TARGET PROPAGATION VIA REGULARIZED INVERSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Target Propagation (TP) algorithms compute targets instead of gradients along neural networks and propagate them backward in a way that is similar yet different than gradient back-propagation (BP). The idea was first presented as a perturbative alternative to back-propagation that may improve gradient evaluation accuracy when training multi-layer neural networks (Le Cun et al., 1989). However, TP may have remained more of a template algorithm with many variations than a well-identified algorithm. Revisiting insights of Le Cun et al. (1989) and more recently of Lee et al. (2015), we present a simple version of target propagation based on a regularized inversion of network layers, easily implementable in a differentiable programming framework. We compare its computational complexity to the one of BP and delineate the regimes in which TP can be attractive compared to BP. We show how our TP can be used to train recurrent neural networks with long sequences on various sequence modeling problems. The experimental results underscore the importance of regularization in TP in practice.

1 INTRODUCTION

Target propagation algorithms can be seen as perturbative learning alternatives to the gradient back-propagation algorithm, where virtual targets are propagated backward instead of gradients (Le Cun, 1986; Le Cun et al., 1989; Rohwer, 1990; Mirowski & LeCun, 2009; Bengio, 2014; Goodfellow et al., 2016). A high-level summary is presented in Fig. 1: while gradient back-propagation considers storing intermediate gradients in a forward pass, target propagation algorithms proceed by computing and storing approximate inverses. The approximate inverses are then passed on backward along the graph of computations to finally yield a weight update for stochastic learning.

Target propagation aims to take advantage of the availability of approximate inverses to compute better descent directions for the objective at hand. Bengio et al. (2013); Bengio (2020) argued that the approach could be relevant for problems involving multiple compositions such as the training of Recurrent Neural Networks (RNNs), which generally suffer from the phenomenon of exploding or vanishing gradients (Hochreiter, 1998; Bengio et al., 1994; **?**). Recently, empirical results indeed showed the potential advantages of target propagation over classical gradient back-propagation for training RNNs on several tasks (Manchev & Spratling, 2020). However, these recent investigations remain built on multiple approximations, which hinder the analysis of the core idea of TP, i.e., using layer inverses.

On the theoretical side, difference target propagation, a modern variant of target propagation, was related to an approximate Gauss-Newton method, suggesting interesting venues to explain the benefits of target propagation (Bengio, 2020; Meulemans et al., 2020). Previous works have considered approximating inverses by adding multiple reverse layers (Manchev & Spratling, 2020; Meulemans et al., 2020; Bengio, 2020). However, it is unclear whether such reverse layers actually learn layer inverses during the training process. Even if they were, the additional cost of computational complexity of learning approximate inverses should be carefully accounted for.

In this work, we propose a simple target propagation approach, revisiting the original insights of Le Cun et al. (1989) on the critical importance of the good conditioning of layer inverses. We define regularized inverses through a variational formulation and we obtain approximate inverses via these regularized inverses. In this spirit, we can also interpret the difference target propagation formula (Lee et al., 2015) as a finite difference approximation of a linearized regularized inverse. We propose a smoother formula that can directly be integrated into a differentiable programming framework.

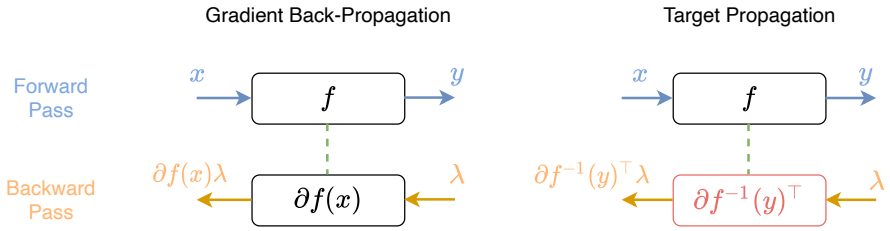


Fig. 1: Our implementation of target propagation uses linearization of gradient inverses instead of gradients in a backward pass akin to gradient back-propagation.

We detail the computational complexity of the proposed target propagation and compare it to the one of gradient back-propagation, showing that the additional cost of computing inverses can be effectively amortized for very long sequences. Following the benchmark of Manchev & Spratling (2020), we observe that the proposed target propagation can perform better than classical gradient-based methods on several tasks involving RNNs.

Related work. Many variations of back-propagation algorithms have been explored; see Werbos (1994); Goodfellow et al. (2016) for an extensive bibliography. Closer to target propagation, penalized formulations of the training problem have been considered to decouple the optimization of the weights in a distributed way or using an ADMM approach (Carreira-Perpinan & Wang, 2014; Taylor et al., 2016; Gotmare et al., 2018). Rather than modifying the backward operations in the layers, one can also modify the weight updates for deep forward networks by using a regularized inverse (Frerix et al., 2018). Wiseman et al. (2017) recast target propagation as an ADMM-like algorithm for language modeling and reported disappointing experimental results. Recently, in a careful experimental benchmark evaluation, Manchev & Spratling (2020) explored further target propagation to train RNNs, mapping a sequence to a single final output, in an attempt to understand the benefits of target propagation to capture long-range dependencies, and obtained promising experimental results. Another line of research has considered synthetic gradients that approximate gradients using an additional layer instead of using back-propagated gradients (Jaderberg et al., 2017; Czarnecki et al., 2017) to speed up the training of deep neural networks. Recently, Ahmad et al. (2020); Dalm et al. (2021) considered using analytical inverses to implement target propagation and blend it with what they called a gradient-adjusted incremental formula. Yet, an additional orthogonality penalty is critical for their approach to work. Recently, Meulemans et al. (2020) considered using as many reverse layers as forwarding operations. We focus here on the optimization gains of using target propagation that cannot be obtained by adding a prohibitive number of reverse layers. Finally, we do not discuss the biological plausibility of TP since we are unable to comment on this. We refer the interested reader to, e.g., (Bengio, 2020).

Notations. For $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^d$, we denote $\partial_x f(x, y) = \left(\partial f^j(x, y)/\partial x_i\right)_{i,j} \in \mathbb{R}^{d\times p}$.

2 Target Propagation with Linearized Regularized Inverses

While target propagation was initially developed for multi-layer neural networks, we focus on its implementation for recurrent neural networks, as we shall follow the benchmark of Manchev & Spratling (2020) in the experiments. Recurrent Neural Networks (RNNs) are also a canonical family of neural networks in which interesting phenomena arise in back-propagation algorithms.

Problem setting. A simple RNN parameterized by $\theta = (W_{hh}, W_{xh}, b_h, W_{hy}, b_y)$ maps a sequence of inputs $x_{1:\tau} = (x_1, \dots, x_\tau)$ to an output $\hat{y} = g_\theta(x_{1:\tau})$ by computing hidden states $h_t \in \mathbb{R}^p$ corresponding to the inputs x_t.

Formally, the output $\hat{y}$ and the hidden states h_t are computed as an output operation following transition operations defined as

$$\begin{aligned} \hat{y} &= c_\theta(h_\tau) := s(W_{hy}h_\tau + b_y), \\ h_t &= f_{\theta,t}(h_{t-1}) := a(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad \text{for } t \in \{1, \dots, \tau\}, \end{aligned}$$

where s is, e.g., the soft-max function for classification tasks, a is a non-linear operation such as the hyperbolic tangent function, and the initial hidden state is generally fixed as $h_0 = 0$. Given samples of sequence-output pairs $(x_{1:\tau}, y)$, the RNN is trained to minimize the error $\ell(y, g_\theta(x_{1:\tau}))$ of predicting $\hat{y} = g_\theta(x_{1:\tau})$ instead of y.

As one considers longer sequences, RNNs face the challenge of exploding/vanishing gradients $\partial g_\theta(x_{1:\tau})/\partial h_t$ (Bengio & Frasconi, 1995); see Appendix A for more discussion. We acknowledge that specific parameterization-based strategies have been proposed to address this issue of exploding/vanishing gradients, such as orthonormal parameterizations of the weights (Arjovsky et al., 2016; Helfrich et al., 2018; Lezcano-Casado & Martınez-Rubio, 2019). The focus here is to simplify and understand target propagation as a backpropagation-type algorithm using RNNs as a workbench. Indeed, training RNNs is an optimization problem involving multiple compositions for which approximate inverses can easily be available. The framework could also be potentially applied to, e.g., time-series or control models (Roulet et al., 2019).

Given the parameters W_{hh}, W_{xh}, b_h of the transition operations, we can get approximate inverses of $f_{\theta,t}(h_{t-1})$ for all $t \in \{1, \dots, \tau\}$, that yield optimization surrogates that can be better performing than the ones corresponding to regular gradients. We present below a *simple version* of target propagation based on *regularized inverses* and *inverse linearizations*.

Back-propagating targets. The idea of target propagation is to compute virtual targets v_t for each layer $t = \tau, \dots, 1$ such that if the layers were able to match their corresponding target at time t, i.e., $f_{\theta,t}(h_{t-1}) \approx v_t$, the objective would decrease. The final target v_τ is computed as a gradient step on the loss w.r.t. h_τ. The targets are then back-propagated using an approximate inverse[1] $f^{-1}_{\theta,t}$ of $f_{\theta,t}$ at each time step.

Formally, consider an RNN that computed τ states $h_1, \dots, h_\tau$ from a sequence $x_1, \dots, x_\tau$ with associated output y. For a given stepsize $\gamma_h > 0$, we propose to back-propagate targets by computing

$$v_\tau = h_\tau - \gamma_h \partial_h \ell(y, c_\theta(h_\tau)), \tag{1}$$

$$v_{t-1} = h_{t-1} + \partial_h f^{-1}_{\theta,t}(h_t)^\top (v_t - h_t), \quad \text{for } t \in \{\tau, \dots, 1\}. \tag{2}$$

The update rule (2) blends two ideas: i) regularized inversion; ii) linear approximation. We shall describe below that our update (2) allows us to interpret the "magic formula" of difference target propagation in Eq. 15 of Lee et al. (2015) as 0th-order finite difference approximation, while ours is a 1st-order linear approximation. We shall also show that (2) puts in practice an insight from Bengio (2020) suggesting to use the inverse of the gradients in the spirit of a Gauss-Newton method.

Once all targets are computed, the parameters of the transition operations are updated such that the outputs of $f_{\theta,t}$ at each time step move closer to the given target. Formally, the update consists of a gradient step with stepsize γ_θ on the squared error between the targets and the current outputs, i.e., for $\theta_h \in \{W_{hh}, W_{xh}, b_h\}$,

$$\theta_h^{\text{next}} = \theta_h - \gamma_\theta \sum_{t=1}^{\tau} \partial_{\theta_h} \|f_{\theta,t}(h_{t-1}) - v_t\|_2^2/2. \tag{3}$$

As for the parameters $\theta_y = (W_{hy}, b_y)$ of the output operation, they are updated by a simple gradient step on the loss with a stepsize γ_θ.

