Block-local learning with probabilistic latent representations

Anonymous Author(s)
Affiliation
Address
email

Abstract

The ubiquitous backpropagation algorithm requires sequential updates across blocks of a network, introducing a locking problem. Moreover, backpropagation relies on the transpose of weight matrices to calculate updates, introducing a weight transport problem across blocks. Both these issues prevent efficient parallelisation and horizontal scaling of models across devices. We propose a new method that introduces a twin network that propagates information backwards from the targets to the input to provide auxiliary local losses. Forward and backward propagation can work in parallel and with different sets of weights, addressing the problems of weight transport and locking. Our approach derives from a statistical interpretation of end-to-end training which treats activations of network layers as parameters of probability distributions. The resulting learning framework uses these parameters locally to assess the matching between forward and backward information. Error backpropagation is then performed locally within each block, leading to "block-local" learning. Several previously proposed alternatives to error backpropagation emerge as special cases of our model. We present results on various tasks and architectures, including transformers, demonstrating state-of-the-art performance using block-local learning. These results provide a new principled framework to train very large networks in a distributed setting and can also be applied in neuromorphic systems.

1 Introduction

Recent developments in machine learning have seen deep neural network architectures scaling to billions of parameters [Touvron et al., 2023, Brown et al., 2020]. This development has boosted the capabilities of these models to unprecedented levels but simultaneously pushed the computing hardware on which large network models are running to its limits. It is therefore becoming increasingly important to distribute learning algorithms over a large number of independent compute nodes. However, today's machine learning algorithms are ill-suited for distributed computing. The error backpropagation (backprop) algorithm requires an alternation of inter-depended forward and backward phases, introducing a locking problem (the two phases have to wait for each other) [Jaderberg et al., 2016a]. Furthermore, the two phases rely on the same weight matrices to calculate updates, introducing a weight transport problem across blocks [Grossberg, 1987, Lillicrap et al., 2014a]. These two issues make efficient parallelisation and horizontal scaling of large machine learning models across compute nodes extremely difficult.

We propose a new method to address these problems by distributing a globally defined optimisation algorithm across a large network of nodes that use only local learning. Our approach uses a message-passing approach that uses results from probabilistic models and communicates uncertainty messages forward and backwards between compute nodes in parallel. To do so, we augment a network

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

architecture with a twin network that propagates information backwards from the targets to the input to provide uncertainty measures and auxiliary targets for local losses. Forward and backward messages comprise information about extracted features and feature uncertainties and are matched against each other using local probabilistic losses. Importantly, forward and backward propagation can work in parallel, reducing the locking problem. Inside each block, conventional error backpropagation is performed locally ("block-local"). These local updates can be used in the forward network and its backward twin for adapting parameters during training. The developed theoretical learning provides a new principled method to distribute very large networks over multiple compute nodes. The solutions emerging from this framework show striking similarities to earlier models that used random feedback weights as local targets [Lillicrap et al., 2020, Frenkel et al., 2021] but also provide a principled way to train these feedback weights.

In summary, the contribution of this paper is threefold:

1. We provide a theoretical framework on how interpreting the representations of deep neural networks as probability distributions provides a principled approach for block-local training of these networks. This can be used to distribute learning and inference over many interacting neural network blocks for various neural network architectures.
2. We demonstrate an instance of this probabilistic learning model on several benchmark classification tasks, where classifiers are split into multiple blocks and trained without end-to-end gradient computation.
3. We demonstrate how this framework can be used to allow deep networks to produce uncertainty estimates over their predictions. This principle is showcased on an autoencoder network that automatically predicts uncertainties alongside pixel intensity values after training.

2 Related work

A number of methods for using local learning in DNNs had been introduced previously. Lomnitz et al. [2022] introduced Target Projection Stochastic Gradient Descent (tpSGD), which uses layer-wise SGD and local targets generated via random projections of the labels, but does not adapt the backward weights. LocoProp [Amid et al., 2022] uses a layer-wise loss that consists of a target term and a regularizer, which is used however to enable 2nd order learning and does not focus on distributing the gradient optimization. Jimenez Rezende et al. [2016] used a generative model and a KL-loss for local unsupervised learning of 3D structures.

Some previous methods are based on probabilistic or energy-based cost functions and use a contrastive approach with positive and negative data samples. Contrastive learning Chen et al. [2020], Oord et al. [2019] can be used to construct block-local losses Xiong et al. [2020], Illing et al. [2021]. Equilibrium propagation replaces target clamping with a target nudging phase [Scellier and Bengio, 2017]. Another interesting contrastive approach was recently introduced [Hinton, 2022, Ororbia and Mali, 2023, Zhao et al., 2023]. However, it needs task-specific negative examples. [Han et al., 2018] uses a local predictive loss to improve recurrent networks' performance. In contrast to these methods, our approach does not need separate positive and negative data samples and focuses on block-local learning.

Feedback alignment [Lillicrap et al., 2020, Sanfiz and Akrout, 2021] uses random projections to propagate gradient information backwards. Jaderberg et al. [2016b] used pseudo-reward functions which are optimized simultaneously by reinforcement learning to improve performance. Random feedback alignment [Amid et al., 2022, Refinetti et al., 2021] and related approaches [Clark et al., 2021, Nøkland, 2016, Launay et al., 2020], use fixed random feedback weights to back-propagate errors. [Jaderberg et al., 2017] used decoupled synthetic gradients for local training. Target propagation demonstrates non-trivial performance with random projections for target labels instead of errors [Frenkel et al., 2021]. In contrast to these methods, we provide a principled way to adapt feedback weights.

