# Recurrent Highway Networks

**Julian Georg Zilly**[*] **Rupesh Kumar Srivastava**[*][†] **Jan Koutník**[†] **Jürgen Schmidhuber**
ETH Zürich
jzilly@ethz.ch
The Swiss AI Lab IDSIA / USI / SUPSI
{rupesh, hkou, juergen}@idsia.ch

## Abstract

Many sequential processing tasks require complex nonlinear transition functions from one step to the next. However, recurrent neural networks with "deep" transition functions remain difficult to train, even when using Long Short-Term Memory (LSTM) networks. We introduce a novel theoretical analysis of recurrent networks based on Geršgorin's circle theorem that illuminates several modeling and optimization issues and improves our understanding of the LSTM cell. Based on this analysis we propose Recurrent Highway Networks, which are deep not only in time but also in space, extending the LSTM architecture to larger step-to-step transition depths. Experiments demonstrate that the proposed architecture results in powerful and efficient models benefiting from up to 10 layers in the recurrent transition. On the Penn Treebank language modeling corpus, a single network outperforms all previous ensemble results with a perplexity of 66.0 on the test set. On the larger Hutter Prize Wikipedia dataset, a single network again significantly outperforms all previous results with an entropy of 1.32 bits per character on the test set.

## 1 Introduction

Network depth is of central importance in the resurgence of neural networks as a powerful machine learning paradigm [1]. Theoretical evidence indicates that deeper networks can be exponentially more efficient at representing certain function classes (see e.g. [2] and references therein). Due to their sequential nature, Recurrent Neural Networks (RNNs; 3–5) have long credit assignment paths and so are *deep in time*. However, certain internal function mappings in modern RNNs composed of units grouped in layers usually do not take advantage of depth [6]. For example, the state update from one time step to the next is typically modeled using a single non-linear transformation.

Unfortunately, increased depth represents a challenge when neural network parameters are optimized by means of error backpropagation [7–9]. Deep networks suffer from what are commonly referred to as the vanishing and exploding gradient problems [10–12], since the magnitude of the gradients may shrink or explode exponentially during backpropagation. These training difficulties were first studied in the context of standard RNNs where the depth through time is proportional to the length of input sequence, which may have arbitrary size. The widely used Long Short-Term Memory (LSTM; 13, 14) architecture was introduced to specifically address the problem of vanishing/exploding gradients for recurrent networks.

As computational resources grow and complex learning problems are tackled, the vanishing gradient problem also becomes a limitation when training very deep feedforward networks. The recently introduced *Highway Layers* [15] based on the LSTM cell address this limitation enabling the training of networks even with hundreds of stacked layers. These layers have been used to improve performance in speech recognition [16] and language modeling [17], and their variants called *Residual networks* have been widely useful for many computer vision problems[18].

[*] These authors contributed equally.

[†] These authors are now affiliated with NNAISENSE SA, Lugano, Switzerland.

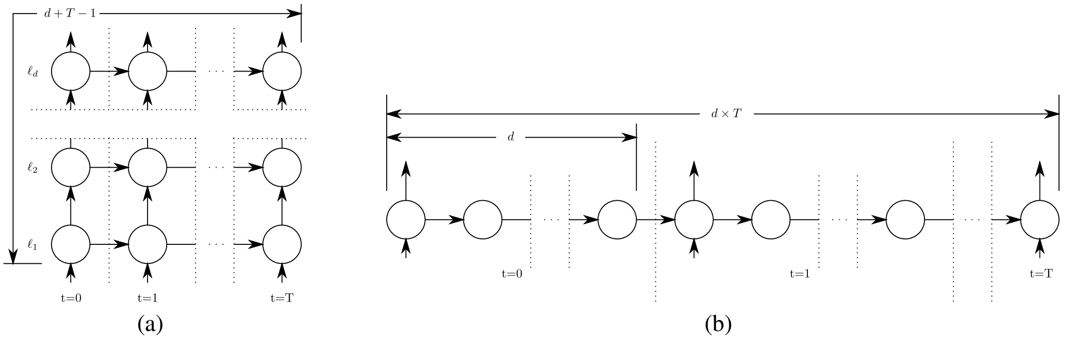


Figure 1: Comparison of (a) stacked RNN with depth $d$ and (b) Deep Transition RNN of recurrence depth $d$, both operating on a sequence of $T$ time steps. The longest credit assignment path between hidden states $T$ time steps apart are longer (deeper) in Deep Transition RNN.

In this paper we first provide a new mathematical analysis of standard RNNs which offers a deeper understanding of various recurrent network architectures. Based on these insights, we introduce LSTM networks that have long credit assignment paths not just in time but also long in space (per time step), called *Recurrent Highway Networks* or *RHNs*. They enable the use of substantially more powerful and trainable sequential models efficiently, and significantly outperform existing architectures on widely used benchmarks.

## 2 Related Work on Deep Recurrent Transitions

In recent years, a common method of utilizing the computational advantages of depth in recurrent networks is *stacking* recurrent layers [19], which is analogous to using multiple hidden layers in feedforward networks. Training stacked RNNs naturally requires credit assignment across both space and time which is difficult in practice. These problems have been recently addressed by architectures utilizing LSTM-based transformations for stacking [16, 20].

A general method to increase the depth of the step-to-step recurrent state transition (the *recurrence depth*) is to let an RNN tick for several *micro time steps* per step of the sequence [21–23]. This method can adapt the recurrence depth to the problem, but the RNN has to learn by itself which parameters to use for memories of previous events and which for standard deep nonlinear processing. It is notable that while Graves [23] reported improvements on simple algorithmic tasks using this method, no performance improvements were obtained on real world data.