2.1 Regularized Inversion

To explore further the original idea of Le Cun et al. (1989), we consider using the variational definition of the inverse,

$$f^{-1}_{\theta,t}(v_t) = \operatorname*{argmin}_{v_{t-1} \in \mathbb{R}^p} \|f_{\theta,t}(v_{t-1}) - v_t\|_2^2 = \operatorname*{argmin}_{v_{t-1} \in \mathbb{R}^p} \|a(W_{xh} x_t + W_{hh} v_{t-1} + b_h) - v_t\|_2^2. \tag{4}$$

As long as v_t belongs to the image $f_{\theta,t}(\mathbb{R}^p)$ of $f_{\theta,t}$, this definition recovers exactly the inverse of v_t by $f_{\theta,t}$. More generally, if $v_t \notin f_{\theta,t}(\mathbb{R}^p)$, Eq. (4) computes the *best approximation* of the

[1] In the following, to ease the presentation, we abuse notations and denote approximate inverses by $f^{-1}_{\theta,t}$.

inverse in the sense of the Euclidean projection. When one considers an activation function a and $\theta_h = (W_{hh}, W_{xh}, b_h)$, the solution of (4) can easily be computed.

Formally, for the sigmoid, the hyperbolic tangent or the ReLU, their inverse can be obtained analytically for any $v_t \in a(\mathbb{R}^p)$. So for $v_t \in a(\mathbb{R}^p)$ and W_{hh} full rank, we get

$$f_{\theta,t}^{-1}(v_t) = (W_{hh}^\top W_{hh})^{-1} W_{hh}^\top (a^{-1}(v_t) - W_{xh} x_t - b_h).$$

If $v_t \notin a(\mathbb{R}^p)$, the minimizer of (4) is obtained by first projecting v_t onto $a(\mathbb{R}^p)$, before inverting the linear operation. To account for non-invertible matrices W_{hh}, we also add a regularization in the computation of the inverse. Overall we consider approximating the inverse of the layer by a regularized inverse of the form

$$f_{\theta,t}^{-1}(v_t) = (W_{hh}^\top W_{hh} + r\,\mathrm{I})^{-1} W_{hh}^\top (a^{-1}(\pi(v_t)) - W_{xh} x_t - b_h),$$

with $r > 0$ and π a projection onto $a(\mathbb{R}^p)$.

Regularized inversion vs. parameterized inversion. Bengio (2014); Manchev & Spratling (2020) parameterize the inverse as a reverse layer such that

$$f_{\theta,t}^{-1}(v_t) = \psi_{\theta',t}(v_t) := a(W_{xh} x_t + V v_t + c),$$

and learn the parameters $\theta' = (V, c)$ for this reverse layer to approximate the inverse of the forward computations. The parameterized layer needs to be learned to get a good approximation which involves numerically solving an optimization problem for each layer. These optimization problems come with a computational cost that can be better controlled by using regularized inversions presented earlier.

However, the approach based on parameterized inverses may lack theoretical grounding, as pointed out by Bengio (2020), as we do not know how close the learned inverse is to the actual inverse throughout the training process. In contrast, the regularized inversion (4) is less *ad hoc* and clearly defined and, as we shall show in the experiments, leads to competitive performance on real datasets.

In any case, the analytic formulation of the inverse gives simple insights on an approach with parameterized inverses. Namely, the analytical formula suggests parameterizing the *reverse layer* s.t. (i) the reverse activation is defined as the inverse of the activation and not any activation, (ii) the layer uses a non-linear operation followed by a linear one instead of the usual scheme, i.e., a linear operation followed by a non-linear one.

2.2 Linearized Inversion

Earlier instances of target propagation used direct inverses of the network layers such that the target propagation update formula would read $v_{t-1} = f_{\theta,t}^{-1}(v_t)$ in (2). Yet, we are unaware of a successful implementation of TP using directly the inverses. To circumvent this issue, Lee et al. (2015) proposed the *difference target propagation* formula that back-propagates the targets as

$$v_{t-1} = h_{t-1} + f_{\theta,t}^{-1}(v_t) - f_{\theta,t}^{-1}(h_t).$$

If the inverses were exact, the difference target propagation formula would reduce to $v_{t-1} = f_{\theta,t}^{-1}(v_t)$. Lee et al. (2015) introduced the difference target propagation formula to mitigate the approximation error of the inverses by parameterized layers. In practice, difference-type target propagation appears to be the only known successful implementation of target propagation we are aware of. The difference target propagation formula can naturally be interpreted as an approximation of the linearization used in (2), as

$$f_{\theta,t}^{-1}(v_t) - f_{\theta,t}^{-1}(h_t) = \partial_h f_{\theta,t}^{-1}(h_t)^\top (v_t - h_t) + O(\|v_t - h_t\|_2^2),$$

where $\partial_h f_{\theta,t}^{-1}(h_t)^\top$ denotes the Jacobian of the inverse of the layer at h_t.

We show in Appendix D that the first-order approximation we propose (2) leads to slightly better training curves than the finite-difference approximation. Moreover, our interpretation illuminates the "mystery" of this formula, which appeared to be critical to the success of target propagation.

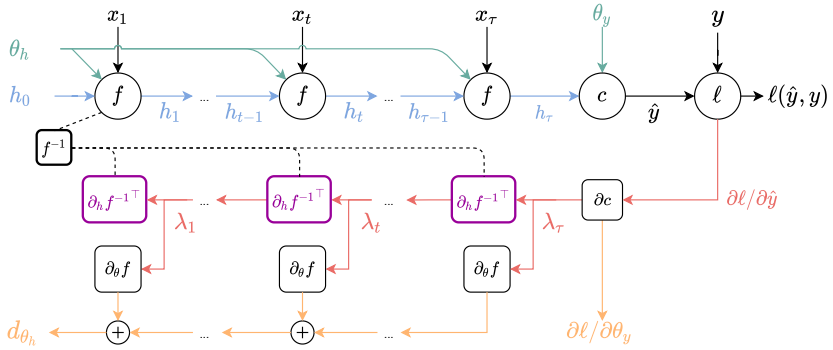


Fig. 2: The graph of computations of target propagation is the same as the one of gradient back-propagation except that f^{-1} needs to be computed and Jacobian of the inverses, $\partial_h f^{-1\top}$ are used instead of gradients $\partial_h f$ in the transition operations.

3 Gradient Back-propagation versus Target Propagation

Graph of computations. Gradient back-propagation and target propagation both compute a descent direction for the objective at hand. The difference lies in the oracles computed and stored in the forward pass, while the graph of computations remains the same. To clarify this view, we reformulate target propagation in terms of displacements $\lambda_t := v_t - h_t$ such that Eq. (1), (2) and (3) read

$$\lambda_\tau = -\gamma_h \partial_h \ell(y, c_\theta(h_\tau)), \qquad \lambda_{t-1} = \partial_h f_{\theta,t}^{-1}(h_t)^\top \lambda_t, \quad \text{for } t \in \{\tau, \dots, 1\},$$
$$d_{\theta_h} = \sum_{t=1}^{\tau} \partial_{\theta_h} f_{\theta,t}(h_{t-1}) \lambda_t, \qquad \theta_h^{\text{next}} = \theta_h + \gamma_h d_{\theta_h}.$$

Target propagation amounts then to computing a descent direction d_{θ_h} for the parameters θ_h with a graph of computations, illustrated in Fig. 2, analogous to that of gradient-back-propagation illustrated in Appendix A. The difference lies in the use of the Jacobian of the inverse

$$\partial_h f_{\theta,t}^{-1}(h_t)^\top \quad \text{instead of} \quad \partial_h f_{\theta,t}(h_{t-1}).$$

The implementation of TP with the formula (2) can be done in a differentiable programming framework, where, rather than computing the gradient of the layer, one evaluates the inverse and keep the Jacobian of the inverse. With the precise graph of computation of TP and BP, we can compare their computational complexity explicitly and bound the difference of the directions they output.

Arithmetic complexity. Clearly, the space complexities of gradient back-propagation (BP) and our implementation of target propagation (TP) are the same since the Jacobians of the inverse, and the original gradients have the same size. In terms of time complexity, TP appears at first glance to introduce an important overhead since it requires the computation of some inverses. However, a close inspection of the formula of the regularized inverse reveals that a matrix inversion needs to be computed only once for all time steps. Therefore the cost of the inversion may be amortized if the length of the sequence is particularly long.

Formally, the time complexity of the forward-backward pass of gradient back-propagation is essentially driven by matrix-vector products, i.e.,

$$\mathcal{T}_{\text{BP}} = \sum_{t=1}^{\tau} \Big[\underbrace{\mathcal{T}(f_{\theta,t}) + \mathcal{T}\left(\partial_h f_{\theta,t}\right) + \mathcal{T}\left(\partial_{\theta_h} f_{\theta,t}\right)}_{\text{Forward}} + \underbrace{\mathcal{T}(\partial_h f_{\theta,t}(h_{t-1})) + \mathcal{T}(\partial_\theta f_{\theta,t}(h_{t-1}))}_{\text{Backward}} \Big]$$
$$\approx \tau(dp + p^2 + pq) + \tau(p^2 + pq),$$

where d is the dimension of the input x_t, q is the dimension of the parameters θ_h, for a function f we denote by $\mathcal{T}(f)$ the time complexity to evaluate f and we consider e.g. $\partial_\theta f_{\theta,t}(h_{t-1}))$ as the linear function $\lambda \to \partial_\theta f_{\theta,t}(h_{t-1}))\lambda$.

On the other hand, the time complexity of target propagation is

$$\mathcal{T}_{\text{TP}} = \sum_{t=1}^{\tau} \Bigg[\underbrace{\mathcal{T}(f_{\theta,t}) + \mathcal{T}(f_{\theta,t}^{-1}) + \mathcal{T}\left(\partial_{\theta_h} f_{\theta,t}\right) + \mathcal{T}(\partial_h f_{\theta,t}^{-1})}_{\text{Forward}} + \underbrace{\mathcal{T}(\partial_h f_{\theta,t}^{-1})(h_t)^\top) + \mathcal{T}(\partial_\theta f_{\theta,t}(h_{t-1}))}_{\text{Backward}} \Bigg] + \mathcal{P}(f_{\theta,t}^{-1}),$$

where $\mathcal{P}(f_{\theta,t}^{-1})$ is the cost of encoding the inverse, which, in our case, amounts to the cost of encoding $g_\theta : z \to (W_{hh}^\top W_{hh} + r\,\mathrm{I})^{-1} W_{hh}^\top$, such that our regularized inverse can be computed as $f_{\theta,t}^{-1}(v_t) = g_\theta(a^{-1}(v_t) - W_{xh}x_t + b_h)$. Encoding g comes at the cost of inverting one matrix of size p. Therefore, the time-complexity of target propagation can be estimated as

$$\mathcal{T}_{\text{TP}} \approx p^3 + \tau(dp + p^2 + pq) + \tau(p^2 + pq) \approx \mathcal{T}_{\text{BP}} \quad \text{if } \tau \geq p,$$

i.e., for long sequences whose length is larger than the dimension of the hidden states, the cost of TP with regularized inverses is approximately the same as the cost of BP. If a parameterized inverse was used rather than a regularized inverse, the cost of encoding the inverse would correspond to the cost of updating the reverse layers by, e.g., a stochastic gradient descent. This update has a cost similar to BP. However, it is unclear whether these updates get us close to the actual inverses.

