3.4 – Learning-to-Learn for Neuromorphic Hardware

Franz Scherr, Wolfgang Maass, Inst. of Theor. Computer Science, TU Graz

Status

An important goal for neuromorphic hardware is to support fast on-chip learning in the hand of a user. Two problems need to be solved for that:

1. A sufficiently powerful learning method has to run on the chip, such as stochastic gradient descent.
2. It needs to be able to generalize from a single example (one-shot learning), or at least from very few.

Evolution has found methods that enable brains to learn a new class from a single or very few examples. For instance, we can recognize the face of a new person in many orientations, scales, and lighting conditions after seeing it just once. But this fast learning is supported by a long series of prior optimization processes of the neural networks in the brain during evolution, development, and prior learning. In addition, insight from cognitive science suggests that the learning and generalization capability of our brains is supported by innate knowledge, e.g. about basic properties of objects, 3D space, and physics. Hence, in contrast to most prior on-chip learning experiments in neuromorphic engineering, neural networks in the brain do not start from a tabula rasa state when they learn something new.

Learning from few examples has already been addressed in modern machine learning and AI [1]. Of particular interest for neuromorphic applications are methods that enable recurrently connected neural networks (RNNs) to learn from single or few examples. RNNs are usually needed for online temporal processing —an application domain of particular interest for energy-efficient neuromorphic hardware. The gold standard for RNN-learning is backpropagation through time (BPTT). While BPTT is inherently an offline learning method that appears to be off-limit for online on-chip learning, it has recently been shown that BPTT can be approximated quite well by computationally efficient online approximations. In particular, one can port the online broadcast alignment heuristic from feedforward to recurrent neural networks [2,3]. In addition, one can emulate the common LSTM (long short-term memory) units of RNNs in machine learning by neuromorphic hardware-friendly adapting spiking neurons. Finally, a computationally efficient online approximation of BPTT —called e-prop— works well for recurrent networks of spiking neurons (RSNNs), also with adapting neurons [3]. The resulting algorithm for on-chip training of the weights W_{ji} for neuron i to neuron j of an RSNN —for some arbitrary but differentiable loss function E— takes there the form $\frac{dE}{dW_{ji}} = \sum_t L_j^t \, e_{ji}^t$. The so-called learning signal L_j^t at time t is some online approximation to the derivative of the loss function E with regard to the spike output of neuron j, and e_{ji}^t is an online and locally computable eligibility trace. If one optimizes the learning signals L_j^t and the initial values of the weights W_{ji} via Learning-to-Learn (L2L) for a range $\mathcal{F}$ of potentially user-relevant tasks C, these can be learnt from very few examples, see Figure 1 and [4, 5].

Current and Future Challenges

The main choices that have to be made for such realization of fast on-chip learning are the choice of the family $\mathcal{F}$ of tasks, the choice of the optimization method for offline priming through the definition of hyperparameters, and the choice of the hyperparameters. Options for the latter are for example

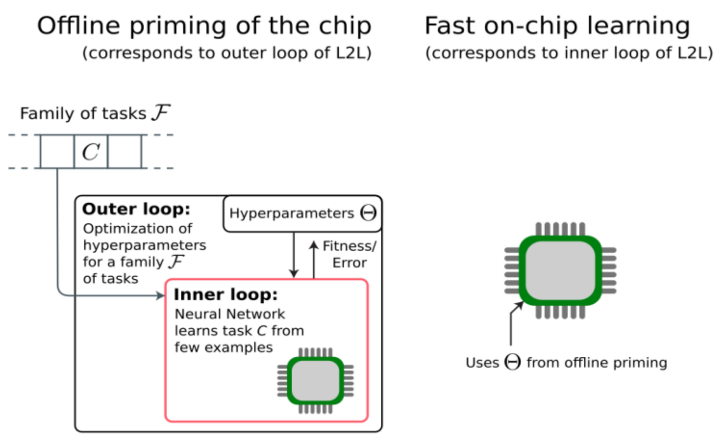
Offline priming of the chip
(corresponds to outer loop of L2L)
Fast on-chip learning
(corresponds to inner loop of L2L)
Family of tasks $\mathcal{F}$
C
Outer loop:
Optimization of hyperparameters for a family $\mathcal{F}$ of tasks
Hyperparameters Θ
Fitness/ Error
Inner loop:
Neural Network learns task C from few examples
Uses Θ from offline priming