Other methods [Belilovsky et al., 2019, Löwe et al., 2019] used greedy local, block- or layer-wise optimization. Notably, Nøkland and Eidnes [2019] achieved good results by combining a matching and a local cross-entropy loss. [Siddiqui et al., 2023] recently used block-local learning based on a cross-correlation metric over feature embeddings [Zbontar et al., 2021], demonstrating promising

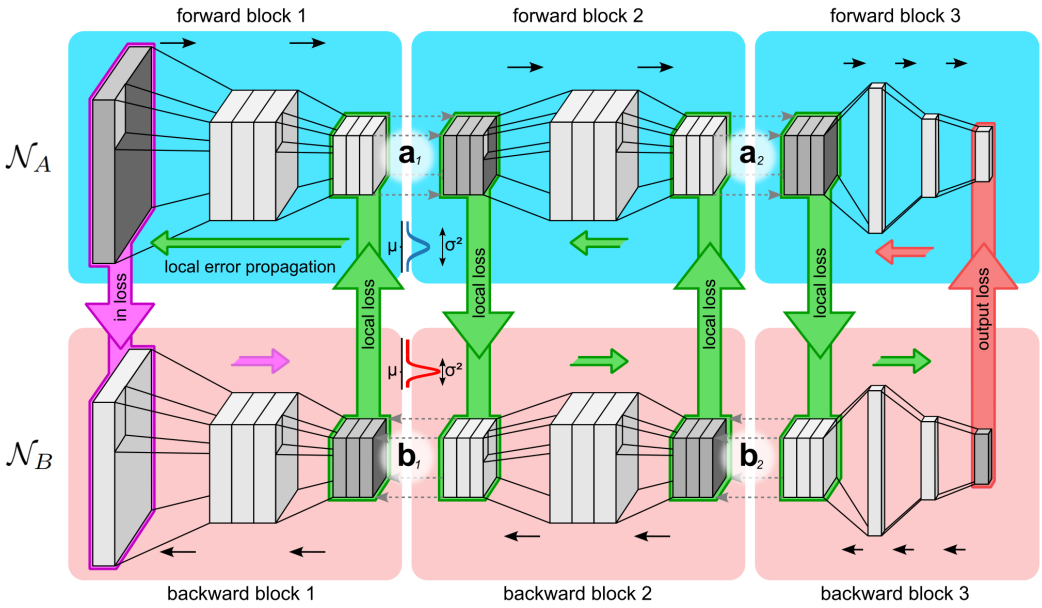

Figure 1: Illustration of use of block-local representations as learning signals on intermediate network layers. A deep neural network architecture $\mathcal{N}_A$ is split into multiple blocks (forward blocks) and trained on an auxiliary local loss. Targets for local losses are provided by a twin backward network $\mathcal{N}_B$.

performance. [Wu et al., 2021] used greedy layer-wise optimization of hierarchical autoencoders for video prediction. [Wu et al., 2022] used an encoder-decoder stage for pretraining. In contrast to these methods, we do not rely solely on local greedy optimization but provide a principled way to combine local losses with feedback information without locking and weight transport across blocks.

3 A probabilistic formulation of distributed learning

At a high level, our method interprets the activations of a neural network as the parameters of probability distributions of latent variables. We use these intermediate representations at each block to derive block local losses. These latent variables over multiple blocks implicitly define a Markov chain, which allows us to tractably minimize the block's local loss. We show that the derived block local losses and the resulting block local learning (BLL) are a general form of various existing local losses and provide an upper bound to a global loss.

3.1 Using latent representations to construct probabilistic block-local losses

Learning in deep neural networks can be formulated probabilistically [Ghahramani, 2015] in terms of maximum likelihood, i.e. the problem is to minimize the negative log-likelihood $\mathcal{L} = -\log p(\mathbf{x}, \mathbf{y}) = -\log p(\mathbf{y} \mid \mathbf{x}) - \log p(\mathbf{x})$ with respect to the network parameters $\boldsymbol{\theta}$. For many practical cases where we may not be interested in the prior distribution $p(\mathbf{x})$, we would like to directly minimize $\mathcal{L} = -\log p(\mathbf{y} \mid \mathbf{x})$.

This probabilistic interpretation of deep learning can be used to define block-local losses and distribute the learning over multiple blocks of networks by introducing intermediate latent representations. The idea is illustrated in Fig. 1. A neural network that computes the distribution $\log p(\mathbf{y} \mid \mathbf{x})$ takes $\mathbf{x}$ as input and outputs the statistical parameters to the conditional distribution. The deep neural network is split at an intermediate layer k (in Fig. 1 we used $k \in (1, 2)$) and end-to-end estimation of the gradient is replaced by two estimators that optimize the sub-networks $\mathbf{x} \rightarrow \mathbf{z}_k$ and $\mathbf{z}_k \rightarrow \mathbf{y}$ separately. To do this, consider the gradient of the log-likelihood loss function

$$-\frac{\partial}{\partial \theta}\mathcal{L} = \frac{\partial}{\partial \theta}\log p(\mathbf{y} \mid \mathbf{x}) \ . \tag{1}$$

