Recurrent Highway Networks

Julian Georg Zilly [*1] **Rupesh Kumar Srivastava** [*2] **Jan Koutník** [2] **Jürgen Schmidhuber** [2]

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 extend the LSTM architecture to allow step-to-step transition depths larger than one. Several language modeling experiments demonstrate that the proposed architecture results in powerful and efficient models. On the Penn Treebank corpus, solely increasing the transition depth from 1 to 10 improves word-level perplexity from 90.6 to 65.4 using the same number of parameters. On the larger Wikipedia datasets for character prediction (`text8` and `enwik8`), RHNs outperform all previous results and achieve an entropy of 1.27 bits per character.

1. Introduction

Network depth is of central importance in the resurgence 复兴 of neural networks as a powerful machine learning paradigm (Schmidhuber, 2015). Theoretical evidence indicates that deeper networks can be exponentially more efficient at representing certain function classes (see e.g. Bengio & LeCun (2007); Bianchini & Scarselli (2014) and references therein). Due to their sequential nature, Recurrent Neural Networks (RNNs; Robinson & Fallside, 1987; Werbos, 1988; Williams, 1989) 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 (Pascanu et al., 2013). For example, the state update from one time step to the next is typically modeled using a single trainable linear transformation followed by a non-linearity.

Unfortunately, increased depth represents a challenge when neural network parameters are optimized by means of error backpropagation (Linnainmaa, 1970; 1976; Werbos, 1982). Deep networks suffer from what are commonly referred to as the vanishing and exploding gradient problems (Hochreiter, 1991; Bengio et al., 1994; Hochreiter et al., 2001), 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; Hochreiter & Schmidhuber, 1997; Gers et al., 2000) architecture was introduced to specifically address the problem of vanishing/exploding gradients for recurrent networks.

The vanishing gradient problem also becomes a limitation when training very deep feedforward networks. *Highway Layers* (Srivastava et al., 2015b) based on the LSTM cell addressed this limitation enabling the training of networks even with hundreds of stacked layers. Used as feedforward connections, these layers were used to improve performance in many domains such as speech recognition (Zhang et al., 2016) and language modeling (Kim et al., 2015; Jozefowicz et al., 2016), and their variants called *Residual networks* have been widely useful for many computer vision problems (He et al., 2015).

In this paper we first provide a new mathematical analysis of RNNs which offers a deeper understanding of the strengths of the LSTM cell. Based on these insights, we introduce LSTM networks that have long credit assignment paths not just in time but also in space (per time step), called *Recurrent Highway Networks* or *RHNs*. Unlike previous work on deep RNNs, this model incorporates Highway layers inside the recurrent transition, which we argue is a superior method of increasing depth. This enables the use of substantially more powerful and trainable sequential models efficiently, significantly outperforming existing architectures on widely used benchmarks.

[*] Equal contribution [1] ETH Zürich, Switzerland [2] The Swiss AI Lab IDSIA (USI-SUPSI) & NNAISENSE, Switzerland. Correspondence to: Julian Zilly <jzilly@ethz.ch>, Rupesh Srivastava <rupesh@idsia.ch>.

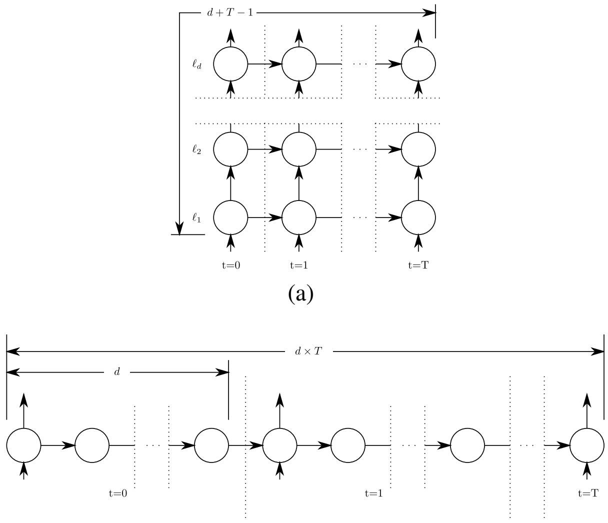
2. Related Work on Deep Recurrent Transitions

In recent years, a common method of utilizing the computational advantages of depth in recurrent networks is *stacking* recurrent layers (Schmidhuber, 1992), 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 (Zhang et al., 2016; Kalchbrenner et al., 2015).

