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.

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 τm)), 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.

Optical IM/DD Link

The simulated IM/DD link (4 km fiber, 200 Gb/s bit rate). A bit sequence B̂_1 B̂_2 is mapped to a signal y^s, transmitted through a fiber and measured at the receiver with a photodiode (PD). The received sequence y^r 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 y^r to bits B̂_1 B̂_2.

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


Fig 1.: IM/DD channel model

Fig 5.: Hardware observables as example

Fig 3.: SNN weights

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

Fig 2.: hxtorch.snn software schematics