Journal of Machine Learning Research 21 (2020) 1-33 Submitted 3/18; Revised 7/19; Published 1/20

Target Propagation in Recurrent Neural Networks

Nikolay Manchev NIKOLAY.MANCHEV@KCL.AC.UK
Department of Informatics
King's College London
London, WC2B 4BG, UK

Michael Spratling MICHAEL.SPRATLING@KCL.AC.UK
Department of Informatics
King's College London
London, WC2B 4BG, UK

Editor: Yoshua Bengio

Abstract

Recurrent Neural Networks have been widely used to process sequence data, but have long been criticized for their biological implausibility and training difficulties related to vanishing and exploding gradients. This paper presents a novel algorithm for training recurrent networks, *target propagation through time* (TPTT), that outperforms standard *backpropagation through time* (BPTT) on four out of the five problems used for testing. The proposed algorithm is initially tested and compared to BPTT on four synthetic time lag tasks, and its performance is also measured using the sequential MNIST data set. In addition, as TPTT uses target propagation, it allows for discrete nonlinearities and could potentially mitigate the credit assignment problem in more complex recurrent architectures.

Keywords: recurrent neural networks, target propagation, biological plausibility

1. Introduction

A Recurrent Neural Network (RNN) represents a type of artificial neural network adapted to sequence data, which can model temporal events by retaining a state that spans a context window of specified length. It has been demonstrated that RNNs can successfully address a wide range of problems in the areas of speech recognition (Graves 2012, Heigold et al. 2015), image captioning (Xu et al. 2015, Karpathy and Fei-Fei 2017), text classification (Liu et al. 2016, Yogatama et al. 2017), word prediction (Mikolov et al. 2013), sentiment analysis (Timmaraju and Khanna 2015), mapping sentences and images (Socher et al. 2014), language modelling (De Mulder et al. 2015) and others.

A diagram of a Simple Recurrent Network (SRN) (Jordan 1989, Elman 1990) is given in Figure 1a. The network receives as input a sequence of data points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ and produces a sequence of output values $\{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N\}$. The input and output values can be binary or real-valued vectors, depending on the problem the network is designed to solve. The SRN can compute an output value at each time step, or it can produce a single value at its final step: in the latter case the output is a single data point.

The state of the network at time t is given by

©2020 Nikolay Manchev and Michael Spratling.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v21/18-141.html.

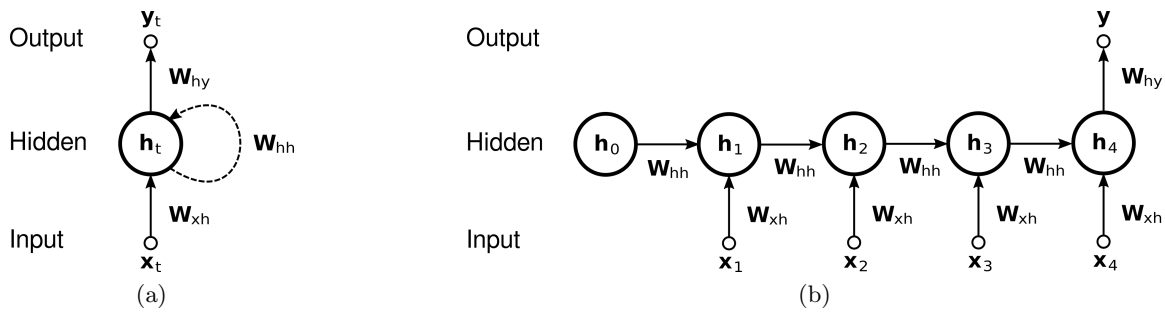

Figure 1: **(a)** A Simple Recurrent Network with one input unit, one output unit, and one recurrent hidden unit **(b)** An SRN "unrolled" for four time steps ($t \in [1, 4]$). The network computes a single output $\mathbf{y}$, and the initial state $\mathbf{h}_0$ is initialised with zeros

$$\mathbf{h}_t = \sigma(\mathbf{W}_{xh} \cdot \mathbf{x}_t + \mathbf{W}_{hh} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h) \tag{1}$$

where $\mathbf{W}_{xh}$ is the matrix of synaptic weights between the input and the hidden layer, $\mathbf{W}_{hh}$ is the matrix of weights between the hidden layer and itself at adjacent time steps, and $\mathbf{b}_h$ is a bias term. Common choices for the non-linear activation function $\sigma(\cdot)$ include the *sigmoid* and *tanh*.

When the network computes a categorical distribution its output is given by

$$\mathbf{y}_t = \text{softmax}(\mathbf{W}_{hy} \cdot \mathbf{h}_t + \mathbf{b}_y) \tag{2}$$

where $\mathbf{W}_{hy}$ is a matrix of weights between the hidden layer and the output, and $\mathbf{b}_y$ is the output bias. If a real-valued output is needed, the normalized exponential function is omitted.

Recurrent Neural Networks are typically trained using backpropagation (Rumelhart et al. 1986), more specifically, a variant of backpropagation known as *backpropagation through time* or BPTT (Werbos 1990).

Despite their popularity, recurrent networks have long been criticised because of the training difficulties associated with learning dependencies in sequences that span long intervals, and because of the lack of biological plausibility in their learning mechanism. The following two sections discuss these issues in detail.

1.1 Training Difficulties

Similar to backpropagation, BPTT computes the gradient of a cost function with respect to the synaptic weights, and uses the gradient to apply corrections that will minimize the cost.

To handle the recurrent connection from $\mathbf{h}_t$ to $\mathbf{h}_{t+1}$, BPTT "unrolls" the network for a fixed number of time steps ($t \in [1, t_{\max}]$), obtaining a simple feedforward network where each time step is represented by one layer (Figure 1b). The initial state of the resulting

network $\mathbf{h}_0$ can be initialised with zeros, and the network can be optimised using standard backpropagation.

Bengio (1991) demonstrates that recurrent networks can outperform static networks for small t_{max} (i.e. when the distance between the output and the input required for a correct prediction is relatively short). It appears, however, that problems that exhibit long-term dependencies present a challenge as the training process settles the network in suboptimal solutions. This issue has been thoroughly investigated by Bengio et al. (1994) and is attributed to the vanishing or exploding gradients of the objective function.

If $\mathcal{E}$ is the error that the network tries to minimize in a supervised setting, then the derivative of the error with respect to the recurrent weights will be a sum of the derivatives at each step in the context window.

$$\frac{\partial \mathcal{E}}{\partial \boldsymbol{W}_{hh}} = \sum_{t=1}^{t_{\max}} \frac{\partial \mathcal{E}}{\partial \boldsymbol{y}} \frac{\partial \boldsymbol{y}}{\partial \mathbf{h}_{t_{\max}}} \frac{\partial \mathbf{h}_{t_{\max}}}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \boldsymbol{W}_{hh}}$$

The $\frac{\partial \mathbf{h}_{t_{\max}}}{\partial \mathbf{h}_t}$ term is a product of Jacobian matrices such that

$$\frac{\partial \mathbf{h}_{t_{\max}}}{\partial \mathbf{h}_t} = \prod_{i=t+1}^{t_{\max}} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \prod_{i=t+1}^{t_{\max}} \boldsymbol{W}_{hh}^T \mathrm{diag}[\sigma\prime(\mathbf{h}_{i-1})]$$

As t_{max} grows, this product of Jacobian matrices can increase to infinity (*exploding gradients*) or exponentially decrease to zero (*vanishing gradients*). Pascanu et al. (2012) show that it is necessary for the spectral radius (operator 2-norm) of $\mathbf{W}_{hh}$ to be larger than 1 for the gradients to explode, and it is sufficient for it to be smaller than 1 for the gradients to vanish.

As explained in Bengio et al. (1994), exploding and vanishing gradients prevent the network from learning dependencies between temporally distant events. Various options have been proposed for solving the vanishing and exploding gradients issues. Popular choices include changing the network architecture (Hochreiter and Schmidhuber 1997), Hessian-free optimisation (Sutskever et al. 2011), gradient clipping and regularisation (Pascanu et al. 2012), and others.

1.2 Biological Implausibility

Most of the objections against biological plausibility in recurrent neural networks stem from the backpropagation-based training. Backpropagation has never been intended to be biologically plausible, and it is deemed biologically unrealistic in almost every aspect (Crick 1989)

Backpropagation requires the passage of error signals back through the synapse and along the axon to each neuron in the upstream layers. Moreover, each neuron has to emit two distinct signals, an output and an error signal, although such error signals have not been observed in neurophysiology (Roelfsema and Van Ooyen 2005).

Another objection is related to the way backpropagation solves the credit assignment problem (Minsky 1961, Hinton et al. 1984). The essence of the problem is that the network should be capable of assigning *credit* or *blame* to its decision elements (i.e. neurons) based on their contribution to the error $\mathcal{E}$. To solve this problem backpropagation uses *weight*

transport, it requires neurons to send each other information using the downstream synaptic weights. This computation would require knowledge of all the synaptic weights in the forward path, implying a symmetric backward connectivity pattern, which is thought to be impossible in the brain (Lillicrap et al. 2016).

It is reasonable to argue that credit assignment is a non-issue for the Simple Recurrent Network model, as the "downstream" and "upstream" neuron classification has significance only in the unrolled version of the network. In the original model (Figure 1a) there is only one neuron and the synapses at time $t+1$, $t+2$, etc. are in fact the same set of synapses as those at time t. However, the Simple RNN model is the most simplistic recurrent network model. Many other RNN architectures exists where credit assignment becomes relevant (i.e. higher order RNNs (Soltani and Jiang 2016), recurrent multilayer perceptrons with multiple hidden layers (Puskorius et al. 1996), recurrent convolutional neural networks with between-layers recurrent connections (Pinheiro and Collobert 2014)).

Another significant arguments against the biological plausibility of BPTT is the need of complex temporal indexing. The computation of the gradients at each time step t requires the network to store and match together time-indexed error signals and activity patterns. It is difficult to imagine a biological mechanism that could satisfy this requirement within a single, recurrently connected neuron.

Other objections include the fact that backpropagation uses continuous values while neurons in the brain communicate using spikes. The algorithm also uses precisely clocked changes between a forward and backward regime, which is also deemed unlikely in biological neural systems (Bengio et al. 2015).

2. Target Propagation Through Time

The model we propose is a form of *target propagation* (LeCun 1986, Bengio 2014), where the network computes targets instead of gradients for each time step and optimises the synaptic weights locally.

The main idea of target propagation is to set local targets that are close to the activation value of each neuron in such a way, that if the targets were produced by the neurons during their forward phase, the global error of the network would decrease. An algorithm that implements target propagation for feedforward networks is given by Lee et al. (2015). This section introduces a similar approach applied to an "unrolled" recurrent network, with certain modifications that handle the network state inputs. The suggested name for this algorithm is *target propagation through time* (TPTT) (Figure 2).

The first question that has to be addressed is how to set local targets $\hat{\mathbf{h}}_t$ for the neurons in the hidden layer at time t. Setting the target for the final time step $\hat{\mathbf{h}}_{t_{max}}$ is trivial, as it can be based on the gradient of the error w.r.t. the activations of $\mathbf{h}_{t_{max}}$

$$\hat{\mathbf{h}}_{t_{max}} = \mathbf{h}_{t_{max}} - \alpha_i * \frac{\partial \mathcal{E}}{\partial \mathbf{h}_{t_{max}}} \tag{3}$$

where α_i is an initial learning rate.

