channels that represent the polarities, leading to a three-dimensional array of size 32x24x2 where each value represents the number of events for a certain pixel and polarity. We scale the feature values by removing the mean and scaling to unit variance, where the mean and variance are computed with respect to the training data.

[1] See https://github.com/lava-nc/lava-dl.

Our logistic classifier predicts the probability of a sample X_i belonging to class A as:

$$\hat{p}_i = \frac{1}{1 + \exp(-X_i w - w_0)} \tag{11}$$

where w and w_0 are parameters of the classifier. We fit a logistic regression classifier by minimizing the function:

$$\sum_{i=1}^{N} (-y_i \log \hat{p}_i - (1 - y_i) \log(1 - \hat{p}_i)) + r(w) \tag{12}$$

where y_i is the true label, N is the number of training samples and $r(w) = \frac{1}{2} w^T w$ is the L2 regularization term. We use an approximation of the Broyden–Fletcher–Goldfarb–Shanno algorithm to solve the optimization problem. This results in a training accuracy of 98.75% and testing accuracy of 96.05%, averaged over all four data folds, see Figure 4. The high accuracy of the linear classifier confirms the optical setup, which acts as a high-dimensional nonlinear feature map, thereby making the imaging data (nearly) linearly separable. It is important to note that we have shown the linear separability only for samples represented by a single frame which aggregates all events during the sample. It is not clear if sparse "frames" with 1µs resolution are linearly classifiable with the same level of accuracy. Indeed, we later show that this is not the case, and performance depends on the sampling time, *i.e.* the time resolution at which events are resolved.

4.2. Neural networks on frames

We also train neural networks on frames of our samples, applying the same zero-mean unit-variance scaling as for the linear classifier. We first train a multi-layer perceptron (MLP) with two hidden layers of 512 units each, with a ReLU activation function. We optimize the weights over 10 epochs using the Adam optimizer on the crossentropy loss with a learning rate of 0.001. We run the training on a GPU with a batch size of 64. The resulting training and testing accuracies are 99.52% and 97.51%, respectively, showing a small but significant improvement over the linear classifier. The results are shown in Figure 4.

We also train a simple convolutional neural network (CNN) containing a convolutional layer with 32 filters, a convolutional layer with 64 filters, and a fully connected layer with 512 units. Both convolutional layers use 3x3 kernels and are followed by max pooling layers with pool size 2x2. All layers use the ReLU activation function, except the last layer which uses a softmax. The network is also optimized using the Adam optimizer on the crossentropy loss with a learning rate of 0.001, and was trained for 20 epochs on a GPU using a batch size of 64. The resulting training and testing accuracies are 99.67% and 97.09%, respectively. Compared to the simpler MLP, the CNN shows increased

Frame-based data	
Linear	98.86% ± 0.21% 96.05% ± 0.80%
1 layer	99.46% ± 0.20% 97.32% ± 0.55%
2 layers	99.52% ± 0.23% **97.51% ± 0.35%**
CNN	99.67% ± 0.05% 97.09% ± 0.70%

Event-based data				
Model		$1\ \mu s$	$10\ \mu s$	$100\ \mu s$
Linear		98.11% ± 0.30% 96.81% ± 0.96%	97.72% ± 0.21% 95.53% ± 2.13%	86.73% ± 0.67% 83.08% ± 2.12%
1 layer	No delay	99.46% ± 0.10% 98.26% ± 0.42%	99.35% ± 0.07% 98.09% ± 0.47%	87.47% ± 0.61% 84.20% ± 4.42%
1 layer	Trained delay	99.43% ± 0.11% **98.45% ± 0.34%**	99.37% ± 0.11% **98.13% ± 0.29%**	91.51% ± 0.76% **88.99% ± 1.56%**
2 layers	No delay	99.60% ± 0.17% 98.12% ± 0.38%	99.18% ± 0.12% 97.25% ± 0.67%	51.97% ± 3.42% 50.72% ± 15.29%
2 layers	Trained delay	99.74% ± 0.12% 98.29% ± 0.41%	99.47% ± 0.03% 97.44% ± 0.58%	68.00% ± 11.34% 68.98% ± 2.43%

Figure 4. Training (top) and testing (bottom) accuracy for all models (mean and standard deviation over all data splits). *Left*: models trained on frame reconstructions of the data. *Right*: models trained directly on events. All neural network layers contain 512 neurons.

training performance but decreased testing performance - likely an indication of overfitting. It is noted that the CNN architecture and hyperparameters were not optimized, and running such an optimization may lead to better results.