Figure 1. Scheme for the application of L2L for offline priming of a neuromorphic chip. Hyperparameters Θ of the RSNN on the chip are optimized for supporting fast learning of arbitrary tasks C from a family $\mathcal{F}$ that captures learning challenges that may arise in the hands of a user. The resulting hyperparameters are then loaded onto the chip. Note that the desired generalization capability is here more demanding than usually: The chip also needs to learn tasks C from the family $\mathcal{F}$ very fast that did not occur during offline priming (but share structural properties with other tasks in the family $\mathcal{F}$).

1. Just the learning rate parameters of on-chip learning rules are hyperparameters.
2. Also the values of all synaptic weights of the RSNN are hyperparameters.
3. Only the initial values of the synaptic weights of the RSNN are hyperparameters.
4. In addition all parameters of an auxiliary NN —the learning signal generator— that generates online learning signals L_j^t for fast convergence of e-prop are hyperparameters.

Option 1 has been explored for RSNNs in [6, 7], and with an application to analog neuromorphic hardware in [8]. Option 2 is arguably the most commonly considered application of L2L in machine learning and computational neuroscience models [5, 9-13]. An attractive feature of this option for realizing fast on-chip learning is that it requires no synaptic plasticity for that. Rather, it uses hidden variables of the RNN for storing information from the few training examples that are needed for fast learning. In the case of machine learning, these hidden variables are the values of memory cells of LSTM units. In spiking neural networks these are the current values of firing thresholds of adapting neurons. An alternative is to choose only some synaptic weights to be hyperparameters, and to leave others open for fast on-chip learning [14]. Option 3 is used by the MAML approach of [15], where only very few updates of synaptic weights via BPTT are required in the inner loop of L2L. It also occurs in [4] in conjunction with option 4, see Figure 2 for an illustration.
One common challenge that underlies the success of all mentioned options, is the efficacy of the training algorithm for the offline priming phase, the outer loop of L2L. While option 1 can often be carried out by gradient-free methods, the more demanding network optimizations of the other options tend to require BPTT for offline priming of the RNN.

Advances in Science and Technology to Meet Challenges

It is quite realistic to enable according to this L2L method fast on-chip learning on neuromorphic hardware. The most demanding aspect for the hardware is to be able to run the on-chip learning algorithm that is required. This can be implemented on most neuromorphic hardware if only simple local rules for synaptic plasticity are required in the inner loop of L2L, as in option 1. In the case of option 2 a spike-based neuromorphic hardware just needs to be able to emulate adapting spiking neurons. This can be done for example on SpiNNaker [16] and Intel's Loihi chip [17]. Using BPTT for on-chip learning appears to be currently infeasible, but on-chip learning with e-prop is supported by

SpiNNaker and the next generation of Loihi. Then option 4 can be used for enabling more powerful fast on-chip learning. The only additional requirements are that an offline primed learning signal generator can be downloaded onto the chip (once and for all), and that the chip supports communication of learning signals for gating local synaptic plasticity rules. A sample application is illustrated in Figure 2: On-chip learning and generalization of a new spoken command from a single example.

Future advances need to address the challenge of training extended learning problems during the offline phase. Besides improved gradient-based algorithms, also gradient-free training methods such as Evolution Strategies [18] are attractive for that. In fact, since the latter paradigm allows to employ neuromorphic hardware directly for evaluating the learning performance, this approach can benefit from the speed and efficiency of fast neuromorphic devices, as in [8]. Particularly fast neuromorphic hardware such as Brainscales [19] might support then even more powerful offline priming with training algorithms that could not be carried out on GPU-based hardware.