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 (Schmidhuber, 1991; Srivastava et al., 2013; Graves, 2016). 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 (2016) reported improvements on simple algorithmic tasks using this method, no performance improvements were obtained on real world data.

Pascanu et al. (2013) 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[1]. In related work Chung et al. (2015) 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 $d+T-1$ 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.



(a)

(b)

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 is $d \times T$ for Deep Transition RNNs.

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 θ of a network can be expanded using the chain rule:

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

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:

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

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

[1] Training of our proposed architecture is compared to these models in subsection 5.1.

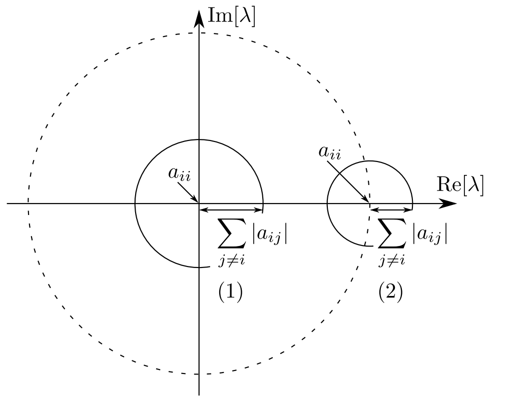


Figure 2: Illustration of the Geršgorin circle theorem. Two Geršgorin circles are centered around their diagonal entries a_{ii}. The corresponding eigenvalues lie within the radius a_{ii}. The corresponding eigenvalues lie within the radius of the sum of absolute values of non-diagonal entries a_{ij}. 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.

satisfies:

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

which together with (2) provides the conditions for vanishing gradients ($\gamma\sigma_{max} < 1$). Note that γ depends on the activation function f, e.g. $|tanh'(x)| \leq 1$, $|\sigma'(x)| \leq \frac{1}{4}, \forall x \in \mathbb{R}$, where σ is a logistic sigmoid. Similarly, we can show that if the spectral radius ρ 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) (Geršgorin, 1931): *For any square matrix* $\mathbf{A} \in \mathbb{R}^{n\times n}$*,*

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

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 L_1/L_2 weight regularization will also limit the magnitude of the eigenvalues.

- 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 we hypothesize can be efficient and necessary for learning complex sequence processing.

Le et al. (2015) 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. (2015).

4. Recurrent Highway Networks (RHN)

Highway layers (Srivastava et al., 2015a) 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 (σ) 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 "$\cdot$" 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 ℓ with $\mathbf{s}_0^{[t]} = \mathbf{y}^{[t-1]}$. Then an RHN 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

$$\mathbf{h}_\ell^{[t]} = tanh(\mathbf{W}_H \mathbf{x}^{[t]} \mathbb{I}_{\{\ell=1\}} + \mathbf{R}_{H_\ell} \mathbf{s}_{\ell-1}^{[t]} + \mathbf{b}_{H_\ell}), \tag{7}$$

$$\mathbf{t}_\ell^{[t]} = \sigma(\mathbf{W}_T \mathbf{x}^{[t]} \mathbb{I}_{\{\ell=1\}} + \mathbf{R}_{T_\ell} \mathbf{s}_{\ell-1}^{[t]} + \mathbf{b}_{T_\ell}), \tag{8}$$

$$\mathbf{c}_\ell^{[t]} = \sigma(\mathbf{W}_C \mathbf{x}^{[t]} \mathbb{I}_{\{\ell=1\}} + \mathbf{R}_{C_\ell} \mathbf{s}_{\ell-1}^{[t]} + \mathbf{b}_{C_\ell}), \tag{9}$$

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^{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 (Cho et al., 2014) and those studied by Greff et al. (2015) and Jozefowicz et al. (2015), 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

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

where

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

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

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

and has a spectrum of:

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

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 σ is saturated). This results in

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

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.