Pascanu et al. [6] proposed to increase the recurrence depth by adding multiple non-linear layers to the recurrent transition, resulting in Deep Transition RNNs (DT-RNNs) and Deep Transition RNNs with Skip connections (DT(S)-RNNs). While being powerful in principle, these architectures are seldom used due to exacerbated gradient propagation issues resulting from extremely long credit assignment paths[3]. In related work Chung et al. [24] added extra connections between all states across consecutive time steps in a stacked RNN, which also increases recurrence depth. However, their model requires many extra connections with increasing depth, gives only a fraction of states access to the largest depth, and still faces gradient propagation issues along the longest paths.

Compared to stacking recurrent layers, increasing the recurrence depth can add significantly higher modeling power to an RNN. Figure 1 illustrates that stacking $d$ RNN layers allows a maximum credit assignment path length (number of non-linear transformations) of $d - 1 + T$ between hidden states which are $T$ time steps apart, while a recurrence depth of $d$ enables a maximum path length of $d \times T$. While this allows greater power and efficiency using larger depths, it also explains why such architectures are much more difficult to train compared to stacked RNNs. In the next sections, we address this problem head on by focusing on the key mechanisms of the LSTM and using those to design RHNs, which do not suffer from the above difficulties.

[3] We compare optimization of our proposed architecture to these models in subsection 5.1

# 3 Revisiting Gradient Flow in Recurrent Networks

Let $\mathcal{L}$ denote the total loss for an input sequence of length $T$. Let $\mathbf{x}^{[t]} \in \mathbb{R}^m$ and $\mathbf{y}^{[t]} \in \mathbb{R}^n$ represent the output of a standard RNN at time $t$, $\mathbf{W} \in \mathbb{R}^{n \times m}$ and $\mathbf{R} \in \mathbb{R}^{n \times n}$ the input and recurrent weight matrices, $\mathbf{b} \in \mathbb{R}^n$ a bias vector and $f$ a point-wise non-linearity. Then $\mathbf{y}^{[t]} = f(\mathbf{W}\mathbf{x}^{[t]} + \mathbf{R}\mathbf{y}^{[t-1]} + \mathbf{b})$ describes the dynamics of a standard RNN. The derivative of the loss $\mathcal{L}$ with respect to parameters $\theta$ of a network can be expanded using the chain rule:

$$\frac{d\mathcal{L}}{d\theta} = \sum_{1 \leq t_2 \leq T} \frac{d\mathcal{L}^{[t_2]}}{d\theta} = \sum_{1 \leq t_2 \leq T} \sum_{1 \leq t_1 \leq t_2} \frac{\partial \mathcal{L}^{[t_2]}}{\partial \mathbf{y}^{[t_2]}} \frac{\partial \mathbf{y}^{[t_2]}}{\partial \mathbf{y}^{[t_1]}} \frac{\partial \mathbf{y}^{[t_1]}}{\partial \theta}. \tag{1}$$

The Jacobian matrix $\frac{\partial \mathbf{y}^{[t_2]}}{\partial \mathbf{y}^{[t_1]}}$, the key factor for the transport of the error from time step $t_2$ to time step $t_1$, is obtained by chaining the derivatives across all time steps:

$$\frac{\partial \mathbf{y}^{[t_2]}}{\partial \mathbf{y}^{[t_1]}} := \prod_{t_1 < t \leq t_2} \frac{\partial \mathbf{y}^{[t]}}{\partial \mathbf{y}^{[t-1]}} = \prod_{t_1 < t \leq t_2} \mathbf{R}^\top \mathrm{diag}\big[f'(\mathbf{R}\mathbf{y}^{[t-1]})\big], \tag{2}$$

where the input and bias have been omitted for simplicity. We can now obtain conditions for the gradients to vanish [10–12]. Let $\mathbf{A} := \frac{\partial \mathbf{y}^{[t]}}{\partial \mathbf{y}^{[t-1]}}$ be the temporal Jacobian, $\gamma$ be a maximal bound on $f'(\mathbf{R}\mathbf{y}^{[t-1]})$ and $\sigma_{max}$ be the largest singular value of $\mathbf{R}^\top$. Then the norm of the Jacobian satisfies:

$$\|\mathbf{A}\| \leq \left\|\mathbf{R}^\top\right\| \left\|\mathrm{diag}\big[f'(\mathbf{R}\mathbf{y}^{[t-1]})\big]\right\| \leq \gamma \sigma_{max}, \tag{3}$$

which together with (2) provides the conditions for vanishing gradients ($\gamma\sigma_{max} < 1$). Note that $\gamma$ depends on the activation function $f$, e.g. $|tanh'(x)| \leq 1$, $|\sigma'(x)| \leq \frac{1}{4}, \forall x \in \mathbb{R}$, where $\sigma$ is a logistic sigmoid. Similarly, we can show that if the spectral radius $\rho$ of $\mathbf{A}$ is greater than 1, exploding gradients will emerge since $\|\mathbf{A}\| \geq \rho$.

This description of the problem in terms of largest singular values or the spectral radius sheds light on boundary conditions for vanishing and exploding gradients yet does not illuminate how the eigenvalues are distributed overall. By applying the Geršgorin circle theorem we are able to provide further insight into this problem.

**Geršgorin circle theorem (GCT) [25]:** *For any square matrix* $\mathbf{A} \in \mathbb{R}^{n \times n}$,

$$\mathrm{spec}(\mathbf{A}) \subset \bigcup_{i \in \{1,\dots,n\}} \left\{ \lambda \in \mathbb{C} | \, \|\lambda - a_{ii}\|_{\mathbb{C}} \leq \sum_{j=1, j \neq i}^{n} |a_{ij}| \right\}, \tag{4}$$

i.e., the eigenvalues of matrix $\mathbf{A}$, comprising the spectrum of $\mathbf{A}$, are located within the union of the complex circles centered around the diagonal values $a_{ii}$ of $\mathbf{A}$ with radius $\sum_{j=1, j\neq i}^{n} |a_{ij}|$ equal to the sum of the absolute values of the non-diagonal entries in each row of $\mathbf{A}$. Two example Geršgorin circles referring to differently initialized RNNs are depicted in Figure 2.

