NNPC 2023: Optimizing Gradient Accumulation in Spiking Neural Networks with AlphaGrad

Jamie Lohoff[1,2], **Emre Neftci**[1,2]
1. Jülich Research Center, Jülich, Germany
2. Department of Electrical Engineering, RWTH Aachen, Germany

Summary. Backpropagation lies at the heart of training spiking neural networks to state-of-the-art performance, but in many cases is not optimal in terms of computational costs. Cross-country evaluation is a method to derive exact gradient accumulation procedures that are tailored to a given function and possibly provide significant savings in compute and memory. However, finding the optimal elimination procedures is an NP-complete problem. Inspired by recent advances in finding optimal matrix-multiplication algorithms [1], we demonstrate that model-based reinforcement learning can be used to overcome this issue and successfully find exact gradient-accumulation procedures with lower computational costs than backpropagation.

Spiking neural networks (SNNs) provide a promising way of implementing AI algorithms at a fraction of the current energy cost. However, current top-performing SNNs still rely on gradient-based learning, i.e. backpropagation or approximations thereof to optimize and update their weights. Apart from not being a bio-plausible learning paradigm, the backpropagation algorithm itself suffers from exceedingly large memory requirements and unnecessary computations for gradient accumulation that become particularly pronounced in SNNs due to the rich neuron dynamics and long time horizons. Several works demonstrated that these problems can be partly alleviated by *mixed-mode automatic differentiation* (AD), where backpropagation is combined with other methods such as forward-mode AD [2, 3]. These methods of mixed-mode AD are part of a much larger subfield of AD called cross-country elimination.

Cross-Country Elimination At the heart of AD lies the concept of a computational graph which is a directed acyclic graph that describes the evaluation procedure of a mathematical function. The computational graph is constructed by decomposing the function into its elemental operations (such as multiplications, logarithms etc.) and then assigning them to the vertices of the graph. The edges determine the flow of data between the vertices of the graph. If we view the interior vertices of the graph as intermediate variables of a step-by-step evaluation procedure, we can compute the partial derivatives of a vertex with respect to the vertices that are connected to it by in-going edges. These partial derivatives are then assigned to the respective edges. To compute the gradient of the function, there exists a number of different *elimination procedures* that redistribute and accumulate the partial derivatives and remove edges according to different schemes. Once all interior edges are eliminated, we are left with a graph of only input and output vertices connected by edges, i.e. a bipartite graph. These edges then contain the components of the gradient or Jacobian of the function. One of the possible elimination procedures is named vertex elimination, since it removes all in-going and out-going edges of a certain vertex at once, which enables us to completely remove this vertex from the graph. In this perspective, backpropagation (or reverse-mode AD) is equivalent to eliminating the vertices in reverse order, starting at the output vertices. Similarly, forward-mode AD eliminates vertices in the same order as the graph is traversed. However, any elimination order is possible, which then gives rise to the method of *cross-country evaluation* [4].

All these modes will produce the same gradient, but at significantly different computational costs in terms of numbers of multiplications, additions and required memory. In general, reverse-mode AD is very efficient for computational graphs with many inputs and few outputs such as neural networks trained with a scalar loss function. However, it is not necessarily the most efficient way for a given computational graph, since it could contain a subgraph where forward-mode AD is more efficient which leads to a suboptimal result. Then, there exists a more efficient cross-country elimination procedure that incurs significant savings. Unfortunately, finding the optimal cross-country elimination order is an NP-hard problem, such that it becomes cumbersome for large computational graphs [5]. These kinds of problems have been demonstrated to be solved successfully using model-based reinforcement learning, e.g. AlphaZero. In particular, the most recent example of this is AlphaTensor, where AlphaZero was employed to find the optimal way of multiplying two matrices in terms of number of computations [1].

Reinforcement Learning and AlphaZero The AlphaZero algorithm uses Monte-Carlo tree search (MCTS) to train a policy and a value network to select the next action for a given game state [6]. For

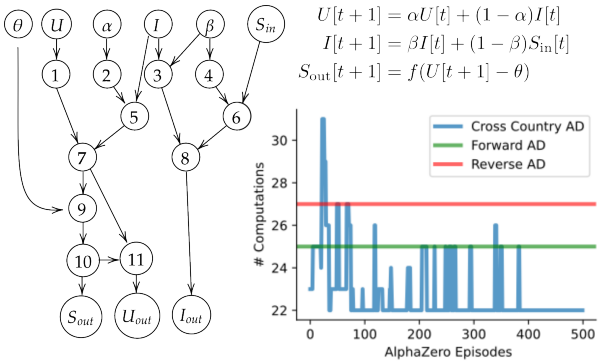

Figure 1: Optimizing the number of computations for the gradient of a single LIF neuron. Left shows the computational graph. Right shows how the number of computations in the gradient accumulation procedure improves with cross-country AD using model-based reinforcement learning (AlphaZero). Forward-mode AD and reverse-mode AD(backpropagation) added for comparison.

every action taken during an episode, MCTS constructs a search tree starting from the current game state which will contain future outcomes to find the best possible action. The search tree is populated by simulating games and the policy and value networks focus the exploration on the branches that give the highest reward based on experience from prior tree searches. Once an episode is finished, the resulting state-policy-reward tuples gained from the tree search are used to update the policy and value networks.

AlphaGrad defines cross-country elimination as a game where the states are matrices of zeros and ones indicating the edges that connect the vertices with each other. This information is sufficient since the number of computations for accumulation depends only on the topology of the graph. The actions are the choices of which vertex to eliminate next and the scalar reward is the negative number of multiplications. A game is complete once all interior vertices are eliminated. To produce estimates for the policy and value, we feed the state matrix into a axial transformer which has a policy head and a value head. Both heads are then trained on the samples obtained from MCTS using a divergence loss and a MSE loss respectively.

Preliminary Results and Prospects Our preliminary results show a reduction of the number of operations from 25 operations to 22 when computing the gradient of one LIF neuron (Fig. 1). While this improvement may seem modest, it is 1) without any loss in exactness 2) generalizable to networks in its current form, and 3) a stepping stone to optimization with engineering constraints (*e.g.* quantization, peak memory use) or biological desiderata (*e.g.* locality) where much larger gains can be achieved. Furthermore, we expect much larger improvements for more intricate neuron models and the use of other elimination procedures such as *edge elimination*.

References

[1] A. Fawzi *et al.*, "Discovering faster matrix multiplication algorithms with reinforcement learning," *Nature*, vol. 610, p. 4753, Oct 2022.

[2] F. Zenke and E. O. Neftci, "Braininspired learning on neuromorphic substrates," *Proceedings of the IEEE*, p. 116, 2021.

[3] J. Kaiser, H. Mostafa, and E. Neftci, "Synaptic plasticity dynamics for deep continuous local learning (DECOLLE)," *Frontiers in Neuroscience*, vol. 14, p. 424, 2020.

[4] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[5] U. Naumann, "Optimal jacobian accumulation is NP-complete," *Mathematical Programming*, vol. 112, no. 2, p. 427441, 2008.

[6] D. Silver *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.