Setting the targets for the earlier time steps relies on a function that acts as an inverse of the forward output function. If $F(\cdot)$ is a function that computes the hidden state of the network at time t, then according to Equation 1, $F(\cdot)$ is a function of $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$:

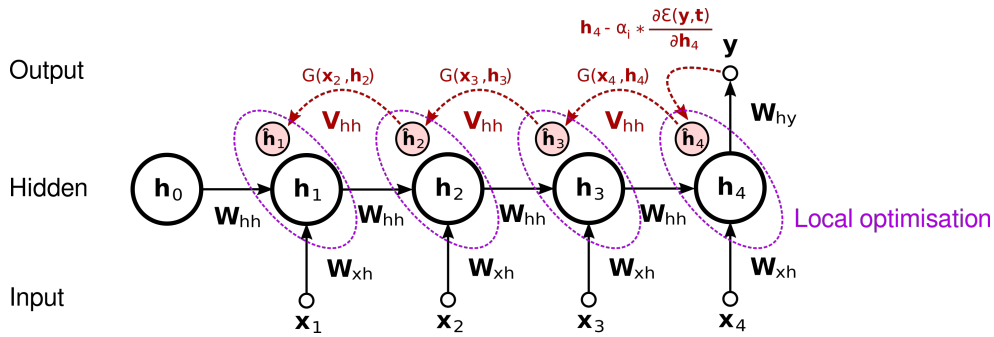

Figure 2: Target propagation through time: Setting the first and the upstream targets and performing local optimisation to bring $\mathbf{h}_t$ closer to $\hat{\mathbf{h}}_t$

$$\mathbf{h}_t = F(\mathbf{x}_t, \mathbf{h}_{t-1}) = \sigma(\mathbf{W}_{xh} \cdot \mathbf{x}_t + \mathbf{W}_{hh} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h)$$

The inverse of $F(\mathbf{x}_t, \mathbf{h}_{t-1})$ should be a function $G(\cdot)$ that takes $\mathbf{x}_t$ and $\mathbf{h}_t$ as inputs and produces an approximation of $\mathbf{h}_{t-1}$:

$$\begin{aligned} \mathbf{h}_{t-1} &\approx G(\mathbf{x}_t, \mathbf{h}_t) \\ \mathbf{h}_{t-1} &\approx G(\mathbf{x}_t, F(\mathbf{x}_t, \mathbf{h}_{t-1})) \end{aligned} \tag{4}$$

If function $G(\cdot)$ can be approximated, then it can serve to set the local targets using:

$$\hat{\mathbf{h}}_t = G(\mathbf{x}_{t+1}, \hat{\mathbf{h}}_{t+1}) \tag{5}$$

The presented model adopts a linearly corrected formula (*difference target propagation*) suggested by Lee et al. (2015), which stabilizes the optimisation problem when $G(\cdot)$ is not a perfect inverse of $F(\cdot)$:

$$\hat{\mathbf{h}}_t = \mathbf{h}_t - G(\mathbf{x}_{t+1}, \mathbf{h}_{t+1}) + G(\mathbf{x}_{t+1}, \hat{\mathbf{h}}_{t+1}) \tag{6}$$

If $G(\cdot)$ is an inverse of $F(\cdot)$ then $G(\mathbf{x}_{t+1}, \mathbf{h}_{t+1}) = \mathbf{h}_t$ and $\mathbf{h}_t - G(\mathbf{x}_{t+1}, \mathbf{h}_{t+1}) = \mathbf{h}_t - \mathbf{h}_t = 0$, thus (6) reduces to (5). The corrected formula stabilizes the optimisation as it guarantees that as $\mathbf{h}_{t+1}$ approaches $\hat{\mathbf{h}}_{t+1}$, $\mathbf{h}_t$ also approaches $\hat{\mathbf{h}}_t$. For a detailed explanation the reader is referred to Section 2.3 in Lee et al. (2015).

The proposed configuration for $G(\cdot)$ is

$$G(\mathbf{x}_{t+1}, \mathbf{h}_{t+1}) = \sigma(\mathbf{W}_{xh} \cdot \mathbf{x}_{t+1} + \mathbf{V}_{hh} \cdot \mathbf{h}_{t+1} + \mathbf{c}_h) \tag{7}$$

where $\mathbf{V}_{hh}$ is a matrix of weights and $\mathbf{c}_h$ is a bias term, which the network must learn so that (4) holds. Plugging (7) into (6) produces the final equation for the upstream targets:

$$\hat{\mathbf{h}}_t = \mathbf{h}_t - \sigma(\mathbf{W}_{xh} \cdot \mathbf{x}_{t+1} + \mathbf{V}_{hh} \cdot \mathbf{h}_{t+1} + \mathbf{c}_h) + \sigma(\mathbf{W}_{xh} \cdot \mathbf{x}_{t+1} + \mathbf{V}_{hh} \cdot \hat{\mathbf{h}}_{t+1} + \mathbf{c}_h) \tag{8}$$

After the local targets have been set using (3) and (8), the network operates by switching between two distinct phases. First, it updates the parameters of $G(\cdot)$ using gradient descent with the following rules:

$$\begin{aligned} \mathbf{V}_{hh} &:= \mathbf{V}_{hh} - \alpha_g \sum_{t=1}^{t_{\max-1}} \frac{\partial \mathrm{MSE}(G(\mathbf{x}_t, F(\mathbf{x}_t, \mathbf{h}_{t-1})), \mathbf{h}_{t-1})}{\partial \mathbf{V}_{hh}} \\ \mathbf{c}_h &:= \mathbf{c}_h - \alpha_g \sum_{t=1}^{t_{\max-1}} \frac{\partial \mathrm{MSE}(G(\mathbf{x}_t, F(\mathbf{x}_t, \mathbf{h}_{t-1})), \mathbf{h}_{t-1})}{\partial \mathbf{c}_h} \end{aligned} \tag{9}$$

where MSE is the mean squared error and α_g is a learning rate. The intuition here is that the network is adapting $\mathbf{V}_{hh}$ and $\mathbf{c}_h$ so that the approximated state for time $t-1$ gets closer to the actual $\mathbf{h}_{t-1}$ values.

In the second phase the network updates the feedforward parameters. The parameters that govern the hidden states are updated using:

$$\begin{aligned} \mathbf{W}_{hh} &:= \mathbf{W}_{hh} - \alpha_f \sum_{t=1}^{t_{\max}} \frac{\partial \mathrm{MSE}(F(\mathbf{x}_t, \mathbf{h}_{t-1}), \hat{\mathbf{h}}_t)}{\partial \mathbf{W}_{hh}} \\ \mathbf{b}_h &:= \mathbf{b}_h - \alpha_f \sum_{t=1}^{t_{\max}} \frac{\partial \mathrm{MSE}(F(\mathbf{x}_t, \mathbf{h}_{t-1}), \hat{\mathbf{h}}_t)}{\partial \mathbf{b}_h} \end{aligned} \tag{10}$$

where α_f is the learning rate for updating the feedforward parameters. The intuition behind the above updates is that outputs closer to $\hat{\mathbf{h}}_t$ reduce the global error. If the network adapts to bring $\mathbf{h}_t$ closer to $\hat{\mathbf{h}}_t$ this should respectively lead to a lower error at the final step.

As the feedforward parameters of the output $\mathbf{W}_{hy}$ and $\mathbf{b}_y$ are not susceptible to vanishing/exploding gradients they can be updated using the gradients w.r.t. $\mathcal{E}$ directly.

$$\begin{aligned} \mathbf{W}_{hy} &:= \mathbf{W}_{hy} - \alpha_f \frac{\partial \mathcal{E}}{\partial \mathbf{W}_{hy}} \\ \mathbf{b}_y &:= \mathbf{b}_y - \alpha_f \frac{\partial \mathcal{E}}{\partial \mathbf{b}_y} \end{aligned} \tag{11}$$

There are two options for handling the feedforward parameters of the input. The updates for $\mathbf{W}_{xh}$ can be obtained equivalently to the updates of $\mathbf{W}_{hy}$ and $\mathbf{b}_y$ by computing their gradients w.r.t. to $\mathcal{E}$

$$\mathbf{W}_{xh} := \mathbf{W}_{xh} - \alpha_f \frac{\partial \mathcal{E}}{\partial \mathbf{W}_{xh}} \tag{12}$$

This mechanism is similar to the approach in Pascanu et al. (2012), where the regularisation is constrained to the transition matrix, and all other parameters are updated using the chain rule. The experiments in Section 3 are conducted using an equivalent method, where the parameters of the output are updated using (11), and the parameters of the input are updated using (12).

Alternatively, $\mathbf{W}_{xh}$ can be updated in a way similar to how the $\mathbf{W}_{hh}$ updates are being performed, pushing the output of the hidden layer closer to the local target.

$$\mathbf{W}_{xh} := \mathbf{W}_{xh} - \alpha_f \sum_{t=1}^{t_{\max}} \frac{\partial \mathrm{MSE}(F(\mathbf{x}_t, \mathbf{h}_{t-1}), \hat{\mathbf{h}}_t)}{\partial \mathbf{W}_{xh}} \tag{13}$$

The results of an identical set of experiments using (13) for the $\mathbf{W}_{xh}$ updates is provided in Appendix A.

The process of setting targets, optimising the parameters of $G(\cdot)$, and optimising the parameters of $F(\cdot)$ is repeated until a selected convergence criterion is met.

2.1 Gradients Stability

As discussed in Section 1.1, vanishing and exploding gradients prevent backpropagation-trained RNNs from learning long-term dependencies. In a deep recurrent network, where the derivatives of $\mathbf{W}_{hh}$ in the lower (decreasing t) layers vanish or explode, the lower layers will not be able to contribute effective corrections to the synaptic weights of the hidden-to-hidden connections.

TPTT alleviates the issue by assigning local targets at every time step, thus allowing the lower layers to continuously learn and contribute to the weight changes in a way that hopefully leads to a lower global error. For the case where $G(\cdot)$ is a perfect inverse of $F(\cdot)$, it has been analytically proven (see Theorem 1 in Lee et al. (2015)) that local updates that improve layer-wise loss also decrease the global loss.

Lee et al. (2015), however, point out that choosing $G(\cdot)$ to be the perfect inverse of $F(\cdot)$ may need heavy computation and lead to instability, hence the model described adopts an alternative approach where an approximate inverse is learnt. In this case, the expectation that bringing the outputs in the lower layers closer to $\hat{\boldsymbol{h}}_t$ also reduces $\mathcal{E}$ is not guaranteed.

2.2 Biological Plausibility

There exists compelling evidence that certain unbiological traits of backpropagation-trained RNNs can be mitigated by using target propagation.

First and foremost, target propagation does not require the passage of error signals back through the synapse towards the upstream layers. Instead, it optimises towards targets, which can be set locally via feedback pathways. There are many feedback connections in the brain (Kaas and Lyon 2001). More specifically, Spratling (2002) illustrates how pyramidal neurons in layers II and III of the cerebral neocortex receive information from different sources, integrating feedforward information at their basal dendrites and feedback information at the apical dendrites. Difference target propagation based training can then be considered a form of error-driven learning, where the basal synaptic weights of a neuron are modified towards a required output presented via the apical input. This can be deemed a form of modulation, where basal activity is suppressed or enhanced to match a certain expectation (target).

A similar interpretation is given by Roelfsema and Holtmaat (2018), who present a model where the synaptic plasticity is explained based on the idea that the error signal factorizes into two components, a steering neuromodulatory signal that determines whether a synapse undergoes potentiation or depression, and a gating signal carried by feedback connections, that determines how much credit or blame should be assigned to individual synapses. In

contrast to Roelfsema and Holtmaat's model, where the gating values are always in $\{0, 1\}$ and the sign of the change is controlled via neuromodulation, the feedback connections in TPTT pass real-valued information to the targeted synapses. This allows the sign of the locally computed error to drive individual synapses towards long-term potentiation (LTP) or long-term depression (LTD).

Assuming that the feedback pathways provide the information required to assign credit or blame allows TPTT to potentially present a more biologically plausible solution to the credit assignment problem. As mentioned in Section 1.2, the common approach to credit assignment in backpropagation-based settings is to use weight transport to explicitly calculate updates for the synaptic weights at earlier time steps. TPTT avoids this non-local transmission of information by optimising towards locally set targets. One could argue that the process of setting of the local target can be viewed as a form of weight transport, however, an argument could be put forward about the feedback connections in the human brain being plastic. In this case a more biologically plausible model is a neuron where all incoming synaptic weights are adaptable (Luo et al. 2017). There is, on the other hand, sufficient evidence that the necessity for updates to $\mathbf{V}_{hh}$ and $\mathbf{c}_h$ might be circumvented by using the random feedback alignment technique suggested by Lillicrap et al. (2016). This could potentially completely remove the requirement for computing gradients and updating the local targets during the feedback phase, although Luo et al. (2017) show that adaptable feedback outperforms feedback-alignment.

