ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

A variational framework for local learning with probabilistic latent representations

Cabrel Teguemne Fokam[1,5], Khaleelulla Khan Nazeer[2], Christian Mayr[2,3], Anand Subramoney[4] and David Kappel[1,5] *

[1] Institut für Neuroinformatik, Ruhr Universität Bochum, Germany

[2] Faculty of Electrical and Computer Engineering, and
[3] Centre for Tactile Internet with Human-in-the-Loop (CeTI), Technische Universität Dresden, Germany

[4] Department of Computer Science, Royal Holloway, University of London, UK

[5] CITEC, Bielefeld University, Germany

Abstract. We introduce a novel method for distributed learning by dividing deep neural networks into blocks and incorporating feedback networks to propagate target information backwards, enabling auxiliary local losses. Forward and backward propagation operate in parallel with independent weights, addressing locking and weight transport problems. Our approach is rooted in a statistical view of training, treating block output activations as parameters of probability distributions to measure alignment between forward and backward passes. Error backpropagation is then performed locally within blocks, hence *block-local learning*. Preliminary results across tasks and architectures showcase state-of-the-art performance, establishing a principled framework for asynchronous distributed learning.

1 Introduction

The error backpropagation algorithm, that dominates today's deep learning methods, requires a sequential alternation of forward and backward phases. This introduces a locking problem because each phase must wait for the other [1]. Furthermore, the two phases rely on the same weight matrices to compute updates, which makes it impossible to separate memory spaces, known as the weight transport problem [2, 3]. Locking and weight transport problems, make efficient parallelization of machine learning models is extremely difficult, especially for low-resource settings.

We propose a new method to address these problems, to distribute a globally defined optimization algorithm across computing devices using only local updates. Our approach, called block-local learning (BLL), is derived from variational inference that provides auxiliary local targets through a separate feedback network that back-propagates information backwards from the targets. Messages are communicated forward and backward in parallel. Updates use local losses

*CTF and KKN are funded by BMBF project EVENTS (16ME0733). DK is funded by the Ministry of Culture and Science of the State of North Rhine-Westphalia under project SAIL (grant no. NW21-059A) and BMWK project ESCADE (01MN23004D). The authors gratefully acknowledge the Gauss Centre for Supercomputing e.V. (`www.gauss-centre.eu`) for providing computing time on GCS JUWELS at Jülich Supercomputing Centre (JSC).

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

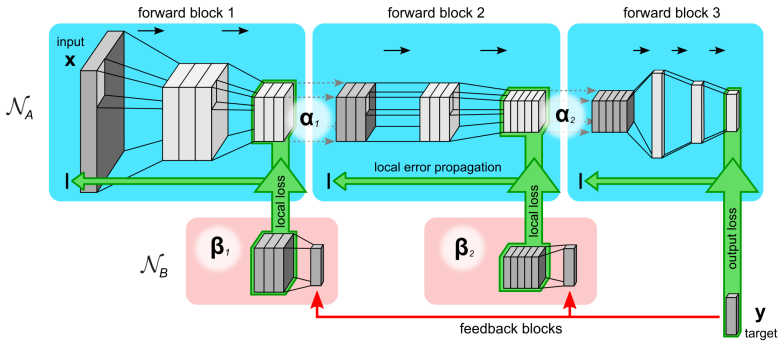

Fig. 1: Block-local learning. A deep neural network architecture $\mathcal{N}_A$ is split into multiple blocks (forward blocks) and trained on an auxiliary local loss. Targets for local losses are provided by a feedback network $\mathcal{N}_B$.

calculated using the targets provided by the feedback messages. Several models have been proposed previously that used random feedback weights to provide local targets, such as feedback alignment [3], target propagation [4] and related approaches [5]. Some previous methods are also based on probabilistic or energy-based cost functions, e.g. contrastive learning [6], equilibrium propagation [7] or forward propagation [8]. Furthermore, [9] have used greedy local, block- or layer-wise optimization, and achieved good results by combining different local losses. In contrast to these previous results, BLL optimizes local losses without requiring a contrastive step where different positive and negative samples are propagated through the network. Within each block, conventional error back-propagation is performed locally ("block local"), thus mitigating the locking problem and solving the weight transport problem. Our method provides a new principled method for distributing network training across multiple computing devices and shows promising first results across several benchmark tasks.

