Demonstrating a Fully Automated Software Framework for PyTorch-based Training on BrainScaleS-2 Using a High-speed Data Communication Example

Elias Arnold†, Philipp Spilger†, Christian Mauch†, Luca Blessing†, Eric Müller†,⚒, Johannes Schemmel†

† Electronic Vision(s) Group, Kirchhoff-Institute for Physics, Heidelberg University, Heidelberg, Germany

⚒ European Institute for Neuromorphic Computing, Heidelberg University, Heidelberg, Germany

Introduction

When approaching ML-inspired tasks with training methods of SNNs on neuromorphic hardware, software allowing users to easily formulate experiments and compute gradients with an auto-differentiation mechanism is essential. We demonstrate the abstraction of training on BrainScaleS-2 (BSS-2) in our `PyTorch`-based [1] framework **`hxtorch.snn`**. As an application we consider an optical data communication intensity-modulation/direct-detection (IM/DD) link. Nowadays data traffic pushes research towards novel energy efficient solutions. We train SNNs emulated on BSS-2 for equalising the IM/DD link [2].

Results

`hxtorch.snn`

The **`hxtorch.snn`** framework is visualised on the left. Models are defined in a `PyTorch` fashion, however, our modules are inherited from class `HXModule` (e.g., `Neuron` representing a population and `Synapse` a projection on BSS-2). The modules are registered in an instance `Instance`. Each module holds a differentiable `function`, defining the computation of the backward pass (and forward pass in mock-mode), returning a `Handle` to future data. By calling the function `run`, `Instance` extracts the network topology and executes the experiment on BSS-2. A `PyTorch` graph is constructed by executing the `functions`, getting the recorded hardware observables injected and thereby filling the handles with measured data. This allows computing the gradient on the hardware data implicitly. In the forward pass, we support both, simulation in software and emulation on BSS-2.

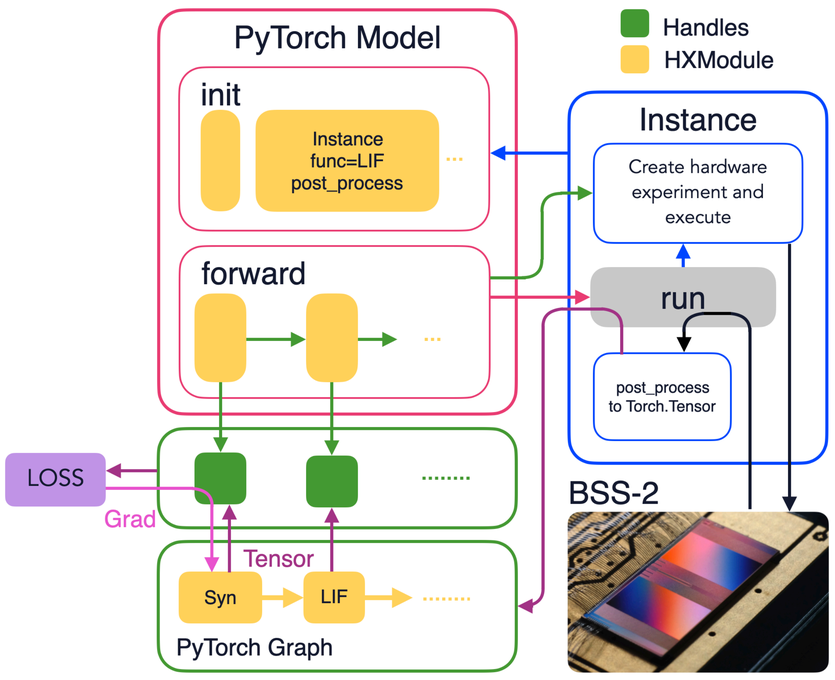

Fig 2.: hxtorch.snn software schematics

Methods

BrainScaleS-2 (BSS-2)

The accelerated mixed-signal neuromorphic BSS-2 system [3], implements 512 AdEx neurons and 131k synapses in analog circuits. Neurons are individually parameterized. Arbitrary topologies can be realised by routing internal and external spike events. Spikes and membranes voltages, sampled at low rate (O(20% of τ_m)), are recorded and read out from the host computer.

Learning on BSS-2

SNNs are often trained using software, like `PyTorch`, allowing calculating weight updates with the backpropagation through time (BPTT) algorithm. On BSS-2 we train SNNs in the loop by emulating the forward pass on-chip and computing the backward pass within `PyTorch`'s ecosystem by injecting recorded observables. For this, the BSS-2 software stack [4] is used.

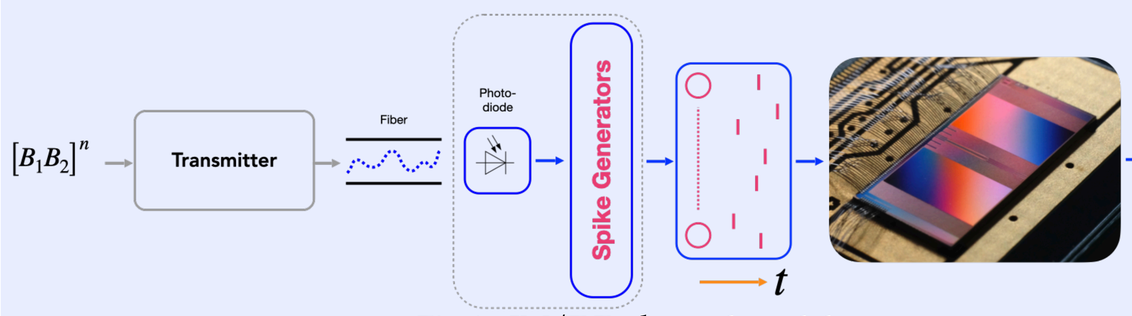

Fig 1.: IM/DD channel model

Optical IM/DD Link

The simulated IM/DD link (4 km fiber, 200 Gb/s bit rate). A bit sequence $[B_1B_2]^n$ is mapped to a signal y^n, transmitted through a fiber and measured at the receiver with a photodiode (PD). The received sequence $\tilde{y}^n$ is impaired linearly by chromatic dispersion, non-linearly by the PD. Additionally amplifier noise is added. A SNN with one hidden LIF and a LI readout layer is trained on BSS-2 to minimize the bit error rate (BER) by equalizing and demapping $\tilde{y}^n$ to bits $\left[\hat{B}_1\hat{B}_2\right]^n$.