As mentioned in Section 1.2, this weight transport resolution provided by TPTT is not applicable to SRNNs, as the unrolled network is a temporally indexed version of the same neuron. It was indicated, however, that target propagation can indeed bring benefits in terms of biological plausibility to recurrent networks comprising multiple neurons with long-range connections. A specific example of such a network could be drawn from the information processing mechanism in the human visual system. It has been suggested that this mechanism employs an underlying recurrent circuit (Drewes et al. 2016, Lamme and Roelfsema 2000), and there is significant evidence that the ventral visual pathway is a complex recurrent network (Kravitz et al. 2013). It has been shown that in primates the primary visual cortex (V1) links not only to the secondary visual cortex (V2), but also to visual areas 3 and 4 (V3, V4), and the middle temporal visual area (MT) (see Van Essen et al. 1986, Nakamura et al. 1993, and Maunsell and van Essen 1983). Based on the assumption that in the human ventral system V2 sends information to V4, which is then send back to V1, and that V4 sends forwards information to the inferior temporal cortex (IT), which is again recurrently connected to V2, Chang et al. (2017) present a simplified RNN model in the form of V1-V2-V4-IT, with V1 receiving recurrent connections from V2, V4, and IT. Unfolding such a network in time, for processing dynamic visual signals, would now entail the repetition of interconnected groups of neurons. In this setting the credit assignment problem reappears in the form of within-the-group weight transport. This can be mitigated by target propagation, however, the need for complex temporal indexing still pertains as group level targets would not be received for a number of time steps. It appears that the storage of temporally indexed data is not a constraint that TPTT, in its present form, can resolve.

Another objection mentioned in Section 1.2 is related to the use of continuous values. By definition, backpropagation uses the error gradients to apply corrections to the synaptic

weights of the network. When the context window increases, learning speed for the early time steps decreases as the gradients get closer to zero. Lee et al. (2015) discuss an extreme case where the error signal becomes discrete: its derivatives are zero almost everywhere and infinite where the function changes discretely. A backpropagation-trained network cannot properly operate in this regime, but target propagation can handle such discrete non-linearities, and it can indeed learn using discrete transmissions between units (LeCun 1986). This property can help with making the model more biologically plausible by implementing spike coding (discrete binary units) instead of coding of analogue variables by firing rate, which seems biologically improbable, especially in the context of fast cortical computations (Maas 1997).

Finally, in contrast to BPTT, TPTT does not require precisely clocked changes between the forward and feedback regimes. The computational dependency between the two phases is sufficiently relaxed to allow them to run in an arbitrary order. In fact, they could potentially run in parallel.

2.3 Related Work

An attempt to use target propagation in a recurrent setting was previously presented by Wiseman et al. (2017). The authors, however, find the results "disappointing", with the presented algorithm underperforming in comparison to BPTT on the two real-world language modelling data sets (Penn Tree Bank and Text8). It should be noted, that the target propagation variant of Wiseman et al. (2017) is based on a constraint in the form of $\lambda C(\hat{h}_t, h_t)$ added to the loss, where $\hat{h}_t$ is a predicted hidden state at time t and C is an $L2$ type penalty. In contrast to the work of Wiseman et al. (2017), this paper demonstrates that Target propagation through time, as outlined in Section 2, generally outperforms BPTT-trained recurrent networks.

Another idea closely related to the target propagation approach adopted in TPTT are synthetic gradients (Czarnecki et al. 2017; Jaderberg et al. 2017). The concept of synthetic gradients is similar to target propagation in the sense that the synaptic weights update process does not have a strict dependency on a backpropagated global error signal.

The different layers of the network are individually updated, but instead of providing local targets and updating the network parameters to bring the activation values close to the targets, the network uses local models to approximate the true error gradients directly. Under this scheme, the model M_t for network layer t is trained by minimising the error between the predicted gradient $\hat{\delta}_t$ and the gradient estimated by the synthetic model in the next layer $\hat{\delta}_{t+1}$. This process is repeated for all upstream layers until the final layer of the network, where the target gradient can be computed directly from the global error $\mathcal{E}$. This training method allows individual segments of the network to be update asynchronously, resulting in an architecture known as Decoupled Neural Interfaces (DNIs).

The DNI architecture gives rise to a group of interesting properties. By removing the sequential backpropagation of error signals through the network, DNI provides a method where the network can be updated asynchronously. The constraint that a layer has to wait until a full forward pass and a backward pass up to the said layer must be completed before the layer is updated is substantially relaxed. Using synthetic gradients the decoupled layers

can be independently updated. This property facilitates parallel learning and in addition enables different segments of the network to learn at different speeds.

3. Experiments

TPTT was tested on a subset of the pathological synthetic problems initially presented in Hochreiter and Schmidhuber (1997). The network was also tested on a sequence classification task based on the MNIST data set (LeCun et al. 1998).

3.1 Pathological Synthetic Problems

The pathological synthetic problems are known to be very challenging for SRNs to solve, as they require the memorization of long-term correlations. Here is a brief description of the four individual problems selected for testing TPTT:

- **The Temporal Order Problem**. The goal in this problem is sequence classification. 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 selected at random and presented at positions $t_1 \in [\frac{T}{10}, \frac{2T}{10}]$ and $t_2 \in [\frac{4T}{10}, \frac{5T}{10}]$. The network must predict the correct order of appearance of X and Y out of four possible options: $\{XX,\ XY,\ YX,\ YY\}$

- **The 3-bit Temporal Order Problem**. This problem is similar to the Temporal Order Problem, but the positions of interest are increased to three—$t_1 \in [\frac{T}{10}, \frac{2T}{10}]$, $t_2 \in [\frac{3T}{10}, \frac{4T}{10}]$, and $t_3 \in [\frac{6T}{10}, \frac{7T}{10}]$. This also leads to an increased number of possible outcomes that the network must learn to predict: $\{XXX,\ XXY,\ XYX,\ XYY,\ YXX,\ YXY,\ YYX,\ YYY\}$

- **The Adding Problem**. The problem presents the network with two input channels of length T. The first channel is a sequence of randomly selected numbers from $[0, 1]$. The second channel is a series of zeros, with the exception of two positions $t_1 \in [1, \frac{T}{10}]$ and $t_2 \in [\frac{T}{10}, \frac{T}{2}]$, where its values are ones. The ones at positions t_1 and t_2 act as markers that select two values from the first channel: X_1 and X_2. The target that the network must learn to predict is the result of $\frac{X_1+X_2}{2}$

- **The Random Permutation Problem**. This task receives a sequence of symbols T, with the symbol at t_1 being either 1 or 0 and also being identical to the symbol at t_{max}. All the other symbols in the sequence are randomly sampled from $[3, 100]$. This condition produces two types of sequences—$(0, a_{t_2}, a_{t_3}, \ldots, a_{t_{max-1}}, 0)$ and $(1, a_{t_2}, a_{t_3}, \ldots, a_{t_{max-1}}, 1)$ where a_t is randomly sampled from $[3, 100]$. The goal is to predict the symbol at t_{max}, which only depends on the symbol at t_1, while the other symbols in the sequence act as distractors.

The problems were tested with a Simple RNN[1] with a hyperbolic tangent non-linear activation function, 100 neuron in the hidden layer, all synaptic weights sampled from $\mathcal{N}(\mu = 0, \sigma = 0.1)$, and the biases initialised with zeros. The optimisation technique used

1. Source code for all experiments is available at `https://github.com/nmanchev/tptt`

for training the network was Nesterov's Accelerated Gradient (Nesterov 1983; Bengio et al. 2013) with the momentum μ_t set to 0.9.

The approach was to start with a relatively short sequence length (T) of 10 time steps and keep increasing it by 10 time steps, in an attempt to see how well the network will handle long test sequences. This approach, however, did not work well with the BPTT-trained SRN. In line with Hochreiter and Schmidhuber (1997)'s observations that BPTT, as many other recurrent net algorithms, "fail miserably on real long time lag problems", the network was unable to solve all but one of the problems, even at a relatively shallow depth of 10 time steps. Moreover, the 3-bit Temporal Order Problem, which the network did manage to solve, was only handled at the initial depth of 10 time steps and failed for larger values of T.

In order to provide a meaningful baseline and to better assess the effect of using TPTT a variant of the SRN was tested, where the synaptic weight matrices were carefully initialised to be orthogonal. Orthogonal matrices preserve gradient norm during backpropagation, hence they can help mitigate issues related to vanishing gradients (Henaff et al. 2016; Le et al. 2015; Arjovsky et al. 2015).

The networks were run for up to 100'000 iterations, each iteration processing a mini-batch of 20 examples: 2 million examples in total. For the Temporal, 3-bit Temporal, and Random Permutation problems the network uses a **softmax** layer as its final layer (see Equation 2), and optimises a cross-entropy cost function between the prediction $\mathbf{y}$ and a target $\mathbf{t}$

$$\mathcal{E} = -\frac{1}{N}\sum_{i=1}^{N}(\mathbf{t}_i \times \log(\mathbf{y}_i))$$

where N is the number of samples in the mini-batch. For the adding problem, which requires a real valued output, the last layer of the network is linear, and the cost function the network minimises is the MSE:

$$\mathbf{y} = \mathbf{W}_{hy} \cdot \mathbf{h}_t + \mathbf{b}_y$$
$$\mathcal{E} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{t}_i - \mathbf{y}_i)^2$$

In this case, an example i in the mini-batch is considered successfully predicted if the error between the prediction and the target is below 0.04. This criterion is identical to the one used by Hochreiter and Schmidhuber (1997).

For all problems, the accuracy of the network was measured every 100 iterations on a validation set of 10'000 samples. The success criterion was a validation error below 0.0001. If the network did not meet this criterion by its last iteration, the run was considered a failure. The results from all runs for the two different networks are shown in Table 1. The network referred to as BPTT is an backpropagation-trained SRN with orthogonal initialisation for $\mathbf{W}_{hh}$; TPTT is a recurrent network with orthogonal initialisation combined with target propagation through time learning. The hyperparameters for both the BPTT and TPTT networks were optimised against each T, using a grid search over $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ for each α. The parameters used by each network to achieve the reported T_{max} are also provided in Table 1.

Problem	BPTT		TPTT	
	t_{max}	parameters	t_{max}	parameters
Temporal Order	120	$\alpha = 0.00001$	**150**	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.001$
3-bit Temporal Order	70	$\alpha = 0.00001$	**150**	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.001$
Adding	**90**	$\alpha = 0.001$	60	$\alpha_i = 0.5$, $\alpha_f = 0.02$, $\alpha_g = 0.07$
Random Permutation	70	$\alpha = 0.01$	**300**	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.001$

Table 1: Maximal depth (sequence length) by model. Each model is trained with an initial sequence length of T=10. If it successfully solves the problem, T is increased by 10 and the model is trained and tested again. The process is repeated until failure and the highest T achieved by the network is reported as t_{max}.