For any deep network, it is possible to choose any intermediate activation at layer k as latent representations $\mathbf{z}_k$, such that $\log p(\mathbf{y} \mid \mathbf{x}) = \left\langle p(\mathbf{y} \mid \mathbf{z}_k)\ p(\mathbf{z}_k \mid \mathbf{x}) \right\rangle_{p(\mathbf{z}_k \mid \mathbf{x}, \mathbf{y})}$, where $\left\langle \ \right\rangle_p$ denotes expectation with respect to p. Therefore, the representations of $\mathbf{y}$ depend on $\mathbf{x}$ only through $\mathbf{z}_k$ as expected for a feed-forward network. Using this conditional independence property, the log-likelihood (1) expands to

$$-\frac{\partial}{\partial \theta}\mathcal{L} \ = \ \frac{\partial}{\partial \theta} \log p(\mathbf{y} \mid \mathbf{x}) \ = \ \left\langle \frac{\partial}{\partial \theta} \log p(\mathbf{y} \mid \mathbf{z}_k) + \frac{\partial}{\partial \theta} \log p(\mathbf{z}_k \mid \mathbf{x}) \right\rangle_{p(\mathbf{z}_k \mid \mathbf{x}, \mathbf{y})} . \tag{2}$$

This well-known result is the foundation of the Expectation-Maximization (EM) algorithm [Dempster et al., 1977]. Computing the marginal with respect to $p(\mathbf{z}_k \mid \mathbf{x}, \mathbf{y})$ corresponds to the E-step and calculating the gradients corresponds to the M-step. The sum inside the expectation separates the gradient estimators into two parts: $\mathbf{x} \rightarrow \mathbf{z}_k$ and $\mathbf{z}_k \rightarrow \mathbf{y}$.

However, the E-step is impractical to compute for most interesting applications because of the combinatorial explosion in the state space of $\mathbf{z}_k$. To get around this, we use a variational lower bound to EM, based on the ELBO loss $\mathcal{L}_V \ = \ -\log p(\mathbf{y} \mid \mathbf{x}) + \mathcal{D}_{KL}(q \mid p)$ [Mnih and Gregor, 2014] and demonstrate that this yields a practical solution to split gradients in a similar fashion to Eq. (2). In the next section, we describe how we construct the variational distribution q.

3.2 Auxiliary latent representations

As described earlier, the output of any layer of a DNN can be interpreted as parameters to a distribution over latent random variable $\mathbf{z}_k$. The sequence of blocks across a network therefore implicitly defines a Markov chain $\mathbf{x} \rightarrow \mathbf{z}_1 \rightarrow \mathbf{z}_2 \rightarrow \ldots$ (see Fig. 2A). This probabilistic interpretation of hidden layer activity is valid under relatively mild assumptions, studied in more detail in the Supplement. It is important to note that the network at no point produces samples from the implicit random variables $\mathbf{z}_k$, but they are introduced here only to conceptualize the mathematical framework. Instead the network outputs the parameters to $\alpha_k(\mathbf{z}_k)$ which is the probability distribution over $\mathbf{z}_k$ (e.g. means and variances if α_k is Gaussian). The network thus translates $\alpha_{k-1} \rightarrow \alpha_k \rightarrow \ldots$ by outputting the statistical parameters of the conditional distribution $\alpha_k(z_k)$ and taking $\alpha_k(z_{k-1})$ parameters as input. More precisely, the network implicitly computes a marginal distribution

$$\alpha_k(\mathbf{z}_k) \ = \ p(\mathbf{z}_k \mid \mathbf{x}) \ = \ \left\langle p_k(\mathbf{z}_k \mid \mathbf{z}_{k-1}) \right\rangle_{p(\mathbf{z}_{k-1} \mid \mathbf{x})} \ = \ \left\langle p_k(\mathbf{z}_k \mid \mathbf{z}_{k-1}) \right\rangle_{\alpha_{k-1}(\mathbf{z}_{k-1})} , \tag{3}$$

where $\left\langle \ \right\rangle_p$ denotes expectation with respect to the probability distribution p. Consequently, the network realizes a conditional probability distribution $p(\mathbf{y} \mid \mathbf{x})$ (where $\mathbf{x}$ and $\mathbf{y}$ are network inputs and outputs, respectively). And by the universal approximator property of deep neural networks, an accurate representation of this distribution can be learnt in the network weights through error back-propagation (as demonstrated for the example in Fig. 2). Eq. (3) is an instance of the belief propagation algorithm to efficiently compute conditional probability distributions.

To construct the variational distribution q we introduce the backward network $\mathcal{N}_B$ that propagates messages β_k backwards according to Eq. 4 (see Fig. 1 for an illustration). Inference over the posterior distribution $p(\mathbf{z}_k \mid \mathbf{x}, \mathbf{y})$ for any latent variable $\mathbf{z}_k$ can be made using the belief propagation algorithm, propagating messages $\alpha_k(\mathbf{z}_k)$ forward through the network using Eq. (3). In addition messages $\beta_k(\mathbf{z}_k)$ need to be propagated backward according to

$$\beta_k(\mathbf{z}_k) \ = \ p(\mathbf{y} \mid \mathbf{z}_k) \ = \ \left\langle p(\mathbf{y} \mid \mathbf{z}_{k+1}) \right\rangle_{p_k(\mathbf{z}_{k+1} \mid \mathbf{z}_k)} \ = \ \left\langle \beta_{k+1}(\mathbf{z}_{k+1}) \right\rangle_{p_k(\mathbf{z}_{k+1} \mid \mathbf{z}_k)} , \tag{4}$$

such that the posterior $p(\mathbf{z}_k \mid \mathbf{x}, \mathbf{y})$ can be computed up to normalization

$$\rho_k(\mathbf{z}_k) \ = \ p(\mathbf{z}_k \mid \mathbf{x}, \mathbf{y}) \quad \propto \quad p(\mathbf{z}_k \mid \mathbf{x})\ p(\mathbf{y} \mid \mathbf{z}_k) \ = \ \alpha_k(\mathbf{z}_k)\ \beta_k(\mathbf{z}_k) \ . \tag{5}$$