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

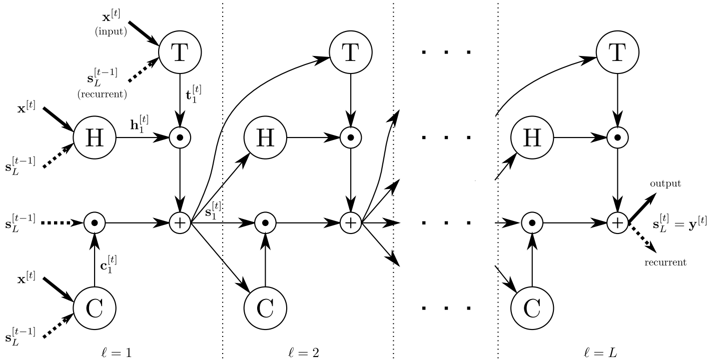


Figure 3: Schematic showing computation within an RHN layer inside the recurrent loop. Vertical dashed lines delimit stacked Highway layers. Horizontal dashed lines imply the extension of the recurrence depth by stacking further layers. H, T & C are the transformations described in equations 7, 8 and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

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 (Kim et al., 2015).

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 suboptimal for certain tasks (Greff et al., 2015; Jozefowicz et al., 2015). 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.

Regularization of RHNs: Like all RNNs, suitable regularization can be essential for obtaining good generalization with RHNs in practice. We adopt the regularization technique proposed by Gal (2015), which is an interpretation of dropout based on approximate variational inference. RHNs regularized by this technique are referred to as variational RHNs. For the Penn Treebank word-level language modeling task, we report results both with and without weight-tying (WT) of input and output mappings for fair comparisons. This regularization was independently proposed by Inan & Khosravi (2016) and Press & Wolf (2016).

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 (Pascanu et al., 2013). Networks with recurrence depth of 1, 2, 4 and 6 are trained for next step prediction on the JSB Chorales polyphonic music prediction dataset

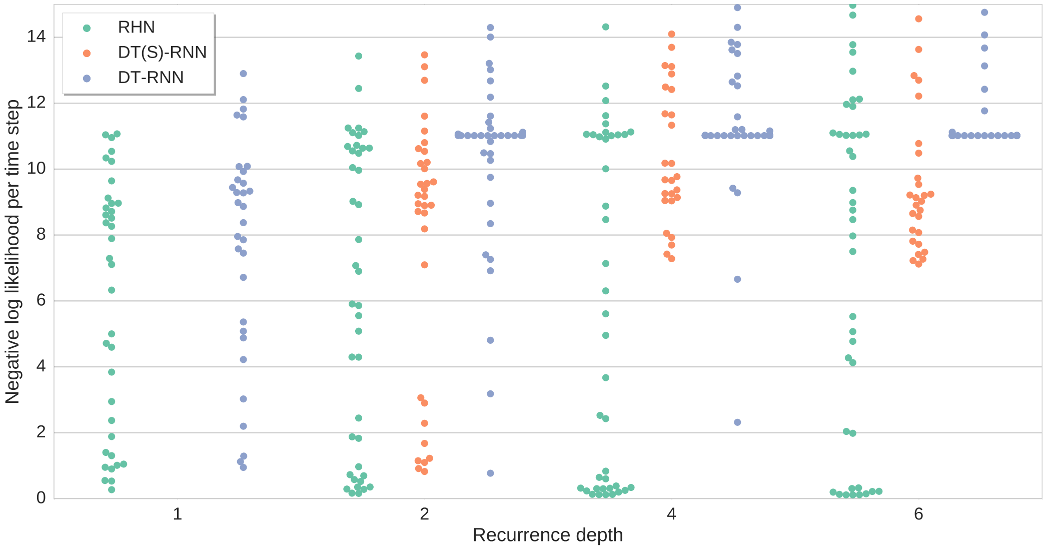


Figure 4: Swarm plot of optimization experiment results for various architectures for different depths on next step prediction on the JSB Chorales dataset. Each point is the result of optimization using a random hyperparameter setting. The number of network parameters increases with depth, but is kept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

(Boulanger-Lewandowski et al., 2012). 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. Similar to feedforward Highway networks, increasing the recurrence depth does not adversely affect optimization of RHNs.

5.2. Sequence Modeling

5.2.1. Penn Treebank

To examine the effect of recurrence depth we train 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 (Marcus et al., 1993) preprocessed by Mikolov et al. (2010). The same hyperparameters are used to train each model. For each depth, we show the test set perplexity of the best model based on performance on the validation set in Figure 5(a). Additionally we also report the results for each model trained with WT regularization. In both cases the test score improves as the recurrence depth increases from 1 to 10. For the best 10 layer model, reducing the weight decay further improves the results to **67.9/65.4** validation/test perplexity.