Bounding the difference between target propagation and gradient back-propagation. As the computational graphs of BP and TP are the same, we can bound the difference between the oracles returned by both methods. First, note that the updates of the parameters of the output functions are the same since, in TP, gradients steps of the loss are used to update these parameters. The difference between TP and BP lies in the updates with respect to the parameters of the transition operations. For BP, the updates are computed by chain rule as

$$\partial_{\theta_h} \ell\left(y, g_\theta(x_{1:\tau})\right) = \sum_{t=1}^{\tau} \partial_{\theta_h} f_{\theta,t}(h_{t-1}) \frac{\partial h_\tau}{\partial h_t} \partial_h \ell(y, c_\theta(h_\tau)),$$

where the term $\partial h_\tau / \partial h_t$ decomposes along the time steps as $\partial h_\tau / \partial h_t = \prod_{s=t+1}^{\tau} \partial_h f_{\theta,s}(h_{s-1})$. The direction computed by TP has the same structure, namely it can be decomposed for $\gamma_h = 1$ as

$$d_\theta = \sum_{t=1}^{\tau} \partial_{\theta_h} f_{\theta,t}(h_{t-1}) \frac{\hat{\partial} h_\tau}{\hat{\partial} h_t} \partial_h \ell(y, c_\theta(h_\tau)),$$

where $\hat{\partial} h_\tau / \hat{\partial} h_t = \prod_{s=t+1}^{\tau} \partial_h f_{\theta,s}^{-1}(h_s)^\top$. We can then bound the difference between the directions given by BP or TP as, for any matrix norm $\|\cdot\|$ as formally stated in the following lemma.

Lemma 3.1. *The difference between the oracle returned by gradient back-propagation $\partial_{\theta_h} \ell\left(y, g_\theta(x_{1:\tau})\right)$ and the oracle returned by target propagation can be bounded as*

$$\|\partial_{\theta_h} \ell\left(y, g_\theta(x_{1:\tau})\right) - d_\theta\| \leq c \sup_{t=1,\dots,\tau} \|\partial_h f_{\theta,t}(h_{t-1}) - \partial_h f_{\theta,t}^{-1}(h_t)^\top\|,$$

where $c = \sum_{t=1}^{\tau} \sum_{s=0}^{t-1} a^s b^{t-1-s}$ *with* $a = \sup_{t=1,\dots\tau} \|\partial_h f_{\theta,t}(h_{t-1})\|, b = \sup_{t=1,\dots\tau} \|\partial_h f_{\theta,t}^{-1}(h_t)^\top\|$.

For regularized inverses, we have, denoting $u_t = W_{xh}x_t + W_{hh}h_{t-1} + b_h$,

$$\|\partial_h f_{\theta,t}(h_{t-1}) - \partial_h f_{\theta,t}^{-1}(h_t)^\top\| \leq \|W_{hh}^\top\| \Big(\|\nabla a(u_t) - \nabla a(u_t)^{-1}\| + \|\,\mathrm{I} - (W_{hh}^\top W_{hh} + r\,\mathrm{I})^{-1}\| \|\nabla a(u_t)^{-1}\| \Big).$$

For the two oracles to be close, we then need the preactivation $u_t = W_{xh}x_t + W_{hh}h_{t-1} + b_h$ to lie in the region of the activation function that is close to being linear s.t. $\nabla a(u_t) \approx \mathrm{I}$. We also need $(W_{hh}^\top W_{hh} + r\,\mathrm{I})^{-1}$ to be close to the identity which can be the case if, e.g., $r = 0$ and the weight matrices W_h were orthonormal. By initializing the weight matrices as orthonormal matrices, the differences between the two oracles can be closer. However, in the long term, target propagation appears to give better oracles, as shown in the experiments below.

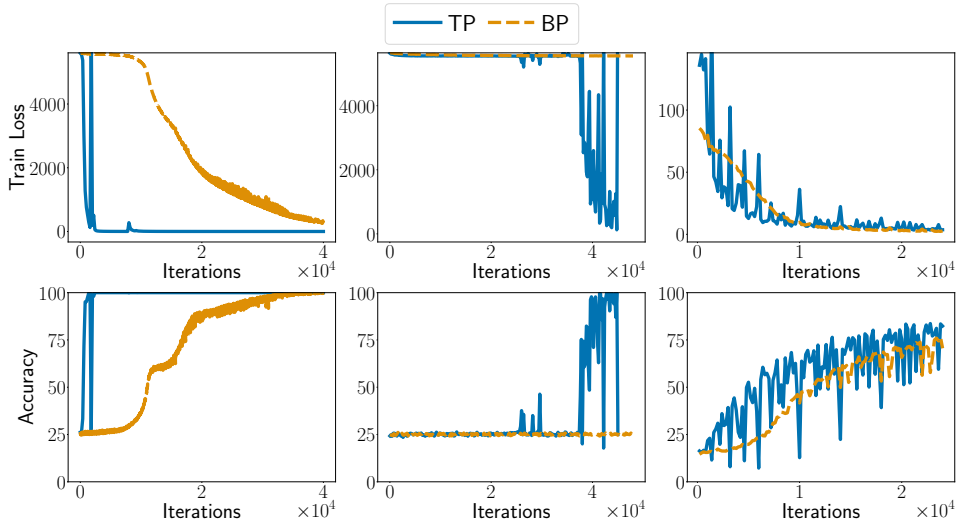


Fig. 3: Temporal order problem $T = 60$, Temporal Problem $T = 120$, Adding problem $T = 30$.

Target propagation as a Gauss-Newton method? Recently target propagation has been interpreted as an approximate Gauss-Newton method, by considering that the difference target propagation formula approximates the linearization of the inverse, which itself is a priori equal to the inverse of the gradients (Bengio, 2020; Meulemans et al., 2020; 2021). Namely, provided that $f_{\theta,t}^{-1}(f_{\theta,t}(h_{t-1})) \approx h_{t-1}$ such that $\partial_h f_{\theta,t}(h_{t-1})\partial_h f_{\theta,t}^{-1}(h_t) \approx \mathrm{I}$, we have

$$\partial_h f_{\theta,t}^{-1}(h_t) \approx (\partial_h f_{\theta,t}(h_{t-1}))^{-1}.$$

By composing the inverses of the gradients, we get an update similar to the one of Gauss-Newton (GN) method. Namely, recall that if n invertible functions $f_1, \dots, f_n$ were composed to solve a least square problem of the form $\|f_n \circ \dots \circ f_1(x) - y\|_2^2$, a Gauss-Newton update would take the form $x^{(k+1)} = x^{(k)} - \partial_{x_0} f_1(x_0)^{-\top} \partial_{x_1} f_2(x_1)^{-\top} \dots \partial_{x_{n-1}} f(x_{n-1})^{-\top}(x_n - y)$ where x_t is defined iteratively as $x_0 = x^{(k)}$, $x_{t+1} = f_t(x_t)$. In other words, GN and TP share the idea of composing the inverse of gradients. However, numerous differences remain: (i) in e.g. RNNs, the gradients w.r.t. to the weights are defined as a sum of the composition of the gradients and the inverse of this sum is a priori not the sum of the inverses, (ii) if the real rationale of GN was used, the gradients w.r.t. the loss should also be inverted, and the gradients w.r.t. to the weights should also be inverted which is not the case in the usual implementation of TP Lee et al. (2015); Bengio (2020). Even if TP was approximating GN, it is unclear whether GN updates are adapted to stochastic problems.

4 Experiments

In the following, we compare our simple target propagation approach, which we shall refer to as **TP**, to gradient Back-Propagation referred to as **BP**. We follow the experimental benchmark of Manchev & Spratling (2020) to which we add results on RNNs on CIFAR and GRUs on FashionMNIST. Additional experiments, details on the initialization and the hyper-parameter selection can be found in Appendix D.

Data. We consider two synthetic datasets generated to present training difficulties for RNNs and several real datasets consisting of scanning images pixel by pixel to classify them (Hochreiter & Schmidhuber, 1997; Le et al., 2015; Manchev & Spratling, 2020).

Temporal order problem. A sequence of length T is generated using a set of randomly chosen symbols $\{a, b, c, d\}$. Two additional symbols X and Y are added at positions $t_1 \in [T/10, 2T/10]$ and $t_2 \in [4T/10, 5T/10]$. The network must predict the correct order of appearance of X and Y out of four possible choices $\{XX, XY, YX, YY\}$.

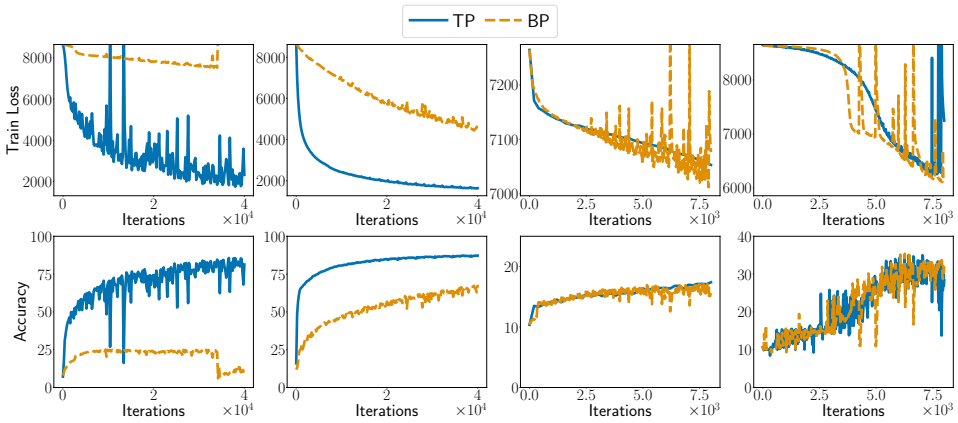


Fig. 4: Image classification pixel by pixel. From left to right: MNIST, MNIST with permuted images, CIFAR10, FashionMNIST with GRU.

Adding problem. The input consists of two sequences: one is made of randomly chosen numbers from $[0, 1]$, and the other one is a binary sequence full of zeros except at positions $t_1 \in [1, T/10]$ and $t_2 \in [T/10, T/2]$. The second position acts as a marker for the time steps t_1 and t_2. The goal of the network is to output the mean of the two random numbers of the first sequence $(X_{t_1} + X_{t_2})/2$.