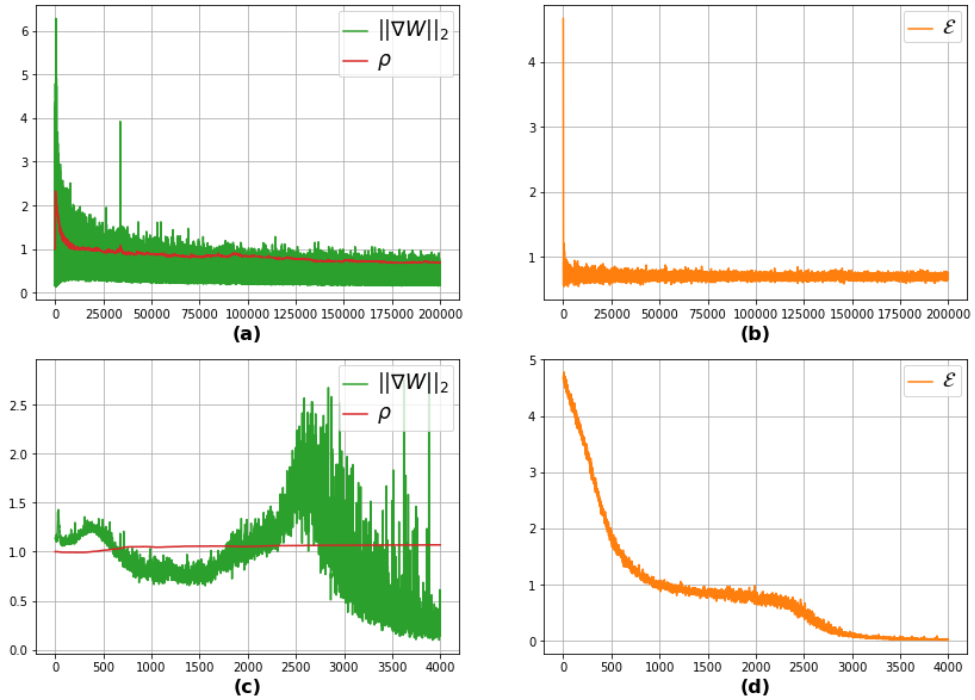

Figure 3: Comparison of BPTT and TPTT-trained networks for the temporal order problem with $T = 100$. For the BPTT network **(a)** shows the change in the norm of the synaptic weight updates ($||\nabla \mathbf{W}||_2$) and the spectral radius of the hidden-to-hidden matrix ($\boldsymbol{\rho}$); **(b)** shows the change in cost ($\mathcal{E}$). For the TPTT the change in $||\nabla \mathbf{W}||_2$ and $\boldsymbol{\rho}$ is given in **(c)**, and $\mathcal{E}$ is shown in **(d)**.

Table 1 reveals that replacing the BPTT learning mechanism with TPTT allows successful training to greater depths (larger t_{max}) in three out of the four selected synthetic

Problem	t_{max}	BPTT best	TPTT common model	TPTT best	TPTT parameters
Temporal Order	120	46,500	**1,600**	**1,600**	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.001$
3-bit Temporal Order	70	56,000	1,300	**800**	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.1$
Addition	90	18,000	-	-	-
Random Permutation	70	12,200	12,700	**700**	$\alpha_i = 0.1$, $\alpha_f = 0.1$, $\alpha_g = 0.01$

Table 2: Difference between BPTT and TPTT-training expressed as number of iterations needed until convergence. "BPTT best" provides the results from the fastest converging BPTT models based on the grid search. "TPTT common model" shows the iterations needed by the TPTT model with identical hyperparameters used in Table 1 ($\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.001$). "TPTT best" and "TPTT parameters" show the results from the fastest TPTT models based on the grid search and their respective parameters.

problems. Moreover, a single TPTT model with identical hyper-parameters was able to successfully handle the Temporal Order, 3-bit Temporal Order, and Random Permutation problems. The adding problem turned out to be the most difficult of the four, and although the network was on the correct path to solving it, its convergence rate was quite slow and would not allow for the training to reach a solution within the limits of the experiment (100'000 iterations). To overcome this, the TPTT learning rates for this specific problem were manually increased and set to $\alpha_f = 0.02$, $\alpha_g = 0.07$, and $\alpha_i = 0.5$. This allowed the TPTT network to perform better in terms of sequence lengths, but it still could not match the results obtained using backpropagation.

The success of TPTT in the classification tasks can be attributed to the mitigation of the vanishing/exploding gradients problem. Figure 3 shows the impact of TPTT on learning and gradient stability for one specific run, however this characteristic situation was consistently observed across different problems and separate runs. In the BPTT-trained network the spectral radius of the hidden-to-hidden matrix gradually decreases until reaching the sufficient criterion for the gradients to vanish ($\rho < 1.0$). This correlates with the decrease in the norm of $\nabla \mathbf{W}$, resulting in the inability of the network to learn efficiently and the lack of reduction in the cost function. In contrast, the TPTT network maintains stable ρ, updates to the synaptic weights are not constrained to the beginning of the training, and the cost decreases until the problem is successfully solved.

Interestingly, the TPTT-trained networks also outperformed BPTT in terms of convergence speed. Table 2 shows the number of iterations needed by BPTT and TPTT for solving the longest sequences that the BPTT-trained network was capable of handling. With the exception of the adding problem, which TPTT could not solve for $t_{max} = 90$, the number of iterations needed was significantly lower when using target propagation for training. The change in the cost function $\mathcal{E}$ for the best performing networks is provided in Figure 4.

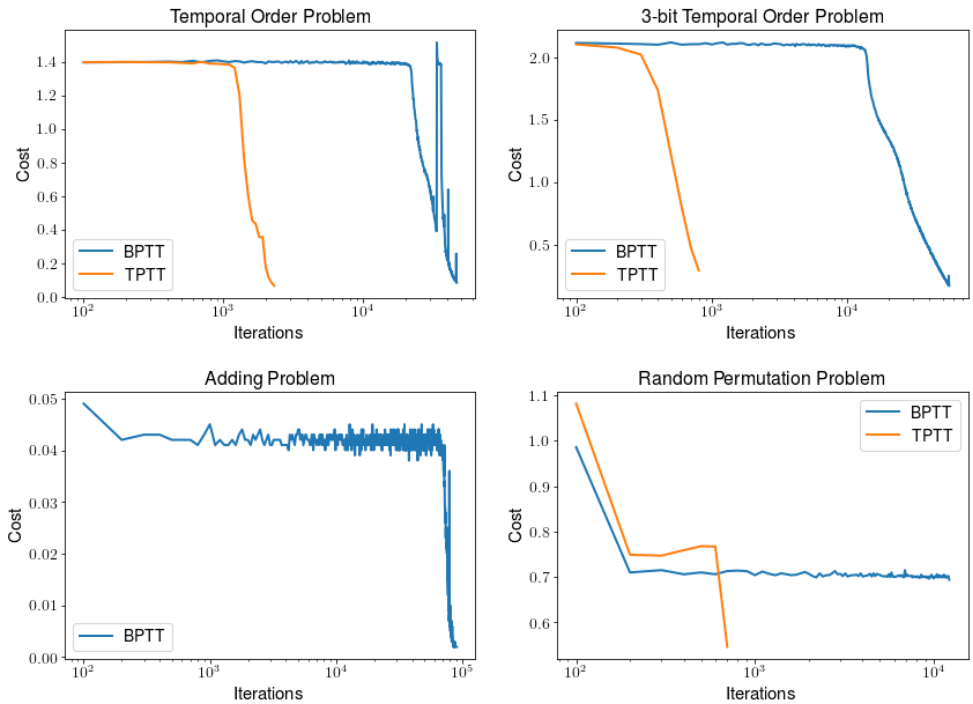

Figure 4: Changes in the training cost for the synthetic problems. The models compared are the ones listed in Table 2. The cost function is averaged over 100 iterations. The iterations are given on a logarithmic scale, as the TPTT network converges significantly faster than the BPTT network.

3.1.1 Comparison of Weight Updates and Layerwise Errors

Lillicrap et al. (2016) demonstrate that after a few epochs of training random feedback alignment can match the synaptic weight updates prescribed by backpropagation. This analysis is performed by exploring the angle between the updates specified by the two learning algorithms.

A similar investigation is presented here in regard to the similarity of the transition matrix updates prescribed by TPTT and BPTT. This analysis is confined to $\boldsymbol{W}_{hh}$, as the remaining network parameters are updated according to Equation 11 and Equation 12, hence they are identical across the backpropagation and target propagation networks. The TPTT-trained network was run as usual and let to compute the update term as given in Equation 10:

$$\Delta \boldsymbol{W}hh_{TPTT} = \alpha_f \sum_{t=1}^{t_{\max}} \frac{\partial \mathrm{MSE}(F(\mathbf{x}_t, \mathbf{h}_{t-1}), \hat{\mathbf{h}}_t)}{\partial \mathbf{W}_{hh}}$$

In parallel, an alternative update term was computed using backpropagation.

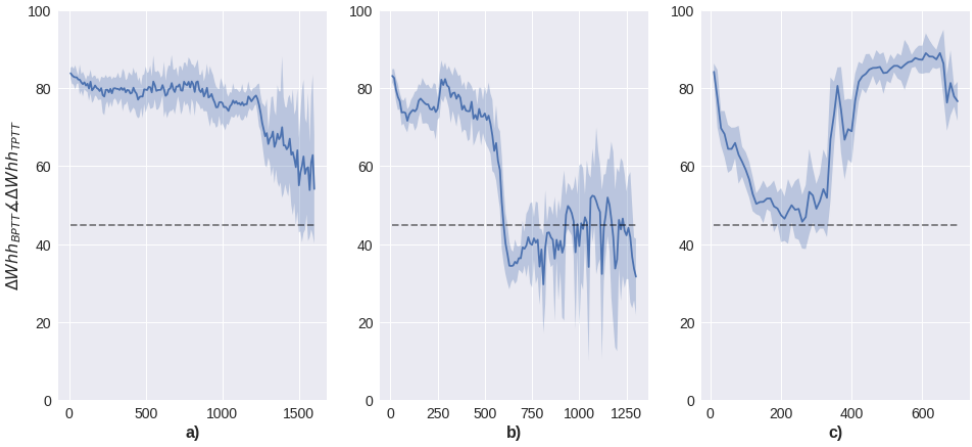

Figure 5: Angle between BPTT and TPTT prescribed updates. The data is averaged over 10 iterations with the error bars indicating two standard deviations. **a)** Temporal order problem with $t_{max} = 120$; **b)** Temporal 3-bit problem $t_{max} = 90$; **c)** Random Permutation problem $t_{max} = 70$;

$$\Delta \boldsymbol{W} hh_{BPTT} = \alpha_f \frac{\partial \mathcal{E}}{\partial \mathbf{W}_{hh}}$$

The angle between the update matrices is then measured using

$$\Delta \boldsymbol{W} hh_{BPTT} \measuredangle \Delta \boldsymbol{W} hh_{TPTT} = \arccos\left(\frac{\langle \Delta \boldsymbol{W} hh_{TPTT}, \Delta \boldsymbol{W} hh_{BPTT} \rangle}{\|\Delta \boldsymbol{W} hh_{TPTT}\| \|\Delta \boldsymbol{W} hh_{BPTT}\|}\right)$$

A set of experiments were conducted, where the angle between the prescribed updates was measured and aggregated over every 10 iterations, and the network was allowed to run until reaching the convergence criterion. The results for the temporal order, 3-bit temporal order, and random permutation problems are presented in Figure 5. For all three problems, the $\Delta \boldsymbol{W} hh_{BPTT} \measuredangle \Delta \boldsymbol{W} hh_{TPTT}$ is initially close to 90°. As training progresses, the update vectors for all three problems become less orthogonal to each other, and in the 3-bit temporal order problem the angle falls under 45°. Interestingly, halfway through the random permutation problem the angle increased again, peaking at about 89°. Combined with the much shorter convergence time, this indication of roughly orthogonal updates suggests that in this case TPTT discovered a better path leading to a local minimum.

A modified version of the network, which tracks the layer-specific cost functions over the period of training was developed, in an attempt to get additional insight on the learning dynamics in TPTT. Figure 6 shows the results for three TPTT networks, working on different problems for $T = 30$. The selection of T was mainly determined with clarity and readability in mind, as deeper networks tend to produce obscured plots that are difficult to analyse. However, experiments with larger values of T where only a subset of layers were plotted (e.g. every third layer) confirmed that the networks exhibit similar behaviour.