Performance IM/DD Link

`hxtorch.snn` is demonstrated by equalizing the IM/DD link. The BER is depicted over noise in the link. Both the SNN in mock-mode and the SNN emulated on BSS-2 outperforms a linear LMMSE equalizer. Also, the membrane traces of the LIF layer, its spikes and the traces of the output layer, together with learned weights, are shown.

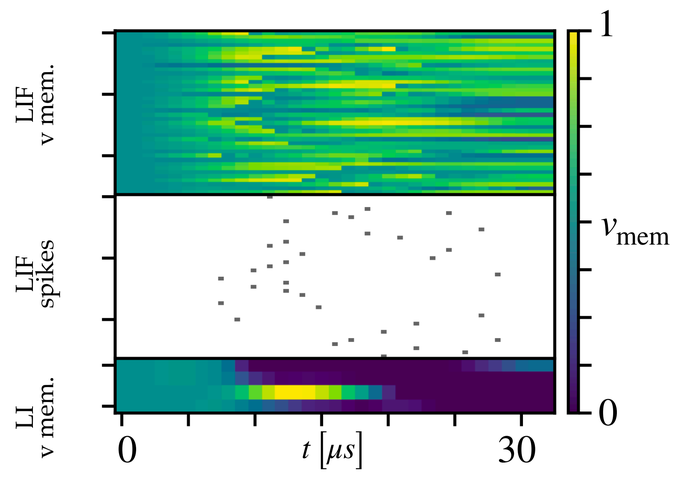

Fig 5.: Hardware observables as example

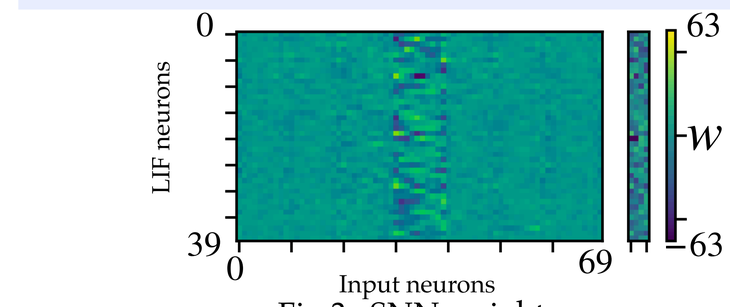

Fig 3.: SNN weights

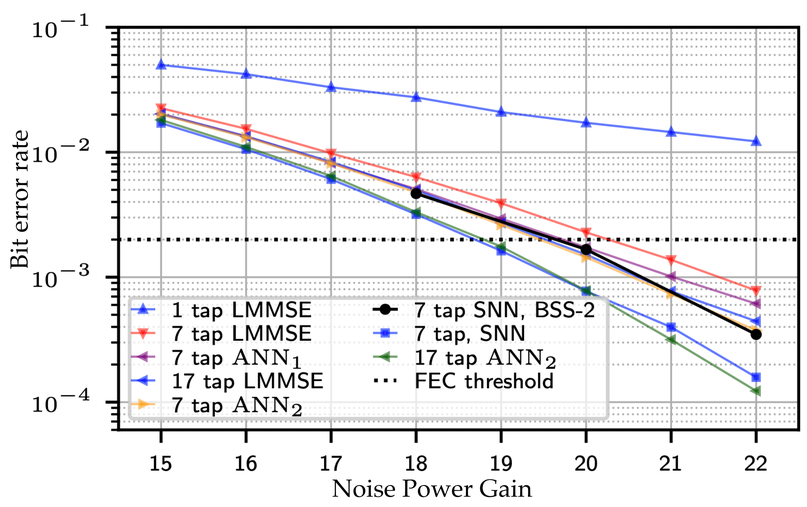

Fig 4.: Signal-to-noise ratio over bit error rate.

Discussion

The `hxtorch.snn` software framework enables arbitrary high-level ML-friendly descriptions of SNN models and their emulation on BSS-2. Due to building upon PyTorch data structures and taking advantage of its auto-differentiation mechanism, weight updates are computed with ease. Hence, **`hxtorch.snn`** facilitates training SNNs on BSS-2 at every level.

References

[1] A. Paszke et al, "Automatic differentiation in pytorch," 2017

[2] E. Arnold, G. Böcherer, et al, "Spiking neural network equalization on neuromorphic hardware for IM/DD optical communication," Proc. Eur. Conf. Optical Commun. (ECOC), Basel, Switzerland, Sep. 2022, paper Th1C.5

[3] C. Pehle et al, "The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity," Front. Neurosci., vol. 16, 2022. [Online]. Available: https://doi.org/10.3389/fnins.2022.795876

[4] E. Müller et al, "A scalable approach to modeling on accelerated neuromorphic hardware," Front. Neurosci., 2022, accepted. [Online]. Available: https://doi.org/10.3389/fnins.2022.884128