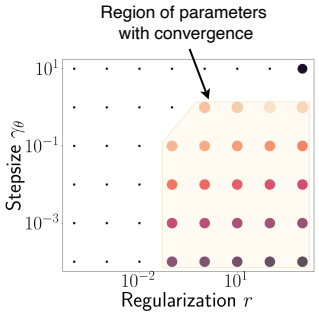
Image classification pixel by pixel. The inputs are images of (i) grayscale handwritten digits given in the database MNIST (LeCun & Cortes, 1998), (ii) colored objects from the database CIFAR10 (Krizhevsky, 2009) or (iii) grayscale images of clothes from the database FashionMNIST (Xiao et al., 2017). The images are scanned pixel by pixel and channel by channel for CIFAR10, and fed to a sequential network such as a simple RNN or a GRU network (Cho et al., 2014). The inputs are then sequences of $28 \times 28 = 784$ pixels for MNIST or FashionMNIST and $32 \times 32 \times 3 = 3072$ pixels for CIFAR with a very long-range dependency problem. We also consider permuting the images of MNIST by a fixed permutation before feeding them into the network, which gives potentially longer dependencies in the sequential data.

Model. In both synthetic settings, we consider randomly generated mini-batches of size 20, a simple RNN with hidden states of dimension 100, and hyperbolic tangent activation. For the temporal order problem, the last layer uses a soft-max function on top of a linear operation, and the loss is the cross-entropy. For the adding problem, the last layer is linear, the loss is the mean-squared error, and a sample is considered to be accurately predicted if the mean squared error is less than 0.04 as done by (Manchev & Spratling, 2020).

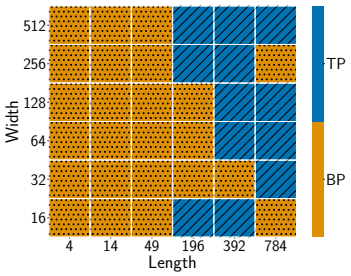
For the classification of images with sequential networks, we consider mini-batches of size 16 and a cross-entropy loss. For MNIST and CIFAR, we consider a simple RNN with hidden states of dimension 100, hyperbolic tangent activation, and a softmax output. For FashionMNIST, we consider a GRU network and adapted our implementation of target propagation to that case while using hidden states of dimension 100 and a softmax output.

Target propagation can tackle long sequences better than gradient back-propagation. In Fig. 3, we observe that TP performs better than BP on the temporal ordering problem: it is able to reach 100% accuracy in fewer iterations than BP for sequences of length 60 and, for sequences of length 120, it is still able to reach 100% accuracy in fewer than 40 000 iterations while BP is not. On the other hand, for the adding problem, TP performs less well than BP. The contrast in performance between the two synthetic tasks was also observed by (Manchev & Spratling, 2020) using difference target propagation with parameterized inverses. The main difference between these tasks is the different nature of the outputs, which are binary for the temporal problem and continuous for the adding problem.

In Fig. 4, we observe that TP generally performs better than BP for image classification tasks. For the MNIST dataset, it reaches around 74% accuracy after $4 \cdot 10^4$ iterations. This phenomenon is also



(6a) Conv. w.r.t. stepsize & regularization



(6b) Perf. vs width & length.

observed with permuted images, where the optimization appears smoother, and TP obtains around 86% accuracy after $4 \cdot 10^4$ iterations and is still faster than BP. On the CIFAR dataset, no algorithms appear to reach a significant accuracy, though TP is still faster. On the FashionMNIST dataset, where a GRU network is used, our implementation of TP performs on par with BP, which shows that our approach can be generalized to more complex networks than a simple RNN.

Target propagation requires a non-zero regularization term. As mentioned in Sec. 3, by using an analytical formula to compute the inverse of the layers, we can question the interpretation of TP as a Gauss-Newton method, which would amount to TP without regularization. To understand the effect of the regularization term, we computed the area under the training loss curve of TP for 400 iterations on a $\log_{10}$ grid of varying step-sizes γ_θ and regularizations r for a fixed $\gamma_h = 10^{-3}$. The results are presented in Fig. 6a, where the smaller the area, the brighter the point and the absence of dots in the grid mean that the algorithm diverged. Fig. 6a shows that without regularization we were not able to obtain convergence of the algorithm. Simply using the gradients of the inverse as in a Gauss-Newton method may not directly work for RNNs. Additional modifications of the method could be added to make target propagation closer to Gauss-Newton, such as inverting the layers with respect to their parameters as proposed by Bengio (2020). For now, the regularization appears to successfully handle the rationale of target propagation.

Target propagation is adapted for long sequences. In Fig. 6b, we compare the performance of BP and TP in terms of accuracy after 400 iterations on the MNIST problem for various widths determined by the size of the hidden states and various lengths determined by the size of the inputs (i.e., we feed the RNN with k pixels at a time, which gives a length $784/k$). Fig 6b shows that TP is generally appropriate for long sequences, while BP remains more efficient for short sequences. TP can then be seen as an interesting alternative for dynamical problems which involve many discretization steps as in RNNs and related architectures.

Conclusion

We proposed a simple target propagation approach grounded in two important computational components, regularized inversion, and linearized propagation. The proposed approach also sheds light on previous insights and successful rules for target propagation. We will publicly release our code to facilitate the reproduction of the results. We have used target propagation within a stochastic gradient outer loop to train neural networks for a fair comparison to stochastic gradient using gradient backpropagation. Developing adaptive stochastic gradient algorithms in the spirit of Adam that lead to boosts in performance when using target propagation instead of gradient backpropagation is an interesting avenue for future work. Continuous counterparts of target propagation in a neural ODE spirit is also an interesting avenue for future work.

References

Nasir Ahmad, Marcel A van Gerven, and Luca Ambrogioni. Gait-prop: A biologically plausible learning rule derived from backpropagation of error. *Advances in Neural Information Processing Systems*, 33, 2020.

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Yoshua Bengio. How auto-encoders could provide credit assignment in deep networks via target propagation. *arXiv preprint arXiv:1407.7906*, 2014.

Yoshua Bengio. Deriving differential target propagation from iterating approximate inverses. *arXiv preprint arXiv:2007.15139*, 2020.

Yoshua Bengio and Paolo Frasconi. Diffusion of context and credit information in markovian models. *Journal of Artificial Intelligence Research*, 3:249–270, 1995.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Miguel Carreira-Perpinan and Weiran Wang. Distributed optimization of deeply nested systems. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, 2014.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Wojciech Marian Czarnecki, Grzegorz Świrszcz, Max Jaderberg, Simon Osindero, Oriol Vinyals, and Koray Kavukcuoglu. Understanding synthetic gradients and decoupled neural interfaces. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Sander Dalm, Nasir Ahmad, Luca Ambrogioni, and Marcel van Gerven. Scaling up learning with gait-prop. *arXiv preprint arXiv:2102.11598*, 2021.

Olivier Devolder, François Glineur, and Yurii Nesterov. First-order methods of smooth convex optimization with inexact oracle. *Mathematical Programming*, 146(1-2):37–75, 2014.

Thomas Frerix, Thomas Möllenhoff, Michael Moeller, and Daniel Cremers. Proximal backpropagation. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.

Akhilesh Gotmare, Valentin Thomas, Johanni Brea, and Martin Jaggi. Decoupling backpropagation using constrained optimization methods. In *Credit Assignment in Deep Learning and Reinforcement Learning Workshop (ICML 2018 ECA)*, 2018.

Kyle Helfrich, Devin Willmott, and Qiang Ye. Orthogonal recurrent neural networks with scaled cayley transform. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02): 107–116, 1998.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

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*, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

Yann Le Cun. Learning process in an asymmetric threshold network. In *Disordered systems and biological organization*. Springer, 1986.

Yann Le Cun, Conrad C Galland, and Geoffrey E Hinton. GEMINI: gradient estimation through matrix inversion after noise injection. In *Advances in neural information processing systems*, pp. 141–148, 1989.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 1998.

Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 2015.

Mario Lezcano-Casado and David Martınez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Nikolay Manchev and Michael Spratling. Target propagation in recurrent neural networks. *Journal of Machine Learning Research*, 21(7):1–33, 2020.

Alexander Meulemans, Francesco Carzaniga, Johan Suykens, João Sacramento, and Benjamin F. Grewe. A theoretical framework for target propagation. In *Advances in Neural Information Processing Systems 33*, 2020.

Alexander Meulemans, Matilde Tristany Farinha, Javier García Ordóñez, Pau Vilimelis Aceituno, João Sacramento, and Benjamin F Grewe. Credit assignment in neural networks through deep feedback control. *arXiv preprint arXiv:2106.07887*, 2021.

Piotr Mirowski and Yann LeCun. Dynamic factor graphs for time series modeling. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 2009.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *arXiv preprint arXiv:1211.5063*, 2012.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. 2019.

Richard Rohwer. The “moving targets" training algorithm. In *Advances in neural information processing systems 3*, 1990.

Vincent Roulet, Siddhartha Srinivasa, Dmitriy Drusvyatskiy, and Zaid Harchaoui. Iterative linearized control: stable algorithms and complexity guarantees. In *International Conference on Machine Learning*, 2019.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*. MIT Press, Cambridge, MA, USA, 1986.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, 2011.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th International conference on machine learning*, 2013.

Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable ADMM approach. In *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

Paul Werbos. *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*. Wiley-Interscience, 1994.

Sam Wiseman, Sumit Chopra, Marc-Aurelio Ranzato, Arthur Szlam, Ruoyu Sun, Soumith Chintala, and Nicolas Vasilache. Training language models using target-propagation. *arXiv preprint arXiv:1702.04770*, 2017.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017. URL https://github.com/zalandoresearch/fashion-mnist.

APPENDIX PLAN

The Appendix is organized as follows.

1. Sec. A recalls how gradient back-propagation works for RNNs.
2. Sec. B details the implementations of target propagation.
3. Sec. C details the differences between TP and gradient back-propagation or Gauss-Newton optimization.
4. Sec. D details the parameters used in our experiments and presents additional experiments.