Using GCT we can understand the relationship between the entries of $\mathbf{R}$ and the possible locations of the eigenvalues of the Jacobian. Shifting the diagonal values $a_{ii}$ shifts the possible locations of eigenvalues. Having large off-diagonal entries will allow for a large spread of eigenvalues. Small off-diagonal entries yield smaller radii and thus a more confined distribution of eigenvalues around the diagonal entries $a_{ii}$.

Let us assume that matrix $\mathbf{R}$ is initialized with a zero-mean Gaussian distribution. We can then infer the following:

- If the values of $\mathbf{R}$ are initialized with a standard deviation close to 0, then the spectrum of $\mathbf{A}$, which is largely dependent on $\mathbf{R}$, is also initially centered around 0. An example of a Geršgorin circle that could then be corresponding to a row of $\mathbf{A}$ is circle (1) in Figure 2. The magnitude of most of $\mathbf{A}$'s eigenvalues $|\lambda_i|$ are initially likely to be substantially smaller than 1. Additionally, employing the commonly used $\mathrm{L}_1/\mathrm{L}_2$ weight regularization will also limit the magnitude of the eigenvalues.

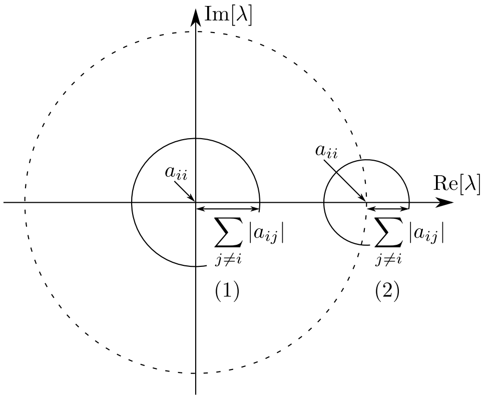


Figure 2: Illustration of the Geršgorin circle theorem. Two Geršgorin circles are centered around their diagonal entries $a_{ii}$. The corresponding eigenvalues lie within the radius of the sum of absolute values of non-diagonal entries $a_{ij}$. Circle $(1)$ represents an exemplar Geršgorin circle for an RNN initialized with small random values. Circle $(2)$ represents the same for an RNN with identity initialization of the diagonal entries of the recurrent matrix and small random values otherwise. The dashed circle denotes the unit circle of radius 1.

- Alternatively, if entries of $\mathbf{R}$ are initialized with a large standard deviation, the radii of the Geršgorin circles corresponding to $\mathbf{A}$ increase. Hence, $\mathbf{A}$'s spectrum may possess eigenvalues with norms greater 1 resulting in exploding gradients. As the radii are summed over the size of the matrix, larger matrices will have an associated larger circle radius. In consequence, larger matrices should be initialized with correspondingly smaller standard deviations to avoid exploding gradients.

In general, unlike variants of LSTM, other RNNs have no direct mechanism to rapidly regulate their Jacobian eigenvalues *across time steps*, which can be efficient and necessary for complex sequence processing. Le et al. [26] proposed to initialize $\mathbf{R}$ with an identity matrix and small random values on the off-diagonals. This changes the situation depicted by GCT – the result of the identity initialization is indicated by circle (2) in Figure 2. Initially, since $a_{ii} = 1$, the spectrum described in GCT is centered around 1, ensuring that gradients are less likely to vanish. However, this is not a flexible remedy. During training some eigenvalues can easily become larger than one, resulting in exploding gradients. We conjecture that due to this reason, extremely small learning rates were used by Le et al. [26].

## 4 Recurrent Highway Networks (RHN)

Highway layers [27] enable easy training of very deep feedforward networks through the use of adaptive computation. Let $\mathbf{h} = H(\mathbf{x}, \mathbf{W}_H), \mathbf{t} = T(\mathbf{x}, \mathbf{W}_T), \mathbf{c} = C(\mathbf{x}, \mathbf{W}_C)$ be outputs of nonlinear transforms $H, T$ and $C$ with associated weight matrices (including biases) $\mathbf{W}_{H,T,C}$. $T$ and $C$ typically utilize a sigmoid ($\sigma$) nonlinearity and are referred to as the *transform* and the *carry* gates since they regulate the passing of the *transformed* input via $H$ or the *carrying* over of the original input $\mathbf{x}$. The Highway layer computation is defined as

$$\mathbf{y} = \mathbf{h} \cdot \mathbf{t} + \mathbf{x} \cdot \mathbf{c}, \tag{5}$$

where "·" denotes element-wise multiplication.

Recall that the recurrent state transition in a standard RNN is described by $\mathbf{y}^{[t]} = f(\mathbf{W}\mathbf{x}^{[t]} + \mathbf{R}\mathbf{y}^{[t-1]} + \mathbf{b})$. We propose to construct a Recurrent Highway Network (RHN) layer with one or multiple Highway layers in the recurrent state transition (equal to the desired recurrence depth). Formally, let $\mathbf{W}_{H,T,C} \in \mathbb{R}^{n \times m}$ and $\mathbf{R}_{H_\ell,T_\ell,C_\ell} \in \mathbb{R}^{n \times n}$ represent the weights matrices of the $H$ nonlinear transform and the $T$ and $C$ gates at layer $\ell \in \{1, \ldots, L\}$. The biases are denoted by $\mathbf{b}_{H_\ell,T_\ell,C_\ell} \in \mathbb{R}^n$ and let $\mathbf{s}_\ell$ denote the intermediate output at layer $\ell$ with $\mathbf{s}_0^{[t]} = \mathbf{y}^{[t-1]}$. Then an RHN

