Can Forward Gradient Match Backpropagation?

Louis Fournier [*1] **Stéphane Rivaud** [*1] **Eugene Belilovsky** [2] **Michael Eickenberg** [3] **Edouard Oyallon** [1]

Abstract

Forward Gradients - the idea of using directional derivatives in forward differentiation mode - have recently been shown to be utilizable for neural network training while avoiding problems generally associated with backpropagation gradient computation, such as locking and memorization requirements. The cost is the requirement to guess the step direction, which is hard in high dimensions. While current solutions rely on weighted averages over isotropic guess vector distributions, we propose to strongly bias our gradient guesses in directions that are much more promising, such as feedback obtained from small, local auxiliary networks. For a standard computer vision neural network, we conduct a rigorous study systematically covering a variety of combinations of gradient targets and gradient guesses, including those previously presented in the literature. We find that using gradients obtained from a local loss as a candidate direction drastically improves on random noise in Forward Gradient methods.

1. Introduction

Stochastic Gradient Descent (SGD) (Amari, 1967; Bottou, 2012) using end-to-end backpropagation for gradient computation is the ubiquitous training method of state-of-the-art Deep Neural Networks. However, one can identify at least two issues with this gradient estimation procedure. For one, it requires a significant amount of computational resources and memory. Indeed, a backpropagation update is performed in two steps: a forward step which computes activations, and a backward step which computes gradients. Until a gradient at a given layer is computed, intermediary computations which lead to this layer must be stored in memory, and computation of the update is blocked. This is referred to as the *backward lock* (Jaderberg et al., 2017), which is also an obstacle in the development of model-parallel asynchronous procedures (Belilovsky et al., 2021). Backpropagation also lacks biological plausibility, which is often considered a direction towards more scalable learning (Ren et al., 2022). Indeed, the backpropagation algorithm suffers from the weight transport problem (Carpenter, 1989), which means that the weights of the above layers must be shared during the backward pass of a given layer. This requires a significant amount of communication and synchronization, which is not observed to be implementable in biological systems (Liao et al., 2016). Since biological systems have achieved learning without these restrictions, researchers hope to isolate and use such principles to improve artificial neural network learning.

Recent contributions have re-examined the converse procedure of computing *Forward Gradients* (Baydin et al., 2022), built on the classical forward mode automatic differentiation (Baydin et al., 2018). It is an alternative to standard backpropagation algorithm, which allows the computation of a directional derivative (i.e., a scalar product with the gradient, as opposed to the gradient vector itself) from a forward pass. The gradient computations in forward mode explicitly use the Jacobian of a given layer through Jacobian-Vector Products but do not require storing any intermediary activations or backward passes. In other words, it is *backward unlocked* (as described by Jaderberg et al. (2017)) in that the computation of the derivative is finished as soon as the forward pass is completed. This leads to a potential memory reduction and does not use the biologically implausible weight transport. A Forward Gradient, as introduced by Baydin et al. (2022), corresponds to an unbiased estimate of an activation or weight gradient (which we will refer to as a Gradient Target) computed via a random, isotropic, directional derivative, i.e., a projection of the Gradient Target onto a direction vector (which we will define as the Gradient Guess). The motivation of Forward Gradient descent is to approximate End-to-End Gradient descent. However, this unbiased gradient approximation generally suffers from high variance.

Several approaches (Ren et al., 2022; Silver et al., 2021) have been proposed to reduce the corresponding variance. Unfortunately, there is often an accuracy gap (Silver et al.,

[*]Equal contribution [1]Sorbonne Université, CNRS, ISIR, Paris, France [2]CCM, Flatiron Institute, New York, USA [3]MILA, Concordia University, Montréal, Canada. Correspondence to: Louis Fournier <louis.fournier@isir.upmc.fr>, Stéphane Rivaud <stephane.rivaud@isir.upmc.fr>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

2021) which is difficult to reduce because the Gradient Guess is not aligned with the End-to-End Gradient Target. Following a different line of ideas, Ren et al. (2022) propose to approximate the gradients computed from Local Loss functions (Pathak et al., 2022; Nøkland & Eidnes, 2019; Belilovsky et al., 2019a) as Gradient Targets. This is combined with a collection of architectural changes allowing the decomposition of losses into subgroups within a layer (Patel et al., 2023). This reduces dimensionality and, thereby, the variance of Forward Gradients. However, these changes lead to architectures that are not broadly applicable. Furthermore, results are still far from competitive with standard backpropagation.

Following Ren et al. (2022), it is also unclear if using local gradients as Gradient Targets combined with a Gaussian Gradient Guess is the optimal strategy. In this work, we propose to study different combinations of Gradient Targets and Gradient Guesses in a way that covers existing work and also explores other possible combinations. More specifically, we investigate whether local gradients, which alone lead to suboptimal performance, can be used to compute a reliable Forward Gradient estimate of the end-to-end gradient. Our goal is to understand better if the forward mode of automatic differentiation can lead to training competitive models while maintaining the above benefits. We believe our study to be of high interest due to our in-depth coverage of possible update signal sources on a well-established neural network architecture.

Contributions. Our contributions are as follows: **(a)** We make explicit the idea that any Gradient Target can be projected onto any Gradient Guess and propose to study a wide variety of these combinations. **(b)** In particular, we propose to use local gradients from auxiliary losses as powerful Gradient Guess directions for the computation of directional derivatives in Forward Gradient mode. We show that this proposal can yield far improved results compared to naive random directions while maintaining the benefits of the Forward Gradient mode. **(c)** To ablate this method, we cover many combinations of possible Gradient Targets and Guesses for a well-established architecture (ResNet-18) applied to standard image classification, using both gradient insertion points commonly described as *activity perturbation* and *weight perturbation*, which indicate whether the directional derivative is taken in activation space or weight space. **(d)** Our evaluations reveal that in the case of Gradient Guesses obtained from supervised Local Losses, a consistent positive alignment between the Gradient Targets and Guesses improves the Forward Gradient estimation, reducing the gap with end-to-end training.

Our paper is structured as follows: Sec. 3 describes Gradient Target approximations with Forward Gradient, using both Random and Local Gradient Guesses. Then, Sec. 4.1 describes all the details of our experimental setup. In Sec. 4.2, we show that Forward Gradient can lead to competitive accuracy with end-to-end training, and in Sec. 4.3 that this is due to having a better estimate of the Global Target than with Random Guesses. However, we show in Sec. 4.4 that the Local Gradient Guess is still a poor estimate due to different training dynamics between Local and Global losses. Finally, we confirm the consistency of our findings across model sizes and other datasets in Sec. 4.5. Our source code is available at: github.com/streethagore/ForwardLocalGradient.

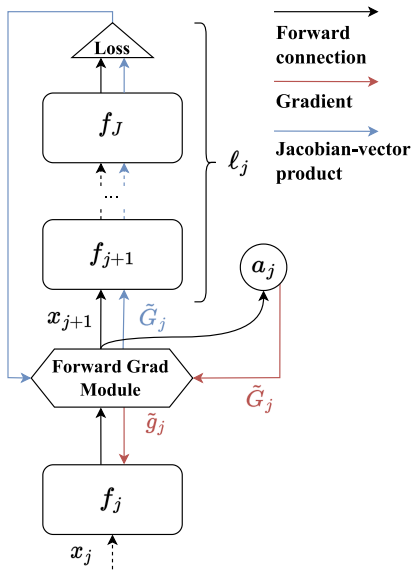

Figure 1. Schematic summarizing the best use-case of our algorithm, which approximates a Global (End-to-End) Target gradient at a block j with forward mode automatic differentiation. A Gradient Guess $\tilde{G}_j$ is provided by the gradient of local loss a_j (red). The Gradient Target is projected on the Gradient Guess by Jacobian-vector products (blue), and we use this estimate to compute the update for block j. Also, in Ren et al. (2022), the Guess $\tilde{G}_j$ is a random vector. More details can be found Sec. 3

2. Related work

Forward Gradient via forward mode automatic differentiation. Forward Gradient has been recently popularized by Baydin et al. (2022), which showed that using a random direction for the input of forward mode automatic differentiation gives an unbiased estimate of the gradient. They studied Forward Gradient descent on example datasets and obtained promising results. Our work attempts to study and understand the limits of Forward Gradient descent on a well-established architecture (Resnet-18) and a difficult computer-vision task (ImageNet32). Note that further theoretical analysis of Forward Gradient was done by Belouze (2022), where they show that a Rademacher random direction offers the best unbiased estimate, but again with no

deep learning experiments. Forward Gradient has also been studied in the context of RNNs (Silver et al., 2021), as it is also an ideal candidate to solve the issue with long-term Backpropagation-through-time. This technique has been applied to small benchmarks, but the question of its benefits on challenging tasks remains open.

Forward Gradient for biologically plausible models. Since Forward Gradient allows the computation of a directional derivative in forward mode, this routine does not require any weight transport or waiting for an update signal as in backpropagation and can be implemented via one Forward pass. Forward Gradient is thus a training mechanism that is more biologically plausible than the Backpropagation mechanism. To scale Forward Gradient, Ren et al. (2022) proposes three techniques that improve the approximation properties of Forward Gradient: architecture modifications that introduce more separability and decorrelation (similar to Patel et al. (2023)), local losses, and to apply Forward Gradient on activations rather than on parameters for reducing the variance of the gradient estimate.