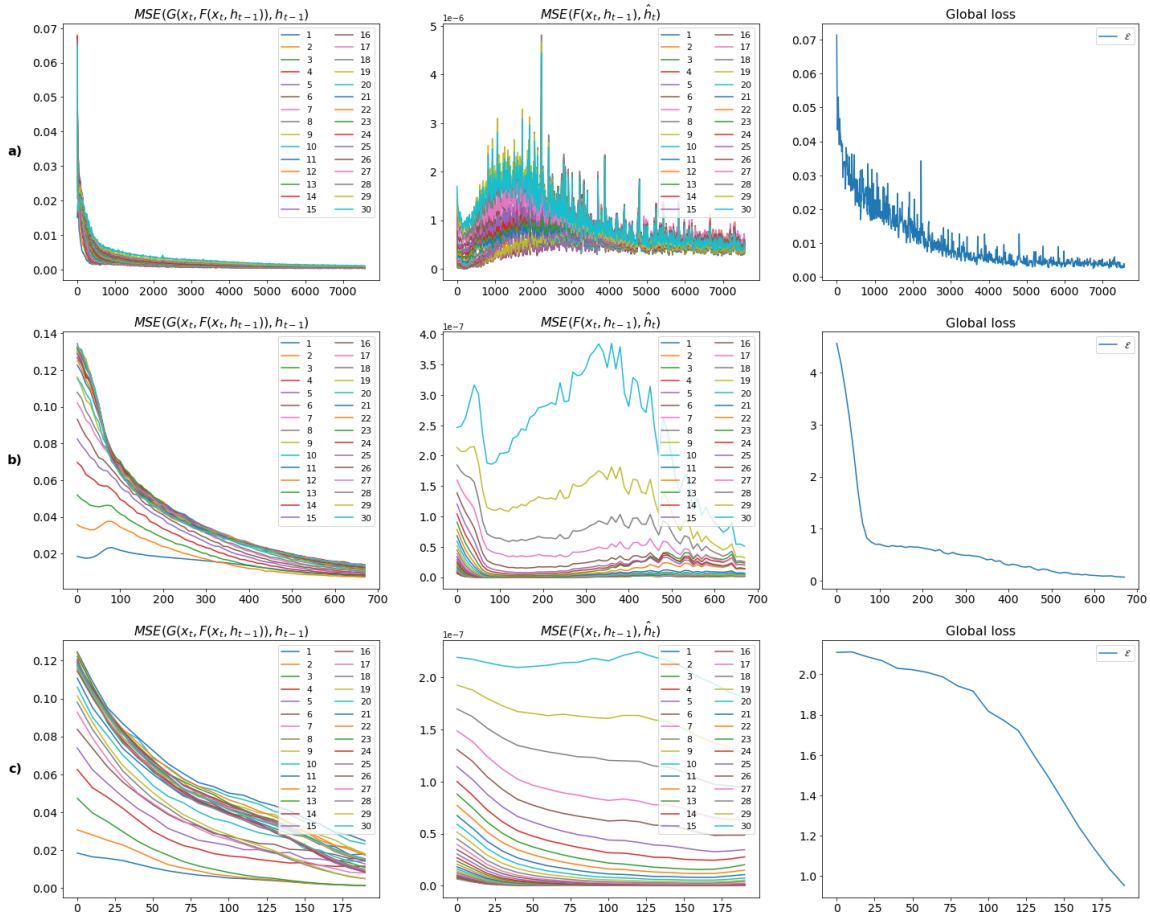

Figure 6: Change in the layer-specific cost in three TPTT trained networks solving the addition problem for $T = 30$ (row **a**), the permutation problem for $T = 30$ (row **b**), and the 3-bit temporal order problem for $T = 30$ (row **c**). The first plot in each row represent the cost used in the update rule for $\mathbf{V}_{hh}$ and $\mathbf{c}_h$. The second plot represents the cost used to learn $\mathbf{W}_{hh}$ and $\mathbf{b}_h$. The third plot shows the change in the global error $\mathcal{E}$. Layers are numbered in a way that reflects the depth of the network with 1 being the cost at $\mathbf{h}_0$ and 30 the cost at $\mathbf{h}_{29}$

The data in Figure 6 reveals that in general the network manages to approximate the inverse without major difficulties. However, it also appears that the difference between the local targets and the feedforward activations becomes smaller towards the lower levels of the network. This likely has a negative effect on the capability of the network to learn and is similar to the effects induced by vanishing gradients. However, it appears that impact in TPTT is not as severe, as the network still outperform its BPTT-trained counterparts on three out of the four synthetic problems.

t_{max}	Parameters	Without noise		With noise injection	
		Accuracy %	ϵ	Accuracy %	ϵ
70	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.01$	82.67	0.0	**94.99**	0.100
80	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.001$	99.09	0.0	**99.59**	0.005
90	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.01$	92.74	0.0	**96.52**	0.006

Table 3: Impact of noise injection on a TPTT-trained network when tested against the adding problem.

3.1.2 Gaussian Noise

Lee et al. (2015) suggest injection of random Gaussian noise in the process of learning $G(\cdot)$. Under this scheme the update rule for the synaptic weights of $G(\cdot)$ given in Equation 9 changes to

$$\mathbf{V}_{hh} := \mathbf{V}_{hh} - \alpha_g \sum_{t=1}^{t_{\max-1}} \frac{\partial \mathrm{MSE}(G(\mathbf{x}_t, F(\mathbf{x}_t, \mathbf{h}_{t-1} + \epsilon)), \mathbf{h}_{t-1} + \epsilon)}{\partial \mathbf{V}_{hh}}$$

$$\mathbf{c}_h := \mathbf{c}_h - \alpha_g \sum_{t=1}^{t_{\max-1}} \frac{\partial \mathrm{MSE}(G(\mathbf{x}_t, F(\mathbf{x}_t, \mathbf{h}_{t-1} + \epsilon)), \mathbf{h}_{t-1} + \epsilon)}{\partial \mathbf{c}_h}$$

where $\epsilon \sim \mathcal{N}(0, \sigma)$. The introduction of noise acts as regularisation, ensuring that the inverse function is not fine-tuned to the training data, but maps to a region around these values in a way that facilitates the computation of $G(\mathbf{x}_t, \hat{\mathbf{h}}_t)$ for unseen $\hat{\mathbf{h}}_t$.

Given that TPTT performed well on the temporal order, 3-bit temporal order, and random permutation problems without the need of noise injection, its impact for these specific problems did not warrant an in-depth investigation. In these three cases the TPTT model presented in Table 1 can be viewed as using the corrected update rule but having $\epsilon = 0$.

A number of experiments were conducted for the adding problem to determine the impact introduced by noise injection. First, a grid search was run for sequences of 70, 80, and 90 time steps. The best performing combination of learning parameters were selected for each sequence, and another grid search was performed over $(0, 1]$ for finding optimal ϵ values. The results from these experiments are given in Table 3, and the impact of introducing noise on the change in the cost function for $t_{max} = 70$ is shown in Figure 7. The experimental results support the suggestion of Lee et al. (2015) that making $F(\cdot)$ and $G(\cdot)$ approximate inverses not just at $\boldsymbol{h}_{t-1}$ but in its neighbourhood leads to better results. For this specific problem, however, the improvement did not provide sufficient boost to the network's accuracy to successfully solve the problem for larger t_{max}.

3.1.3 Asynchronous Learning

One of the limitations of backpropagation that was acknowledged in Section 1.2 is the need of precise alternation between feedforwad and backpropagation phases. In target propagation

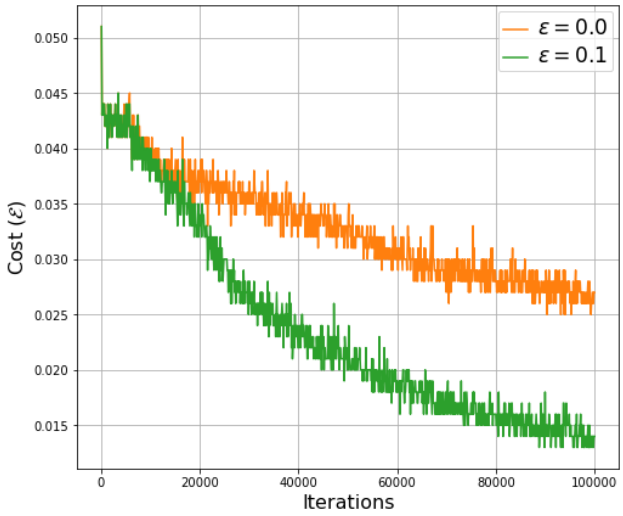

Figure 7: Impact of Gaussian noise on the adding problem with $t_{max} = 70$. The learning parameters are set to $\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.01$, and the network is run with $\epsilon = 0.0$ and $\epsilon = 0.1$ for 100,000 iterations. The value of the cost function is averaged over 100 iterations.

the dependency between the parameter update phases of $G(\cdot)$ and $F(\cdot)$ is not as strict, relaxing the requirement for precisely clocked transitions between the two phases.

This suggestion was experimentally confirmed by running a modified version of the TPTT network, where the probability of executing either of the update phases is controlled by a tunable parameter. The network was used in a series of tests on the pathological synthetic problems where the change of the cost function was compared between runs with strictly alternating phases and runs where the update phases were probabilistically determined.

Figure 8 shows the change in cost for the Temporal and the 3-bit Temporal Order Problem when the phase change is synchronised (i.e. $G(\cdot)$ update is always followed by an $F(\cdot)$ update). The two alternative runs per network show results for when the phase is randomly determined with the network running a $G(\cdot)$ update with a probability of $\frac{1}{2}$ in the first run, and a probability of $\frac{1}{3}$ in the second run.

The empirical evidence upholds the statement that precise phase alternation is not required in TPTT-trained networks. Although there is a small penalty in terms of the onset of decrease of the cost, the fact that the two phases do not strictly depend on each other hints towards the possibility of a TPTT implementation where the parameter update for $G(\cdot)$ and $F(\cdot)$ are computed in a parallel fashion.

3.1.4 Generalisation Capability

It has been shown (Pollack 1990; Pascanu et al. 2012) that recurrent networks sometimes generalise beyond the length of the training examples they were exposed to. It appears

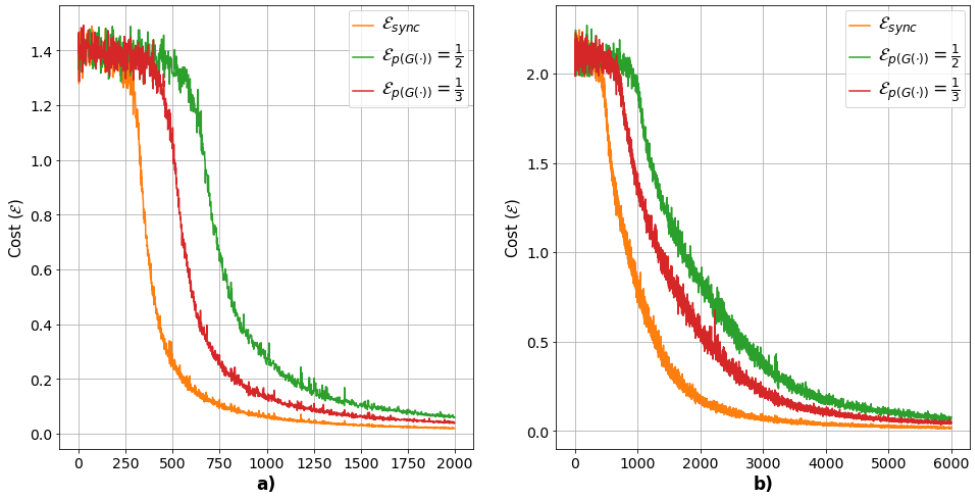

Figure 8: Impact of randomised transition between updating the parameters of $G(\cdot)$ and $F(\cdot)$ on **a)** temporal order problem with $t_{max} = 60$; **b)** 3-bit temporal order problem with $t_{max} = 60$; $\mathcal{E}_{\text{sync}}$ is the cost change in a network with constant alteration between the $G(\cdot)$ and $F(\cdot)$ phases. $\mathcal{E}_{p(G(\cdot))=\frac{1}{2}}$ and $\mathcal{E}_{p(G(\cdot))=\frac{1}{3}}$ represent networks that switch to updating the parameters of $G(\cdot)$ with probability $\frac{1}{2}$ and $\frac{1}{3}$ respectively.