A GRADIENT BACK-PROPAGATION IN RECURRENT NEURAL NETWORKS

Given differentiable activation functions a, the training of recurrent neural networks is amenable to optimization by gradient descent. The gradients can be computed by gradient back-propagation implemented in modern differentiable programming software (Rumelhart et al., 1986; Werbos, 1994; Paszke et al., 2019). The gradient back-propagation algorithm is illustrated in Fig. 6. Formally, the gradients are computed by the chain rule such that, for a sample $(y, x_{1:\tau})$ and $\theta_h = (W_{hh}, W_{xh}, b_h)$,

$$\frac{\partial \ell\left(y, g_\theta(x_{1:\tau})\right)}{\partial \theta_h} = \sum_{t=1}^{\tau} \frac{\partial h_t}{\partial \theta_h} \frac{\partial h_\tau}{\partial h_t} \frac{\partial \hat{y}}{\partial h_\tau} \frac{\partial \ell}{\partial \hat{y}}.$$

The term $\partial h_\tau / \partial h_t$ decomposes along the time steps as

$$\frac{\partial h_\tau}{\partial h_t} = \prod_{s=t+1}^{\tau} \frac{\partial h_s}{\partial h_{s-1}}.$$

As τ grows, the norm of the term $\partial h_\tau / \partial h_t$ can then either increase to infinity (*exploding gradients*) or exponentially decrease to 0 (*vanishing gradients*). This phenomenon may prevent the RNN from learning from dependencies between temporally distant events (Hochreiter, 1998). Several solutions were proposed to tackle this issue, including changing the network architecture (Hochreiter & Schmidhuber, 1997), Hessian-free optimization (Sutskever et al., 2011), gradient clipping and regularization (Pascanu et al., 2012), or orthonormal parametrizations (Arjovsky et al., 2016; Helfrich et al., 2018; Lezcano-Casado & Martınez-Rubio, 2019). We consider here propagating targets instead of gradients as first presented by LeCun and co-workers (Le Cun, 1986; Le Cun et al., 1989) and recently revisited by Bengio and co-workers (Bengio, 2014; Lee et al., 2015).

B DETAILED IMPLEMENTATION

B.1 TARGET PROPAGATION FOR RNNS

As detailed in Sec. 3, target propagation with linearized regularized inverses amounts to move along a descent direction computed by a forward-backward algorithm akin to gradient propagation. The iterations of linearized target propagation are then summarized in Algo. 1. The iterations of Algo. 1 make calls to any algorithm providing a descent direction which is computed by Algo. 2.

In the implementation of the regularized inverses, since the inverse of activation functions such as the sigmoid or the tangent hyperbolic is numerically unstable, we consider projecting on a subset of $a(\mathbb{R}^p)$. For the hyperbolic tangent, we clip the target to $[-1+\varepsilon, 1-\varepsilon]$ for $\varepsilon = 10^{-3}$. Concretely, for an hyperbolic tangent activation function, the projection is then $\pi(x) = (\min(\max(x_i, -1+\varepsilon), 1-\varepsilon))_{i=1}^d$ for $x \in \mathbb{R}^d$. To read Algo. 2, we recall our notations for $\theta = (W_{hh}, W_{xh}, b_h, W_{hy}, b_y)$:

$$c_\theta(h_\tau) = \alpha(W_{hy} h_\tau + b_y), \tag{5}$$

$$f_{\theta,t}(h_{t-1}) = a(W_{xh} x_t + W_{hh} h_{t-1} + b_h), \tag{6}$$

$$f_{\theta,t}^{-1}(v_t) = (W_{hh}^\top W_{hh} + r\,\mathrm{I})^{-1} W_{hh}^\top (a^{-1}(\pi(v_t)) - W_{xh} x_t - b_h). \tag{7}$$

Note that Algo. 2 can also be used for mini-batches of sequence-output pairs since all operations are either element-wise or linear with respect to the sample of sequence-output pair.

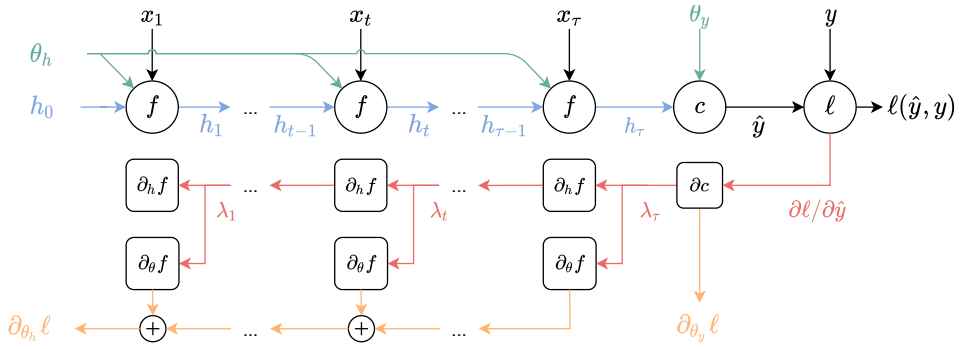


Fig. 6: Gradient back-propagation for RNN.

Algorithm 1 Stochastic learning with target propagation

1: **Inputs:** Initial parameters $\theta^{(0)} = (W_{hh}, W_{xh}, b_h, W_{hy}, b_y)$ of an RNN defined by Eq. (5) and (6), stepsize γ_θ, total number of iterations K
2: **for** $k = 1 \dots K$ **do**
3: Draw a sample or a mini-batch of sequences-output pairs $(x_{1:\tau}, y)$.
4: Compute

$$d_\theta = (d_{\theta_h}, d_{\theta_y}) = \text{TP}(\theta^{(k-1)}, x_{1:\tau}, y),$$

where TP is Algo. 2
5: Update the parameters as $\theta^{(k)} = \theta^{(k-1)} + \gamma_\theta d_\theta$.
6: **end for**

B.2 Target-propagation for GRU networks

B.2.1 Formulation

Starting from $h_0 = 0$, given an input sequence $x_1, \dots, x_\tau$, the GRU network (as implemented in Pytorch[2] (Paszke et al., 2019)), iterates for $t = 1, \dots, \tau$,

$$m_t = f_{m,t}(h_{t-1}) := \sigma(W_{im} x_t + W_{hm} h_{t-1} + b_m) \tag{8}$$
$$z_t = f_{z,t}(h_{t-1}) := \sigma(W_{iz} x_t + W_{hz} h_{t-1} + b_z) \tag{9}$$
$$n_t = f_{n,t}(h_{t-1}, m_t) := \tanh(W_{in} x_t + b_{in} + m_t \odot (W_{hn} h_{t-1} + b_{hn})) \tag{10}$$
$$h_t = f_{h,t}(h_{t-1}, z_t, n_t) := (1 - z_t) \odot h_{t-1} + z_t \odot n_t, \tag{11}$$

where $\odot$ is the Hadamard product, σ is a sigmoid. In the following, we will denote simply $\theta = (\theta_m, \theta_z, \theta_n)$ the parameters of the network with

$$\theta_m = (W_{im}, W_{hm}, b_m), \qquad \theta_z = (W_{iz}, W_{hz}, b_z), \qquad \theta_n = (W_{in}, b_{in}.W_{hn}, b_{hn}).$$

The output of the network is e.g. a soft-max operation on the hidden state computed at the last step (if applied to an image scanned pixel by pixel for example). See the main paper for the expression of the output in that case.

B.2.2 Modifying the chain rule

The underlying idea of our implementation of target propagation in a differentiable programming framework is to mix classical gradients and Jacobians of the inverse of the functions. Denote for a given output loss $\mathcal{L}$ computed on a given mini-batch with the current parameters θ, $\hat{\partial}\mathcal{L}/\hat{\partial} h_t$ the direction back-propagated by our implementation of target propagation until the step h_t. The direc-

[2]Compared to https://pytorch.org/docs/stable/generated/torch.nn.GRU.html, we used a single variable $b_m = b_{im} + b_{hm}$, same for b_z.

Algorithm 2 Proposed target propagation algorithm

Parameters: π a projection onto a susbet of $a(\mathbb{R}^p)$, stepsize γ_h, regularization r.
Inputs: Current parameters $\theta = (\theta_h, \theta_y)$ with $\theta_h = (W_{hh}, W_{xh}, b_h), \theta_y = (W_{hy}, b_y)$ of the RNN, sample of sequences-output pairs $(x_{1:\tau}, y)$.
Forward Pass:
Compute and store $V = (W_{hh}^\top W_{hh} + r\,\mathrm{I})^{-1} W_{hh}^\top$ giving access to $f_{\theta,t}^{-1}(v_t)$ defined in Eq. (7).
Initialize $h_0 = 0$.
for $t = 1, \dots, \tau$ **do**
    Compute and store $h_t = f_{\theta,t}(h_{t-1}), \quad \partial_{\theta_h} f_{\theta,t}(h_{t-1}), \quad \partial_{h_t} f_{\theta,t}^{-1}(h_t)$.
end for
Compute and store $\ell(y, c_\theta(h_\tau)), \quad \partial_{\partial h_\tau} \ell(y, c_\theta(h_\tau)), \quad \partial_{\theta_y} \ell(y, c_\theta(h_\tau))$.
Backward Pass:
Define $\lambda_\tau = -\gamma_h \partial_{h_\tau} \ell(y, c_\theta(h_\tau)), \quad d_{\theta_y} = -\partial_{\theta_y} \ell(y, c_\theta(h_\tau))$.
for $t = \tau, \dots, 1$ **do**
    Compute $\lambda_{t-1} = \partial_{h_t} f_{\theta,t}^{-1}(h_t)^\top \lambda_t$.
end for
Outputs: Descent directions for θ_h, θ_y:

$$d_{\theta_h} = \sum_{t=1}^{\tau} \partial_{\theta_h} f_{\theta,t}(h_{t-1}) \lambda_t, \qquad d_{\theta_y} = -\partial_{\theta_y} \ell(y, c_\theta(h_\tau)).$$

tions for the parameters of the network can be output as

$$\frac{\hat{\partial}\mathcal{L}}{\hat{\partial}\theta} = \sum_{t=1}^{\tau} \frac{\partial h_t}{\partial \theta} \frac{\hat{\partial}\mathcal{L}}{\hat{\partial} h_t}.$$

The main task is to define $\hat{\partial}\mathcal{L}/\hat{\partial} h_{t-1}$ given $\hat{\partial}\mathcal{L}/\hat{\partial} h_t$ and appropriate regularized inverses. For that, we start with the chain rule for $\partial h_t / \partial h_{t-1}$ and we will replace some of the gradients by Jacobians of regularized inverses at some places.

Classical chain rule. We have

$$\frac{\partial h_t}{\partial h_{t-1}} = \left(-\frac{\partial z_t}{\partial h_{t-1}} \right) \mathrm{diag}(h_{t-1}) + \mathrm{I}\,\mathrm{diag}(1 - z_t) + \frac{\partial z_t}{\partial h_{t-1}} \mathrm{diag}(n_t) + \frac{\partial n_t}{\partial h_{t-1}} \mathrm{diag}(z_t) \tag{12}$$

$$= \mathrm{diag}(1 - z_t) + \frac{\partial z_t}{\partial h_{t-1}} (\mathrm{diag}(n_t) - \mathrm{diag}(h_{t-1})) + \frac{\partial n_t}{\partial h_{t-1}} \mathrm{diag}(z_t). \tag{13}$$

Now for $\partial n_t / \partial h_{t-1}$, we further decompose the function $f_{n,t}(h_{t-1})$ as

$$f_{n,t}(h_{t-1}) = g_t(m_t \odot a_t),$$

with $g_t(u) = \tanh(W_{in} x_t + b_{in} + u)$ and $a_t = \ell(h_{t-1}) := W_{hn} h_{t-1} + b_{hn}$. We then have, denoting $u = m_t \odot a_t$

$$\frac{\partial n_t}{\partial h_{t-1}} = \left(\frac{\partial m_t}{\partial h_{t-1}} \mathrm{diag}(a_t) + \frac{\partial a_t}{\partial h_{t-1}} \mathrm{diag}(m_t) \right) \nabla g_t(u), \tag{14}$$

with $\nabla g_t(u) = \mathrm{diag}(\tanh'(W_{in} x_t + b_{in} + u))$.