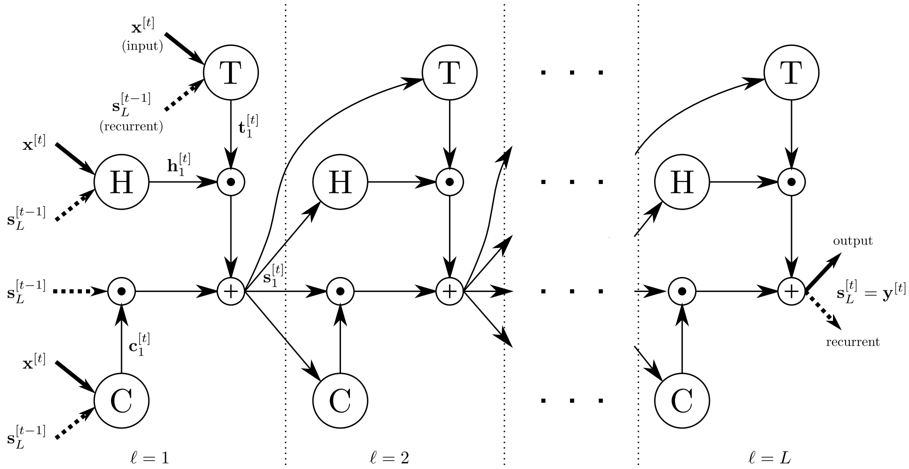


Figure 3: Schematic showing computation within an RHN layer inside the recurrent loop. Vertical dashed lines delimit stacked Highway layers. Horizontal dashed lines imply the extension of the recurrence depth by stacking further layers. $H$, $T$ & $C$ are the transformations described in equations 7, 8 and 9, respectively.

layer with a recurrence depth of $L$ is described by

$$\mathbf{s}_\ell^{[t]} = \mathbf{h}_\ell^{[t]} \cdot \mathbf{t}_\ell^{[t]} + \mathbf{s}_{\ell-1}^{[t]} \cdot \mathbf{c}_\ell^{[t]}, \tag{6}$$

where

$$\begin{aligned}
\mathbf{h}_\ell^{[t]} &= tanh(\mathbf{W}_H \mathbf{x}^{[t]} \mathbb{I}_{\{\ell=1\}} + \mathbf{R}_{H_\ell} \mathbf{s}_{\ell-1}^{[t]} + \mathbf{b}_{H_\ell}), && (7)\\
\mathbf{t}_\ell^{[t]} &= \quad \sigma(\mathbf{W}_T \mathbf{x}^{[t]} \mathbb{I}_{\{\ell=1\}} + \mathbf{R}_{T_\ell} \mathbf{s}_{\ell-1}^{[t]} + \mathbf{b}_{T_\ell}), && (8)\\
\mathbf{c}_\ell^{[t]} &= \quad \sigma(\mathbf{W}_C \mathbf{x}^{[t]} \mathbb{I}_{\{\ell=1\}} + \mathbf{R}_{C_\ell} \mathbf{s}_{\ell-1}^{[t]} + \mathbf{b}_{C_\ell}), && (9)
\end{aligned}$$

and $\mathbb{I}_{\{\}}$ is the indicator function.

A schematic illustration of the RHN computation graph is shown in Figure 3. The output of the RHN layer is the output of the $L^{\text{th}}$ Highway layer i.e. $\mathbf{y}^{[t]} = \mathbf{s}_L^{[t]}$.

Note that $\mathbf{x}^{[t]}$ is directly transformed only by the first Highway layer ($\ell = 1$) in the recurrent transition[1] and for this layer $\mathbf{s}_{\ell-1}^{[t]}$ is the RHN layer's output of the previous time step. Subsequent Highway layers only process the outputs of the previous layers. Dotted vertical lines in Figure 3 separate multiple Highway layers in the recurrent transition.

For conceptual clarity, it is important to observe that an RHN layer with $L = 1$ is essentially a basic variant of an LSTM layer. Similar to other variants such as GRU [28] and those studied by Greff et al. [29] and Jozefowicz et al. [30], it retains the essential components of the LSTM – multiplicative gating units controlling the flow of information through self-connected additive cells. However, an RHN layer naturally extends to $L > 1$, extending the LSTM to model far more complex state transitions. Similar to Highway and LSTM layers, other variants can be constructed without changing the basic principles, for example by fixing one or both of the gates to always be *open*, or coupling the gates as done for the experiments in this paper.

The simpler formulation of RHN layers allows for an analysis similar to standard RNNs based on GCT. Omitting the inputs and biases, the temporal Jacobian $\mathbf{A} = \partial \mathbf{y}^{[t]} / \partial \mathbf{y}^{[t-1]}$ for an RHN layer with recurrence depth of 1 (such that $\mathbf{y}^{[t]} = \mathbf{h}^{[t]} \cdot \mathbf{t}^{[t]} + \mathbf{y}^{[t-1]} \cdot \mathbf{c}^{[t]}$) is given by

$$\mathbf{A} = \mathrm{diag}(\mathbf{c}^{[t]}) + \mathbf{H}'\mathrm{diag}(\mathbf{t}^{[t]}) + \mathbf{C}'\mathrm{diag}(\mathbf{y}^{[t-1]}) + \mathbf{T}'\mathrm{diag}(\mathbf{h}^{[t]}), \tag{10}$$

[1] This is not strictly necessary, but simply a convenient choice.

where

$$\mathbf{H}' = \mathbf{R}_H^\top \mathrm{diag}\big[tanh'(\mathbf{R}_H \mathbf{y}^{[t-1]})\big], \tag{11}$$

$$\mathbf{T}' = \mathbf{R}_T^\top \mathrm{diag}\big[\sigma'(\mathbf{R}_T \mathbf{y}^{[t-1]})\big], \tag{12}$$

$$\mathbf{C}' = \mathbf{R}_C^\top \mathrm{diag}\big[\sigma'(\mathbf{R}_C \mathbf{y}^{[t-1]})\big], \tag{13}$$

and has a spectrum of:

$$\mathrm{spec}(\mathbf{A}) \subset \bigcup_{i \in \{1,\dots,n\}} \Big\{ \lambda \in \mathbb{C} \Big| \Big\| \lambda - \mathbf{c}_i^{[t]} - \mathbf{H}'_{ii}\mathbf{t}_i^{[t]} - \mathbf{C}'_{ii}\mathbf{y}_i^{[t-1]} - \mathbf{T}'_{ii}\mathbf{h}_i^{[t]} \Big\|_{\mathbb{C}} \leq \sum_{j=1, j \neq i}^{n} \big| \mathbf{H}'_{ij}\mathbf{t}_i^{[t]} + \mathbf{C}'_{ij}\mathbf{y}_i^{[t-1]} + \mathbf{T}'_{ij}\mathbf{h}_i^{[t]} \big| \Big\}. \tag{14}$$