Problem	**Accuracy (%)**	
	BPTT	TPTT
Temporal Order	26.67	**45.76**
3-bit Temporal Order	13.65	**28.36**
Adding	65.63	**66.23**
Random Permutation	51.52	**100.00**

Table 4: Accuracy of the BPTT and TPTT networks trained on samples with $t_{\max} = 100$ and tested on sequences of length 200. The learning rate used for the backpropagation network is the corresponding problem specific value given in Table 1. The TPTT network uses the common model parameters ($\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.001$)

relevant to investigate this generalisation capability by comparing the solutions found by the TPTT and BPTT-trained networks.

A series of experiments were conducted, where a TPTT and a BPTT networks were trained on the synthetic problems using training sequences of length $T = 100$. The two networks were then tested using longer sequences with T set to 200. The accuracy achieved

Problem	Learning rate	Accuracy (%)
Temporal Order	0.0001	75.28
3-bit Temporal Order	0.001	29.03
Adding	0.0001	91.90
Random Permutation	0.0001	50.54

Table 5: Accuracy of an LSTM network trained on samples with $t_{\max} = 100$ and tested on sequences of length 200.

by the networks is reported in Table 4. The data reveals that the TPTT-trained network outperforms the equivalent network trained with backpropagation on all of the synthetic problems, suggesting that the problem-specific solutions found using target propagation generalise better.

An identical experiment was performed for assessing the generalisation capability of an LSTM network. The configuration of the LSTM network was analogous to the simple recurrent network, with the activation function set to hyperbolic tangent and using orthogonal initialisation for the synaptic weights. The learning rate α was optimised using a grid search over $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ for sequences with $T = 100$. The network was then trained using the optimal learning rate for each of the synthetic problems and its accuracy was measured on sequences of length 200. The results of this experiment are given in Table 5. It is evident that the LSTM network outperforms both the BPTT and TPTT networks on three of the synthetic problems. It should be noted, however, that this result is largely provided for reference, as the improvement demonstrated is likely driven by the LSTM architecture and not so much by the specific optimisation technique—the LSTM network used in this specific experiment was optimised using backpropagation.

3.2 MNIST Sequence of Pixels

The MNIST database contains a large collection of images of handwritten digits. The data set was assembled by LeCun et al. (1998) and is derived from the NIST Special Database 19 (Grother 1995). It contains 60'000 training and 10'000 test images with dimensions of 28x28 pixels each (784 pixels in total).

The MNIST data was used to define the MNIST classification from a sequence of pixels problem, originally devised by Le et al. (2015). In this challenge, the images are presented to the network one pixel at a time in a scanline order. This results in a very long range dependency problem as 784 pixels translate to 784 time steps (i.e. $T = 784$).

To make the results from the TPTT network comparable to the results of Le et al. (2015), the training configuration was kept as similar as possible. The number of images per mini-batch was set to 16, the training was capped to 1'000'000 iterations (≈ 267 epochs), and the number of neurons in the hidden layer was set 100.

A grid search was performed for finding optimal α_i, α_g, and α_f, however, due to the size and computational complexity of the data, the grid search was carried out using only 10'000

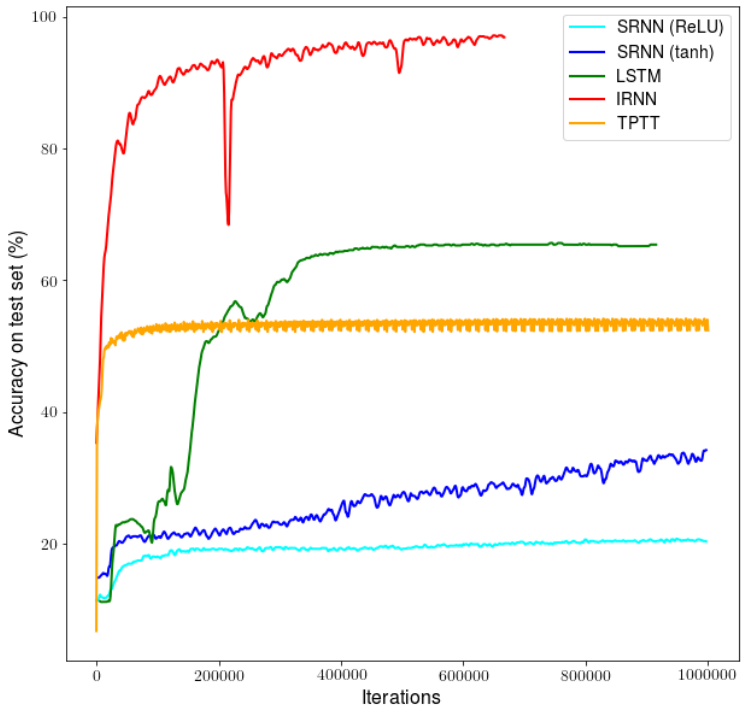

Figure 9: Accuracy results from the "MNIST sequence of pixels problem"

training and 1'000 test images, which were randomly selected from the complete data set. The runs were also limited to a maximum of 375'000 iterations (100 epochs). The hyper-parameter space for α_i, α_g, and α_f was constrained to $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}\}$. The best accuracy was determined to be produced when using $\alpha_i = 10^{-7}$, $\alpha_f = 10^{-2}$, and $\alpha_g = 10^{-8}$.

The accuracy results achieved by a TPTT-trained SRNN on the test MNIST set are compared to four other networks—a BPTT-trained SRNN with ReLU activations, a BPTT-trained SRNN with *tanh* activations, a BPTT-trained SRNN with ReLU activations and identity matrix initialisation (IRNN), and an LSTM network. The performance data from the four networks is directly obtained[2] from Le et al. (2015). The maximal accuracy achieved by all networks is given in Table 6, and the change over time is shown in Figure 9.

It is evident, that the TPTT-trained network outperforms both the ReLU and tanh-based RNNs, although it does not perform as well as the LSTM or IRNN networks when constrained to the same number of neurons in the hidden layer.

It has been reported that "too large" networks trained with backpropagation rarely do worse, as backpropagation tends to ignore excess parameters (Caruana 1993 as cited in Lawrence et al. 1997). This prompts the question if the TPTT network would experience difficulties as the number of neurons in the hidden layer increases, as this would also increase

2. The data used for the SRNN, LSTM, and IRNN networks presented in Table 6 and Figure 9 was extracted from the plot presented in Le et al. (2015) and potentially suffers from a minor loss of precision.

Network	Accuracy %
SRNN (ReLU)	20.74
SRNN (tanh)	34.20
LSTM	65.68
IRNN	97.20
TPTT	54.20

Table 6: Maximal accuracy on the MNIST data set for SRNN (using backpropagation with ReLU or tanh activation), LSTM, IRNN, and TPTT.

Size of $\mathbf{W}_{hh}$	Accuracy %
64	49.49
128	53.28
256	54.60
512	56.77
1024	74.02

Table 7: Maximal accuracy on the MNIST data set in a TPTT network for various sizes of the transition matrix $\mathbf{W}_{hh}$.

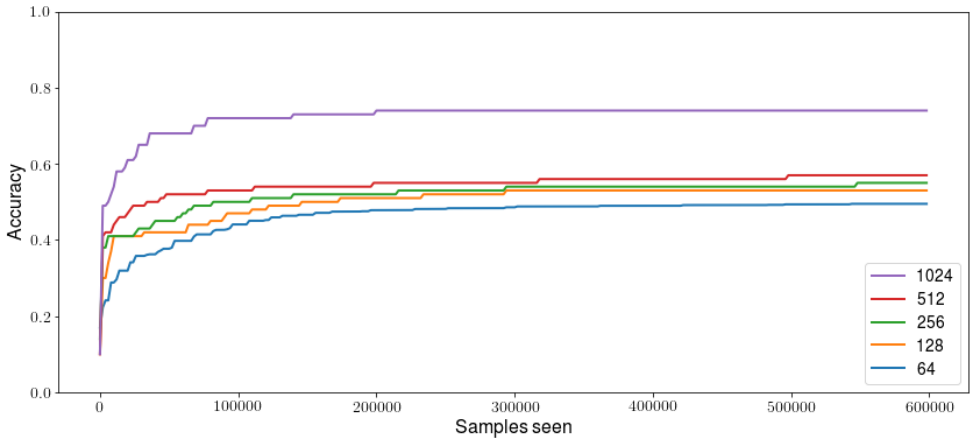

Figure 10: Impact of varying the number of neurons in the hidden layer on the accuracy achieved by a TPTT-trained network on the "MNIST sequence of pixels problem"

the variance in the gradient approximation. It could be therefore argued that the superior results shown by TPTT in comparison to BPTT are dependent on the fairly small number of neurons in the hidden layer where getting a good target is relatively easy.

To assess this possibility a series of experiments were conducted on the MNIST data set, with the size of the transition matrix $\mathbf{W}_{hh}$ set to 2^6, 2^7, 2^8, 2^9, and 2^{10}. The results of the runs are given in Table 7 and Figure 10. The change in accuracy indicates that increasing the complexity of the TPTT-trained network does not harm its performance. On the contrary, increasing the number of neurons in the hidden layer leads to an increase in the network's accuracy, with the 2^{10} network surpassing the accuracy achieved by the LSTM

network (although this is not a fair comparison provided that the number of parameters of the TPTT network is greater by one order of magnitude).

4. Discussion and Future Work

This work presented *target propagation through time*: a difference target propagation-based algorithm for learning in recurrent neural networks. It was demonstrated that TPTT performs generally better when compared to backpropagation, can be combined with other techniques that look to reduce the impact of vanishing/exploding gradients, and provides an advantage in training deeper recurrent networks. The experiments provide solid evidence that the introduction of target propagation-based updates for the transition matrix provides clear advantage not only on the sequence length but also in terms of convergence speed. The paper also introduced a training scheme where the $\mathbf{W}_{xh}$ updates are computed to minimize the distance between the output of the hidden layer and the local target (i.e. see TPTT_{hx} in Appendix A), thus avoiding the chain rule through $\mathbf{W}_{hh}$. This variant of the network, where the updates for both $\mathbf{W}_{hh}$ and $\mathbf{W}_{xh}$ are based on local targets, performed even better in terms of depth, although it lagged slightly behind on speed of convergence. Both target propagation variants, however, were significantly quicker to converge compared to BPTT.

In addition, there is evidence that in certain aspects TPTT is more biologically plausible compared to Back-Propagation Through Time. This is in agreement with the widely accepted idea that incorporation of additional neuro-biological mechanisms results in enhanced computational abilities (Hassabis et al. 2017; Spoerer et al. 2017; Lotter et al. 2016). Indeed, the feedback connections in the building blocks of TPTT mimic the layout of a neuron with functionally distinct apical and basal dendrites. This neuro-biologically inspired mechanism is the main driver that helps TPTT to outperform other BPTT-based approaches, as it was experimentally confirmed.

Another factor that supports the more biologically plausible nature of TPTT is that this training mechanism is better suited to handle discrete error signals. It has been shown by Lee et al. (2015) that target propagation can be used in networks with discretized transmission between units, and it would be interesting to see if TPTT can be adapted for recurrent networks with discretized layers. This is, however left to future work.

A question that remains open is the impact of orthogonal initialisation on biological plausibility. The literature on biological plausibility of initialisation tricks is unusually scarce. The decision to use orthogonal initialisation was driven by the preference to keep the BPTT and TPTT network architectures as close as possible, so that the impact of architecture-related differences on the learning performance can be confined to the difference between the BPTT and TPTT learning algorithms. It is worth noting that Lee et al. (2015) also use orthogonal initialisation in their presentation of *difference target propagation*. One could speculate that it is in theory possible to have a separate biological process that aligns the weights before the actual learning commences. An example to illustrate this idea could be the work presented in Bengio et al. (2016), where a feedforward pass is used to obtain initialization for a deep Boltzmann machine. In a more general case it could be argued that careful initialisation can be considered a form of pre-training. Furthermore Henaff et al. (2016) demonstrate that different types of initialisation are more or less effective for different

types of tasks. For example, identity initialization should perform better on the Addition task compared to orthogonal initialisation. A more in-depth investigation on the impact of different initialisation techniques in a TPTT context and the biological plausibility of orthogonal initialisation is left for future investigations.