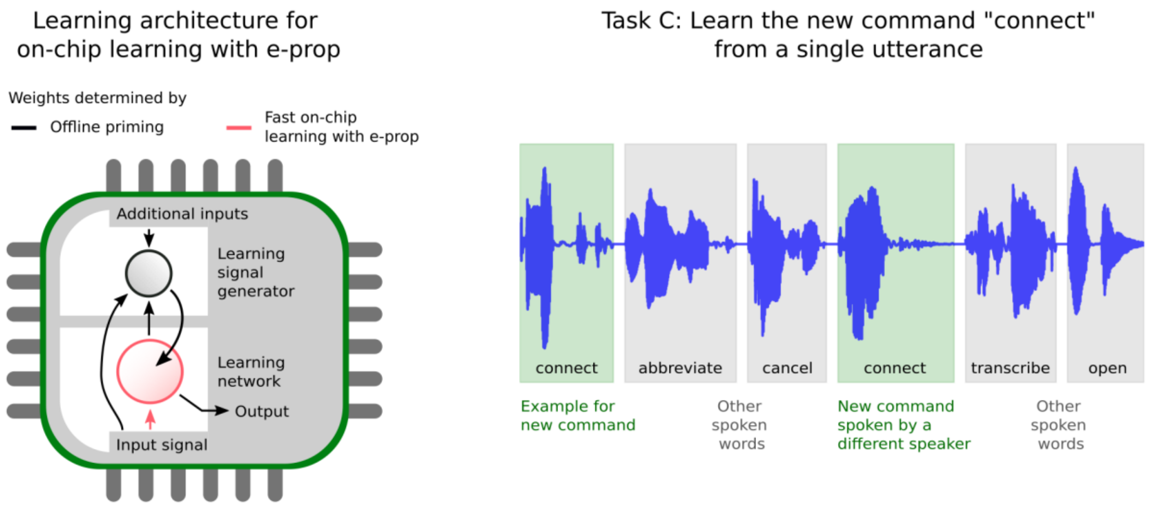
Learning architecture for on-chip learning with e-prop
Weights determined by
Offline priming
Fast on-chip learning with e-prop
Additional inputs
Learning signal generator
Learning network
Output
Input signal
Task C: Learn the new command "connect" from a single utterance
connect
abbreviate
cancel
connect
transcribe
open
Example for new command
Other spoken words
New command spoken by a different speaker
Other spoken words

Figure 2. Left: Learning architecture for fast on-chip learning with e-prop. A learning signal generator produces online learning signals for fast on-chip learning. The weights of the learning signal generator as well as the initial weights of the learning network result from offline priming. **Right: Example application for fast learning**. In this task C, the learning network has to learn the new command "connect" from a single utterance, so that it recognizes it also from other speakers. The learning signal generator is only activated when the new command is learnt (leftmost green segment).

Concluding Remarks

Learning in neuromorphic hardware is likely to become split into two phases that each have different goals and require different learning methods: An extensive offline priming phase —either on the actual hardware or a software model for it— that optimizes selected hyperparameters but possibly also the network architecture for a large family of potential on-chip learning tasks in the hands of the user. The resulting hyperparameters and network architectures will be downloaded onto the neuromorphic hardware before it gets into the hands of the user. The hardware is then primed so that remaining open parameters can be learnt on-chip from very few examples, possibly even just one example. It is conceivable that this method can be expanded to provide another useful property for neuromorphic hardware in the hands of the user: That on-chip learning cannot bring the chip into an operating regime which is unsafe, or undesired for other reasons. It has already been verified that the outer loop of L2L can impose powerful priors for subsequent computing and learning of RSNNs [13].

Acknowledgements
This research was supported by the Human Brain Project (Grant Agreement number 785907) of the European Union and a grant from Intel. We thank Christoph Stöckl for helpful comments.