We make use of the fact that, through Eq. (3), the parameters of a probability distribution $p(\mathbf{z}_k \mid \mathbf{x})$ are a function of the parameters to $p(\mathbf{z}_i \mid \mathbf{x})$, for $0 < i < k$, e.g. if α is assumed to be Gaussian we have $\left(\mu(\alpha_k), \sigma^2(\alpha_k)\right) \ = \ f\left(\mu(\alpha_i), \sigma^2(\alpha_i)\right)$, where $\mu(.)$ and $\sigma^2(.)$ are the mean and variance of the distribution respectively. Thus, if a network outputs $\left(\mu(\alpha_i), \sigma^2(\alpha_i)\right)$ on layer i and $\left(\mu(\alpha_k), \sigma^2(\alpha_k)\right)$ on layer k, a suitable probabilistic loss function will allow the network to learn

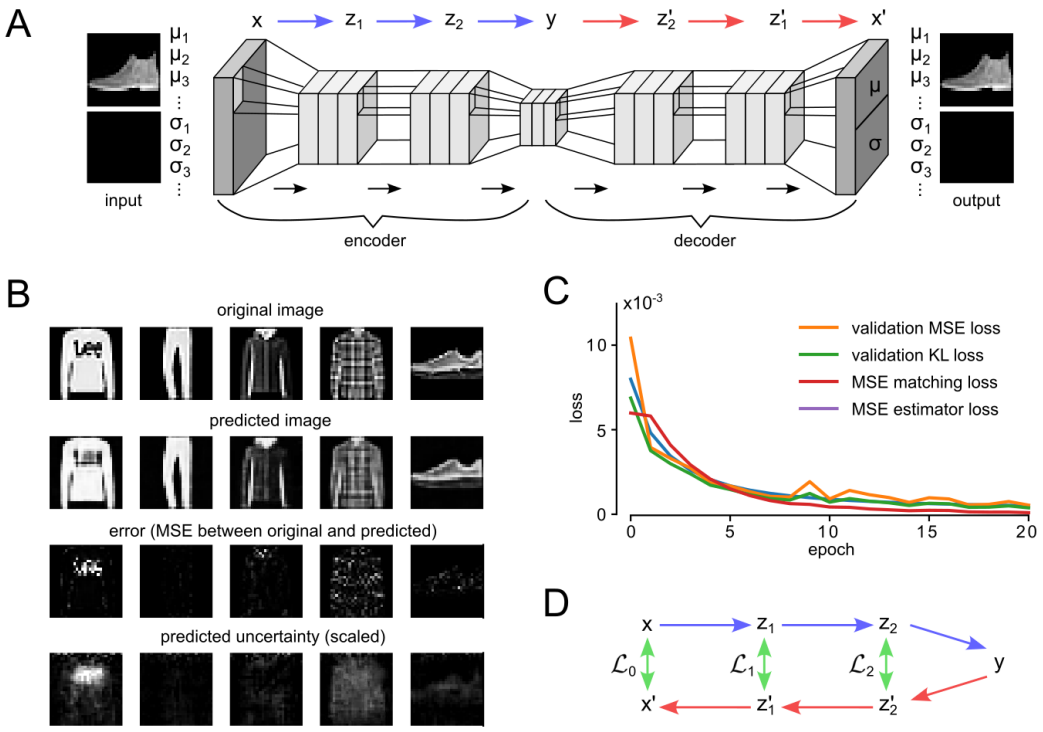

Figure 2: Zero shot learning of predicted uncertainties. **A:** Gaussian convolutional autoencoder network. Variance inputs and outputs are set to a constant during the whole training process. The network implements an implicit Markov chain. **B:** Example images showing self-prediction of uncertainties. **C:** Uncertainty mismatch metrics throughout learning. **D:** The network in (A) can be 'folded' to provide targets for local losses $\mathcal{L}_0, \mathcal{L}_1, \ldots$

f from examples. Therefore, the conditional distributions $p_k\left(\mathbf{z}_k \,|\, \mathbf{z}_{k-1}\right)$ and the expectation in Eq. (3) are only implicitly encoded in the network weights. We will study the exponential family of probability distributions for which this observation can be formalized more thoroughly.

Exponential family distributions: To derive concrete losses and update rules for the forward and backward networks, we assume that α_k are from the exponential family (EF) of probability distributions, given by

$$\alpha_k\left(\mathbf{z}_k\right) \;=\; \prod_j \alpha_{kj}\left(z_{kj}\right) \;=\; \prod_j h(z_{kj}) \exp\left(T\left(z_{kj}\right)\phi_{kj} - A\left(\phi_{kj}\right)\right) \;, \tag{6}$$

with base measure h, sufficient statistics T, log-partition function A, and natural parameters ϕ_{kj}. This rich class contains the most common distributions, such as Gaussian, Poisson or Bernoulli, as special cases. For the example of a Bernoulli random variable we have $z_{kj} \in \{0, 1\}$, $T\left(z_{kj}\right) = z_{kj}$ and $A\left(\phi_{kj}\right) = \log\left(1 + e^{\phi_{kj}}\right)$ [Koller and Friedman, 2009]. A network directly implements an EF distribution if the activations a_{kj} encode the natural parameters, $a_{kj} = \phi_{kj}$. Using this result, a feed-forward DNN $\mathcal{N}_A : \mathbf{x} \to \mathbf{y}$, can be split into N blocks by introducing implicit latent variables $\mathbf{z}_k : \mathbf{x} \to \mathbf{z}_k \to \mathbf{y}$, and generating the respective natural parameters. In principle, blocks can be separated after any arbitrary layer, but some splits may turn out more natural for a particular network architecture.