Jaderberg et al. (2017) suggest that target propagation is a more restrictive training scheme, as it relies on local targets that must be generated sequentially, thus a network trained with target propagation must remain update- and backwards-locked. This limitation should be relatively easy to address. Let us consider an unrolled SRN with t_{max} number of time steps ($t_{max} > 3$) and local targets $\hat{\boldsymbol{h}}_t$ ($t \in [1, t_{max}]$). Let k be arbitrary chosen so that $1 < k < t_{max}$. This network can be treated as comprised of two separate computational graphs—the first spanning over hidden units in $[1, k]$ and the second one over $[k+1, t_{max}]$. These two graphs can be treated as separate TPTT networks. Network $[k+1, t_{max}]$ sets its $\hat{\boldsymbol{h}}_{t_{max}}$ using the global SRN error and Equation 3. The first target for network $[1, k]$ ($\hat{\boldsymbol{h}}_k$) can be initialised at random or, if waiting for one sequential setting of targets for network $[k+1, t_{max}]$ is not an issue, it can be computed using

$$\hat{\mathbf{h}}_k = \mathbf{h}_k - \alpha_i * \frac{\partial \hat{\boldsymbol{h}}_{t_{k+1}}}{\partial \mathbf{h}_{t_k}}$$

Under this regime the two networks operate independently, with network $[1, k]$ still providing updates in the generally correct direction, whilst waiting for more precise guidance from the second network. In this case, the two networks can run asynchronously and in theory, they could also run at different speeds. Combined with the asynchronous execution of the $G(\cdot)$ and $F(\cdot)$ phases, this mechanism has the potential of completely unlocking the TPTT network so that it can be fully trained in parallel. Moreover, there is no need to limit the separation of the network to only two subgraphs. As long as t_{max} is large enough, the same approach can be applied to split the network into three, four, or more independent TPTT networks. In the extreme case, each subnetwork can be made as small as consisting of two hidden units only—an architecture close to the local representations alignment suggested by Ororbia et al. (2018)

Another investigation related to "unlocking" the network is the effect of removing the approximation of $G(\cdot)$ phase, and relying on some form of random feedback alignment instead. Said modification should be fairly straightforward to implement. It could further simplify the training scheme, and it can provide insights on the performance difference of plastic vs. fixed feedback synaptic weights.

Lee et al. (2015) also demonstrate that injecting random noise is beneficial for stabilizing learning in target propagation. This was experimentally confirmed for the adding problem, where the addition of fixed Gaussian noise did indeed improve the accuracy of the network. Ororbia et al. (2018), however, show that difference target propagation struggles to select good local targets in the upstream layers, leading to a decrease of the global loss in the first epochs, followed by a collapse afterwards. This effect was also observed experimentally in the grid search runs for the MNIST sequence of pixels problem. Ororbia and Mali (2018) suggest an improved variation of difference target propagation called DTP-σ, which introduces an adaptive approach to noise injection scheme where the standard deviation of the noise is a function of the local loss. Future research could look into adapting TPTT to

leverage the DTP-σ modification and verify the positive impact on accuracy in a recurrent context.

Finally, this work provides empirical evidence of the superiority of target propagation over backpropagation in simple recurrent neural networks. The TPTT network does not performs as well as LSTM, however, it was suggested in Section 3.2 that there is a high possibility that the LSTM performance is driven exclusively by the advantages provided by the LSTM architecture. A fair LSTM-based comparison of BPTT and TPTT would require an LSTM model trained with target propagation.

Taking into account the work of Wiseman et al. (2017) and given that TPTT outperforms BPTT in four out of the five problems it was tested on, it would be interesting to see how a TPTT trained network would fare on language modelling data sets. One could easily imagine a modification of TPTT that computes $\boldsymbol{y}_t$ for every $\boldsymbol{h}_t$, and thus can be used for "next character prediction" type of tasks. It would also be interesting to see how a TPTT-trained LSTM network performs in this setting. This investigation, however, is left to future work.

Appendix A. Alternative Update Rule

To distinguish between the two update schemes we refer to the original network configuration as TPTT, and introduce the TPTT_{hx} notation to denote a network where $\mathbf{W}_{hh}$ and $\mathbf{b}_h$ are updated using (10), and $\mathbf{W}_{xh}$ is updated using (13) (i.e. both the hidden and input weight matrices are updated using local targets). The key experiments from Section 3 were repeated and all results are provided below.

The configuration of the TPTT_{hx} network for the synthetic problems was kept identical to the configuration of the network used in Section 3.1—hyperbolic tangent non-linear activation function, 100 neuron in the hidden layer, orthogonal initialisation, and the biases set to zeros. All runs were restricted to 100,000 iterations. An optimal combination of hyperparameters for the TPTT_{hx} network was discovered by running a grid search over $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ for each α. This is also identical to the arrangement used for the network in Section 3.1. A comparison between the maximal depth achieved by the TPTT and TPTT_{hx} networks is given in Table A1. The optimal parameters discovered by the grid search are given in Table A2.

The results in Table A1 suggest that applying target based updates to $\mathbf{W}_{xh}$ is beneficial, as the TPTT_{hx} network outperforms the original TPTT network on all but the adding problem. It could be theorized that because updating $\mathbf{W}_{xh}$ using $\frac{\partial \mathcal{E}}{\partial \mathbf{W}_{xh}}$ mandates a chain rule through $\mathbf{W}_{hh}$, similar instabilities come into play. Removing the chain rule through $\mathbf{W}_{hh}$ from the computation of $\mathbf{W}_{xh}$ therefore improves the overall stability and helps the network achieve convergence at greater depth.

Similar to the results from Section 3.1, the adding problem proved to be the most difficult challenge for the TPTT_{hx} network. Again, the network was on convergence path, but wouldn't complete within the 100,000 iterations limit, hence the learning rates for this specific problem were manually increased to $\alpha_i = 0.02$, $\alpha_f = 0.01$, and $\alpha_g = 0.05$. This allowed the TPTT_{hx} network to achieve performance identical to TPTT, but it still couldn't match the results obtained using backpropagation.

Problem	t_{max}	
	TPTT	TPTT_{hx}
Temporal Order	150	**210**
3-bit Temporal Order	150	**160**
Adding	60	60
Random Permutation	300	**800**

Table A1: Comparison of the maximal depth (sequence length) achieved by the TPTT and TPTT_{hx} models. The TPTT results come from Section 3.1, and the TPTT_{hx} depth is obtained using the same technique (initial sequence length of T=10, which is increased by 10 upon successful solving of the given problem).

Problem	TPTT Network
Temporal Order	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.001$
3-bit Temporal Order	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.001$
Adding	$\alpha_i = 0.5$, $\alpha_f = 0.02$, $\alpha_g = 0.07$
Random Permutation	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.001$

Problem	TPTT_{hx} Network
Temporal Order	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.01$
3-bit Temporal Order	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.01$
Adding	$\alpha_i = 0.2$, $\alpha_f = 0.01$, $\alpha_g = 0.05$
Random Permutation	$\alpha_i = 0.1$, $\alpha_f = 0.01$, $\alpha_g = 0.001$

Table A2: Learning rates used for the TPTT and TPTT_{hx} networks listed in Table A1.

The impact of Gaussian noise in the TPTT_{hx} network was assessed under the same conditions as in Section 3.1.2. After selecting an optimal value for ϵ from $(0, 1]$, the accuracy of the TPTT_{hx} network on the adding problem with $t_{max} = 70$ increased by approximately 6.7%. As with the TPTT network, however, this improvement was not sufficient to allow the network to fulfil the solving criteria. The impact of running the TPTT_{hx} network with and without injected noise is illustrated in Figure A1.

It is interesting to note that contrary to the results in Section 3.1, the grid search did not discover a common set of learning rates that solves the temporal order, 3-bit temporal order, and permutation problems (i.e. a common model). It was also observed, that the convergence speed of the TPTT_{hx} network was on par with the BPTT implementation, and was not exhibiting the faster convergence times achieved by TPTT (see Table 2). This prompted an expansion of the search space towards larger learning rates, which quickly uncovered the set $\alpha_i = 0.1$, $\alpha_f = 0.2$, $\alpha_g = 0.01$ that solves all three problems quicker than BPTT and constitutes a common model. Similarly to the TPTT results, the modified search uncovered other parameter combinations that solve the problems even quicker. These

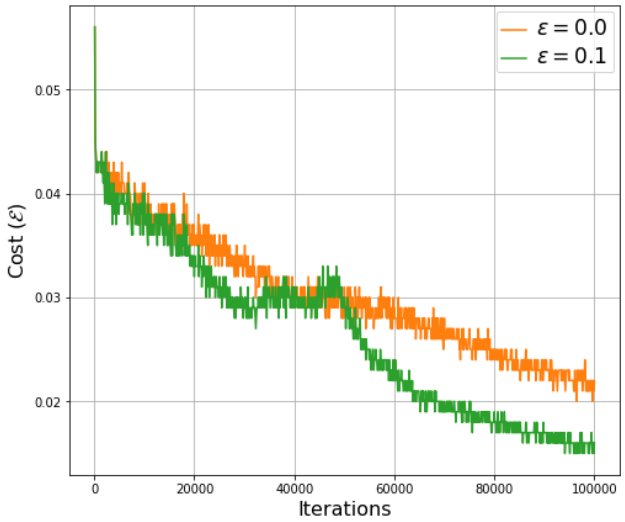

Figure A1: Impact of Gaussian noise on the adding problem with $t_{max} = 70$. The learning parameters are set to $\alpha_i = 0.2$, $\alpha_f = 0.01$, $\alpha_g = 0.05$, and the network is run with $\epsilon = 0.0$ and $\epsilon = 0.1$ for 100,000 iterations. The value of the cost function is averaged over 100 iterations.

Problem	t_{max}	BPTT best	TPTT$_{\text{hx}}$ common model	TPTT$_{\text{hx}}$ best	TPTT$_{\text{hx}}$ parameters
Temporal Order	120	46,500	**5,800**	**5,800**	$\alpha_i = 0.1$, $\alpha_f = 0.2$, $\alpha_g = 0.01$
3-bit Temporal Order	70	56,000	13,800	**12,500**	$\alpha_i = 0.2$, $\alpha_f = 0.2$, $\alpha_g = 0.001$
Addition	90	18,000	-	-	-
Random Permutation	70	12,200	40,300	**100**	$\alpha_i = 0.1$, $\alpha_f = 0.2$, $\alpha_g = 0.1$

Table A3: Difference between BPTT and TPTT$_{\text{hx}}$-training expressed as number of iterations needed until convergence. "BPTT best" provides the results from the fastest converging BPTT models based on the grid search. "TPTT$_{\text{hx}}$ common model" shows the iterations needed by the TPTT$_{\text{hx}}$ model with identical hyper-parameters for all problems ($\alpha_i = 0.1$, $\alpha_f = 0.2$, $\alpha_g = 0.01$). "TPTT$_{\text{hx}}$ best" and "TPTT$_{\text{hx}}$ parameters" show the results from the fastest TPTT$_{\text{hx}}$ models based on the extended grid search and their respective parameters.

combinations and their respective convergence times, measured as number of iterations, are listed in Table A3.

Network	Accuracy %	Parameters
TPTT	54.20	$\alpha_i = 10^{-7}$, $\alpha_f = 10^{-2}$, $\alpha_g = 10^{-8}$
TPTT_{hx}	54.75	$\alpha_i = 10^{-7}$, $\alpha_f = 10^{-2}$, $\alpha_g = 10^{-8}$

Table A4: Accuracy on the MNIST sequence of pixels problem achieved by the TPTT and TPTT_{hx} networks.

The TPTT_{hx} was also tested on the MNIST sequence of pixels problem (see Section 3.2). The grid search performed over the same parameter space selected a combination of learning rates identical to the TPTT network used on the same problem. In addition, the accuracy results shown in Table A4 appear too close to call, therefore the performance of the two schemes on this specific problem can be considered nearly identical.