Local losses. In essence, Forward Gradient allows the removal of the backward lock, as introduced by Jaderberg et al. (2017), that is inherent to the backpropagation algorithm. Before updating its weights, a given block need not wait for a full backward pass. This idea has been extensively tested in Belilovsky et al. (2021; 2019b); Löwe et al. (2019); Nøkland & Eidnes (2019); Gomez et al. (2022) by using local losses, which also remove other computational locks, increasing the potential for asynchrony of the training procedure. However, such methods fail to reach the same performance as the end-to-end backpropagation when too much parallelism is desired: incorporating feedback, such as the feedback of the Forward Gradient, is a promising direction for reducing the gap in the accuracy of such methods. In our work, we will use gradients from local losses as candidates of directions for computing Forward Gradients.

3. Gradient approximation via Forward Gradient

The objective of this paper is to study computationally efficient and accurate estimates $g(x) \in \mathbb{R}^d$ of a **Gradient Target** $\nabla f(x) \in \mathbb{R}^d$ of some objective function $f : \mathbb{R}^d \to \mathbb{R}$, using Forward Gradients. The main idea is to use a **Gradient Guess** vector $G(x) \in \mathbb{R}^d$ onto which we project the Gradient Target to obtain an approximation:

$$g(x) = \langle \nabla f(x), G(x) \rangle G(x) \approx \nabla f(x)\,. \tag{1}$$

Once $G(x)$ is provided, then $g(x)$ can be efficiently computed as it can be implemented as a Forward Gradient. This induces limited computational overhead compared to a standard forward pass since evaluating the directional derivative costs around as much as a forward pass (especially for non-linearities whose derivatives cost as much as the function itself). The choice of G will thus greatly impact the quality of the approximation. Note that while Gradient Descent aims at determining $x^* \in \mathbb{R}^d$ such that $\nabla f(x^*) = 0$, the stationary points of (1) are given by:

$$G(x^*) = 0 \text{ or } \langle \nabla f(x^*), G(x^*) \rangle = 0\,.$$

3.1. Gradient Guesses

Gradient Guesses come in different flavors of randomness. Hence the term "approximation" describing (1) can take on both probabilistic and deterministic forms. We discuss guesses that are purely random, purely deterministic, and one intermediate case.

Random Guesses. By drawing G from a zero-mean, unit-covariance probability distribution, we can create an unbiased estimator of the gradient. We propose to use either Rademacher variables $\mathbb{P}(G_i = 1) = \mathbb{P}(G_i = -1) = \frac{1}{2}$ as proposed by Belouze (2022), or isotropic Gaussians $G \sim \mathcal{N}(0, \mathbf{I})$, as proposed in Ren et al. (2022). In both cases, g is an unbiased estimate of ∇f as we have $\mathbb{E}[GG^T] = \mathbf{I}$. However, such estimates potentially have high variance, leading to large errors in individual gradient estimates and slow optimization progress.

Local Guesses. To improve the quality of this estimate, we consider deterministic gradient guesses. We obtain these guesses by computing local update signals from local auxiliary losses, as in Belilovsky et al. (2021); Nøkland & Eidnes (2019). This gives access to a gradient $\nabla a(x)$ from a small local model $a(x)$ with minimal computational effort. Multiple choices for $a(x)$ are possible, for example, a CNN, an MLP, or a linear layer. The intuition behind using local gradients is that they should provide a better one-shot approximation of $\nabla f(x)$ than uncorrelated noise. To obtain the best linear approximation of $\nabla f(x)$ from $\nabla a(x)$, we will then use $G(x) = \frac{\nabla a(x)}{\|\nabla a(x)\|}$ as the projection direction.

NTK Guesses. Conceptually, comparing random and deterministic projection directions via local auxiliary losses is not straightforward. We include a transitional setting using random auxiliary networks to test whether the inductive bias of the random network is sufficient for improvement or whether training of the auxiliary is required. To do so, we re-initialize the Local Model $a(x)$ introduced above at each iteration of the algorithm. This corresponds to a Guess obtained from the Neural Tangent Kernel (in finite width) of our model (Jacot et al., 2018). Note that this can be understood as a weaker version of DFA (Nøkland, 2016; Bartunov et al., 2018), which we expand on in App. B.2.

3.2. Gradient Targets

Global Target. To update any set of weights, the most common gradient target in a supervised Neural Network setting is the gradient of the loss of the final layer of our model. The use of this gradient is standard practice (Krizhevsky, 2009), and it is present in most Deep Learning models. For Forward Gradients, Baydin et al. (2018) uses this gradient as the target, as illustrated in Fig. 1. We will study this setting extensively, and also call this Target the End-to-End Gradient Target.

Local Target. However, it is also possible to take the gradient from the auxiliary loss of the current layer as a gradient target, in order to perform fully local learning. Ren et al. (2022) uses such Local Targets in their experiments and obtains their best results with it. We refer to local learning as the case where the Gradient Guess for the Local Target is equal to the Local Guess. (As for end-to-end training, this is a particular case of Forward Gradient with the Guess exactly equal to the Target).

Intermediate Target. One could also obtain gradient targets from auxiliary losses attached to any intermediate layer of the model. Such a target is likely to be better correlated with the Local Guess while also being closer to the global target, possibly offering a middle ground.

3.3. Gradient computation and insertion points

The candidate direction for the directional derivative obtained using the Forward Gradient technique can be computed in different spaces, with two natural candidates: parameters or activations.

To propose an update of the j-th block $f_j(x_j, \theta_j)$ where θ_j describes the parameters of the j-th block and x_j is the output of the block f_{j-1}, the strategy is to train a model via gradient descent. The output $f_j(x_j, \theta_j)$ is fed to a (target) loss ℓ_j, where ℓ_j can represent a local loss obtained by attaching an auxiliary network to the current or a subsequent block, or it can represent the standard loss at the output of the network. The procedure uses a sample estimate $\nabla_{\theta_j}(\ell_j \circ f_j)$ of the gradient aggregated into a mini-batch $\mathcal{B} = \{x_j^1, ..., x_j^n\}$ of data:

$$\nabla_{\mathcal{B}}(\ell_j \circ f_j)(\theta_j) \triangleq \frac{1}{n}\sum_{i=1}^{n} \nabla_{\theta_j}(\ell_j \circ f_j)(x_j^i, \theta_j) \tag{2}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \partial_{\theta_j} f_j(x_j^i, \theta_j)^T \nabla_{x_{j+1}} \ell_j(x_{j+1}^i) \tag{3}$$

Weight perturbation. A first natural strategy is to use the so-called weight perturbation of the gradient, which means that the estimate obtained in Eq. (2) is replaced by:

$$g_j(x) = \frac{1}{n}\langle \nabla_{\mathcal{B}}(\ell_j \circ f_j)(\theta_j), \frac{1}{n}\sum_{i=1}^{n} G_j^i \rangle \sum_{i=1}^{n} G_j^i \tag{4}$$

where G_j^i is a gradient guess for $\nabla_{\theta_j}(\ell_j \circ f_j)(x_j^i, \theta_j)$.

Activity perturbation. However, if G_j^i is pure noise, Ren et al. (2022) observed that this estimate has high variance. In this case, estimating the gradient via *activity perturbation* is preferable. Indeed, observe that the update of Eq. (3) can be approximated by:

$$\tilde{g}_j(x) = \frac{1}{n}\sum_{i=1}^{n} \partial_{\theta_j} f_j(x_j^i, \theta_j)^T \langle \tilde{G}_j^i, \nabla_{x_{j+1}} \ell_j(x_{j+1}^i) \rangle \tilde{G}_j^i \tag{5}$$

where $\tilde{G}_j^i$ is a guess for $\nabla_{x_{j+1}} \ell_j(x_{j+1}^i)$, and which is often of dimension smaller than G_j^i, leading to a reduced variance in the case of e.g., pure noise (Ren et al., 2022).

Compute trade-off. We summarize in Table 1 the unlocking capabilities as well as compute tradeoffs for the different pairs of Guesses and Targets we introduced. As described by Jaderberg et al. (2017), standard backpropagation is backward-locked and requires a global vector-Jacobian product. Forward Gradient however allows backward unlocking with a Global Target and even update unlocking with a Local Target. As noted in Ren et al. (2022), weight perturbation has favorable memory usage, as activity perturbation requires storing intermediate activations for the Forward Gradient estimator. Activity perturbation also necessitates a backpropagation step in the associated block to compute the gradient used to update the weights.

Table 1. Unlocking and compute tradeoffs for different Guess and Target pairs. Here, vJp and Jvp stand for vector-Jacobian product and Jacobian-vector product respectively. The flag (local) means that the output of the operation (vJp or Jvp) for a given module is not needed by subsequent modules.

Guess		Global Target	Local Target
Local & NTK	Unlocking	Backward	Update
	Compute	vJp (local)+Jvp	vJp (local)
Random	Unlocking	Backward	Update
	Compute	Jvp	Jvp (local)

4. Numerical experiments

4.1. Experimental setup

We now describe how we implemented our models, training procedure, and the implementations of Gradient Targets and Guesses to study the accuracy of a given model under the variations of those parameters.