Conveniently, if both α_{kj} and β_{kj} are members of the EF with natural parameters a_{kj} and b_{kj}, then ρ_{kj} is also EF with parameters $a_{kj} + b_{kj}$. We will use this property to deconstruct a single global loss into multiple block-local losses.

3.3 Illustrative example: forward-backward networks as an autoencoder

Probability representations in DNNs are useful since they provide a principled way to represent uncertainties in the network. Before we establish our main result to show how a DNN can be deconstructed into local blocks, we first demonstrate how representations of Bayesian uncertainty can emerge in DNNs by using appropriate probabilistic losses. We consider the autoencoder network illustrated in Fig. 2A and use it to learn representations for the Fashion-MNIST dataset [Xiao et al., 2017]. The CNN comprises a bottleneck layer $\mathbf{y}$ that implicitly splits the architecture into a decoder and encoder part (Fig. 2A). It is well known that such a network is able to learn compact representations and features that allow it to reconstruct the gray scale pixel intensities of a given input [Kingma and Welling, 2013]. Here we demonstrate that autoencoders are also able to learn representations of uncertainties, i.e. to automatically output high uncertainties for pixel values that are poorly represented in the learnt features.

To show this, we augmented the pixel representations on the inputs and outputs with additional channels that represented the logarithms of the variances of a Gaussian distribution (see Supplement for details). The input and outputs now represent the parameters of probability distributions, where the variances are proxies for the uncertainties. An appropriate loss function for this architecture is one that measures the distance between probability distributions. We used the Kullback-Leibler (KL) divergence between Gaussian distributions. This augmentation to conventional deep auto-encoders requires us to also provide uncertainty values for training data samples. Since the Fashion-MNIST dataset does not contain this information, we set the variances of pixels for all training samples to the same small constant values, reflecting high confidence (low variance) in the training set. Thus, during training, the network has only seen the same constant inputs (and outputs) for the variance channels.

Fig. 2B shows representative sample outputs for the test dataset after training. As expected, the network is able to represent the means of gray scale values in the dataset well and generalize to new images. Interestingly, the network also learned meaningful representations of the variances. Although the network has only seen constant values for the variances during training, it is able to infer information about its own uncertainty during testing. The true MSE errors between inputs and predictions qualitatively match the pixel-level variance predictions across a wide variety of inputs. For example, the network poorly represents the logo on the shirt (leftmost example) and predicts high variance in the output for these pixels. Other samples like the trousers (second from left) that are well represented correctly predict low variance. To further quantify this result, we developed additional metrics that measure the mismatch between estimated and true prediction errors (Fig. 2C, see Supplement for details). These metrics consistently decrease throughout training even though they were not directly minimized. These results suggest that DNNs are able to represent uncertainties well enough that they show zero-shot generalizations to unseen data from very limited training data.

3.4 Modularized learning using local variational losses

The autoencoder example described in Section 3.3 shows that DNNs can represent probability distributions well in principle, and also provides an idea of how probabilistic losses could be constructed locally at any layer. By 'folding' the network along the bottleneck layer y we are able to construct a sequence of pairs of auxiliary targets $(\mathbf{z}_1, \mathbf{z}'_1), (\mathbf{z}_2, \mathbf{z}'_2), \ldots$ (see Fig. 2D). Finally, by introducing suitable loss functions $\mathcal{L}_0, \mathcal{L}_1, \ldots$, the mismatch between the encoder and decoder parts of the network can be minimized on a per-layer basis.

The forward and backward networks $\mathcal{N}_A$ and $\mathcal{N}_B$ can be used to construct local loss functions $\mathcal{L}_V^{(k)}$ at blocks k. In the Supplement, we show in detail that minimizing $\mathcal{L}_V^{(k)}$ locally and in parallel optimizes a lower bound to the log-likelihood loss $\mathcal{L}$ (Eq. 1), without propagating gradients end-to-end. To arrive at this result, we take the forward α_k and posterior messages ρ_k to be given by EF distributions with natural parameters ϕ_{kj} and γ_{kj}. Using this we show in the Supplement that the local loss can be optimized using the modularized gradient estimator

$$-\frac{\partial}{\partial\theta}\mathcal{L}_V^{(k)} \;=\; \sum_j \underbrace{\big(\mu\left(\rho_{kj}\right) - \mu\left(\alpha_{kj}\right)\big)}_{\textit{forward weight}} \frac{\partial}{\partial\theta}\phi_{kj} \;+\; \underbrace{\sigma^2\left(\rho_{kj}\right)\left(\phi_{kj} - \gamma_{kj}\right)}_{\textit{posterior weight}} \frac{\partial}{\partial\theta}\gamma_{kj}\;, \tag{7}$$

where $\mu(\cdot)$ and $\sigma^2(\cdot)$ are means and variances of EF distribution. Note that the gradients of the natural parameters ϕ_{kj} and γ_{kj} are computed independently and modulated by the *forward* and *posterior weight*, respectively.

The result in Eq. (7) holds for general EF distributions. For the special case of Bernoulli random variables we get

$$-\frac{\partial}{\partial\theta}\mathcal{L}_V^{(k)} \;=\; \sum_{k,j}\left(\rho_{kj}-\alpha_{kj}\right)\frac{\partial}{\partial\theta}a_{kj} \;-\; \rho_{kj}\;\left(1-\rho_{kj}\right)b_{kj}\left(\frac{\partial}{\partial\theta}a_{kj}+\frac{\partial}{\partial\theta}b_{kj}\right)\;, \tag{8}$$

where $a_{kj} = f_j(\mathbf{a}_{k-1})$ and $b_{kj} = g_j(\mathbf{b}_{k+1})$, are the outputs of the forward and backward network at block k,

$$\rho_{kj} = S\left(a_{kj} + m\, b_{kj}\right) \quad \text{and} \quad \alpha_{kj} = S\left(a_{kj}\right)\;, \tag{9}$$

where m is a mixing parameter described below and $S(x) = 1/1 + e^{-x}$ is the sigmoid/logistic function.