Equation 14 captures the influence of the gates on the eigenvalues of $\mathbf{A}$. Compared to the situation for standard RNN, it can be seen that an RHN layer has more flexibility in adjusting the centers and radii of the Geršgorin circles. In particular, two limiting cases can be noted. If all carry gates are fully open and transform gates are fully closed, we have $\mathbf{c} = \mathbf{1}_n, \mathbf{t} = \mathbf{0}_n$ and $\mathbf{T}' = \mathbf{C}' = \mathbf{0}_{n \times n}$ (since $\sigma$ is saturated). This results in

$$\mathbf{c} = \mathbf{1}_n, \quad \mathbf{t} = \mathbf{0}_n \Rightarrow \lambda_i = 1 \quad \forall i \in \{1, \dots, n\}, \tag{15}$$

i.e. all eigenvalues are set to 1 since the Geršgorin circle radius is shrunk to 0 and each diagonal entry is set to $\mathbf{c}_i = 1$. In the other limiting case, if $\mathbf{c} = \mathbf{0}_n$ and $\mathbf{t} = \mathbf{1}_n$ then the eigenvalues are simply those of $\mathbf{H}'$. As the gates vary between 0 and 1, each of the eigenvalues of $\mathbf{A}$ can be dynamically adjusted to any combination of the above limiting behaviors.

The key takeaways from the above analysis are as follows. Firstly, GCT allows us to observe the behavior of the full spectrum of the temporal Jacobian, and the effect of gating units on it. We expect that for learning multiple temporal dependencies from real-world data efficiently, *it is not sufficient to avoid vanishing and exploding gradients*. The gates in RHN layers provide a more versatile setup for *dynamically* remembering, forgetting and transforming information compared to standard RNNs. Secondly, it becomes clear that through their effect on the behavior of the Jacobian, highly non-linear gating functions can facilitate learning through rapid and precise regulation of the network dynamics. Depth is a widely used method to add expressive power to functions, motivating us to use multiple layers of $H$, $T$ and $C$ transformations. In this paper we opt for extending RHN layers to $L > 1$ using Highway layers in favor of simplicity and ease of training. However, we expect that in some cases stacking plain layers for these transformations can also be useful. Finally, the analysis of the RHN layer's flexibility in controlling its spectrum furthers our theoretical understanding of LSTM and Highway networks and their variants. For feedforward Highway networks, the Jacobian of the layer transformation ($\partial \mathbf{y} / \partial \mathbf{x}$) takes the place of the temporal Jacobian in the above analysis. Each Highway layer allows increased flexibility in controlling how various components of the input are transformed or carried. This flexibility is the likely reason behind the performance improvement from Highway layers even in cases where network depth is not high [17].

# 5 Experiments

**Setup:** In this work, the carry gate was coupled to the transform gate by setting $C(\cdot) = \mathbf{1}_n - T(\cdot)$ similar to the suggestion for Highway networks. This coupling is also used by the GRU recurrent architecture. It reduces model size for a fixed number of units and prevents an unbounded blow-up of state values leading to more stable training, but imposes a modeling bias which may be sub-optimal for certain tasks [29, 30]. An output non-linearity similar to LSTM networks could alternatively be used to combat this issue. For optimization and Wikipedia experiments, we bias the transform gates towards being closed at the start of training. All networks use a single hidden RHN layer since we are only interested in studying the influence of recurrence depth, and not of stacking multiple layers, which is already known to be useful. Detailed configurations for all experiments are included in the supplementary material. Source code for the experiments is available at `https://github.com/julian121266/RecurrentHighwayNetworks`.

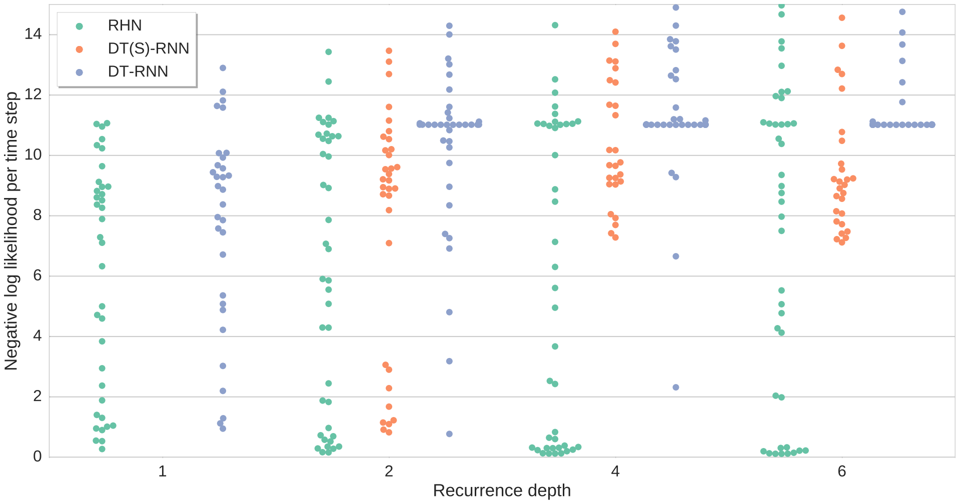


Figure 4: Swarm plot of optimization experiment results for various architectures for different depths on next step prediction on the JSB Chorales dataset. Each point is the result of optimization using a random hyperparameter setting. The number of network parameters increases with depth, but is kept the same across architectures for each depth. For architectures other than RHN, the random search was unable to find good hyperparameters when depth increased.