Inverses. Now, the variables z_t, m_t and a_t are functions of h_t that incorporate a linear operation and that can be inverted. Namely, we can define the following regularized inverses

$$f_{m,t}^{-1}(v_t) = (W_{hm}^\top W_{hm} + r\,\mathrm{I})^{-1} W_{hm}^\top (\sigma^{-1}(v_t) - W_{ir} x_t - b_m)$$
$$f_{z,t}^{-1}(v_t) = (W_{hz}^\top W_{hz} + r\,\mathrm{I})^{-1} W_{hz}^\top (\sigma^{-1}(v_t) - W_{iz} x_t - b_z)$$
$$\ell^{-1}(v_t) = (W_{hn}^\top W_{hn} + r\,\mathrm{I})^{-1} W_{hn}^\top (v_t - b_{hn}).$$

We can then do the following substitutions in Eq.(12) and (14)

$$\frac{\partial m_t}{\partial h_{t-1}} \leftarrow \frac{\hat{\partial} m_t}{\hat{\partial} h_{t-1}} = \nabla f_{m,t}^{-1}(m_t)^\top$$
$$\frac{\partial z_t}{\partial h_{t-1}} \leftarrow \frac{\hat{\partial} z_t}{\hat{\partial} h_{t-1}} = \nabla f_{z,t}^{-1}(z_t)^\top$$
$$\frac{\partial a_t}{\partial h_{t-1}} \leftarrow \frac{\hat{\partial} a_t}{\hat{\partial} h_{t-1}} = \nabla \ell^{-1}(a_t)^\top$$

to define the quantity back-propagated by target propagation.

Note that by taking the gradient of the inverse we can ignore the biases and the inputs. Namely, we have for example

$$\nabla f_{m,t}^{-1}(m_t) = \operatorname{diag}((\sigma^{-1})'(m_t)) W_{hm} (W_{hm}^\top W_{hm} + r\,\mathrm{I})^{-1},$$

hence

$$\nabla f_{m,t}^{-1}(m_t)^\top = (W_{hm}^\top W_{hm} + r\,\mathrm{I})^{-1} W_{hm}^\top \operatorname{diag}((\sigma^{-1})'(m_t))$$

The expression for $\nabla f_{z,t}^{-1}(z_t)$ is identical. Since ℓ is affine, we have simply

$$\nabla \ell^{-1}(a_t)^\top = (W_{hn}^\top W_{hn} + r\,\mathrm{I})^{-1} W_{hn}^\top . s$$

Summary. Combined together, we get, denoting $d_t = \frac{\hat{\partial}\mathcal{L}}{\hat{\partial} h_t}$,

$$\begin{aligned}
\frac{\hat{\partial}\mathcal{L}}{\hat{\partial} h_{t-1}} &= (1 - z_t) \odot d_t + \nabla f_{z,t}^{-1}(z_t)^\top ((n_t - h_{t-1}) \odot d_t) \\
&\quad + \nabla f_{m,t}^{-1}(m_t)^\top (a_t \odot \tanh'(W_{in} x_t + b_{in} + u) \odot z_t \odot d_t) \\
&\quad + \nabla \ell^{-1}(a_t)^\top (m_t \odot \tanh'(W_{in} x_t + b_{in} + u) \odot z_t \odot d_t) \\
&= (1 - z_t) \odot d_t + (W_{hz}^\top W_{hz} + r\,\mathrm{I})^{-1} W_{hz}^\top \left((\sigma^{-1})'(z_t) \odot (n_t - h_{t-1}) \odot d_t\right) \\
&\quad + (W_{hm}^\top W_{hm} + r\,\mathrm{I})^{-1} W_{hm}^\top \left((\sigma^{-1})'(m_t) \odot a_t \odot \tanh'(W_{in} x_t + b_{in} + u) \odot z_t \odot d_t\right) \\
&\quad + (W_{hn}^\top W_{hn} + r\,\mathrm{I})^{-1} W_{hn}^\top (m_t \odot \tanh'(W_{in} x_t + b_{in} + u) \odot z_t \odot d_t).
\end{aligned}$$

This provides a rule to propagate targets through linearized regularized inverses.

C Target Propagation vs Gradient or Gauss-Newton descent

C.1 Gradient Back-Propagation vs Target Propagation

Lemma 3.1. *The difference between the oracle returned by gradient back-propagation $\partial_{\theta_h} \ell\left(y, g_\theta(x_{1:\tau})\right)$ and the oracle returned by target propagation can be bounded as*

$$\|\partial_{\theta_h} \ell\left(y, g_\theta(x_{1:\tau})\right) - d_\theta\| \leq c \sup_{t=1,\dots,\tau} \|\partial_h f_{\theta,t}(h_{t-1}) - \partial_h f_{\theta,t}^{-1}(h_t)^\top\|,$$

where $c = \sum_{t=1}^{\tau} \sum_{s=0}^{t-1} a^s b^{t-1-s}$ *with* $a = \sup_{t=1,\dots\tau} \|\partial_h f_{\theta,t}(h_{t-1})\|, b = \sup_{t=1,\dots\tau} \|\partial_h f_{\theta,t}^{-1}(h_t)^\top\|$.

For regularized inverses, we have, denoting $u_t = W_{xh} x_t + W_{hh} h_{t-1} + b_h$,

$$\|\partial_h f_{\theta,t}(h_{t-1}) - \partial_h f_{\theta,t}^{-1}(h_t)^\top\| \leq \|W_{hh}^\top\| \Big(\|\nabla a(u_t) - \nabla a(u_t)^{-1}\| + \|\,\mathrm{I} - (W_{hh}^\top W_{hh} + r\,\mathrm{I})^{-1}\| \|\nabla a(u_t)^{-1}\| \Big).$$

Proof. The first claim is a direct application of Lemma C.1 and the second claim follows from the formulation of the regularized inverse, using that $\nabla a^{-1}(h_t) = \nabla a(a^{-1}(h_t))^{-1} = \nabla a(u_t)^{-1}$. □

Lemma C.1. *Given $A_1, \dots, A_n, B_1, \dots, B_n \in \mathbb{R}^{n\times n}$, for any matrix norm $\|\cdot\|$, and any $1 \leq t \leq n$,*

$$\left\| \prod_{i=1}^{t} A_i - \prod_{i=1}^{t} B_i \right\| \leq \delta \sum_{i=0}^{t-1} a^i b^{t-1-i}$$

where $a = \sup_{i=1,\dots,n} \|A_i\|$, $b = \sup_{i=1,\dots n} \|B_i\|$ and $\delta = \sup_{i=1,\dots,n} \|A_i - B_i\|$.

Proof. Define for $t \geq 1$, $\delta_t = \| \prod_{i=1}^{t} A_i - \prod_{i=1}^{t} B_i\|$, we have

$$\delta_t \leq \left\| A_t \left(\prod_{i=1}^{t-1} A_i - \prod_{i=1}^{t-1} B_i \right) + (A_t - B_t) \prod_{i=1}^{t-1} B_i \right\| \leq a\delta_{t-1} + \delta b^{t-1} \leq \delta \sum_{i=0}^{t-1} a^i b^{t-1-i}.$$

□

A convergence to a stationary point for TP can be derived from classical results on an approximate gradient descent detailed below (the proof is akin to the results of Devolder et al. (2014)).

Corollary C.2 (Corollary of Lemma C.3)**.** *Denote ε_k a bound on the difference between the oracle returned by gradient back-propagation and by target-propagation both applied to the whole dataset. Provided that the objective is L-smooth and the stepsizes of TP are chosen such that $\gamma = \gamma_h \gamma_y < 1/2L$, after k iterations, we get*

$$\min_{i\in\{0,\dots,k-1\}} \|\nabla F(\theta_i)\|_2^2 \leq c_1 \frac{F(\theta_0) - \min_{\theta\in\mathbb{R}^d} F(\theta)}{\gamma k} + \frac{c_2}{k} \sum_{i=0}^{k-1} \varepsilon_i^2.$$

where $F(\theta_i) = \frac{1}{n} \sum_{i=1}^{n} \ell(\phi(x_i, \theta), y_i)$ with $\phi(x_i, \theta)$ the output of the RNN on a sample x_i and ℓ the chosen loss.

Lemma C.3. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a L-smooth function. Consider an ε-approximate gradient descent on f with step-size $\gamma \leq 1/(2L)$, i.e., $x_{k+1} = x_k - \gamma \widehat{\nabla} f(x_k)$, where $\|\widehat{\nabla} f(x_k) - \nabla f(x_k)\|_2 \leq \varepsilon_k$. After k iterations, this method satisfies, for c_1, c_2 two universal constants,*

$$\min_{i\in\{0,\dots,k-1\}} \|\nabla f(x_i)\|_2^2 \leq c_1 \frac{f(x_0) - \min_{x\in\mathbb{R}^d} f(x)}{\gamma k} + \frac{c_2}{k} \sum_{i=0}^{k-1} \varepsilon_i^2.$$

Proof. Denote $g_k = \widehat{\nabla} f(x_k) - \nabla f(x_k)$ for all $k \geq 0$. By L-smoothness of the objective, the iterations of the approximate gradient descent satisfy

$$\begin{aligned}
f(x_{k+1}) &\leq f(x_k) + \nabla f(x_k)^\top (x_{k+1} - x_k) + \frac{L}{2} \|x_{k+1} - x_k\|_2^2 \\
&= f(x_k) - \gamma \|\nabla f(x_k)\|_2^2 - \gamma \nabla f(x_k)^\top g_k + \frac{L\gamma^2}{2} \|\nabla f(x_k) + g_k\|_2^2 \\
&= f(x_k) - \gamma \left(1 - \frac{L\gamma}{2}\right) \|\nabla f(x_k)\|_2^2 + \frac{L\gamma^2}{2} \|g_k\|_2^2 + \gamma(L\gamma - 1) \nabla f(x_k)^\top g_k \\
&\leq f(x_k) - \gamma \left(1 - \frac{L\gamma}{2}\right) \|\nabla f(x_k)\|_2^2 + \frac{L\gamma^2}{2} \|g_k\|_2^2 + \gamma(1 - L\gamma) \|\nabla f(x_k)\|_2 \|g_k\|_2,
\end{aligned}$$

where in the last inequality we bounded the absolute value of the last term and used that $\gamma L \leq 1$. Now we use that for any $a, b \in \mathbb{R}$ and $\theta > 0$, $2ab \leq \theta a^2 + \theta^{-1} b^2$, which gives for $\theta > 0$, $a = \sqrt{\gamma(1 - L\gamma)/2} \|\nabla f(x_k)\|_2$ and $b = \sqrt{\gamma(1 - L\gamma)/2} \|g_k\|_2$,

$$f(x_{k+1}) \leq f(x_k) - \gamma \left(1 - \frac{L\gamma + \theta(1 - L\gamma)}{2}\right) \|\nabla f(x_k)\|_2^2 + \frac{L\gamma^2 + \theta^{-1}\gamma(1 - L\gamma)}{2} \|g_k\|_2^2.$$