The Bernoulli solution in Eq. (8) is convenient because it is a single parameter distribution (mean and variance share one parameter) such that all channels in $\mathbf{z}$ can be treated independently. Also the structure of Eq. 9 is well suited for a DNN implementation. In our experiments, we focus on this Bernoulli variant of the general result in Eq. (7). In the Supplement, we study a number of other relevant members of the EF. Furthermore, it is interesting to study the structure of Eq. (8) more carefully. The first term minimizes the mismatch between the forward and the posterior distribution with respect to the forward blocks. The second term is the uncertainty-weighted backward activation b_{kj} which modulates local gradients (see Supplement). Therefore, the backward activations b_{kj} act directly as learning signals for local updates. The BLL method is therefore related to feedback alignment [Lillicrap et al., 2020] and target propagation [Frenkel et al., 2021] where backward information is provided through random weights. However, since the gradients of the backward blocks appear in the second term, our model also provides a principled way to optimize the backward flow of information from the targets.

Data mixing schedule: The equation for the posterior distribution Eq. 9 contains a data mixing parameter m, with $0 \leq m \leq 1$, that scales the influence of the backward messages in the posterior distribution. This parameter serves two important functions, (1) It scales the balance between forward and backward messages in the posterior distribution ρ and (2) it scales the first term in the parameter updates Eq. 8. We found that a annealing schedule for this parameter that decreases m slowly during learning works well in practice. If not stated otherwise, we used $m = (1 + \tau\, M)^{-1}$ in our experiments, where M is the index of the current epoch and τ is a scaling parameter (see the Supplement for further details).

4 Experimental results

We evaluated the BLL model on a number of vision and sequence learning tasks. All models used the Bernoulli BLL gradients described in Eq. (8) for local optimization. Additional details of the network models can be found in the Supplement.

4.1 Block-local learning of vision benchmark tasks

We compare the performance of our block local learning (BLL) algorithm with that of end-to-end backprop (BP) and Feedback Alignment (FA) Lillicrap et al. [2014b]. Three datasets are considered: MNIST, Fashion MNIST and CIFAR10 together with two residual network architectures [He et al., 2016]: ResNet-18 and ResNet-50, each trained with one of the three methods (BP, FA, BLL).

The BLL architectures were split into 4 blocks that were trained locally using the Bernoulli loss in Eq. (8). Splits were introduced after residual layers of the ResNet architecture by grouping subsequent layers into blocks. Group sizes were (4,5,4,5) for ResNet-18 and (12,13,12,13) for ResNet-50. Backward twin networks were here constructed simply by using the same network architecture (ResNet-18 or ResNet-50) in reverse order, introducing appropriate splits to provide intermediate targets. For CIFAR-10 gradients were propagated between two neighboring blocks (see Supplement for details and a comparison with purely local gradients). The kernels of ResNet-18/ResNet-50 + FA architectures used during backpropagation are fixed and uniformly initialised following the Kaiming He et al. [2015] initialisation method. The bias is set to one.

The results are summarized in Table 1. Test top-1, top-3 and train top-1 accuracies are shown. Top-3 accuracies count the number of test samples for which the correct class was among the network's

	MNIST			Fahion-MNIST			CIFAR-10		
	test-1	test-3	train-1	test-1	test-3	train-1	test-1	test-3	train-1
ResNet-18 + BP	99.5	100	99.7	92.7	99.3	96.0	95.2	99.3	100
ResNet-50 + BP	99.5	99.9	100	89.0	98.9	92.7	94.0	99.2	99.8
ResNet-18 + FA	99.0	99.9	100	87.9	98.6	92.1	70.4	92.5	80.9
ResNet-50 + FA	98.9	99.9	100	83.1	97.9	83.7	70.3	92.0	79.3
ResNet-18 + BLL	99.4	100	99.6	91.2	98.8	91.0	72.2	93.0	98.8
ResNet-50 + BLL	99.4	99.8	99.2	88.7	99.0	85.9	73.4	92.7	99.7

Table 1: Classification accuracy (% correct) on vision tasks. BP: end-to-end backprop, FA: feedback alignment, BLL: block local learning. Test-1, test-3 and train-1 represent the top-1, top-3 test accuracy and top-1 training accuracy respectively.