**Regularization of RHNs:** Like all RHNs, suitable regularization can be essential for obtaining good generalization with RHNs in practice. We adopt the regularization technique proposed by Gal [31], which is an interpretation of dropout based on approximate variational inference. RHNs regularized by this technique are referred to as variational RHNs. Additionally, for the word-level language modeling task, we report results both with and without weight-tying (WT) of input and output mappings [32] for fair comparisons.

## 5.1 Optimization

RHN is an architecture designed to enable the optimization of recurrent networks with deep transitions. Therefore, the primary experimental verification we seek is whether RHNs with higher recurrence depth are easier to optimize compared to other alternatives, preferably using simple gradient based methods.

We compare optimization of RHNs to DT-RNNs and DT(S)-RNNs [6]. Networks with recurrence depth of 1, 2, 4 and 6 are trained for next step prediction on the JSB Chorales polyphonic music prediction dataset [33]. Network sizes are chosen such that the total number of network parameters increases as the recurrence depth increases, but remains the same across architectures. A hyperparameter search is then conducted for SGD-based optimization of each architecture and depth combination for fair comparisons. In the absence of optimization difficulties, larger networks should reach a similar or better loss value compared to smaller networks. However, the swarm plot in Figure 4 shows that both DT-RNN and DT(S)-RNN become considerably harder to optimize with increasing depth. Increasing the recurrence depth does not adversely affect optimization of RHNs. These results are similar to those obtained in an optimization study on feedforward Highway networks [27].

## 5.2 Sequence Modeling

### 5.2.1 Penn Treebank

To examine the effect of recurrence depth we trained RHNs with fixed total parameters (32 M) and recurrence depths ranging from 1 to 10 for word level language modeling on the Penn TreeBank dataset [34]. For each depth, we show the test set perplexity of the best model based on performance on the validation set in Figure 5. Additionally we also report the results for each model trained with WT regularization which further reduces parameters. In both cases the test score improves as the recurrence depth increases from 1 to 10, dramatically at first, then levelling out at 9-10 layers.

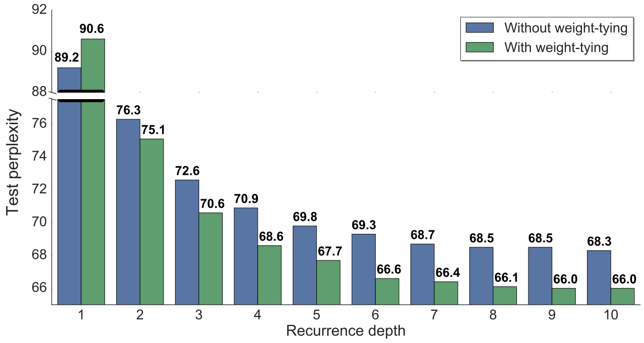


Figure 5: Test set perplexity on Penn Treebank word-level language modeling using RHNs with fixed parameter budget and increasing recurrence depth. Increasing the depth improves performance up to 9 layers.

As the recurrence depth increased from 1 to 10 layers the “width" of the network decreased from 1275 to 830 units since the number of parameters was kept fixed. Thus, these results demonstrate that even for small datasets utilizing parameters to increase depth can yield much greater benefits than increasing width. Table 1 compares our result with the best published results on this dataset. The directly comparable baseline is Variational LSTM+WT, which only differs in network architecture and size from our models. RHNs far outperform all single models as well as all previous ensembles, and also benefit from WT regularization similar to LSTMs.

Table 1: Validation and test set perplexity of recent state of the art word-level language models on the Penn Treebank dataset. The model from Kim et al. [17] uses feedforward highway layers to transform a character-aware word representation before feeding it into LSTM layers. *dropout* indicates the regularization used by Zaremba et al. [35] which was applied to only the input and output of recurrent layers. *Variational* refers to the dropout regularization from Gal [31] based on approximate variational inference. RHNs with large recurrence depth significantly outperform all previous single model and ensemble model results.

| Model | Size | Best Val. | Test |
|---|---|---|---|
| Conv.+Highway+LSTM+dropout [17] | 19 M | – | 78.9 |
| LSTM+dropout [35] | 66 M | 82.2 | 78.4 |
| Variational LSTM [31] | 66 M | 77.3 | 75.0 |
| Variational LSTM + WT [32] | 66 M | 75.8 | 73.2 |
| Pointer Sentinel networks [36] | 21 M | 72.4 | 70.9 |
| Ensemble of 38 large LSTMs [35] | – | 71.9 | 68.7 |
| **Variational RHN** | **32 M** | **71.2** | **68.5** |
| **Variational RHN + WT** | **24 M** | **68.1** | **66.0** |

### 5.2.2 Wikipedia

The task for this experiment is next symbol prediction on the challenging Hutter Prize Wikipedia dataset (`enwik8`) [37] with 205 unicode symbols in total. Due to its size (100 M characters in total) and complexity (inclusion of Latin/non-Latin alphabets, XML markup and various special characters) this dataset allows us to stress the learning and generalization capacity of RHNs. We train a variational RHN with recurrence depth of 5 and 1500 units per hidden layer, resulting in validation/test set BPC of **1.31/1.32**. Table 2 shows RHNs outperform all previous work on this dataset. However, we note that Table 2 lists models with widely different sizes, architecture and regularization techniques, and cannot be used for comparison of architectures on its own. In particular, our limited experiments suggest further improvement in scores using larger models and optimization of regularization hyperparameters.

Table 2: Entropy in Bits Per Character (BPC) on the `enwik8` test set (without dynamic evaluation). A variational RHN with recurrence depth of 5 comfortably outperforms all other methods. (*) Model size based on our estimate.