Models. For this set of experiments, we consider two models which allow us to test various hypotheses on the scalability of gradient computations in forward mode: a standard ResNet-18 (He et al., 2016) (divided into 8 blocks with local losses) and the same ResNet-18 without skip-connections, to avoid side-effects when combining this model with layerwise local losses (divided into 16 blocks with local losses to reach a fully layerwise subdivision of the network, except for the first block which contains two layers).

Hyper-parameters. We considered the CIFAR-10 and ImageNet32 datasets, used with standard data augmentation. The ImageNet32 dataset is smaller than the full-resolution ImageNet dataset, making it more efficient to use during training and testing while still providing a challenging task for model performance. Chrabaszcz et al. (2017) has demonstrated that, in general, the conclusions drawn from the ImageNet32 dataset are also applicable to the full-resolution ImageNet dataset. We followed a standard training procedure: SGD with a momentum of 0.9 and weight decay of 5×10^{-4}. For CIFAR-10, we train the model for 100 epochs, with a learning rate decayed by 0.2 every 30 epochs. For ImageNet32, we also first try a shorter training of 70 epochs, decaying the learning rate by 0.1 every 20 epochs. The initial learning rate was chosen among $\{0.05, 0.01, 0.005\}$ for CIFAR-10 and $\{0.1, 0.05, 0.01, 0.005, 0.0001\}$ for ImageNet32. We report the validation accuracy at the last epoch of training using the best training procedure and learning rate. More details are given in App. A.

Table 2. Test accuracy of a ResNet-18 using a Global Target, split into 16 local-loss blocks, on CIFAR-10 and ImageNet32 for both activity (Activ.) and weight perturbations. Weight perturbation systematically outperforms activity perturbation with a Local Guess. The MLP auxiliary provides the best Local Guess in all configurations. We report the mean and standard deviation over 4 runs for CIFAR-10.

Dataset	CIFAR-10		ImageNet32	
End-to-End	94.3 ± 0.1		53.7	
Space	Activ.	Weight	Activ.	Weight
Local, CNN	79.0 ± 1.0	88.0 ± 0.4	7.3	40.0
Local, MLP	84.7 ± 0.3	88.7 ± 1.2	21.8	37.4
Local, Linear	46.7 ± 2.5	86.1 ± 1.4	10.0	23.3
NTK, CNN	37.7 ± 0.1	50.1 ± 1.0	2.0	3.6
NTK, MLP	50.3 ± 0.4	49.7 ± 0.6	7.5	3.9
NTK, Linear	49.9 ± 1.0	48.3 ± 0.7	4.2	3.9
Gaussian	38.9 ± 0.9	50.0 ± 0.8	4.9	4.9
Rademacher	38.0 ± 1.5	49.8 ± 0.2	5.5	4.6

Local models and losses. We considered 3 types of trainable local auxiliary models: a CNN, an MLP, and a Linear Layer, designed and developed for use with CIFAR-10, to add no more than 10% compute overhead (in FLOPS) while leading to good accuracy. Again, we used a subset of the training data to cross-validate this architecture, which we then fixed during our experiments. Our CNN local auxiliary model is a 3-layer CNN with ReLU non-linearities followed by a 2×2 adaptive average pooling and a projection onto the classification space. The MLP is a 3-layer MLP using ReLU non-linearities, followed by a projection on the classification space. The linear auxiliary net consists of a 2×2 adaptive average pooling preceded by a batch normalization module and followed by a projection onto the classification space as in Belilovsky et al. (2021). More details on our methodology to cross-validate the hyper-parameters can be found in App. A. Each local loss can then be used to obtain Gradient Guesses via an abridged backpropagation procedure or to serve as Gradient Target for computing Forward Gradients.

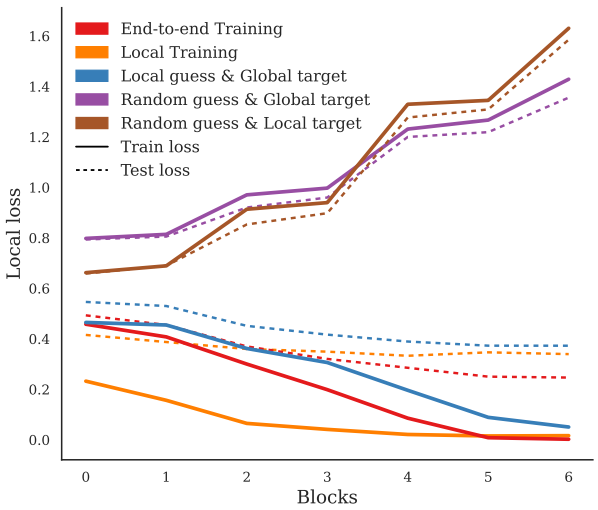

Figure 2. Local train losses at the end of training at each block for a ResNet-18 split in 8 blocks, with CNN auxiliary, for different training algorithms. In the Gaussian and end-to-end cases, the auxiliary training is detached from the main module training and is only for logging purposes. With Gaussian Guesses, losses increase with depth. The Local Target allows better convergence than the Global Target at the first block but an even worse convergence for the whole network. With local learning, local losses converge to local minima.

4.2. The gap between backpropagation and Forward Gradient is narrowed with Local Guesses

Tab. 2 and Tab. 3 report our results for a ResNet-18 split into 8 or 16 blocks, on both CIFAR-10 and ImageNet32. First, we note that accuracies of Forward Gradient with a Random Guess as studied in Ren et al. (2022) are extremely low. Despite extensive hyper-parameter searches (for learning rate notably), it proved difficult to achieve reasonable accuracies for ImageNet32 using random Gradient Guesses. This is in line with expectations from Belouze (2022), and suggests that the architecture modifications performed in

Ren et al. (2022) are essential to its success.

Table 3. Test accuracy of a ResNet-18 using a Global Target, split into 8 local-loss blocks, on CIFAR-10 and ImageNet32 for both activation (Activ.) and weight perturbations. Local Guesses improve on Random Guesses and NTK Guesses. Activity perturbation only provides an advantage for Random Guesses, otherwise, weight perturbation performs better in all CIFAR-10 configurations. We report the mean and standard deviation over 4 runs for CIFAR-10.

Dataset	CIFAR-10		ImageNet32	
End-to-End	94.5 ± 0.2		54.6	
Guess	Activ.	Weight	Activ.	Weight
Local, CNN	90.4 ± 0.2	92.0 ± 0.3	33.1	38.3
Local, MLP	89.2 ± 0.2	91.6 ± 0.2	38.0	45.8
Local, Linear	65.9 ± 6.7	88.9 ± 0.3	27.9	34.9
NTK, CNN	55.2 ± 0.4	66.8 ± 2.6	5.5	8.0
NTK, MLP	61.6 ± 0.5	63.1 ± 1.1	17.4	15.5
NTK, Linear	61.6 ± 0.9	62.6 ± 1.0	9.8	15.4

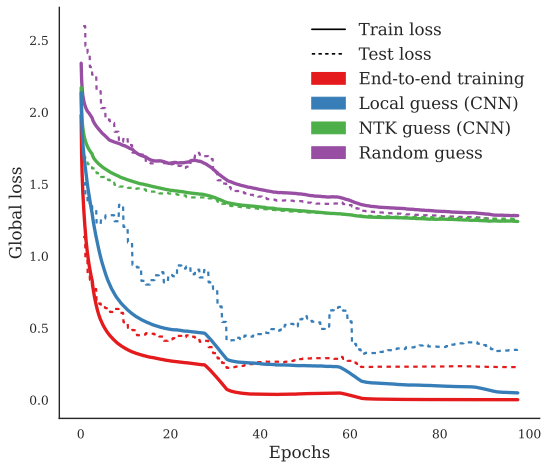

Figure 3. Comparison of train and test losses for end-to-end training (red), and Forward Gradient with Global Target and Local Guess (blue), NTK Guess (green) and Random Guess (purple), on CIFAR-10 with a ResNet-18 split in 8 blocks. Local and NTK Guesses are derived from a CNN auxiliary. Random and NTK Guesses fail to optimize the Global Loss. Local Guess performs much better, with a consistent gap with respect to end-to-end throughout training.

Local Guesses reduce the gap with end-to-end training. Tab. 2 and Tab. 3 indicate that Local Guesses systematically outperform the naive Random Guesses strategy, in some cases by about 40% for CIFAR-10 and 20% for ImageNet32. This is not surprising, as a pure Gaussian Guess leads to random directions which are not aligned with the directions important for the classification task. Even NTK Guesses, which are random but benefit from the inductive bias of the auxiliary network, yield a slight improvement in most cases. Reusing the same random network for a full mini-batch permits us to obtain a gradient direction which is less naive than random noise and proves that NTK (Jacot et al., 2018) has a good inductive bias for this task. We report in Figure 3 the train and test losses for Forward Gradient training with Global Target and several Guesses, showing that only Local Guesses allow convergence of the global loss.