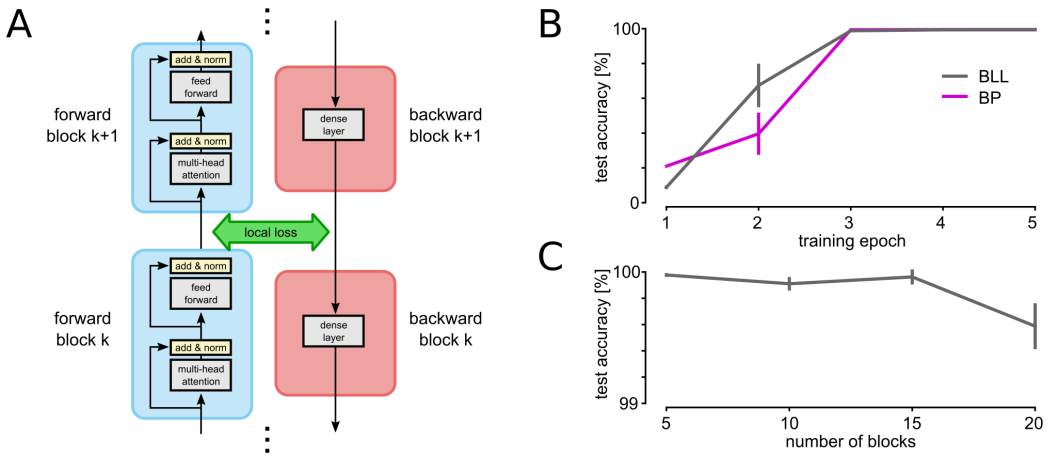

Figure 3: Block local learning of transformer architecture. **A:** Illustration of the transformer twin network. **B:** Learning curves of block local (BLL) and backprop (BP) training. **C:** Test accuracy vs. number of blocks in the transformer model. Error bars show standard deviations over 5 independent runs.

3 highest output activations. See Supplement for results over multiple runs. BLL achieved good performance on MNIST and Fashion-MNIST, closely matching end-to-end training and outperforming FA networks. Note that in contrast to FA and BP, BLL does not need to compute error gradients at the output but can work directly with the target labels. Performance on CIFAR-10 was significantly lower than BP but outperformed FA. Interestingly the performance on the training set was close to perfect for ResNet-50 suggesting over-fitting the task.

4.2 Block-local transformer architecture for sequence-to-sequence learning

Transformer architectures are in principle well suited for distributed computing due to their modular network structure that comprises a repetition of homogeneous blocks. We demonstrate a proof-of-concept result on training a transformer with BLL. We used a transformer model with 20 self-attention blocks with a single attention head each. Block local losses were added after each layer and blocks were trained locally. A backward twin network was constructed by projecting targets through dense layers and used the Bernoulli loss Eq. (8) for local training (see Fig. 3 A for an illustration). The transformer was trained on a sequence-to-sequence task, where a random permutation of numbers 0..9 was presented on the input and had to be re-generated at the output in reverse order. We trained the network for 5 epochs.

BLL achieves convergence speed that is comparable to that of end-to-end BP on this task. Fig. 3 B shows learning curves of BLL and BP. Both algorithms converge after around 3 epochs to nearly perfect performance. BLL also achieved good performance for a wide range of network depths. Fig. 3 C shows the performance after 5 epochs for different transformer architectures. Using only 5 transformer blocks yields performance of around 99.9% (average over five independent runs). The

test accuracy on this task for the 20 block transformer was 99.6%. These results suggest that the BLL method is equally applicable to transformer architectures.

5 Discussion

In this work, we have demonstrated a general purpose probabilistic framework for rigorously defining block-local losses for deep architectures. This not only provides a novel way of performing distributed training of large models but also hints at new paradigms of self-supervised training that are biologically plausible. We have also shown that our block-local training approach outperforms existing local training approaches while still getting around the locking and weight transport problems. Our method introduces a twin network that propagates information backwards from the targets to the input and automatically estimates uncertainties on intermediate layers. This is achieved by representing probability distributions in the network activations. The forward network and its backward twin can work in parallel and with different sets of weights.

The proposed method may also help further blur the boundary between deep learning and probabilistic models. A number of previous models have shown that DNNs are capable of representing probability distribution [Abdar et al., 2021, Pawlowski et al., 2017, Tran et al., 2019, Malinin and Gales, 2019]. Unlike these previous methods, our method does not require Monte Carlo sampling or contrastive training, but instead exploits the log-linear structure of exponential family distributions to efficiently propagate uncertainty-aware messages through a network using a belief-propagation strategy. We have demonstrated that implicit uncertainty messages can be learnt from sparse data and accurately represent the network's performance.

Greedy block-local learning has recently shown compelling performance on a number of tasks [Nøkland and Eidnes, 2019, Siddiqui et al., 2023]. These methods use local losses with an information-theoretic motivation but are agnostic to global back-propagating information. In future work, it may be interesting to combine these approaches with the proposed model to get the best of both worlds. Being able to produce block-level uncertainty predictions can also be useful for enhancing the sparsity of the network and using optimal amount of compute for predictions. The uncertainty predictions can also be used to handle missing labels, and for evaluating the model's confidence about its predictions. Since the framework is flexible enough to apply to self-supervised training, it can be used on unlabelled and multi-modal datasets as well. Due to the local nature of the training process, our method is particularly attractive for application on neuromorphic systems that co-locate memory and compute and use orders of magnitude less energy if the computation is local.

This work addresses potential problems of modern ML: The estimation of uncertainties in neural networks is an important open problem and understanding the underlying mechanisms better will likely help to make ML models safer and more reliable. Also the main focus of this work, which is on distributing large ML models over many compute nodes may make these model more energy efficient in the future. The energy consumption and resulting carbon footprint of ML is a major concern and the proposed model may provide a new direction to approach this problem. This method may enable training of larger models which also come with associated risks in terms of biases and inappropriate use in the real world. It is also not known what biases using this method itself and extensions with sparsity may introduce in the models predictions.

References

Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021.