2 Probabilistic formulation of block-local learning

The BLL architecture is illustrated in Fig. 1. The neural network $\mathcal{N}_A$, that forward-propagates inputs $\mathbf{x}$ in the conventional fashion, is split into $N+1$ blocks. Intermediate outputs α_k, at each block k are matched against auxiliary targets β_k provided through feedback blocks $\mathcal{N}_B$. The feedback blocks only use the target vector $\mathbf{y}$ as input and can therefore be computed in parallel and independently. Error back-propagation is used locally within each forward block, i.e. block local learning (BLL). Backward blocks can be updated using convex optimization on each training batch, adding only a small compute overhead.

To derive BLL we use a probabilistic interpretation of deep learning. The problem of minimizing any loss function $\mathcal{L}$ in a deep neural networks can be reformulated in terms of maximum likelihood (ML), by defining the probability

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

of data samples $(\mathbf{x}, \mathbf{y})$ as probability[1] $p(\mathbf{y} \,|\, \mathbf{x}) \propto e^{-\mathcal{L}}$. The equivalent learning problem is to minimize the negative log-likelihood $\mathcal{L} = -\log p(\mathbf{y} \,|\, \mathbf{x})$ with respect to the network parameters $\boldsymbol{\theta}$ [10]. This probabilistic interpretation of deep learning can be used to define block-local losses and distribute learning across the network. Splits of the network into blocks k can be interpreted in the ML formalism by introducing latent variables[2] $\mathbf{z}_k$ such that $p(\mathbf{y} \,|\, \mathbf{x}) = \mathbb{E}\left[p(\mathbf{y} \,|\, \mathbf{z}_k)\, p(\mathbf{z}_k \,|\, \mathbf{x})\right]$. At block k, the network outputs the parameters of a probability distribution $\alpha_k(\mathbf{z}_k) = p(\mathbf{z}_k \,|\, \mathbf{x})$ (e.g., means and variances if α_k is Gaussian). The network thus translates $\alpha_{k-1} \to \alpha_k \to \ldots$ by outputting the statistical parameters of the conditional distribution $\alpha_k(\mathbf{z}_k)$ and taking the $\alpha_{k-1}(\mathbf{z}_{k-1})$ parameters as input.

More specifically, the network implicitly calculates the following expectation

$$\alpha_k(\mathbf{z}_k) \;=\; p(\mathbf{z}_k \,|\, \mathbf{x}) \;=\; \mathbb{E}\left[p_k(\mathbf{z}_k \,|\, \mathbf{z}_{k-1})\, \alpha_{k-1}(\mathbf{z}_{k-1})\right] \;=\; f_k(\alpha_{k-1}, \boldsymbol{\theta}_k)\;, \quad (1)$$

with state transition probability p_k and block-local network parameters $\boldsymbol{\theta}_k$. Eq. (1) is an instance of the belief propagation algorithm for the chain of latent variables $\mathbf{z}_k, \mathbf{z}_{k+1}, \ldots$ Consequently, the whole network realizes a conditional probability distribution $p(\mathbf{y} \,|\, \mathbf{x})$ as in the maximum-likelihood formulation above, where $\mathbf{x}$ and $\mathbf{y}$ are network inputs and outputs, respectively.

2.1 Distributed variational learning

We construct and use an upper bound $\mathcal{F}$ on the log-likelihood loss $\mathcal{L}$ for training the model. $\mathcal{F}$ uses the backward network $\mathcal{N}_B$ to introduce a variational distribution q. If constructed suitably, the variational upper bound can replace $\mathcal{L}$ with local losses. The variational posterior $\rho_k(\mathbf{z}_k)$ can be computed up to normalization by propagating forward messages $\alpha_k(\mathbf{z}_k)$ and feedback messages $\beta_k(\mathbf{z}_k)$ forward and backward through the network. Importantly, $\alpha_k(\mathbf{z}_k)$ and $\beta_k(\mathbf{z}_k)$ can have separated parameter spaces, which we denote by $\boldsymbol{\theta}_k$ and $\boldsymbol{\phi}_k$, respectively. We used a single linear layer to back-propagate β-messages. The variational posterior is then given by

$$\rho_k(\mathbf{z}_k) \;=\; q_k(\mathbf{z}_k \,|\, \mathbf{x}, \mathbf{y}) \quad \propto \quad p(\mathbf{z}_k \,|\, \mathbf{x})\; q(\mathbf{y} \,|\, \mathbf{z}_k) \;=\; \alpha_k(\mathbf{z}_k)\; \beta_k(\mathbf{z}_k)\;. \quad (2)$$

The variational objective minimizes the ML loss $\mathcal{L}$ *and* the mismatch between p and q

$$\mathcal{F} \;=\; -\log p(\mathbf{y} \,|\, \mathbf{x}) + \frac{1}{N}\sum_{k=1}^{N} \mathcal{D}_{KL}\left(q_k \,|\, p_k\right) \quad \geq\; \mathcal{L}\,, \quad (3)$$

Simple version:

$$\mathcal{F} \;=\; \mathcal{L} \;+\; \frac{1}{N}\sum_{k=1}^{N} \mathcal{D}_{KL}\left(\mathbf{a}_k \,|\, \mathbf{b}_k\right) \quad \geq\; \mathcal{L}\,, \quad (4)$$

[1] Learning the prior $p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y})$ over input samples $\mathbf{x}$ is often of lower relevance and is ignored here for brevity. Therefore, we focus on directly optimizing $p(\mathbf{y} \,|\, \mathbf{x})$.