Weight perturbation outperforms activity perturbation. For the case of supervised gradient guesses, we note that weight perturbation significantly outperforms activity perturbation. This shows that the best setup for the setting studied in Ren et al. (2022) may not generalize to standard neural networks.

Table 4. Test accuracy on CIFAR-10 of each variation of a ResNet-18, using a Local Target with Gaussian, Rademacher, and Local Guesses, for both activity and weight perturbations. We report the mean and standard deviation over 4 runs.

8 blocks	CNN		MLP		Linear	
Local Target	92.0 ± 0.2		92.1 ± 0.2		89.2 ± 0.1	
Guesses	Activity	Weight	Activity	Weight	Activity	Weight
Gaussian	41.2 ± 1.9	52.1 ± 1.5	45.6 ± 1.6	53.2 ± 0.6	36.9 ± 1.9	56.5 ± 0.8
Rademacher	39.0 ± 1.3	53.0 ± 0.7	43.6 ± 2.1	53.3 ± 0.9	34.6 ± 1.9	56.6 ± 0.6
16 blocks	CNN		MLP		Linear	
Local Target	87.7 ± 0.1		89.4 ± 0.3		86.7 ± 0.3	
Guesses	Activity	Weight	Activity	Weight	Activity	Weight
Gaussian	23.5 ± 2.4	37.5 ± 1.6	21.5 ± 4.4	39.7 ± 0.8	23.8 ± 1.6	43.5 ± 0.5
Rademacher	22.3 ± 1.5	37.8 ± 0.5	23.3 ± 1.4	41.3 ± 1.4	24.1 ± 1.3	45.5 ± 0.5

Training by residual blocks obtains a significantly better accuracy than training layerwise Comparing the results of Tab. 2 and Tab. 3 indicates that systematically, for the same pair of algorithms, training by blocks of two layers (corresponding to residual blocks) with one auxiliary per block outperforms the corresponding version with one auxiliary per layer. This observation is consistent with the fact that the layers encapsulated in a given block are jointly trained in the first case. By contrast, the layerwise strategy is fully decoupled. The fact that using joint training systematically improves the accuracy of the method is known from previous work (Belilovsky et al., 2021; Patel et al., 2023; Belilovsky et al., 2019a). However, it comes at the cost of not obtaining the full potential of computational and memory savings associated with Forward Gradient. For Forward Gradient methods, we note that Local Guesses with weight perturbation are the Guesses that minimize the accuracy drop when transitioning from a 8 blocks split to a 16 blocks one.

4.3. Reliable estimating the Global Target is necessary for Forward Gradient to succeed

Accuracy degrades in upper layers with Random Guesses. Fig. 2 shows the evolution of the separability of representations with depth via the training loss of a local (CNN) auxiliary for a ResNet-18 split into 8 blocks. In the case of end-to-end training and Random Guesses for Global targets, the auxiliary losses were trained ad-hoc

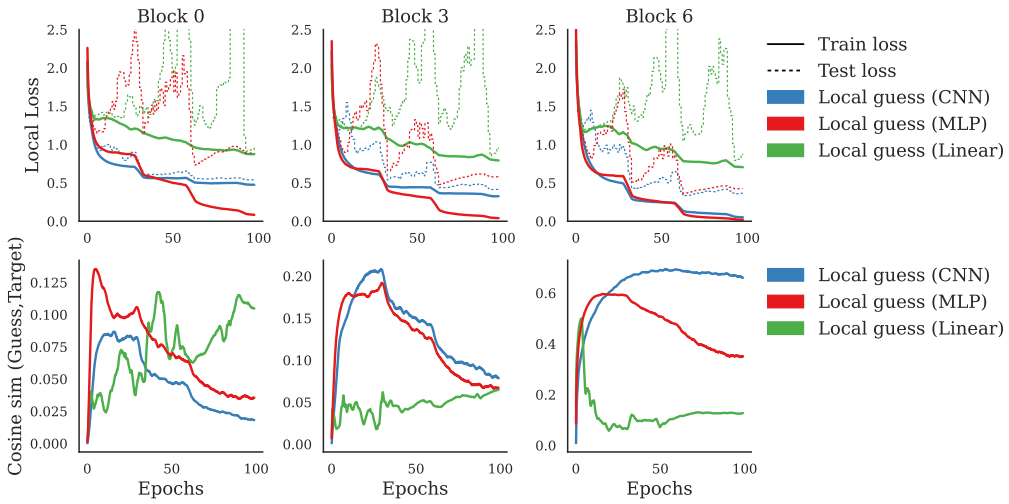

Figure 4. Train and test local losses (top row) and mean cosine similarity between Local Guess and Global Target in the activation space (bottom row), for blocks 0, 3, and 6 (left, middle, and right columns) during training. The model is a ResNet-18 divided into 8 blocks trained on CIFAR-10. The similarity values are low but consistently positive and increase with depth as the target gets closer (and less deep). Learning improves with depth, as expected. As both target and local losses converge to a minimum, their similarity decreases.

without interacting with the training of the global model. Without any surprise, the end-to-end progressively separate classes, a phenomenon already observed in Zeiler & Fergus (2014); Jacobsen et al. (2018); Oyallon (2017). Random isotropic Guesses do not follow this trend. On the contrary, the training loss progressively degrades with depth, both for Global and Local Targets. In comparison, Local Guesses with Global Target accuracies improve with depth, as we expect from a deep model. Furthermore, Random Guesses with Local Target give worse results than Random Guesses with Global Target, as reported in Tab. 4. This constitutes a further difference between this work and Ren et al. (2022).

Intermediate Targets perform worse than Global Targets. To bridge the gap between Global and Local Targets, we proposed an Intermediate Target, the gradient of the Local loss at block $j+1$. The idea is to have a target closer to the Gradient Guess than the Global Target, allowing for easier convergence (since similarity increases with depth) as well as improving on Local Target, where subsequent blocks do not provide any feedback. However, despite more consistent alignment between Targets and Guesses across blocks, we observed a consistent decrease in accuracy compared to both Global and Local Targets, as reported in Tab 5. Thus, despite good alignment between Guess and Target, not using a Global Target penalizes the learning of the network.

4.4. Local Guesses do not align with the Global Target

We now study the difference between the optimization path of the Forward Gradient and the end-to-end gradient to understand if Forward Gradient can be understood as an approximation of the Global Target i.e., the End-to-End Gradient Target.

Table 5. Test accuracy on CIFAR-10 of a Resnet-18 using the loss of the block $j+1$ as the Intermediate Target for the block j, for both activity (Activ.) and weight perturbations. The model has been split into 8 and 16 blocks as in the experiments of Tab. 3 and Tab. 2. We added the Target (tgt) used for the Random Guesses.

ResNet-18 split	8 blocks		16 blocks	
Gradient Guesses	Activ.	Weight	Activ.	Weight
Local, CNN	91.6	91.6	80.0	87.8
Local, MLP	91.5	91.8	87.6	89.5
Local, Linear	76.3	89.6	59.8	86.9
NTK, CNN	55.7	60.0	33.0	18.4
NTK, MLP	57.4	52.5	40.7	20.1
NTK, Linear	55.3	44.2	32.4	20.3
Gaussian, CNN tgt	45.3	51.5	28.0	47.1
Gaussian, MLP tgt	53.7	53.2	31.4	48.0
Gaussian, Linear tgt	51.4	53.9	42.5	46.4
Rademacher, CNN tgt	43.8	51.6	24.8	47.5
Rademacher, MLP tgt	54.7	52.9	30.4	49.5
Rademacher, Linear tgt	49.0	54.2	43.6	45.6

Local Guesses and Global Targets are weakly aligned. Fig. 4 and 5 display the evolution of the local train and test losses, and the cosine similarity between Local Guesses and the Global Target, for a ResNet-18 split into 8 blocks. For the cosine similarity, we report a batch estimate at every iteration given by:

$$\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\frac{\langle \nabla_{x_{j+1}}\ell_j(x^i_{j+1}), \tilde{G}^i_j\rangle}{\|\nabla_{x_{j+1}}\ell_j(x^i_{j+1})\|\|\tilde{G}^i_j\|}\,. \tag{6}$$

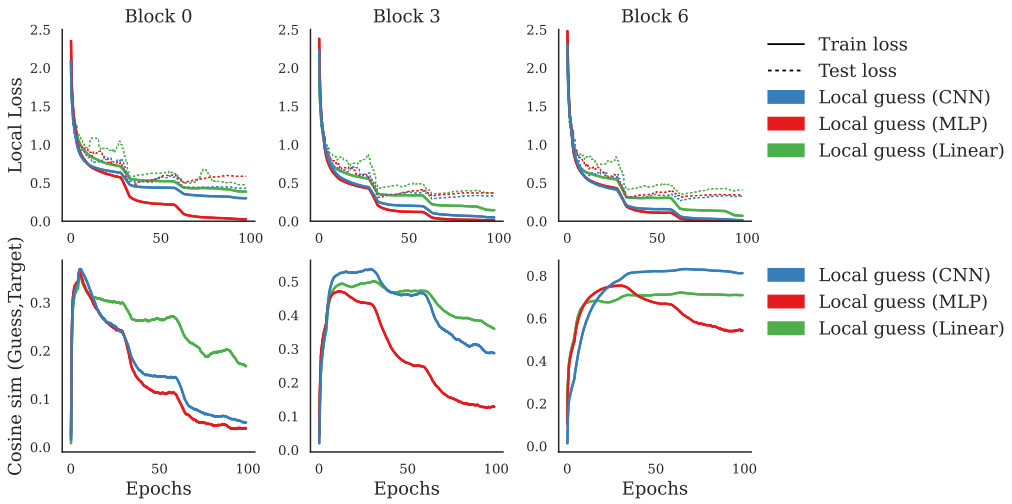

Figure 5. Train and test local losses (top row) and mean cosine similarity between Local Guess and Global Target in the weight space (bottom row, averaged over the parameters), for blocks 0, 3, and 6 (left, middle, and right columns) during training. The generalization gap is more consistent during training than with activation gradients. The cosine similarity is also consistently higher than for activation gradients but falls more drastically during training.