As the recurrence depth increases from 1 to 10 layers the "width" of the network decreases 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 large benefits even though the size of the RNN "state" is reduced. 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 outperform most single models as well as all previous ensembles, and also benefit from WT regularization similar to LSTMs. Solely the yet to be analyzed architecture found through reinforcement learning and hyperparamater search by Zoph & Le (2016) achieves better results.

5.2.2. Wikipedia

The task for this experiment is next symbol prediction on the challenging Hutter Prize Wikipedia datasets `text8` and `enwik8` (Hutter, 2012) with 27 and 205 unicode symbols in total, respectively. Due to its size (100 M characters in total) and complexity (inclusion of Latin/non-Latin alphabets, XML markup and various special characters for `enwik8`) these datasets allow us to stress the learning and generalization capacity of RHNs. We train various variational RHNs with recurrence depth of 5 or 10 and 1000 or 1500 units per hidden layer, obtaining state-of-the-art results. On `text8` a

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. (2015) 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. (2014) which was applied to only the input and output of recurrent layers. *Variational* refers to the dropout regularization from Gal (2015) based on approximate variational inference. RHNs with large recurrence depth achieve highly competitive results and are highlighted in bold.

Model	Size	Best Val.	Test
RNN-LDA + KN-5 + cache (Mikolov & Zweig, 2012)	9 M	–	92.0
Conv.+Highway+LSTM+dropout (Kim et al., 2015)	19 M	–	78.9
LSTM+dropout (Zaremba et al., 2014)	66 M	82.2	78.4
Variational LSTM (Gal, 2015)	66 M	77.3	75.0
Variational LSTM + WT (Press & Wolf, 2016)	51 M	75.8	73.2
Pointer Sentinel-LSTM (Merity et al., 2016)	21 M	72.4	70.9
Variational LSTM + WT + augmented loss (Inan et al., 2016)	51 M	71.1	68.5
Variational RHN	**32 M**	**71.2**	**68.5**
Neural Architecture Search with base 8 (Zoph & Le, 2016)	32 M	–	67.9
Variational RHN + WT	**23 M**	**67.9**	**65.4**
Neural Architecture Search with base 8 + WT (Zoph & Le, 2016)	25 M	–	64.0
Neural Architecture Search with base 8 + WT (Zoph & Le, 2016)	54 M	–	62.4

Table 2: Entropy in Bits Per Character (BPC) on the `enwik8` test set (results under 1.5 BPC & without dynamic evaluation). LN refers to the use of layer normalization (Lei Ba et al., 2016).

Model	BPC	Size
Grid-LSTM (Kalchbrenner et al., 2015)	1.47	17 M
MI-LSTM (Wu et al., 2016)	1.44	17 M
mLSTM (Krause et al., 2016)	1.42	21 M
LN HyperNetworks (Ha et al., 2016)	1.34	27 M
LN HM-LSTM (Chung et al., 2016)	1.32	35 M
RHN - Rec. depth 5	**1.31**	**23** M
RHN - Rec. depth 10	**1.30**	**21** M
Large RHN - Rec. depth 10	**1.27**	**46** M

Table 3: Entropy in Bits Per Character (BPC) on the `text8` test set (results under 1.5 BPC & without dynamic evaluation). LN refers to the use of layer normalization (Lei Ba et al., 2016).