[2] At no point does the network produce samples of the implicit random variables $\mathbf{z}_k$, they are introduced here only to conceptualize the mathematical framework.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

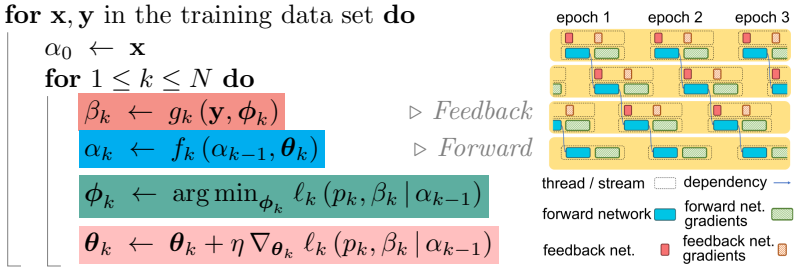

Fig. 2: Left: Pseudo code of the BLL training algorithm. f_k and g_k are the transfer functions of forward and feedback blocks, respectively. The *for*-loops can be interleaved and run in parallel. Right: Timeline of execution for BLL.

where p_k and q_k are true and variational posterior distributions, and the Kullback-Leibler divergence $\mathcal{D}_{KL}(p \,|\, q) = \mathbb{E}_p[\log p - \log q]$ measures their mismatch. Conveniently, using the Markov property of the network blocks outlined above we can rewrite this form and identify an upper bound on $\mathcal{L}$, that only consists of block-local losses ℓ_k at all network blocks k, and that can be constructed using the forward and feedback networks $\mathcal{N}_A$ and $\mathcal{N}_B$, given by

$$\ell_k(p_k, \beta_k \,|\, \alpha_{k-1}) \;=\; \mathcal{D}_{KL}(q_k \,|\, \alpha_k) \;+\; \mathcal{H}(p_k \,|\, \alpha_{k-1}) \;, \tag{5}$$

Simple version:

$$\ell_k(\mathbf{a}_k, \mathbf{b}_k) \;\approx\; \ell_k(\mathbf{a}_k, \mathbf{b}_k)\, \mathcal{D}_{KL}(q_k \,|\, \alpha_k) \;+\; \mathcal{H}(p_k \,|\, \alpha_{k-1}) \;, \tag{6}$$

where the first term measures the mismatch between the variational posterior ρ_k and the forward message α_k. The second term is the entropy loss of the forward network $\mathcal{H}(p_k \,|\, \alpha_{k-1}) \;=\; -\mathbb{E}[\alpha_{k-1}(\mathbf{z}_{k-1}) \log p_k(\mathbf{z}_k \,|\, \mathbf{z}_{k-1})]$.

The loss in Eq. (6) is local in the sense that it is completely determined by the information available at block k, i.e., the forward message from the previous block α_{k-1}, and the feedback β_k. Furthermore, the loss is local with respect to learning, i.e. it doesn't require global signals to be communicated to each block. In this sense, our approach differs from previous contrastive methods that need to distinguish between positive and negative samples. Any sample that passes through a block can be used directly for weight updating, by local greedy optimization of the losses (6). For linear backward networks, the loss (6) is convex such that optimal $\boldsymbol{\phi}_k$ can be directly computed over a mini-batch, adding little extra compute for feedback messages. Forward network parameters $\boldsymbol{\theta}_k$ are optimized by conventional error back-propagation locally and in parallel.