Using $0 \leq L\gamma \leq 1/2$, $\theta = 1/4$ and $\|g_k\|_2^2 \leq \varepsilon_k^2$, we get $f(x_{k+1}) \leq f(x_k) - \frac{11}{16}\gamma \|\nabla f(x_k)\|_2^2 + 2\gamma\varepsilon_k^2$. Rearranging the terms, summing from $i = 0, \dots, k-1$, taking the minimum, dividing by k we get the result. □

C.2 TARGET PROPAGATION VS GAUSS-NEWTON UPDATES

We discuss the interpretation of Target Propagation (TP) as a Gauss-Newton (GN) method which was proposed by Bengio (2020); Meulemans et al. (2020). As already mentioned in Sec. 3, the main similarity between TP and GN is the fact that both TP and GN use the inverse or approximations of inverses of the gradients. In this section, we shall discuss this interpretation for feed-forward networks to follow the claims of Meulemans et al. (2020). Namely, we consider here a network defined by L weights $W_1, \dots, W_L$ and L activation functions $a_1, \dots, a_L$ which transform an input x_0 into an output x_L by computing (no biases were considered by Meulemans et al. (2020)),

$$x_t = f_t(x_{t-1}) = a_t(W_t x_{t-1}) \quad \text{for } t \in \{1, \dots, L\}$$

Denoting $\phi(x; \theta)$ the output of the network for an input $x = x_0$, with $\theta = (W_1, \dots, W_L)$ being the parameters of the network, the objective consists in minimizing the loss between the outputs of the network and the sample outputs, i.e., minimizing $\mathcal{L}(y, \phi(x; \theta))$ for pairs of inputs outputs samples (x, y).

GN step. Recall first the rationale of a GN step for such feed-forward networks with a squared-loss, which amount to solving

$$\min_{\theta \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^{n} \|\phi(x_i; \theta) - y_i\|_2^2,$$

with $y_i \in \mathbb{R}^K$ (for classification in K classes) and $\phi(x_i, \theta) \in \mathbb{R}^{d_L}$. A GN step amounts to linearize the non-linear function ϕ around a current set of parameters $\theta^{(k)}$ and solve the corresponding least-square problems to define the next set of parameters, i.e,

$$\begin{aligned}
\theta^{(k+1)} &= \operatorname*{argmin}_{\theta} \frac{1}{n} \sum_{i=1}^{n} \|\phi(x_i; \theta^{(k)}) + \partial_\theta \phi(x_i; \theta^{(k)})^\top (\theta - \theta^{(k)}) - y_i\|_2^2 \\
&= \theta^{(k)} - \left(\sum_{i=1}^{n} \partial_\theta \phi(x_i; \theta^{(k)}) \partial_\theta \phi(x_i; \theta^{(k)})^\top \right)^{-1} \left(\sum_{i=1}^{n} \partial_\theta \phi(x_i; \theta^{(k)}) \left(\phi(x_i, \theta^{(k)}) - y_i \right) \right).
\end{aligned}$$

Now,

1. Consider the iteration on a mini-batch of size 1, s.t.
$$\theta^{(k+1)} = \theta^{(k)} - \left(\partial_\theta \phi(x_i; \theta^{(k)}) \partial_\theta \phi(x_i; \theta^{(k)})^\top \right)^{-1} \left(\partial_\theta \phi(x_i; \theta^{(k)}) \left(\phi(x_i, \theta^{(k)}) - y_i \right) \right).$$
2. Consider that the gradients of the networks are invertible, s.t.
$$\theta^{(k+1)} = \theta^{(k)} - \left(\partial_\theta \phi(x_i; \theta^{(k)}) \right)^{-\top} \left(\phi(x_i, \theta^{(k)}) - y_i \right).$$
3. Consider updating only one set of parameters $\theta_l = W_l$, s.t.,
$$\theta_l^{(k+1)} = \theta_l^{(k)} - \left(\partial_{\theta_l} \phi(x_i; \theta^{(k)}) \right)^{-\top} \left(\phi(x_i, \theta^{(k)}) - y_i \right).$$
with
$$\partial_{\theta_l} \phi(x_i; \theta^{(k)}) = \partial_{\theta_l} f_l(x_{l-1}) \partial_x f_{l+1}(x_l) \dots \partial_x f_L(x_{L-1})$$
so that, provided that all matrices inside the matrix multiplication are invertible, we get
$$\partial_{\theta_l} \phi(x_i; \theta^{(k)})^{-T} = \partial_{\theta_l} f_l(x_{l-1})^{-T} \partial_x f_{l+1}(x_l)^{-T} \dots \partial_x f_L(x_{L-1})^{-T}$$
4. Finally, ignore the last inversion and replace it by a gradient step on the parameters θ_l, then we get an iteration similar to TP, with
$$\theta_l^{(k+1)} = \theta_l^{(k)} - \partial_{\theta_l} f_l(x_{l-1}) \partial_x f_{l+1}(x_l)^{-T} \dots \partial_x f_L(x_{L-1})^{-T} \partial_{x_L} \mathcal{L}(y, x_L)$$
for $\mathcal{L}$ a squared loss. Namely, we keep the inversion of the gradients of the intermediate functions.

Our objective here is to question whether viewing TP as a GN step with the approximations explained above is meaningful or not.

Does the original TP formulation approximate GN? Meulemans et al. (2020) start by considering the original TP formulation, i.e., targets computed as $v_t = \psi_t(v_{t+1})$ for ψ_t an approximate inverse of f_t and with $v_L = x_L - \eta \partial_{x_L} \ell(y, x_L)$. Meulemans et al. (2020, Lemma 1) show then that, provided that we use the exact inverse, $\psi_t = f_t^{-1}$,

$$\Delta x_t = v_t - x_t = -\eta \prod_{s=t}^{L-1} \partial_{x_s} f_{s+1}(x_s)^{-\top} \partial_{x_L} \ell(y, x_L) + O(\eta^2).$$

(Meulemans et al., 2020, Theorem 2) conclude that (i) for mini-batches of size 1, (ii) for a squared loss, (iii) for invertible f_t, as $\eta \to 0$ TP uses a Gauss-Newton optimization with block diagonal approximation to compute the targets in the sense that as $\eta \to 0$,

$$\Delta x_t \approx -\eta \partial_{x_t} (f_{t+1} \circ \ldots \circ f_L)^{-\top}(x_t).$$

As the stepsize of any optimization algorithm tends to 0, they all are the same, since the update would be 0 in all cases. An optimization algorithm aims not to have infinitesimal stepsizes. To make the claim of Meulemans et al. (2020) more precise, the constants hidden in $O(\eta^2)$ need to be detailed in order to understand in which regimes of the stepsize the approximation is meaningful. Assuming the inverses ψ_t to be ℓ Lipschitz continuous and L-smooth (i.e. with L-Lipschitz continuous gradients), a quick look at the proof of Lemma 1 of Meulemans et al. (2020) shows that

$$\begin{aligned}
v_t - x_t &= -\eta \prod_{s=t}^{L-1} \partial_{x_s} f_{s+1}(x_s)^{-\top} \partial_{x_L} \ell(y, x_L) + \xi_t \\
\|\xi_t\|_2 &\leq a_t \\
a_s &\leq L a_{s+1}^2 + \ell a_{s+1} + L\ell^2 \eta^2 \|\partial_{x_L} \ell(y, x_L)\|_2^2 \quad \text{for } s \in \{t, \ldots, L-1\} \\
a_L &\leq \frac{L}{2} \eta^2 \|\partial_{x_L} \ell(y, x_L)\|_2^2.
\end{aligned}$$

The above bound shows that $\|\xi_t\|_2$ grows w.r.t. the stepsize η as a polynomial with leading term $\eta^{2^{L-t}}$. So unless the stepsize is extremely small, it seems unclear whether the original TP formulation approximates GN in this case. Though the above bound may be pessimistic, it captures correctly the dependency of the error w.r.t. η for η that does not simply converge to 0.

Finally, if the similarity of TP with GN could explain its efficiency, then by the reasoning of Meulemans et al. (2020), the original TP formulation should be efficient. Yet, the original TP formulation has never been shown to produce satisfying results.

Does TP with the difference target propagation approximate GN? Meulemans et al. (2020) make a similar claim for TP with the Difference Target Propagation formula, i.e., $v_t = x_t + \psi_t(v_{t+1}) - \psi_t(x_{t+1})$. Namely, Meulemans et al. (2020, Lemma 3) show that

$$\Delta x_t = v_t - x_t = -\eta \prod_{s=t}^{L-1} \partial_{x_s} \psi_s(x_s)^{\top} \partial_{x_L} \ell(y, x_L) + O(\eta^2).$$

Once again, for the claim to be meaningful beyond infinitesimal stepsizes, the terms in $O(\eta^2)$ need to be detailed. A quick look at the proof of Meulemans et al. (2020, Lemma 3) shows that under appropriate smoothness assumptions the error can be bounded as a polynomial in η with a leading term $\eta^{2^{L-t}}$. So again, unless we consider infinitesimal stepsizes, it is unclear whether this approximation is useful.

Linearized target propagation and GN. If we use a linearized version of the difference target propagation formula as presented in (2), namely $v_t - x_t = \partial_{x_{t+1}} \psi_t(x_{t+1})^{\top}(v_{t+1} - x_{t+1})$, then we have the *equality*

$$\Delta x_t = v_t - x_t = -\eta \prod_{s=t}^{L-1} \partial_{x_s} \psi_s(x_s)^{\top} \partial_{x_L} \ell(y, x_L)$$

and the idea that TP could be seen as an approximate GN method may be pursued in a meaningful way. However the error of approximation of the inverse of the gradients must be taken into account in order to understand the validity of the approach.

Propagating the approximation error of the gradient inverses. We compute the approximation error incurred by composing gradients of the inverse instead of inverses of gradients. Formally, the approximation error for one layer can be estimated under the assumption that

$$\psi_t(f_t(x_{t-1})) = x_{t-1} + e(x_{t-1}), \tag{15}$$

with e an ε-Lipschitz continuous function and the assumption that the minimal singular value σ of $\partial_{x_t} f_t(x_{t-1})$ is positive.