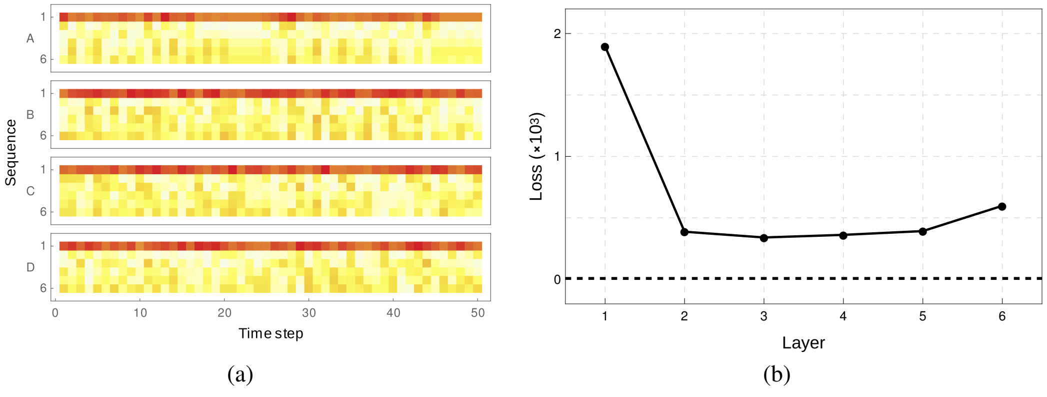
| Model | BPC | Size | Test Data |
|---|---|---|---|
| Stacked LSTM [38] | 1.67 | 27.0 M | last 4 MB |
| GF-RNN [24] | 1.58 | 20.0 M | last 5 MB |
| Grid-LSTM [20] | 1.47 | 16.8 M | last 5 MB |
| MI-LSTM [39] | 1.44 | ≈17 M | last 5 MB |
| mLSTM [40] | 1.42 | ≈21 M | last 5 MB |
| HM-LSTM* [41] | 1.40 | ≈48 M | last 5 MB |
| HyperLSTM [42] | 1.38 | 17.9 M | last 5 MB |
| **RHN** | **1.32** | **27.6** M | **last 5 MB** |

## 6 Analysis

We analyze the inner workings of RHNs through inspection of gate activations, and their effect on network performance. For the RHN with a recurrence depth of six optimized on the JSB Chorales dataset (subsection 5.1), 6(a) shows the mean transform gate activity in each layer over time steps for 4 example sequences. We note that while the gates are biased towards zero (white) at initialization, all layers are utilized in the trained network. The gate activity in the first layer of the recurrent transition is typically high on average, indicating that at least one layer of recurrent transition is almost always utilized. Gates in other layers have varied behavior, dynamically switching their activity over time in a different way for each sequence.

The contributions of the layers towards network performance can be quantified through a *lesioning* experiment similar to that introduced by Srivastava et al. [27]. For one layer at a time, all the gates are pushed towards carry behavior by setting the bias to a large negative value, and the resulting loss is measured. Figure 6(b) shows the change in loss due to the biasing of each layer, and hence its contribution to the network performance. In this case, we find that the first layer contributes several times more to the overall performance compared to others. It is notable that removing any layer hurts the performance substantially due to the recurrent nature of the network.

Similar to the feedforward case, the Highway layers in RHNs perform *adaptive computation*, i.e. the effective amount of transformation is dynamically adjusted for each sequence and time step. Unlike the general methods mentioned in section 2, the maximum depth is limited to the recurrence depth of the RHN layer.



(a)

(b)

Figure 6: (a) Activations of the transform (T) gates for different recurrence depths in 4 different sequences. An active transform gate indicates that the recurrence layer is used to process input at a particular time step, as opposed to passing it to the next layer. (b) Changes in loss when the recurrence layers are biased towards carry behavior (effectively removed), one layer at a time.

## 7 Conclusion

We developed a new analysis of the behavior of RNNs based on the Geršgorin Circle Theorem. The analysis provided insights about the ability of gates to variably influence learning in a simplfied version of LSTMs. We introduced Recurrent Highway Networks, a powerful new model designed to take advantage of increased depth in the recurrent transition while retaining the ease of training of LSTMs. Experiments confirmed the theoretical optimization advantages as well as improved performance on well known sequence modeling tasks.

**Acknowledgements:** This research was partially supported by the H2020 project "Intuitive Natural Prosthesis UTilization" (INPUT; #687795) and SNSF grant "Advanced Reinforcement Learning" (#156682). We thank Klaus Greff, Sjoerd van Steenkiste, Wonmin Byeon and Bas Steunebrink for many insightful discussions.

## References

[1] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[2] Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks*, 2014.

[3] A. J. Robinson and F. Fallside. The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department, 1987.

[4] Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1988.

[5] R. J. Williams. Complexity of exact gradient computation algorithms for recurrent neural networks. Technical Report NU-CCS-89-27, Boston: Northeastern University, College of Computer Science, 1989.

[6] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio. How to construct deep recurrent neural networks. *ArXiv e-prints*, December 2013.

[7] S. Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. Master's thesis, Univ. Helsinki, 1970.

[8] Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, 16 (2):146–160, 1976. ISSN 1572-9125.

[9] Paul J. Werbos. *System Modeling and Optimization: Proceedings of the 10th IFIP Conference New York City, USA, August 31 – September 4, 1981*, chapter Applications of advances in nonlinear sensitivity analysis, pages 762–770. Springer Berlin Heidelberg, 1982.

[10] S Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Master's thesis, Institut f. Informatik, Technische Univ. Munich, 1991.

[11] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In S. C. Kremer and J. F. Kolen, editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press, 2001.

[12] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *ArXiv e-prints*, November 2012.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[14] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, 2016/02/18 2000.

[15] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[16] Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yao, Sanjeev Khudanpur, and James Glass. Highway long short-term memory RNNS for distant speech recognition. In *2016 IEEE, ICASSP*, 2016.

[17] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*, 2015.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[19] Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

[20] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *CoRR*, abs/1507.01526, 2015. URL `http://arxiv.org/abs/1507.01526`.

[21] Jürgen Schmidhuber. Reinforcement learning in markovian and non-markovian environments. In *Advances in Neural Information Processing Systems 3*. Morgan-Kaufmann, 1991.