4.3. Spiking neural networks

The results of our experiments with spiking neural networks are summarized in Figure 4 for a direct comparison with the baseline classifiers on frames.

We begin by training a feed-forward neural network with one hidden layer of 512 neurons. The output layer of the spiking MLP (sMLP) contains two spiking neurons, one for each class. We train the sMLP's output neurons to spike with a firing rate of $r_{true} = 0.2$ or $r_{false} = 0.03$ for the correct and incorrect class, respectively, over the sample duration of 1000 timesteps. We train the sMLP using Adam with a learning rate of 0.001 for 50 epochs on a GPU with a batch size of 64. This sMLP reaches better performance than any model we have trained on frames of the data, namely 99.43% ± 0.11% training and 98.45% ± 0.34% testing accuracy (mean and standard deviation computed over all four data splits, see Section 3.3). This performance increase likely stems from the addition of the temporal dimension for classification based on spatiotemporal features rather than only spatial features.

4.3.1 Effect of more layers

We further train a spiking neural network with two hidden layers of 512 neurons each, with the same training setup as the network above. This yields a slightly higher training acurracy of 99.60% ± 0.17% as well as a slighly lower testing accuracy of 98.12% ± 0.38%, indicating that the larger capacity of the two-layer network allows for overfitting on the training data. This is in line with our expectation that a simpler network architecture should be enough, as much of the computation is happening already in the optical domain. This allows us to use a simple network architecture with only one hidden layer, effectively decreasing the latency that processing data through multiple hidden layers would introduce.

4.3.2 Effect of delay

We also train a spiking neural network without delay, which shows a small performance decrease of approximately 0.2% in both training and testing accuracy, resulting in a test accuracy of 98.26% ± 0.42%. This may be explained by the fact that trainable delays allow the network to select a richer set of spatiotemporal features than would be possible without delays. We conjecture that recurrent connections in the spiking neural network may further increase performance - and may indeed be necessary to solve more complex classification tasks that require longer memory, see Section 5.

4.3.3 Effect of longer sampling time

So far we have trained every spiking neural network on events with the highest temporal resolution available to us, i.e. a sampling time of 1µs. However, to run our system in real time, this puts a large burden on the neuromorphic processor: it has only 1µs to read data from the event-based camera and process the data through the spiking neural network. This is not feasible with our current setup. Therefore, we also train spiking neural network with a sampling time of 10µs and 100µs. This still provide us with an input spike train of 100 and 10 timesteps, respectively, which allows us to use *some* temporal dynamics of the flow cytometry for classification. We use an identical training setup as before, except that with a sampling time of 100µs, we change the target spiking rates to $r_{true} = 0.8$ for the correct class and $r_{false} = 0.2$ for the incorrect class, as the network only runs for ten timesteps on each sample.