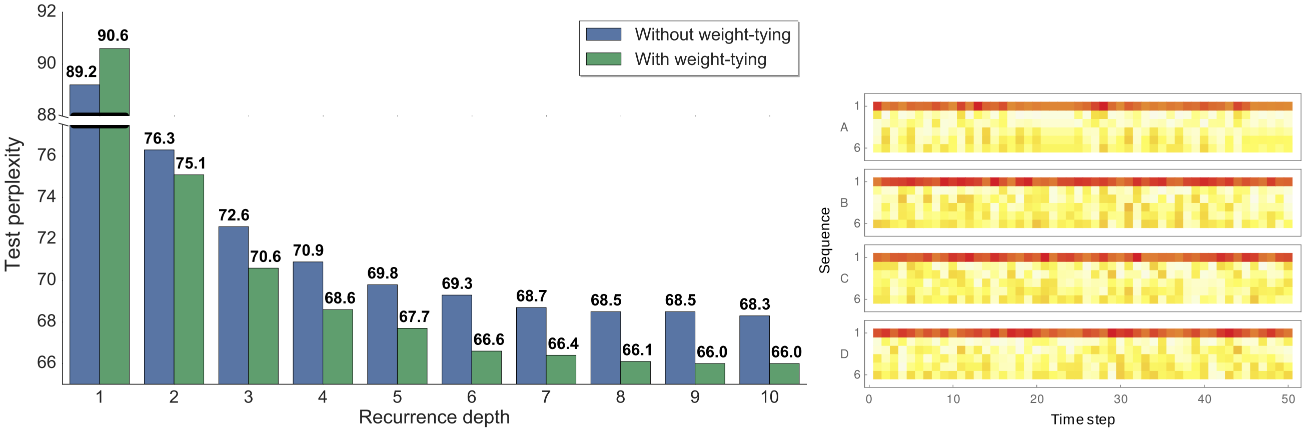
Model	BPC	Size
MI-LSTM (Wu et al., 2016)	1.44	17 M
mLSTM (Krause et al., 2016)	1.40	10 M
BN LSTM (Cooijmans et al., 2016)	1.36	16 M
HM-LSTM (Chung et al., 2016)	1.32	35 M
LN HM-LSTM (Chung et al., 2016)	1.29	35 M
RHN - Rec. depth 10	**1.29**	**20** M
Large RHN - Rec. depth 10	**1.27**	**45** M

validation/test set BPC of **1.19/1.27** for a model with 1500 units and recurrence depth 10 is achieved. Similarly, on `enwik8` a validation/test set BPC of **1.26/1.27** is achieved for the same model and hyperparameters. The only difference between the models is the size of the embedding layer, which is set to the size of the character set. Table 2 and Table 3 show that RHNs outperform the previous best models on `text8` and `enwik8` with significantly fewer total parameters. A more detailed description of the networks is provided in the supplementary material.

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), Figure 5(b) 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.

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 concrete description of such computations in feedforward networks has recently been offered in terms of learning *unrolled iterative estimation* (Greff et al., 2016). This description carries over to RHNs – the first layer in the recurrent transition computes a rough estimation of how the memory state should change



(a) (b)

Figure 5: (a) 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 10 layers. (b) Mean activations of the transform (T) gates in an RHN with a recurrence depth of 6 for 4 different sequences (A-D). The activations are averaged over units in each Highway layer. A high value (red) indicates that the layer transforms its inputs at a particular time step to a larger extent, as opposed to passing its input to the next layer (white).

given new information. The memory state is then further refined by successive layers resulting in better estimates.

The contributions of the layers towards network performance can be quantified through a *lesioning* experiment (Srivastava et al., 2015a). For one Highway 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 on the training set is measured. The change in loss due to the biasing of each layer measures its contribution to the network performance. For RHNs, we find that the first layer in the recurrent transition contributes much more to the overall performance compared to others, but removing any layer in general lowers the performance substantially due to the recurrent nature of the network. A plot of obtained results is included in the supplementary material.

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 simplified 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. We are grateful to NVIDIA Corporation for providing a DGX-1 computer to IDSIA as part of the Pioneers of AI Research award.

References

Bengio, Yoshua and LeCun, Yann. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34(5):1–41, 2007.

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

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

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

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

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

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

Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, Ç., and Courville, A. Recurrent Batch Normalization. *ArXiv e-prints*, March 2016.

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

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

Geršgorin, S. 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.

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

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

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

Greff, Klaus, Srivastava, Rupesh K, and Schmidhuber, Jürgen. Highway and residual networks learn unrolled iterative estimation. *arXiv preprint arXiv:1612.07771*, 2016.

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

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

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

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

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

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

Inan, H., Khosravi, K., and Socher, R. Tying Word Vectors and Word Classifiers: A Loss Framework for Language Modeling. *ArXiv e-prints*, November 2016.

Inan, Hakan and Khosravi, Khashayar. Improved learning through augmenting the loss, 2016.

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

Jozefowicz, Rafal, Vinyals, Oriol, Schuster, Mike, Shazeer, Noam, and Wu, Yonghui. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

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

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

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

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

Lei Ba, J., Kiros, J. R., and Hinton, G. E. Layer Normalization. *ArXiv e-prints*, July 2016.