[22] Rupesh Kumar Srivastava, Bas R Steunebrink, and Jürgen Schmidhuber. First experiments with powerplay. *Neural Networks*, 41:130–136, 2013.

[23] A. Graves. Adaptive Computation Time for Recurrent Neural Networks. *ArXiv e-prints*, March 2016.

[24] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, 2015.

[25] S. Geršgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Bulletin de l'Acadèmie des Sciences de l'URSS. Classe des sciences mathèmatiques*, no. 6:749–754, 1931.

[26] Q. V. Le, N. Jaitly, and G. E. Hinton. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. *ArXiv e-prints*, April 2015.

[27] Rupesh K Srivastava, Klaus Greff, and Juergen Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems 28*, pages 2368–2376. Curran Associates, Inc., 2015.

[28] 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.

[29] K. Greff, R. K. Srivastava, J. Koutník, B. R Steunebrink, and J. Schmidhuber. LSTM: A Search Space Odyssey. *arXiv preprint arXiv:1503.04069*, 2015.

[30] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. 2015.

[31] Yarin Gal. A theoretically grounded application of dropout in recurrent neural networks. *arXiv preprint arXiv:1512.05287*, 2015.

[32] O. Press and L. Wolf. Using the Output Embedding to Improve Language Models. *ArXiv e-prints*, August 2016.

[33] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. *ArXiv e-prints*, June 2012.

[34] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, June 1993. ISSN 0891-2017.

[35] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent Neural Network Regularization. *ArXiv e-prints*, September 2014.

[36] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer Sentinel Mixture Models. *ArXiv e-prints*, September 2016.

[37] M. Hutter. The human knowledge compression contest. http://prize.hutter1.net/, 2012.

[38] S. Fernandez, A. Graves, and J. Schmidhuber. Sequence labelling in structured domains with hierarchical recurrent neural networks. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[39] Y. Wu, S. Zhang, Y. Zhang, Y. Bengio, and R. Salakhutdinov. On Multiplicative Integration with Recurrent Neural Networks. *ArXiv e-prints*, June 2016.

[40] B. Krause, L. Lu, I. Murray, and S. Renals. Multiplicative LSTM for sequence modelling. *ArXiv e-prints*, September 2016.

[41] J. Chung, S. Ahn, and Y. Bengio. Hierarchical Multiscale Recurrent Neural Networks. *ArXiv e-prints*, September 2016.

[42] D. Ha, A. Dai, and Q. V. Le. HyperNetworks. *ArXiv e-prints*, September 2016.

[43] A. Graves. Generating sequences with recurrent neural networks. *ArXiv e-prints*, August 2013.

# 8 Supplementary Material

## 8.1 Details of Experimental Setups

The following paragraphs describe the precise experimental settings used to obtain results in this paper. The repository at `https://github.com/julian121266/RecurrentHighwayNetworks` contains code for reproducing the results on Penn Treebank and enwik8 experiments with Tensorflow or Torch7 packages.

**Optimization**

In these experiments, we compare RHNs to Deep Transition RNNs (DT-RNNs) and Deep Transition RNNs with Skip connections (DT(S)-RNNs) introduced by [6]. We ran 60 random hyperparamter settings for each architecture and depth. The number of units in each layer of the recurrence was fixed to $\{1.5 \times 10^5, 3 \times 10^5, 6 \times 10^5, 9 \times 10^5\}$ for recurrence depths of $1, 2, 4$ and 6, respectively. The batch size was set to 32 and training for a maximum of 1000 epochs was performed, stopping earlier if the loss did not improve for 100 epochs. $tanh(\cdot)$ was used as the activation function for the nonlinear layers. For the random search, the initial transform gate bias was sampled from $\{0, -1, -2, -3\}$ and the initial learning rate was sampled uniformly (on logarithmic scale) from $[10^0, 10^{-4}]$. Finally, all weights were initialized using a Gaussian distribution with standard deviation sampled uniformly (on logarithmic scale) from $[10^{-2}, 10^{-8}]$. For these experiments, optimization was performed using stochastic gradient descent with momentum, where momentum was set to $0.9$.

**Penn Treebank**

The Penn Treebank text corpus [34] is a comparatively small standard benchmark in language modeling. The and pre-processing of the data was same as that used by Gal [31] and our code is based on Gal's [31] extension of Zaremba's [35] implementation. To study the influence of recurrence depth, we trained and compared RHNs with 1 layer and recurrence depth of from 1 to 10. with a total budget of 32 M parameters. This leads to RHN with hidden state sizes ranging from 1275 to 830 units. Batch size was fixed to 20, sequence length for truncated backpropagation to 35, learning rate to 0.2, learning rate decay to 1.02 which started after 20 epochs, weight decay to 1e-7 and maximum gradient norm to 10. Dropout rates were chosen to be 0.25 for the embedding layer, 0.75 for the input to the gates, 0.75 for the hidden units and 0.25 for the output activations. All weights were initialized from a uniform distribution between $[-0.04, 0.04]$.

**Wikipedia**

The Wikipedia dataset [37] was split into training/validation/test splits of 90 M, 5 M and 5 M characters similar to other recent work. We trained an RHN with 5 stacked layers in the recurrent state transition with 1500 units, resulting in a network with ≈27.6 M parameters. An initial learning rate of 0.2 and learning rate decay of 1.04 after 5 epochs was used. Training was performed on mini-batches of 100 sequences of length 50 with a weight decay of 1e-7. The activation of the previous sequence was kept to enable learning of very long-term dependencies [43]. To regularize, variational dropout [31] was used with dropout probabilities of 0.1 at input embedding, 0.3 at the output layer and input to the RHN and 0.05 for the hidden units of the RHN. Weights were initialized uniformly from the range [-0.04, 0.04] and an initial bias of $-4$ was set for the transform gate to facilitate learning early in training. Similar to the Penn Treebank experiments, the gradients were rescaled to a norm of 10 whenever this value was exceeded.