References
[1] Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J., “Building machines that learn and think like people” in Behavioral and brain sciences, 2017, 40.
[2] Murray, J. M., “Local online learning in recurrent networks with random feedback” in eLife, 8, 2019, e43299.
[3] Bellec, G., Scherr, F., Subramoney, A., Hajek, E., Salaj, D., Legenstein, R., & Maass, W., “A solution to the learning dilemma for recurrent networks of spiking neurons” in Nature Communications, 11, 2020, 3652.
[4] Scherr, F., Stöckl, C., & Maass, W., “One-shot learning with spiking neural networks”, bioRxiv, 2020.
[5] Hochreiter, S., Younger, A. S., & Conwell, P. R., “Learning to learn using gradient descent” in International Conference on Artificial Neural Networks (pp. 87-94). Springer, Berlin, Heidelberg, 2020.
[6] Confavreux, B., Zenke, F., Agnes, E. J., Lillicrap, T., & Vogels, T. P., “A meta-learning approach to (re) discover plasticity rules that carve a desired function into a neural network”, bioRxiv, 2020.
[7] Jordan, J., Schmidt, M., Senn, W., & Petrovici, M. A., “Evolving to learn: discovering interpretable plasticity rules for spiking networks”, arXiv preprint, 2020, arXiv:2005.14149.
[8] Bohnstingl, T., Scherr, F., Pehle, C., Meier, K., & Maass, W., “Neuromorphic hardware learns to learn” in Frontiers in neuroscience, 13, 2019, 483.
[9] Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., ... & Botvinick, M., “Learning to reinforcement learn”, arXiv preprint, 2016, arXiv:1611.05763.
[10] Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., & Abbeel, P., “Rl $^ 2$: Fast reinforcement learning via slow reinforcement learning”, arXiv preprint, 2016, arXiv:1611.02779.
[11] Wang, J. X., Kurth-Nelson, Z., Kumaran, D., Tirumala, D., Soyer, H., Leibo, J. Z., ... & Botvinick, M., “Prefrontal cortex as a meta-reinforcement learning system” in Nature neuroscience, 21(6), 2018, 860-868.
[12] Bellec, G., Salaj, D., Subramoney, A., Legenstein, R., & Maass, W., “Long short-term memory and Learning-to-learn in networks of spiking neurons” in Advances in Neural Information Processing Systems, 2018, pp. 787–797
[13] Subramoney, A., Bellec, G., Scherr, F., Legenstein, R., & Maass, W., "Revisiting the role of synaptic plasticity and network dynamics for fast learning in spiking neural networks", bioRxiv, 2020.
[14] Subramoney, A., Scherr, F., & Maass, W., “Reservoirs learn to learn”, arXiv preprint, 2019, arXiv:1909.07486.
[15] Finn, C., Abbeel, P. & Levine, S., “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks” in Proceedings of the 34th International Conference on Machine Learning” in PMLR, 2017, 70:1126-1135
[16] Furber, S. B., Lester, D. R., Plana, L. A., Garside, J. D., Painkras, E., Temple, S., & Brown, A. D., “Overview of the spinnaker system architecture” in IEEE Transactions on Computers, 62(12), 2012, 2454-2467.
[17] Davies, M., Srinivasa, N., Lin, T. H., Chinya, G., Cao, Y., Choday, S. H., ... & Wang, H., “Loihi: A neuromorphic manycore processor with on-chip learning” in Ieee Micro, 38(1), 2018, 82-99.

[18] Salimans, T., Ho, J., Chen, X., Sidor, S., & Sutskever, I., “Evolution strategies as a scalable alternative to reinforcement learning”, arXiv preprint, 2017, arXiv:1703.03864.
[19] Grübl, A., Billaudelle, S., Cramer, B., Karasenko, V., & Schemmel, J., “Verification and design methods for the BrainScaleS neuromorphic hardware system” in Journal of Signal Processing Systems, 92(11), 2020, 1277-1292.