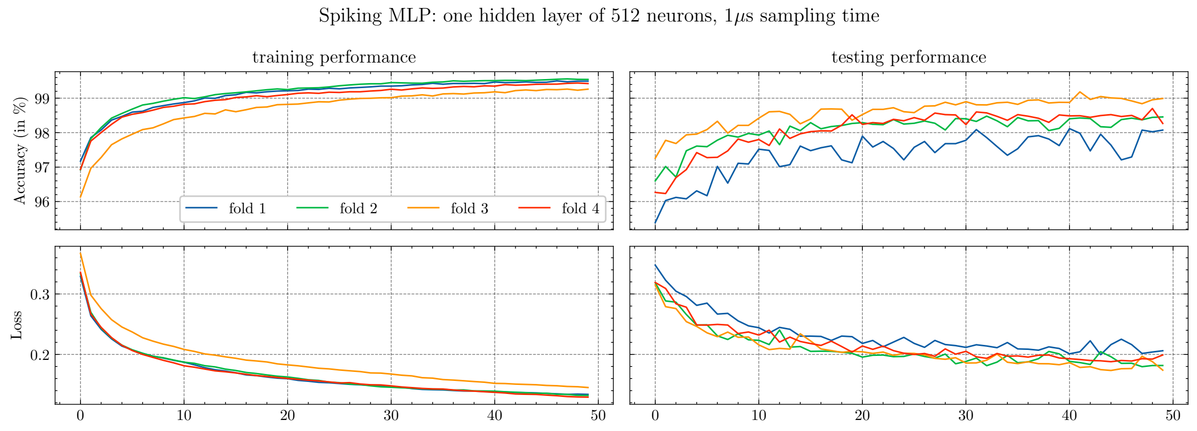

Figure 5. Training curves for a feed-forward spiking neural network with one hidden layer of 512 neurons and trained axonal delays. As explained in Section 3.3, the network is trained on different data splits, indicated by the legend. See Section 4.3 for details.

Note that the input to the spiking neural network is binary. At every timestep, there is only one binary value for every pixel and polarity. We thus discard information about the rate of events, which was present in the data that the baseline classifiers were trained on (see Sections 4.1-4.2), and in the data for the sMLPs trained previously. This issue can be avoided by working with *graded* spikes which is supported by Loihi 2 but requires further pre-processing of the data, which is not desirable for our setup.

It shows that a sampling time of 100µs is too long, with the network reaching only 88.99% ± 1.56% which is significantly below our linear baseline classifiers on the frame representation of the data. Clearly, 10 timesteps are not quite enough for the network to classify the particles. For a sampling time of 10µs and identical training conditions as before, the spiking neural network reaches a training accuracy of 99.37% ± 0.11% and testing accuracy of 98.13% ± 0.29%, which is competitive to the performance reached with a sampling time of 1µs.

4.3.4 Comparison

For completeness, we trained spiking neural networks with all mentioned variations. For the three different sampling times of 1µs, 10µs and 100µs, we train spiking neural networks with one hidden layer of 512 neurons and two hidden layers of 512 neurons. We train each network with and without trainable axonal delay. The results are shown in the table of Figure 4. The best performance was achieved by a one-layer spiking neural network with trained axonal delays at a sampling time of 1µs, with a final testing accuracy of 98.45% ± 0.34%. With a higher sampling time of 10µs, the best-performing network was also a one-layer network with trained axonal delays, reaching a final testing accuracy of 98.13% ± 0.29%. Depending on the available hardware and real-time processing constraint it may be worth trading off some accuracy for a higher sampling time, as we will show next. The best-performing network for the sampling time of 100µs also has only one hidden layer, reaching a final testing accuracy of 88.99% ± 1.56%. This is well below our baseline classifier, thus to use a spiking neural network for this sampling time we suppose that graded input spikes must be used.

Finally, we also train linear classifiers on the input spike trains of different sampling times. As expected, the linear classifiers typically perfom well below the spiking MLPs, especially for longer sampling times. This can be explained by the memory and nonlinearity that the spiking network adds.

4.4. Neuromorphic hardware implementation

The spiking neural networks from the previous section were trained offline using a GPU. We use the trained weights and delays of two spiking neural networks with trained delays to configure the Loihi 2 chip for inference. We use the best-performing networks for a sampling time of 10μs with one and two hidden layers. We chose this sampling time to make real-time processing easier, see Section 5.1. The accuracy of the spiking neural networks on the Loihi 2 chip is identical to the results obtained in simulation, shown in Figure 4. See Section 3.4.3 for more details.