2.2 Variational greedy block-local learning

To derive concrete losses and update rules for the forward and backward networks, we assume that α_k's and β_k's are channel-wise independent univariate Gaussian distributions with known variance. For the Gaussian case considered

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

	Fashion-MNIST ResNet-50	CIFAR-10 ResNet-50	CIFAR-100 ResNet-50
BP	**93.4 ± 0.60**	**94.8 ± 0.25**	**77.8 ± 0.21**
FA	86.6 ± 0.70	62.5 ± 0.40	-
Pred-Sim	**94.2 ± 0.20**	92.1 ± 0.20	71.7 ± 0.76
BLL	94.1 ± 0.24	**94.6 ± 0.17**	**77.9 ± 0.10**

Table 1: Classification accuracy (% correct) on vision tasks. BP: end-to-end backpropagation, FA: Feedback Alignment, Sim Loss: Local learning with similarity matching loss [9], BLL: block local learning.

here ρ_{kj}'s are simply $\rho_{kj} = \frac{1}{2}(\alpha_{kj} + \beta_{kj})$, however, the framework outlined above can easily be generalized to a more complex probability distribution. A feed-forward DNN $\mathcal{N}_A : \mathbf{x} \to \mathbf{y}$, can be split into $N+1$ blocks by introducing implicit latent variables $\mathbf{z}_k : \mathbf{x} \to \mathbf{z}_k \to \mathbf{y}$. Blocks can be separated after any arbitrary layer and we used 4 blocks throughout all experiments.

In summary, the BLL algorithm is shown in Figure 2. The two *for*-loops can be interleaved and parallelized by pipelining the propagation of data samples through the network (see illustration: right). Updates can be computed as soon as propagation through a given block is complete. There is no locking since only the data labels $\mathbf{y}$ are needed to compute the output of the backward network. Furthermore, there is no weight transport problem since parameter spaces are separated and updates are computed only locally.

3 Results

We evaluated the BLL algorithm on three vision tasks: Fashion-Mnist, CIFAR-10 and CIFAR-100. Its performance is compared for the ResNet50 architecture with that of Backpropagation (BP), Feedback Alignment [3] (FA) and Local learning using similarity matching loss, Pred-Sim [9]. The network was divided into four blocks for BLL and Pred-Sim. The splits were introduced after residual layers by grouping subsequent layers into blocks. We included the predictive loss as suggested in [9] as an additional target in our BLL method. Gradient flow was stopped at block boundaries for all losses.

Block sizes were (12,13,12,13) for ResNet-50. Backward networks for BLL were constructed as linear layers with label size as input and output size equal to the number of channels in the corresponding ResNet block output. The kernels of ResNet-50 used by FA architectures during backpropagation were fixed and uniformly initialized following the Kaiming initialisation method [11].

The results are summarized in Table 1, showing the mean and standard deviation over 3 runs for all methods. We do not report results for FA on CIFAR-100 due to limitations in the current PyTorch implementation. BLL performs on par or slightly better than Pred-Sim overall for all tasks. FA was outperformed by BLL, where the gap became wider as the task and model complexity increased [12]. BLL performs comparably to end-to-end backpropagation for all datasets.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

4 Discussion

This work tackles a key challenge in modern ML: distributing and horizontally scaling ML models across multiple compute nodes for training models too large for a single node. This approach enhances efficiency and enables the use of smaller, energy-efficient devices, such as edge hardware. We propose a probabilistic framework for defining block-local losses in deep architectures, achieving comparable or slightly better performance than prior methods, and reaching the performance of standard end-to-end error back-propagation on small-scale and middle-scale vision tasks. However, performance may degrade with excessive splits, as local losses may lack sufficient feedback. Our framework's flexibility allows for multi-layer feedback networks, opening avenues for new block-local parallel training models in energy-constrained, distributed settings.

References

[1] Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled Neural Interfaces using Synthetic Gradients. *arXiv:1608.05343 [cs]*, August 2016.

[2] Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.

[3] Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random feedback weights support learning in deep neural networks. *arXiv:1411.0247*, November 2014.

[4] Charlotte Frenkel, Martin Lefebvre, and David Bol. Learning without feedback: Fixed random learning signals allow for feedforward training of deep neural networks. *Frontiers in Neuroscience*, 15, February 2021.

[5] Arild Nøkland. Direct Feedback Alignment Provides Learning in Deep Neural Networks. *arXiv:1609.01596 [cs, stat]*, September 2016.

[6] Bernd Illing, Jean Ventura, Guillaume Bellec, and Wulfram Gerstner. Local plasticity rules can learn deep representations using self-supervised contrastive predictions. In *Advances in Neural Information Processing Systems*, volume 34, pages 30365–30379, 2021.

[7] Benjamin Scellier and Yoshua Bengio. Equilibrium propagation: Bridging the gap between energy-based models and backpropagation. *Frontiers in computational neuroscience*, 11:24, 2017.

[8] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2022.

[9] Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. In *International conference on machine learning*, pages 4839–4850. PMLR, 2019.

[10] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459, 2015.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

[12] Sergey Bartunov, Adam Santoro, Blake Richards, Luke Marris, Geoffrey E Hinton, and Timothy Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. *Advances in neural information processing systems*, 31, 2018.