Ehsan Amid, Rohan Anil, and Manfred Warmuth. LocoProp: Enhancing BackProp via local loss optimization. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, pages 9626–9642. PMLR, 2022. URL `https://proceedings.mlr.press/v151/amid22a.html`.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to ImageNet. In *Proceedings of the 36th International Conference on Machine Learning*, pages 583–593. PMLR, 2019. URL `https://proceedings.mlr.press/v97/belilovsky19a.html`.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*, July 2020. URL http://arxiv.org/abs/2005.14165.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

David Clark, L F Abbott, and Sueyeon Chung. Credit assignment through broadcasting a global error vector. In *Advances in Neural Information Processing Systems*, volume 34, pages 10053–10066. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/532b81fa223a1b1ec74139a5b8151d12-Abstract.html.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the *EM* algorithm. 39(1):1–22, 1977. ISSN 00359246. doi: 10.1111/j.2517-6161.1977.tb01600.x. URL https://onlinelibrary.wiley.com/doi/10.1111/j.2517-6161.1977.tb01600.x.

Charlotte Frenkel, Martin Lefebvre, and David Bol. Learning without feedback: Fixed random learning signals allow for feedforward training of deep neural networks. 15, 2021. ISSN 1662-453X. URL https://www.frontiersin.org/articles/10.3389/fnins.2021.629892.

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

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

Kuan Han, Haiguang Wen, Yizhen Zhang, Di Fu, Eugenio Culurciello, and Zhongming Liu. Deep predictive coding network with local recurrent processing for object recognition, 2018. URL http://arxiv.org/abs/1805.07526.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

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. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/feade1d2047977cd0cefdafc40175a99-Abstract.html.

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 2016a. URL http://arxiv.org/abs/1608.05343.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks, 2016b. URL http://arxiv.org/abs/1611.05397.

Max Jaderberg, Wojciech Marian Czarnecki, Simon Osindero, Oriol Vinyals, Alex Graves, David Silver, and Koray Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1627–1635. PMLR, 2017. URL https://proceedings.mlr.press/v70/jaderberg17a.html.

Danilo Jimenez Rezende, S. M. Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/hash/1d94108e907bb8311d8802b48fd54b4a-Abstract.html.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, December 2013. URL http://arxiv.org/abs/1312.6114.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Julien Launay, Iacopo Poli, François Boniface, and Florent Krzakala. Direct feedback alignment scales to modern deep learning tasks and architectures. In *Advances in Neural Information Processing Systems*, volume 33, pages 9346–9360. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/69d1fc78dbda242c43ad6590368912d4-Abstract.html.

Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random feedback weights support learning in deep neural networks. *arXiv:1411.0247 [cs, q-bio]*, November 2014a. URL http://arxiv.org/abs/1411.0247.

Timothy P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random feedback weights support learning in deep neural networks, 2014b.

Timothy P. Lillicrap, Adam Santoro, Luke Marris, Colin J. Akerman, and Geoffrey Hinton. Backpropagation and the brain. 21(6):335–346, 2020. ISSN 1471-0048. doi: 10.1038/s41583-020-0277-3. URL https://www.nature.com/articles/s41583-020-0277-3.

Michael Lomnitz, Zachary Daniels, David Zhang, and Michael Piacentino. Learning with local gradients at the edge, 2022. URL http://arxiv.org/abs/2208.08503.

Sindy Löwe, Peter O' Connor, and Bastiaan Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/851300ee84c2b80ed40f51ed26d866fc-Abstract.html.

Andrey Malinin and Mark Gales. Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness. *Advances in Neural Information Processing Systems*, 32, 2019.

Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. In *International Conference on Machine Learning*, pages 1791–1799. PMLR, 2014.

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

Arild Nøkland. Direct feedback alignment provides learning in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/hash/d490d7b4576290fa60eb31b5fc917ad1-Abstract.html.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. URL http://arxiv.org/abs/1807.03748.

Alexander Ororbia and Ankur Mali. The predictive forward-forward algorithm. *arXiv preprint arXiv:2301.01452*, 2023.

Nick Pawlowski, Andrew Brock, Matthew CH Lee, Martin Rajchl, and Ben Glocker. Implicit weight uncertainty in neural networks. *arXiv preprint arXiv:1711.01297*, 2017.

Maria Refinetti, Stéphane d'Ascoli, Ruben Ohana, and Sebastian Goldt. Align, then memorise: the dynamics of learning with feedback alignment, 2021. URL http://arxiv.org/abs/2011.12428.

Albert Jiménez Sanfiz and Mohamed Akrout. Benchmarking the accuracy and robustness of feedback alignment algorithms, 2021. URL http://arxiv.org/abs/2108.13446.

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

Shoaib Ahmed Siddiqui, David Krueger, Yann LeCun, and Stéphane Deny. Blockwise self-supervised learning at scale, 2023. URL http://arxiv.org/abs/2302.01647.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and Efficient Foundation Language Models, February 2023. URL http://arxiv.org/abs/2302.13971.

Dustin Tran, Mike Dusenberry, Mark Van Der Wilk, and Danijar Hafner. Bayesian layers: A module for neural network uncertainty. *Advances in neural information processing systems*, 32, 2019.

Bohan Wu, Suraj Nair, Roberto Martin-Martin, Li Fei-Fei, and Chelsea Finn. Greedy hierarchical variational autoencoders for large-scale video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2318–2328, 2021.

Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. TinyViT: Fast pretraining distillation for small vision transformers, 2022. URL http://arxiv.org/abs/2207.10666.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yuwen Xiong, Mengye Ren, and Raquel Urtasun. LoCo: Local contrastive representation learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 11142–11153. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/7fa215c9efebb3811a7ef58409907899-Abstract.html.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction, 2021. URL http://arxiv.org/abs/2103.03230.

Gongpei Zhao, Tao Wang, Yidong Li, Yi Jin, Congyan Lang, and Haibin Ling. The cascaded forward algorithm for neural network training. *arXiv preprint arXiv:2303.09728*, 2023.