The Loihi 2 chip has 128 neurocores in total. The 1-layer SNN occupies 44 neurocores with a total of 6,620 synapses, while the 2-layer SNN occupies 57 neurocoes with a total of 13,039 synapses. For a single sample of 10ms with a sampling time of 10μs (containing 948 events in total), the 1-layer SNN requires 392,200 synaptic operations, while the 2-layer SNN requires 6,072,829 synaptic operations. Furthermore, the 1-layer SNN is 4x faster than the 2-layer SNN. These metrics are visualized in Figure 6.

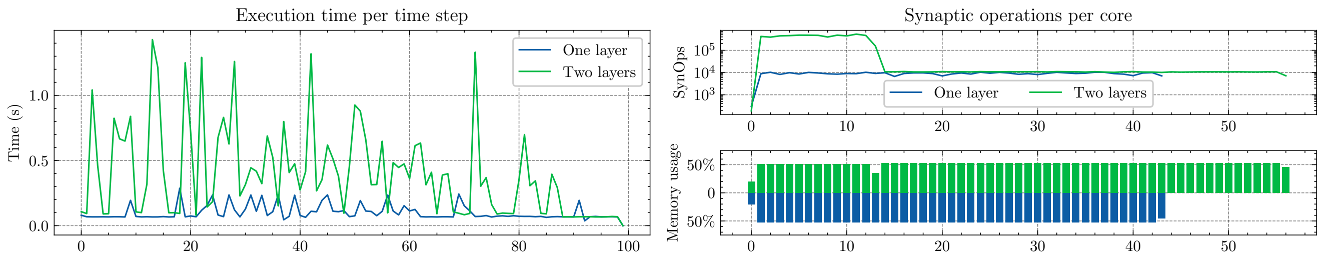

Figure 6. Profiling results from inference of two trained spiking neural networks on the Loihi 2 chip. See text for explanation.

5. Outlook

We point out two promising directions for the nascent field of neuromorphic flow cytometry [3, 5, 12].

5.1. Real-time processing

For the experiments in this paper, we have not been able to use the event-based vision interface on Loihi, as it was not fully supported by the Lava software framework at the time of writing. Therefore, we are not able to run the classification in real-time. It is difficult to estimate if our network will be able to run in real-time on Loihi 2. We deem real-time classification with a sampling time of $1\mu s$ unrealistic. To the best of our knowledge, running in real time with a sampling time of $100\mu s$ should be possible. Running in real-time with a sampling time of $10\mu s$ may also be possible. The performance of our networks was not competitive with a sampling time of $100\mu s$, thus for this setting the data needs to be pre-processed into graded input spike trains (see discussion in Section 4.3).

Alternatively, a system-on-chip (SoC) event-based camera with integrated processing on-chip (such as SynSense's Speck[2]) may be used to run our network. The Speck SoC can run the events produced from the event-based camera through its integrated processing unit in real time.

5.2. Local and online learning

We presented results obtained by training a feed-forward spiking neural network through backpropagation. This procedure requires collecting training data and training a spiking neural network through surrogate gradient methods on a conventional computer. This is suitable for flow cytometry, where machine learning algorithms are typically trained offline. However, it is also possible to use more hardware-friendly methods for training our spiking neural network. As we are able to solve the classification task sufficiently well using only a feed-forward spiking neural network with one hidden layer, using a simple one-layer spiking *recurrent* neural network (sRNN) will yield comparable, or better, performance. The recurrence gives the network longer memory, which can help with more complex particle classifications, such as for biological cells. Although sRNNs are difficult to train with backpropagation through time, we conjecture that a version of forward propagation through time that was recently proposed for spiking recurrent neural networks will solve this task [11], while also allowing for an online learning setup. Alternatively, a reservoir combined with local plasticity, such as spike-timing dependent plasticity (STDP), has been shown to perform well on tasks of similar complexity, and can be extended to a deep reservoir architecture [10]. This is fully compatible with on-chip learning on Loihi 2 and may even be extended to three-factor learning rules such as reward-modulated STDP.