We note that the cosine similarity between the end-to-end gradient and the gradients obtained from local auxiliaries show three main tendencies: **(a)** With increasing block index, the alignment is higher (see y-axis scale on each plot). **(b)** Most of the time, the convolutional auxiliary shows the strongest correlation and the lowest test loss value. **(c)** Finally, although low, the similarity is consistently positive. The first statement is a reflection of the depth proximity of the auxiliary to the Global Target; the second is likely due to the convolutional nature of the subsequent network. We further note that MLP and linear auxiliaries share similarly erratic test loss curves and that most similarity curves tend to descend toward the end of training. The former may be due to an architectural misalignment. The latter observation likely indicates that different local minima have been reached and that the optimization trajectories will, from then, drift apart.

Projecting on the entire Gradient Guess space still does not estimate the Gradient Target. The low alignment between Local Guesses and Global Targets indicates a poor approximation. We propose projecting each Global Target on the subspace spanned by Local Guesses, for each sample to test this hypothesis further. In other words, we replace the gradient estimate of Eq. (5) with the best linear approximation of the Global Target using (the entire subspace of) Local Guesses. In practice, for a batch of gradients, we project each Global Target onto the span of the batch of Local Guesses. Our results are aggregated in Tab. 6, and we trained each model using this novel dynamic. The improvement over projecting on a single Local Guess is marginal, and the accuracy is still far from end-to-end training. This means that approximating the Global Target by Local Guesses is insufficient to recover the dynamic of backpropagation training. Generally, the task of approximating a Global Target by Gradient Guesses is likely a difficult task (Montufar et al., 2014).

Table 6. Test accuracy on CIFAR-10 of a Resnet-18 trained by using the best linear approximation of the end-to-end gradients by local guesses, with activity perturbation. We report the mean and standard deviation over 4 runs.

Model	Gradient Guess subspace	Accuracy
ResNet-18, 8 blocks	Local Guess, CNN	92.3 ± 0.1
	Local Guess, MLP	89.6 ± 0.3
	Local Guess, Linear	84.0 ± 0.6
ResNet-18, 16 blocks	Local Guess, CNN	89.6 ± 0.6
	Local Guess, MLP	77.9 ± 3.3
	Local Guess, Linear	73.4 ± 1.8

Alignment between Guess and Target matters. Experimental results indicate that the Local Guess does not approximate well the Global Target. Comparing Tab. 4 and Tab. 6 shows that using the best linear approximation of the Global Target on the space spanned by the Local Guesses still performs on par or worse than local learning (i.e., Local Guess on Local Target). Thus, despite the Global Target being a better target than the Local Target in general, the adequation between the Guess and the Target has a stronger impact than the reweighting of the Guess. However, it is known that local learning saturates at early layers and does not benefit from the depth of the model (Wang et al., 2021), indicating that global feedback may be necessary to recover backprop-

Table 7. Test accuracy of a ResNet-18 using a Global Target, split into 8 or 16 local-loss blocks, on Fashion-MNIST, CIFAR-100 and Imagenette datasets for both activity and weight perturbations. We report the mean and standard deviation over 4 runs.

Dataset	CIFAR-100				Fashion-MNIST				Imagenette			
Model	16 blocks		8 blocks		16 blocks		8 blocks		16 blocks		8 blocks	
End-to-End	74.6 ± 0.3		75.7 ± 0.2		94.1 ± 0.2		93.9 ± 0.1		88.9 ± 0.4		90.0 ± 0.4	
Guess	Activity	Weight	Activity	Weight	Activity	Weight	Activity	Weight	Activity	Weight	Activity	Weight
Local, CNN	44.6 ± 1.7	58.2 ± 0.6	64.5 ± 0.9	67.9 ± 0.4	92.5 ± 0.3	93.7 ± 0.1	93.7 ± 1.3	93.6 ± 0.2	71.8 ± 1.8	84.2 ± 0.6	84.9 ± 0.3	88.0 ± 0.1
Local, MLP	59.5 ± 0.5	64.3 ± 0.4	67.5 ± 0.3	70.0 ± 0.3	92.3 ± 0.2	93.8 ± 0.1	93.6 ± 0.3	94.0 ± 0.1	77.3 ± 1.3	82.8 ± 0.3	84.9 ± 0.4	85.9 ± 0.4
Local, Linear	46.1 ± 1.1	62.1 ± 0.4	50.0 ± 9.8	65.9 ± 0.4	73.4 ± 1.7	93.4 ± 0.2	87.4 ± 1.5	94.0 ± 0.1	53.2 ± 2.0	82.8 ± 0.4	70.7 ± 1.7	86.1 ± 0.5
NTK, CNN	15.0 ± 0.7	22.9 ± 0.3	29.9 ± 0.7	40.3 ± 1.5	82.0 ± 0.2	88.2 ± 0.3	87.2 ± 0.4	91.1 ± 0.3	47.0 ± 1.0	53.7 ± 2.1	65.9 ± 0.7	68.2 ± 1.0
NTK, MLP	23.5 ± 0.3	23.1 ± 0.4	35.9 ± 1.0	36.1 ± 0.2	86.8 ± 0.6	88.0 ± 0.5	88.3 ± 0.3	90.8 ± 0.2	56.3 ± 0.8	53.3 ± 0.9	69.7 ± 1.0	67.9 ± 0.7
NTK, Linear	23.9 ± 0.5	22.9 ± 0.7	36.4 ± 0.4	37.8 ± 2.2	83.7 ± 0.2	88.8 ± 0.3	88.5 ± 0.4	90.8 ± 0.3	55.3 ± 1.4	51.3 ± 0.8	68.7 ± 0.6	68.6 ± 0.5
Gaussian	10.0 ± 1.2	22.4 ± 0.5	27.5 ± 0.5	35.3 ± 0.5	79.1 ± 0.7	88.3 ± 0.3	87.5 ± 0.2	89.5 ± 0.3	17.8 ± 1.5	47.6 ± 1.4	52.7 ± 3.1	64.1 ± 1.1
Rademacher	8.7 ± 1.1	22.6 ± 0.4	27.5 ± 0.4	33.2 ± 1.2	79.5 ± 0.2	88.3 ± 0.5	87.1 ± 0.7	89.7 ± 0.2	18.4 ± 2.6	47.9 ± 1.7	51.8 ± 1.7	63.8 ± 0.9

agation performances. These issues suggest that deriving a good estimate of the end-to-end gradient is a difficult task that might need to be addressed to obtain performance on par with standard backpropagation.

Table 8. Test accuracy of a Wide ResNet-18 using a Global Target, split into 16 local-loss blocks, on CIFAR-10 for both activity and weight perturbations for different width factors, i.e., k=0.5, 2 and 4. Table 2 refers to the case where $k = 1$. We report the mean and standard deviation over 4 runs.

Width factor k	0.5		2		4	
End-to-End	93.0 ± 0.4		94.9 ± 0.0		95.3 ± 0.1	
Guesses	Activity	Weight	Activity	Weight	Activity	Weight
Local, CNN	63.3 ± 1.6	72.4 ± 0.6	87.0 ± 0.4	85.5 ± 0.3	90.6 ± 0.2	93.3 ± 1.5
Local, MLP	81.5 ± 0.5	79.1 ± 0.3	86.2 ± 0.2	84.1 ± 0.2	87.0 ± 0.2	91.7 ± 0.2
Local, Linear	26.9 ± 3.0	78.5 ± 0.3	62.7 ± 2.8	84.6 ± 0.2	62.5 ± 4.3	92.3 ± 0.3
NTK, CNN	33.8 ± 1.3	48.9 ± 0.2	41.1 ± 0.4	50.6 ± 0.3	44.2 ± 0.3	51.6 ± 0.3
NTK, MLP	48.9 ± 1.3	47.7 ± 0.4	50.9 ± 0.5	50.6 ± 0.2	51.8 ± 0.2	51.6 ± 0.3
NTK, Linear	47.9 ± 1.2	47.6 ± 1.2	51.4 ± 0.4	48.7 ± 0.7	52.1 ± 0.3	49.3 ± 0.7
Random, Gaussian	38.9 ± 0.4	49.6 ± 0.9	36.2 ± 1.4	48.9 ± 0.5	29.6 ± 2.3	48.7 ± 0.5
Random, Rademacher	38.8 ± 0.1	49.5 ± 0.4	35.3 ± 0.6	49.0 ± 0.7	30.2 ± 2.1	49.0 ± 0.8

4.5. Consistency across model size and datasets

Wide ResNets further show the curse of dimensionality. We study the effect of parameter sizes on our findings by applying different Target and Guess pairs on Wide ResNets. We provide in Tab. 8 the accuracies we obtain for different guesses for a Global Target with a Wide ResNet-18 for different width factors k (for $k = 0.5$, 2, and 4, with $k = 1$ being the standard ResNet-18 we studied earlier), by scaling the number of features according to k (for both the model and the auxiliary network). We observe that Local Guess accuracy improves with width. However, Random Guess accuracy decreases, confirming that Forward Gradient estimation with random directions deteriorates as the size of the gradient increases due to the curse of dimensionality. This in turn suggests pursuing good direction guesses. We also provide in Tab. 10 in App. B.1 results for a Local Target.