The function e a priori depends on θ; we ignore this dependency and simply consider e to be ε-Lipschitz continuous for all θ. For a multivariate function such as e, we define its Lipschitz continuity constant ε as $\varepsilon = \sup_x \sup_{\|\lambda\|_2 \leq 1} \|\partial_x e(x)^\top \lambda\|_2 = \sup_x \|\partial_x e(x)\|$, where $\|\cdot\|$ denotes the spectral norm. By differentiating both sides of Eq. (15), we get

$$\partial_x f_t(x_{t-1}) \partial_x \psi_t(x_t) = \mathrm{I} + \partial_x e(x_{t-1}).$$

By assuming the minimal singular value σ of $\partial_x f_t(x_{t-1})$ to be positive, we get that $\partial_x f_t(x_{t-1})$ is invertible and so

$$\partial_x \psi_t(x_t) = \partial_x f_t(x_{t-1})^{-1}(\mathrm{I} + \partial_x e(x_{t-1})).$$

Hence

$$\| \left(\partial_x f_t(x_{t-1})\right)^{-1} - \partial_x \psi_t(x_t) \| \leq \frac{\varepsilon}{\sigma}, \tag{16}$$

and $\partial_x \psi_t(x_t)$ is $\sigma^{-1}(1+\varepsilon)$ Lipschitz-continuous.

Now for multiple compositions, using Lemma C.1, we get

$$\| \left(\partial_h f_1(x_0)\right)^{-1} \dots \left(\partial_x f_L(x_{L-1})\right)^{-1} - \partial_x \psi_1(x_1) \dots \partial_x \psi_L(x_L) \| \leq \frac{(1+\varepsilon)^L}{\sigma^L}.$$

Therefore the accumulation error diverges with L as soon as $\varepsilon \geq \sigma - 1$.

Testing the hypothesis that TP could be interpreted as using GN updates directions. Here we come back to the setting of RNNs presented in the paper. In this case the length of the compositions of layers is τ and according to the previous discussion, the error of approximation of the product of the inverse of the gradients by the product of the gradients of the approximate inverses could easily diverge as τ grows (long sequences). Nevertheless, by using analytical formulas for the inverses, we can ensure that the approximation error is zero, which would correspond then to the ideal setting where TP uses GN update directions for the hidden states.

Formally, in the context of RNNs, a Gauss-Newton update direction for the hidden states is given as (ignoring the inverse of the output function)

$$-\gamma_h \prod_{s=t+1}^{\tau-1} \left(\partial_h f_{t+1,\theta}(h_t)\right)^{-\top} \partial_h \ell(y, c_\theta(h_\tau)),$$

If no regularization is used in the definition of the regularized inverse, i.e., if we use

$$f_{\theta,t}^{-1}(h_t) = (W_{hh}^\top W_{hh})^{-1} W_{hh}^\top (a^{-1}(h_t) - W_{xh} x_t - b_h)$$

which requires the inverse of W_{hh} to be well defined, we would get

$$\partial f_{\theta,t}^{-1}(h_t) = \partial_h f_{t+1,\theta}(h_t)^{-1}.$$

The updates of TP using the formula (2) would then be exactly the ones of a GN update direction, i.e.,

$$v_t - h_t = -\gamma_h \prod_{s=t+1}^{\tau-1} \left(\partial_h f_{t+1,\theta}(h_t)\right)^{-\top} \partial_h \ell(y, c_\theta(h_\tau)).$$

So by considering our implementation without regularization, we can test whether the interpretation of TP as an approximate GN method is meaningful in terms of optimization convergence. As shown in Fig. 6b, it appears that regularizing the inverses is necessary to obtain convergence, hence the interpretation of TP as GN may not be sufficient to explain why TP can converge.

	BP	TP		
	γ	γ_h	γ_θ	κ
Temporal Order Problem length 60	10^{-5}	10^{-2}	10^{-1}	10
Temporal Order Problem length 120	10^{-5}	10^{-2}	10^{-2}	1
Adding Problem	10^{-3}	10^{-1}	10^{-1}	1
MNIST pixel by pixel	10^{-6}	10^{-4}	10^{-1}	1
MNIST pixel by pixel permuted	10^{-4}	10^{-4}	10^{-1}	1
CIFAR	10^{-3}	10^{-2}	10^{-2}	10
FashionMNIST with GRU	10^{-2}	10^{-1}	10^{-2}	1

Table 1: Hyper-parameters chosen for Fig. 3 and 4.

D Experimental Details

D.1 Initialization and hyper-parameters

Initialization and data generation. In all experiments, the weights of the RNN are initialized as random orthogonal matrices, and the biases are initialized as 0 as presented by Le et al. (2015) and Manchev & Spratling (2020). For all experiments, the data was not normalized, as done by Manchev & Spratling (2020). We kept a setting as similar as possible as the one of Manchev & Spratling (2020) to be able to compare target propagation with regularized or parameterized inverses.

Hyper-parameters. In the synthetic tasks, for BP we used a momentum of 0.9 with Nesterov accelerated gradient scheme as done by Manchev & Spratling (2020). Otherwise, we did not use any momentum for the experiment on MNIST pixel by pixel presented in the main paper. The learning rates of BP and the parameters of TP were found by a grid-search on a $\log_{10}$ basis and are presented in Table 1. We did not add a regularization term in the training of the RNNs.

For the Fig. 6b, we used batch sizes of size 512 and performed a grid search for the stepsizes of BP and for the stepsizes γ_h of TP while keeping the same regularization r and stepsize γ_θ to the parameters found for the length 784.

Software. We used Python 3.8 and PyTorch 1.6. The RNN was coded using the cuDNN implementation available in PyTorch that is highly optimized for computing forward passes on the network or gradient back-propagation.

Hardware. All experiments were performed on GPUs using Nvidia GeForce GTX 1080 Ti (12G memory). Each experiment only used one gpu at a time (clock speed 1.5 Ghz).

Time evaluation. On our GPU, we observed that for the MNIST pixel by pixel experiment, 200 iterations (each iteration considering 16 samples) were taking approximately 60s for BP and 800s for TP. Note that with larger batch-sizes the cost of the regularized inversion would be amortized by the fact that more samples are treated simultaneously. We kept the setting of Manchev and Spartling (Manchev & Spratling, 2020) for ease of comparison.

D.2 Additional experiments

The overhead of TP can be worth its performance. To account for the additional cost of inversion for each mini-batch, we consider the convergence of the algorithms in time rather than in iterations. We found that, on average, 1 iteration of BP takes approximately 13 times less time than one iteration of TP in our implementation (note that BP benefits from highly optimized implementations for GPU machines, and TP could potentially also benefit from the same optimized implementations). Therefore we ran BP for 13 times more iterations than TP and multiplied the number of iterations by the approximate time needed for each iteration for all algorithms. In the

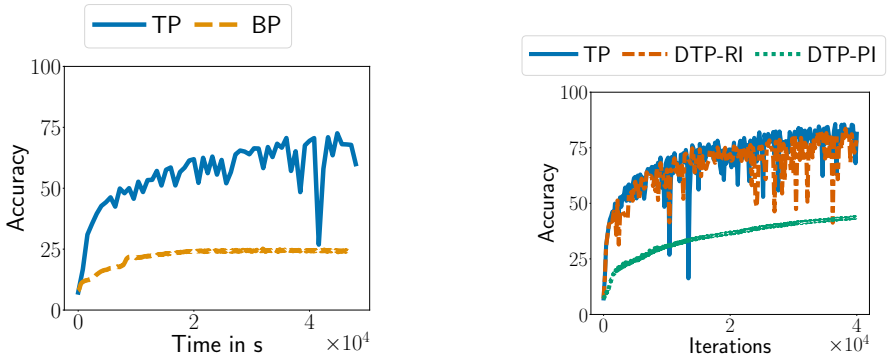


Fig. 7: Left: MNIST in time. Right: Comparison of different implementations of TP.

right panel of Fig. 7, we observe that in time too, TP performs better than BP, which stays stuck at an accuracy of approximately 22

Regularized inverses outperform parameterized inverses. We evaluate the impact of using regularized inverses as opposed to parameterized inverse and linearized propagation as opposed to finite-difference-based propagation. The variant of target propagation with parameterized inverse and finite-difference propagation corresponds to the approach of Lee et al. (2015) recently implemented by Manchev & Spratling (2020) and referred to in the figure above as **DTP-PI**. The variant of target propagation with regularized inverse and finite-difference propagation is referred to in the figure above as **DTP-RI**. Recall that our approach involves regularized inverses and linearized propagation, referred as **TP**. In Fig. 7, we observe that both TP and DTP-RI outperform DTP-PI, demonstrating the benefits of using regularized inverses. On the other hand, both TP and DTP-RI perform on par overall, with the former being slightly better for the given parameters.

Target propagation is robust to the choice of the target stepsize γ_h. In Fig. 6a, we observed how the convergence could be affected by the choice of the regularization and the step-size γ_θ. In the left panel of Fig. 8, we observe that varying γ_h does not lead to significant changes in the convergence behavior.

Target propagation does not benefit from momentum techniques. Numerous methods have been proposed to enhance the performance of a classical stochastic gradient descent by using, e.g., a momentum term akin to Nesterov's accelerated gradient formula (Sutskever et al., 2013). Since TP also produces a priori a descent direction, we can wonder whether an additional momentum provides faster convergence. In the right panel of Fig. 8, we observe that adding a momentum on TP is possible but does not seem to provide significantly faster convergence while being less stable. On the other hand, on this same figure, the momentum seems to help the gradient descent. Our preliminary experiments using Adam with TP did not conclude; namely, we were not able to obtain a convergence similar to the one illustrated in Fig. (4) that simply used Algo. 2. We leave for future work the implementation of appropriate adaptive stepsizes strategies for TP.

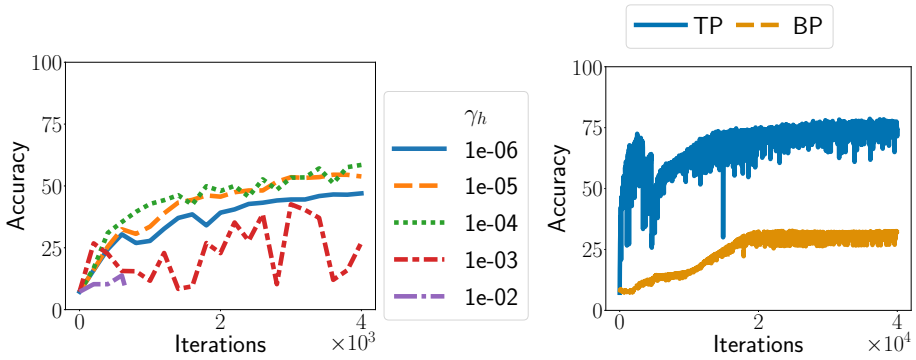


Fig. 8: Left: MNIST for varying γ_h and fixed $\gamma_\theta = 1$, $r = 1$. Right: MNIST with momentum.