[2]See https://www.synsense.ai/products/speck/

6. Conclusion

We presented a spiking neural network trained with SLAYER and running on Loihi 2 that solves a binary particle classification task in a flow cytometry setup. We highlighted the promise of this setup for accurate and fast flow cytometry, and suggest that our experimental setup - particularly our free-space optical setup - will scale favorably to more complex classification tasks and faster flow rates of the particles. In accordance with literature [3–5, 12], we demonstrate that event-based cameras are suitable for flow cytometry and implement, for the first time, a spiking neural network on neuromorphic hardware for particle classification. We further demonstrated a neuromorphic advantage in terms of accuracy, as none of the tested artificial neural networks using frames outperformed our spiking neural network trained directly on events from the event-based camera. We illustrated the promise of further research in this area, in particular real-time classification and on-chip learning.

Acknowledgements

This work was performed in the context of the European projects Neoteric (grant agreement 871330), Prometheus (grant agreement 101070195) and Post-Digital (grant agreement 860360).

References

[1] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J. Davison, Jorg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):154–180, 2022. 2

[2] Muhammed Gouda, Steven Abreu, Alessio Lugnan, and Peter Bienstman. Neuromorphic particle flow cytometry dataset, 2023. Zenodo. 2, 3

[3] Muhammed Gouda, Steven Abreu, Alessio Lugnan, and Peter Bienstman. Training a spiking neural network on an event-based label-free flow cytometry dataset. *arXiv:2303.10632*, 2023. 2, 8

[4] Muhammed Gouda, Alessio Lugnan, Joni Dambre, Gerd van den Branden, Christoph Posch, and Peter Bienstman. Improving the classification accuracy in label-free flow cytometry using event-based vision and simple logistic regression. *IEEE Journal of Selected Topics in Quantum Electronics*, 29(2: Optical Computing):1–8, 2023. 1, 2, 8

[5] Weihua He, Yongxiang Feng, Junwen Zhu, Huichao Chai, and Wenhui Wang. Neuromorphic-Enabled Event-Based Deep Imaging Flow Cytometry. In *26th International Conference on Miniaturized Systems for Chemistry and Life Sciences*, 10 2022. 2, 8

[6] Lee Kamentsky, Thouis R Jones, Adam Fraser, Mark-Anthony Bray, David J Logan, Katherine L Madden, Vebjorn Ljosa, Curtis Rueden, Kevin W Eliceiri, and Anne E Carpenter. Improved structure, function and compatibility for cellprofiler: modular high-throughput image analysis software. *Bioinformatics*, 27(8):1179–1180, 2011. 1, 2

[7] Alessio Lugnan, Emmanuel Gooskens, Jeremy Vatin, Joni Dambre, and Peter Bienstman. Machine learning issues and opportunities in ultrafast particle classification for label-free microflow cytometry. *Scientific Reports*, 10(1):1–13, 2020. 3

[8] Howard M. Shapiro. *Practical Flow Cytometry, 4th edition*. John Wiley & Sons, Ltd, 2003. 1

[9] Sumit Bam Shrestha and Garrick Orchard. SLAYER: Spike Layer Error Reassignment in Time. In *Advances in Neural Information Processing Systems*, volume 31, 2018. 4, 5

[10] Nicholas Soures and Dhireesha Kudithipudi. Deep liquid state machines with neural plasticity for video activity recognition. *Frontiers in Neuroscience*, 13, 2019. 8

[11] Bojian Yin, Federico Corradi, and Sander M. Bohte. Accurate online training of dynamical spiking neural networks through forward propagation through time. *arXiv:2112.11231*, 2022. 8

[12] Ziyao Zhang, Maria Sabrina Ma, Jason K. Eshraghian, Daniele Vigolo, Ken-Tye Yong, and Omid Kavehei. Work in Progress: Neuromorphic Cytometry, High-throughput Event-based flow Flow-Imaging. In *8th International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP)*. IEEE, 2022. 2, 8