Our findings extend to other datasets. We also showcase results for other image classification datasets with varying difficulty. We consider the following datasets: Fashion-MNIST, a more complex replacement for MNIST, CIFAR-100, the same dataset as CIFAR-10 with 100 labels, and Imagenette, an easier subset of Imagenet with only 10 labels. We summarize the accuracies for a Global Target in Tab. 7 and a Local Target in Tab. 9 in App. B.1 for a ResNet-18 divided into 8 and 16 blocks. We notice the same trends discussed in this study for CIFAR-10, thus confirming our findings.

5. Conclusion

We have extensively studied various Forward Gradient training variants for a ResNet-18 architecture divided into 8 and 16 blocks on the standard object recognition tasks CIFAR-10 and ImageNet32. In particular, we introduce the use of gradients from locally supervised auxiliary losses as a better candidate direction for the Target than random noise. We varied Gradient Targets, Guesses, auxiliary networks, and feedback insertion points. The Gradient Guesses selected were either Random isotropic directions, NTK gradients, or local auxiliary net gradients. We confirmed our findings for other image datasets and varying model sizes.

We determined several unambiguous trends. Firstly, Gradient Guesses obtained from Local Guesses exceeded the performance of Random Guesses. Secondly, our study confirms that estimating consistently the Global Target should be the main focus of Forward Gradient algorithms. Thirdly, we conclude that the limits of our method are due to the limited alignment between the local loss gradients and end-to-end gradients. Still, we found that Local Gradient Guesses reduces the gap with end-to-end training for Forward Gradient and that subsequent works need to improve alignment between Guess and Target to reach end-to-end accuracy.

Acknowledgements

This work was supported by Project ANR-21-CE23-0030 ADONIS, EMERG-ADONIS from Alliance SU, and Sorbonne Center for Artificial Intelligence (SCAI) of Sorbonne University (IDEX SUPER 11-IDEX-0004). This work was granted access to the AI resources of IDRIS under the allocations 2022-AD011013095 and 2022-AD011013769 made by GENCI. We acknowledge funding and support from NSERC Discovery Grant RGPIN-2021-04104 and resources from Compute Canada and Calcul Quebec.

References

Amari, S. A theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, EC-16(3):299–307, 1967. doi: 10.1109/PGEC.1967.264666.

Bartunov, S., Santoro, A., Richards, B., Marris, L., Hinton, G. E., and Lillicrap, T. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Advances in Neural Information Processing Systems*, pp. 9389–9399, 2018.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

Baydin, A. G., Pearlmutter, B. A., Syme, D., Wood, F., and Torr, P. Gradients without backpropagation. *arXiv preprint arXiv:2202.08587*, 2022.

Belilovsky, E., Eickenberg, M., and Oyallon, E. Greedy layerwise learning can scale to imagenet. In *International conference on machine learning*, pp. 583–593. PMLR, 2019a.

Belilovsky, E., Eickenberg, M., and Oyallon, E. Greedy layerwise learning can scale to imagenet. *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019b.

Belilovsky, E., Leconte, L., Caccia, L., Eickenberg, M., and Oyallon, E. Decoupled greedy learning of cnns for synchronous and asynchronous distributed learning. *arXiv preprint arXiv:2106.06401*, 2021.

Belouze, G. Optimization without backpropagation, 2022. URL https://arxiv.org/abs/2209.06302.

Bottou, L. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pp. 421–436. Springer, 2012.

Carpenter, G. A. Neural network models for pattern recognition and associative memory. *Neural networks*, 2(4): 243–257, 1989.

Chrabaszcz, P., Loshchilov, I., and Hutter, F. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

Gomez, A. N., Key, O., Perlin, K., Gou, S., Frosst, N., Dean, J., and Gal, Y. Interlocking backpropagation: Improving depthwise model-parallelism. *Journal of Machine Learning Research*, 23(171):1–28, 2022.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jacobsen, J.-H., Smeulders, A., and Oyallon, E. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., and Kavukcuoglu, K. Decoupled neural interfaces using synthetic gradients. *International Conference of Machine Learning*, 2017.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Liao, Q., Leibo, J., and Poggio, T. How important is weight symmetry in backpropagation? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Löwe, S., O'Connor, P., and Veeling, B. S. Putting an end to end-to-end: Gradient-isolated learning of representations, 2019. URL https://arxiv.org/abs/1905.11786.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27, 2014.

Nøkland, A. Direct feedback alignment provides learning in deep neural networks. *Advances in neural information processing systems*, 29, 2016.

Nøkland, A. and Eidnes, L. H. Training neural networks with local error signals. In *International conference on machine learning*, pp. 4839–4850. PMLR, 2019.

Oyallon, E. Building a regular decision boundary with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5106–5114, 2017.

Patel, A., Eickenberg, M., and Belilovsky, E. Local learning with neuron groups. *arXiv preprint arXiv:2301.07635*, 2023.

Pathak, P., Zhang, J., and Samaras, D. Local learning on transformers via feature reconstruction. *arXiv preprint arXiv:2212.14215*, 2022.

Refinetti, M., d'Ascoli, S., Ohana, R., and Goldt, S. Align, then memorise: the dynamics of learning with feedback alignment, 2021.

Ren, M., Kornblith, S., Liao, R., and Hinton, G. Scaling forward gradient with local losses. *arXiv preprint arXiv:2210.03310*, 2022.

Silver, D., Goyal, A., Danihelka, I., Hessel, M., and van Hasselt, H. Learning by directional gradient descent. In *International Conference on Learning Representations*, 2021.

Wang, Y., Ni, Z., Song, S., Yang, L., and Huang, G. Revisiting locally supervised learning: an alternative to end-to-end training. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=fAbkE6ant2.

Zeiler, M. D. and Fergus, R. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pp. 818–833. Springer, 2014.

A. Additional details

A.1. ResNet-18 Design

We follow the standard ResNet-18 implementation when decoupling with 8 blocks, and remove all skip-connections in the 16 blocks case as explained. For all datasets (except Imagenette, see Sec. B.1), we use the standard modification of the first layers of the ResNet to accommodate the smaller image size of our datasets: The first layers are replaced by a convolutional layer with kernel size 3 and stride 1 and no bias, a BatchNorm layer with affine output, and a ReLU layer (note that we do not have a Max Pooling layer).

A.2. Auxiliary Net Design

The auxiliary nets used to train our model in the experiments are designed to keep the ratio of FLOPS between the auxiliary net and the main module under 10%. We investigate three architectures: a convolutional neural network (CNN), a multi-layer perceptron (MLP), and a linear classifier. We evaluate each architecture by using it as a local loss to train a ResNet-18 split in 8 blocks with the DGL algorithm. The optimization is carried out by stochastic gradient descent with a momentum of 0.9, a weight decay of 5×10^{-4}, and mini-batch size 256. The initial learning rate is set to 0.1 and is decayed by a factor of 0.2 every 30 epochs.

The linear classifier applies batch normalization followed by an adaptive average pooling which yields an output with spatial resolution 2×2. This output is then flattened into a 1-dimensional tensor, which is then projected onto the classification space. The MLP architecture first employs an adaptive average pooling yielding an output with spatial resolution 2×2. This output is then flattened and propagated through a number n_{depth} of the fully-connected layer using batch normalization and ReLU nonlinearity. Finally, the output is then projected onto the classification space. The first fully-connected layer outputs a vector with h_{chan} dimension, which parameterizes the width of the auxiliary net, and this number of channels is kept fixed until the projection onto the classification space. The number n_{depth} of fully-connected layers instead parameterizes the depth of the MLP auxiliary net. Similarly, CNN first employs a 1x1 convolution to obtain a spatial output with h_{chan}, where h_{chan} essentially parameterizes the width of the architecture. It is followed by a number n_{depth} of convolutional layers with kernel 3×3 and stride equal to 2. Using strided convolution helps reduce the computational footprint while maintaining similar accuracies.

We tested n_{depth} between 1 to 8 to test the effect of the depth. We tested h_{chan} = 64, 128, 256, 512, 1024, 2048, 4096 for the MLP, and h_{chan} = 4, 8, 16, 32, 64 for CNN. The ResNet-18 was trained on CIFAR-10 using standard data augmentation with the DGL training procedure for 90 epochs. Among the configurations complying with the 10% FLOPS ratio constraint, we kept the parameters yielding the best accuracy on CIFAR-10. This yields $h_{\text{chan}} = 1024$ and 32 for the MLP and CNN respectively, and $n_{\text{depth}} = 3$ for both architectures.