In conclusion, the empirical results suggest that in terms of sequence length the alternative update rule (i.e. a network that is fully optimised using local targets) outperforms both BPTT and a TPTT network that updates $\mathbf{W}_{xh}$ using the global error directly. Similarly to the original TPTT network, the TPTT_{hx} converges quicker than BPTT, discovers a common model for the synthetic classification problems, and has comparable accuracy on the MNIST sequence of pixels problem.

References

M. Arjovsky, A. Shah, and Y. Bengio. Unitary evolution recurrent neural networks. *CoRR*, abs/1511.06464, 2015.

Y. Bengio. *Artificial Neural Networks and Their Application to Sequence Recognition*. PhD thesis, McGill University, Montreal, Quebec, Canada, 1991.

Y. Bengio. How auto-encoders could provide credit assignment in deep networks via target propagation. *CoRR*, abs/1407.7906, 2014.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, March 1994. ISSN 1045-9227. doi: 10.1109/72.279181. URL `http://dx.doi.org/10.1109/72.279181`.

Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu. Advances in optimizing recurrent networks. *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pages 8624–8628, 2013. ISSN 15206149. doi: 10.1109/ICASSP.2013.6639349.

Y. Bengio, D. H. Lee, J. Bornschein, and Z. Lin. Towards biologically plausible deep learning. *CoRR*, abs/1502.04156, 2015.

Y. Bengio, B. Scellier, O. Bilaniuk, J. Sacramento, and W. Senn. Feedforward initialization for fast inference of deep generative networks is biologically plausible. *CoRR*, abs/1606.01651, 2016.

R. Caruana. Generalization vs. net size, 1993. Neural Information Processing Systems, Tutorial, Denver, CO.

J. Chang, P. Kuo, and Y. Chen. Neuroscience-inspired recurrent network for object recognition. In *2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pages 729–734, Nov 2017. doi: 10.1109/ISPACS.2017.8266572.

F. Crick. The recent excitement about neural networks. *Nature*, 337:129–132, 01 1989. doi: 10.1038/337129a0. URL `http://dx.doi.org/10.1038/337129a0`.

W. M. Czarnecki, G. Swirszcz, M. Jaderberg, S. Osindero, O. Vinyals, and K. Kavukcuoglu. Understanding synthetic gradients and decoupled neural interfaces. In Precup and Teh (2017), pages 904–912. URL `http://proceedings.mlr.press/v70/czarnecki17a.html`.

W. De Mulder, S. Bethard, and M. F. Moens. A survey on the application of recurrent neural networks to statistical language modeling. *Comput. Speech Lang.*, 30(1):61–98, March 2015. ISSN 0885-2308. doi: 10.1016/j.csl.2014.09.005. URL `http://dx.doi.org/10.1016/j.csl.2014.09.005`.

J. Drewes, G. Goren, W. Zhu, and J. H. Elder. Recurrent processing in the formation of shape percepts. *Journal of Neuroscience*, 36(1):185–192, 2016. ISSN 0270-6474. doi: 10.1523/JNEUROSCI.2347-15.2016. URL `http://www.jneurosci.org/content/36/1/185`.

J. L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

A. Graves. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711, 2012. URL `http://arxiv.org/abs/1211.3711`.

P. J. Grother. NIST special database 19 handprinted forms and character database. Available: `https://www.nist.gov/sites/default/files/documents/srd/nistsd19.pdf`. National Institute of Standards and Technology, Gaithersburg, MD., 1995.

D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick. Review Neuroscience-Inspired Artificial Intelligence. *Neuron*, 95(2):245–258, 2017. ISSN 0896-6273. doi: 10.1016/j.neuron.2017.06.011. URL `http://dx.doi.org/10.1016/j.neuron.2017.06.011`.

G. Heigold, I. Moreno, S. Bengio, and N. Shazeer. End-to-end text-dependent speaker verification. *CoRR*, abs/1509.08062, 2015.

M. Henaff, A. Szlam, and Y. LeCun. Orthogonal rnns and long-memory tasks. *CoRR*, abs/1602.06662, 2016.

G. E. Hinton, T. J. Sejnowski, and D. H. Ackley. Boltzmann machines: Constraint satisfaction networks that learn. Technical Report CMU-CS-84-119, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1984.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

M. Jaderberg, W. M. Czarnecki, S. Osindero, O. Vinyals, A. Graves, D. Silver, and K. Kavukcuoglu. Decoupled neural interfaces using synthetic gradients. In Precup and Teh (2017), pages 1627–1635. URL http://proceedings.mlr.press/v70/jaderberg17a.html.

M. I. Jordan. Serial order: A parallel, distributed processing approach. In Jeffrey L. Elman and David E. Rumelhart, editors, *Advances in Connectionist Theory: Speech*. Erlbaum, Hillsdale, NJ, 1989.

J.H. Kaas and D. C. Lyon. Chapter 18 visual cortex organization in primates: theories of v3 and adjoining visual areas. In *Vision: From Neurons to Cognition*, volume 134 of *Progress in Brain Research*, pages 285 – 295. Elsevier, 2001. doi: https://doi.org/10.1016/S0079-6123(01)34019-0. URL http://www.sciencedirect.com/science/article/pii/S0079612301340190.

A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):664–676, April 2017. ISSN 0162-8828. doi: 10.1109/TPAMI.2016.2598339. URL https://doi.org/10.1109/TPAMI.2016.2598339.

D. J. Kravitz, K. S. Saleem, C. I. Baker, L. G. Ungerleider, and M. Mishkin. The ventral visual pathway: an expanded neural framework for the processing of object quality. *Trends Cogn. Sci. (Regul. Ed.)*, 17(1):26–49, Jan 2013.

V. Lamme and P. R. Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in Neurosciences*, 23(11):571 – 579, 2000. ISSN 0166-2236. doi: https://doi.org/10.1016/S0166-2236(00)01657-X. URL http://www.sciencedirect.com/science/article/pii/S016622360001657X.

S. Lawrence, C. Lee Giles, and Ah. Chung Tsoi. Lessons in neural network training: Overfitting may be harder than expected. In *Proceedings of the National Conference on Artificial Intelligence*, pages 540–545, 01 1997.

Q. V. Le, N. Jaitly, and G. E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *CoRR*, abs/1504.00941, 2015. URL http://arxiv.org/abs/1504.00941.

Y. LeCun. Learning process in an asymmetric threshold network. In E. Bienenstock, F. Fogelman Soulié, and G. Weisbuch, editors, *Disordered Systems and Biological Organization*, pages 233–240, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

D. H. Lee, S. Zhang, A. Fischer, and Y. Bengio. Difference target propagation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9284:498–515, 2015. ISSN 16113349. doi: 10.1007/978-3-319-23528-8_31.

T. P. Lillicrap, Daniel Cownden, Douglas B. Tweed, and Colin J. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7, Nov 2016. URL `http://dx.doi.org/10.1038/ncomms13276`.

P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2873–2879, 2016. URL `http://www.ijcai.org/Abstract/16/408`.

W. Lotter, G. Kreiman, and D. D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *CoRR*, abs/1605.08104, 2016.

H. Luo, J. Fu, and J.R. Glass. Bidirectional backpropagation: Towards biologically plausible error signal transmission in neural networks. *CoRR*, abs/1702.07097, 2017.

W. Maas. Networks of spiking neurons: The third generation of neural network models. *Trans. Soc. Comput. Simul. Int.*, 14(4):1659–1671, December 1997. ISSN 0740-6797. URL `http://dl.acm.org/citation.cfm?id=281543.281637`.

J. H. Maunsell and D. C. van Essen. The connections of the middle temporal visual area (MT) and their relationship to a cortical hierarchy in the macaque monkey. *J. Neurosci.*, 3(12):2563–2586, Dec 1983.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL `http://arxiv.org/abs/1301.3781`.

M. Minsky. Steps toward Artificial Intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961. ISSN 0096-8390. doi: 10.1109/JRPROC.1961.287775. URL `http://ieeexplore.ieee.org/document/4066245/`.

H. Nakamura, R. Gattass, R. Desimone, and L. G. Ungerleider. The modular organization of projections from areas V1 and V2 to areas V4 and TEO in macaques. *J. Neurosci.*, 13(9):3681–3691, Sep 1993.

Y. Nesterov. A method of solving a convex programming problem with convergence rate o(1/sqr(k)). *Soviet Mathematics Doklady*, 27, 1983.

A. G. II Ororbia and A. Mali. Biologically motivated algorithms for propagating local target representations. *CoRR*, abs/1805.11703, 2018. URL `http://arxiv.org/abs/1805.11703`.

A. G. II Ororbia, A. Mali, D. Kifer, and C. L. Giles. Conducting credit assignment by aligning local representations. *CoRR*, abs/1803.01834, 2018. URL `http://arxiv.org/abs/1803.01834`.

R. Pascanu, T. Mikolov, and Y. Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

P. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 82–90, Bejing, China, 22–24 Jun 2014. PMLR. URL `http://proceedings.mlr.press/v32/pinheiro14.html`.

J. B. Pollack. Language induction by phase transition in dynamical recognizers. In R. Lippmann, J. E. Moody, and D.S. Touretzky, editors, *Advances in Neural Information Processing Systems 3, [NIPS Conference, Denver, Colorado, USA, November 26-29, 1990]*, pages 619–626. Morgan Kaufmann, 1990. ISBN 1-55860-184-8. URL `http://papers.nips.cc/paper/298-language-induction-by-phase-transition-in-dynamical-recognizers`.

D. Precup and Y.W. Teh, editors. *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, 2017. PMLR. URL `http://jmlr.org/proceedings/papers/v70/`.

G. V. Puskorius, L. A. Feldkamp, and L. I. Davis. Dynamic neural network methods applied to on-vehicle idle speed control. *Proceedings of the IEEE*, 84(10):1407–1420, Oct 1996. ISSN 0018-9219. doi: 10.1109/5.537107.

P. R. Roelfsema and A. Holtmaat. Control of synaptic plasticity in deep cortical networks. *Nature Reviews Neuroscience*, 19(3):166–180, 2 2018. ISSN 1471-003X. doi: 10.1038/nrn.2018.6.

P. R. Roelfsema and A. R. Van Ooyen. Attention-gated reinforcement learning of internal representations for classification. *Neural Comput.*, 17(10):2176–2214, October 2005. ISSN 0899-7667. doi: 10.1162/0899766054615699. URL `http://dx.doi.org/10.1162/0899766054615699`.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 0-262-68053-X. URL `http://dl.acm.org/citation.cfm?id=104279.104293`.

R. Socher, A. Karpathy, Q. V. Le, C. D. Manning, and A. Y. Ng. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2:207–218, 2014.

R. Soltani and H. Jiang. Higher order recurrent neural networks. *CoRR*, abs/1605.00064, 2016. URL `http://arxiv.org/abs/1605.00064`.

C. Spoerer, P. McClure, and N. Kriegeskorte. Recurrent convolutional neural networks: A better model of biological object recognition under occlusion. *bioRxiv*, 2017. doi: 10.1101/133330. URL `https://www.biorxiv.org/content/early/2017/05/02/133330`.

M. Spratling. Cortical region interactions and the functional role of apical dendrites. *Behavioral and cognitive neuroscience reviews*, 1(3):219–228, 2002.

I. Sutskever, J. Martens, and G. E. Hinton. Generating text with recurrent neural networks. In *ICML*, pages 1017–1024. Omnipress, 2011.

A. Timmaraju and V. Khanna. Sentiment analysis on movie reviews using recursive and recurrent neural network architectures. 2015.

D. C. Van Essen, W. T. Newsome, J. H. Maunsell, and J. L. Bixby. The projections from striate cortex (V1) to areas V2 and V3 in the macaque monkey: asymmetries, areal boundaries, and patchy connections. *J. Comp. Neurol.*, 244(4):451–480, Feb 1986.

P. J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

S. Wiseman, S. Chopra, M. Ranzato, A. Szlam, R. Sun, S. Chintala, and N. Vasilache. Training Language Models Using Target-Propagation. *ArXiv e-prints*, February 2017.

K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

D. Yogatama, C. Dyer, W. Ling, and P. Blunsom. Generative and Discriminative Text Classification with Recurrent Neural Networks. *ArXiv e-prints*, March 2017.