A.3. Implementation details

Practical implementation In practice, we compute the projection between Guess and Target in our implementation by computing two backward passes, with both losses. For activity perturbation, a backward hook is used to first log the Gradient Target, and then to log the Gradient Guess and compute the projection. We then let the new estimated activation gradient backpropagate through the model. For weight perturbation, we store the (batch-wise) weight gradients after both target and guess backpropagation, and compute weight-wise the projection.

Of course, both implementations are implementable using forward mode automatic differentiation with modern deep learning libraries. Similarly, we note that for the experiment of Tab. 6, where we project the Target on a batch of Guesses, this projection is still possible by using k Jacobian-vector products by projecting the Global Target on the k principal components of the batch of Local Guesses.

Data augmentation The data augmentation procedure we use is a standard Random Crop with padding 4 and a Random Horizontal Flip (with probability 0.5), plus a normalization step. Fashion-MNIST do not use any data augmentation.

Learning rates chosen In practice, the learning rate (lr) chosen for CIFAR-10 is 0.005 for Random Guesses (except for the activity-perturbed with a Global Target which has a lr of 0.01), as well as the 16-blocks Local Linear Guess (activity-perturbed). NTK Guesses uses a lr of 0.01 in the 16 blocks case (as for the Wide ResNet). With all other Gradient Guesses, the lr is 0.05.

For the additional CIFAR-100, Fashion-MNIST and Imagenette datasets, end-to-end learning, local learning and local guesses use a lr of 0.05. CIFAR-100 uses the same lr as CIFAR-10 for the NTK guesses. Fashion-MNIST uses a lr of 0.01 for activity-perturbed NTK guesses and 0.05 for the weight-perturbed ones. Imagenette uses a lr of 0.05 for all NTK guesses. The Random Guesses have a lr of 0.005.

ImageNet32 proved trickier to optimize. The learning rate and scheduler steps size chosen for the reported accuracy are: For the End-to-End training, a lr of 0.05 with step size 20. For the 16 blocks model, local guess activity perturbed

have a lr of 0.01 with step size 30 and weight perturbed a lr of 0.1 with step size 20. The NTK guess have a lr of 0.005 with step size 30 (except the CNN auxiliary, with lr 0.01). Random guesses have a step size of 30 with a lr of 0.001 for the activity perturbed and 0.005 for the weight perturbed. For the 8 blocks model, local guesses have a lr of 0.05 for the CNN auxiliary, 0.01 for the linear auxiliary and the activity-perturbed MLP auxiliary, and 0.1 for the weight-perturbed MLP auxiliary. Activity-perturbed CNN and Linear auxiliary and the weight-perturbed MLP auxiliary have a step size of 20, and the other 30. NTK guesses have a step size of 30 and lr of 0.001 for the CNN auxiliary, and the MLP and Linear activity-perturbed auxiliaries. The others have a lr of 0.005.

B. Additional results

B.1. Local Target results

We provide in this subsection the Table results discussed in Sec. 4.5.

Additional datasets We report in Tab. 9 the results of different guesses with a Local Target for the additional datasets Fashion-MNIST, CIFAR-100, and Imagenette. The results are consistent with our findings on CIFAR-10 and ImageNet32. We revert the first layers of the ResNet-18 to its original design with Imagenette since its image size is bigger than our other datasets.

Wide ResNet We report in Tab. 10 the results of different guesses with a Local Target for a Wide ResNet-18 with width factors $k = 0.5, 2$ and 4. We observe similar tendencies to the ones observed for the Global Target.

B.2. Additional local losses and guesses

Predsim To propose a slightly different supervised loss, we refer to the setup of Nøkland & Eidnes (2019), and more particularly the 'predsim' local loss for comparison with our more simple cross-entropy local losses. We directly adapt the architecture of the predsim local auxiliary net and loss to our framework, as a Target or Guess. Using predsim for local learning corresponds to the Local Error Signal framework. We report the accuracies using the predsim local loss in Tab. 12. We find predsim to be competitive with the deeper local losses we used (for 8 blocks), despite the predictive auxiliary of predsim being linear. We also note that despite the Global Target Loss being different from the Local Loss in that case (minimizing a supplementary similarity matching term), the Local Guess method does not seem affected.

Fixed NTK One can note that the dynamics of cosine similarity when training with Local Guess and Global Target (see Fig. 4 and 5) are similar to the alignment behavior of Direct Feedback Alignment (DFA) which can be observed in (Refinetti et al., 2021). We can thus propose an alternative to our random NTK Guess that is even further inspired by DFA to produce similar gradient feedback. Rather than reinitializing the local loss at each batch randomly, we keep a single fixed random local loss throughout training. In this case, the linear Fixed NTK local guess can be computed with no backpropagation and seen as close to DFA. We use the Fixed NTK local gradient as a type of Local Guess for Forward Gradients to estimate the Global Target. We report in Tab. 11 the test accuracy obtained for a Resnet-18 on CIFAR-10, for the three types of auxiliary networks. Despite the much more limited Gradient Guess space compared to the random NTK, results are competitive between the two methods. Local learning results also show strong accuracy despite the auxiliary network being fixed compared to the local learning we proposed in Tab. 4.

B.3. Figures for a ResNet-18 split in 16 blocks

We also provide additional Figures equivalent to the Figures in the main paper but for a ResNet-18 trained in 16 blocks and without skip connection.

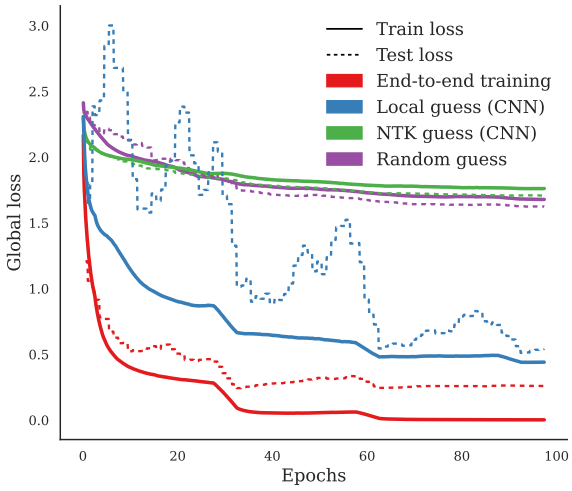

Figure 6. Comparison of train and test losses for end-to-end training (red), and Forward Gradient with Global Target and Local Guess (blue), NTK Guess (green) and Random Guess (purple), on CIFAR-10 with a ResNet-18 split in 16 blocks. Local Guess and NTK Guess are derived from a CNN auxiliary.

Table 9. Test accuracy of a ResNet-18 using a Local Target, split into 8 or 16 local-loss blocks, on Fashion-MNIST, CIFAR-100 and Imagenette datasets for both activity and weight perturbations. We report the mean and standard deviation over 4 runs.

Dataset	Model	16 blocks						8 blocks					
	Local auxiliary	CNN		MLP		Linear		CNN		MLP		Linear	
	Gradient Guess	Activity	Weight	Activity	Weight	Activity	Weight	Activity	Weight	Activity	Weight	Activity	Weight
Fashion-MNIST	Local learning	93.6 ± 0.1		94.1 ± 0.2		93.4 ± 0.1		94.1 ± 0.2		94.4 ± 0.2		93.8 ± 0.2	
	Gaussian Guess	58.3 ± 0.5	85.4 ± 0.6	68.1 ± 2.6	86.1 ± 0.7	67.6 ± 1.8	88.3 ± 4.8	85.2 ± 1.5	89.0 ± 0.4	87.1 ± 0.7	88.9 ± 0.2	82.8 ± 1.4	90.3 ± 0.1
	Radem. Guess	61.6 ± 5.5	85.3 ± 0.5	66.7 ± 1.9	86.2 ± 0.5	65.0 ± 2.3	87.9 ± 0.1	81.5 ± 7.7	89.1 ± 0.1	86.4 ± 0.8	89.1 ± 0.1	82.9 ± 1.5	90.3 ± 0.1
CIFAR-100	Local learning	57.3 ± 0.3		64.6 ± 0.3		62.5 ± 0.2		67.3 ± 0.2		70.3 ± 0.5		65.8 ± 0.5	
	Gaussian Guess	3.7 ± 0.6	10.7 ± 0.4	3.3 ± 0.8	11.2 ± 1.2	3.9 ± 0.8	14.3 ± 2.7	15.7 ± 1.2	25.3 ± 0.4	6.6 ± 1.8	24.9 ± 0.7	11.8 ± 2.1	26.3 ± 0.4
	Radem. Guess	4.4 ± 0.3	7.2 ± 1.5	3.2 ± 0.8	10.4 ± 1.1	4.2 ± 0.7	12.1 ± 0.3	14.4 ± 2.1	24.0 ± 0.9	10.7 ± 2.0	23.1 ± 0.5	4.2 ± 0.7	24.4 ± 0.3
Imagenette	Local learning	84.6 ± 0.3		82.4 ± 0.2		83.2 ± 0.6		88.6 ± 0.4		86.0 ± 0.3		86.2 ± 0.1	
	Gaussian Guess	21.6 ± 2.8	36.7 ± 1.5	21.0 ± 2.9	36.8 ± 1.5	25.8 ± 2.3	39.0 ± 0.8	39.1 ± 4.4	55.6 ± 0.8	44.7 ± 1.9	55.9 ± 0.4	36.5 ± 4.7	58.2 ± 1.2
	Radem. Guess	22.8 ± 3.2	36.3 ± 1.3	20.3 ± 2.7	37.6 ± 2.5	24.9 ± 1.8	40.1 ± 0.8	38.5 ± 1.9	55.9 ± 0.8	42.3 ± 0.8	56.1 ± 1.2	37.4 ± 3.2	59.0 ± 1.1

Table 10. Test accuracy of a Wide ResNet-18 using a Local Target, split into 16 local-loss blocks, on CIFAR-10 for both activity and weight perturbations for different width factors, i.e. k=0.5, 2 and 4. Table 4 refers to the case where $k = 1$. We report the mean and standard deviation over 4 runs.

Width	Local auxiliary	CNN		MLP		Linear	
	Gradient Guess	Activity	Weight	Activity	Weight	Activity	Weight
0.5	Local learning	81.1 ± 0.4		86.9 ± 0.4		83.5 ± 0.4	
	Gaussian Guess	21.0 ± 2.2	34.1 ± 2.3	23.2 ± 4.0	37.7 ± 0.9	13.1 ± 6.3	44.1 ± 1.0
	Rademacher Guess	21.8 ± 3.5	34.4 ± 1.8	24.0 ± 1.2	38.3 ± 0.4	13.0 ± 6.4	43.0 ± 0.9
2	Local Learning	91.1 ± 0.1		90.8 ± 0.1		88.5 ± 0.1	
	Gaussian Guess	17.9 ± 1.0	40.8 ± 1.0	20.5 ± 2.1	42.0 ± 0.6	24.9 ± 7.8	44.3 ± 1.1
	Rademacher Guess	17.9 ± 2.6	41.0 ± 0.4	20.3 ± 1.4	41.7 ± 0.4	23.1 ± 2.3	44.0 ± 2.6
4	Local learning	93.1 ± 0.1		91.7 ± 0.3		89.4 ± 0.1	
	Gaussian Guess	18.7 ± 2.8	41.8 ± 0.5	18.6 ± 3.4	43.1 ± 0.0	23.2 ± 2.1	44.7 ± 0.6
	Rademacher Guess	17.3 ± 3.0	42.3 ± 0.4	18.6 ± 2.4	42.8 ± 0.5	20.7 ± 2.9	44.8 ± 0.5

Table 11. Test accuracy of a ResNet-18 split into 8 or 16 local-loss blocks, using a Fixed NTK (Fixed randomly parameterized local loss) as a Gradient Guess for the Global Target.

Model	8 blocks		16 blocks	
Gradient Guesses	Activity	Weight	Activity	Weight
Fixed NTK, CNN	51.2	57.2	27.5	45.1
Fixed NTK, MLP	52.3	64.6	40.2	45.5
Fixed NTK, Linear	57.7	76.4	36.9	60.5

Table 12. Test accuracy of a ResNet-18 using a predsim auxiliary loss, on CIFAR-10 for both activity and weight perturbations. The Local learning case corresponds to the Local Error Signals framework.

Model	8 blocks		16 blocks	
Global Target	Activity	Weight	Activity	Weight
Local Guess	84.0	89.8	64.8	85.3
NTK Guess	31.8	65.7	11.5	45.5
Local Target	Activity	Weight	Activity	Weight
Local learning	89.2		86.8	
Gaussian Guess	56.2	37.9	37.3	30.8
Rademacher Guess	55.9	39.4	37.2	30.2

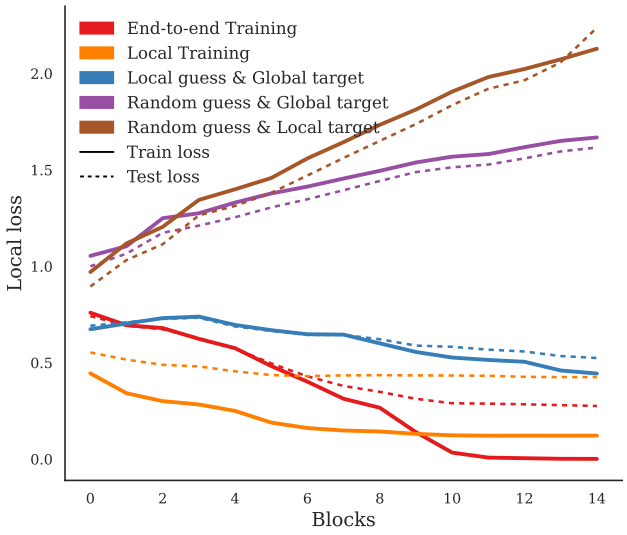

Figure 7. Local train losses at the end of training at each block for a ResNet-18 split into 16 blocks, with CNN auxiliary, for different training algorithms. In the Gaussian and End-to-End cases, the auxiliary training is detached from the main module training and is only for logging purposes.

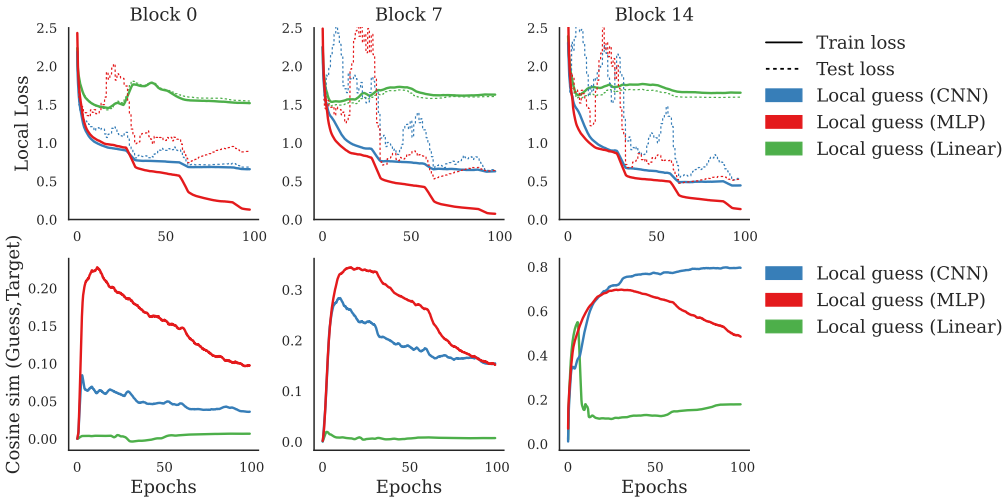

Figure 8. Train and test local losses (top row) and mean cosine similarity between a Local Guess and Global Target in the activation space (bottom row), for blocks 0, 7, and 14 (left, middle, and right columns) during training. The model is a ResNet-18 divided into 16 blocks trained on CIFAR-10.

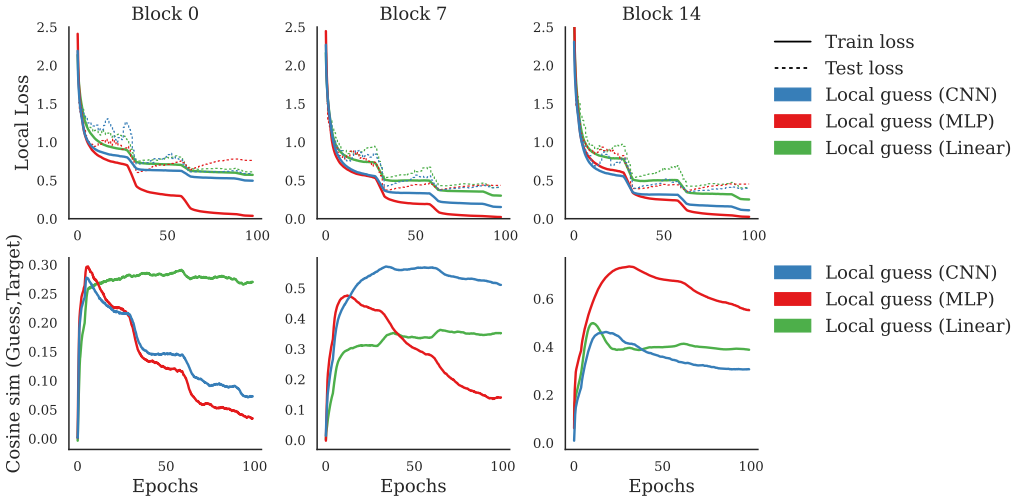

Figure 9. Train and test local losses (top row) and mean cosine similarity between a local guess and global target weight gradients (bottom row, averaged over the parameters), for blocks 0, 7, and 14 (left, middle, and right columns) during training. The generalization gap is more consistent during training than with activation gradients. The cosine similarity is also consistently higher than activation gradients but falls more drastically during training. The model is a ResNet-18 divided into 16 blocks trained on CIFAR-10.