Linnainmaa, S. 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.

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

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

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

Mikolov, Tomas and Zweig, Geoffrey. Context dependent recurrent neural network language model. *SLT*, 12:234–239, 2012.

Mikolov, Tomas, Karafiát, Martin, Burget, Lukas, Cernockỳ, Jan, and Khudanpur, Sanjeev. Recurrent neural network based language model. In *Interspeech*, volume 2, pp. 3, 2010.

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

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

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

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

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

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

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

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

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

Werbos, Paul J. *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, pp. 762–770. Springer Berlin Heidelberg, 1982.

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

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

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

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

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

Zoph, Barret and Le, Quoc V. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

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 source code for reproducing the results on Penn Treebank, enwik8 and text8 experiments is available at https://github.com/julian121266/RecurrentHighwayNetworks on Github.

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 Pascanu et al. (2013). 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 (Marcus et al., 1993) is a comparatively small standard benchmark in language modeling. The and pre-processing of the data was same as that used by Gal (2015) and our code is based on Gal's (Gal, 2015) extension of Zaremba's (Zaremba et al., 2014) 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 starting at 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, 0.25 for the hidden units and 0.75 for the output activations. All weights were initialized from a uniform distribution between $[-0.04, 0.04]$. For the best 10-layer model obtained, lowering the weight decay to 1e-9 further improved results.

Enwik8

The Wikipedia enwik8 dataset (Hutter, 2012) was split into training/validation/test splits of 90 M, 5 M and 5 M characters similar to other recent work. We trained three different RHNs. One with 5 stacked layers in the recurrent state transition with 1500 units, resulting in a network with ≈23.4 M parameters. A second with 10 stacked layers in the recurrence with 1000 units with a total of ≈20.1 M parameters and a third with 10 stacked layers and 1500 units with a total of of ≈46.0 M parameters. An initial learning rate of 0.2 and a learning rate decay of 1.04 after 5 epochs was used. Only the large model with 10 stacked layers and 1500 units used a learning rate decay of 1.03 to ensure for a proper convergence. Training was performed on mini-batches of 128 sequences of length 50 with a weight decay of 0 for the first model and 1e-7 for the other two. The activation of the previous sequence was kept to enable learning of very long-term dependencies (Graves, 2013). To regularize, variational dropout (Gal, 2015) was used. The first and second model used 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. The larger third model used dropout probabilities of 0.1 at input embedding, 0.4 at the output layer and input to the RHN and 0.1 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 re-scaled to a norm of 10 whenever this value was exceeded. The embedding size was set to 205 and weight-tying (Press & Wolf, 2016) was not used.

Text8

The Wikipedia text8 dataset (Hutter, 2012) was split into training/validation/test splits of 90 M, 5 M and 5 M characters similar to other recent work. We trained two RHNs with 10 stacked layers in the recurrent state transition. One with 1000 units and one with 1500 units, resulting in networks with ≈20.1 M and ≈45.2 M parameters, respectively. An initial learning rate of 0.2 and a learning rate decay of 1.04 for the 1000 unit model and 1.03 for the 1500 units model was used after 5 epochs. Training was performed on mini-batches of 128 sequences of length 100 for the model with 1000 units and 50 for the model with 1500 units with a weight decay of 1e-7. The activation of the previous sequence was kept to enable learning of very long-term dependencies (Graves, 2013). To regularize, variational dropout (Gal, 2015) was used with dropout probabilities of 0.05 at the input embedding, 0.3 at the output layer and input to the RHN and 0.05 for the hidden units of the RHN for the model with 1000 units. The model with 1500 units used dropout probabilities of 0.1 at the input embedding, 0.4 at the output layer and at the input to the RHN and finally 0.1 for the dropout probabilities of 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. The embedding size was set to 27 and weight-tying (Press & Wolf, 2016) was not used.

Lesioning Experiment Figure 6 shows the results of the lesioning experiment from section 6. This experiment was conducted on the RHN with recurrence depth 6 trained on the JSB Chorales dataset as part of the Optimization experiment in subsection 5.1. The dashed line corresponds to the training error without any lesioning. The x-axis denotes the index of the lesioned highway layer and the y-axis denotes the log likelihood of the network predictions.

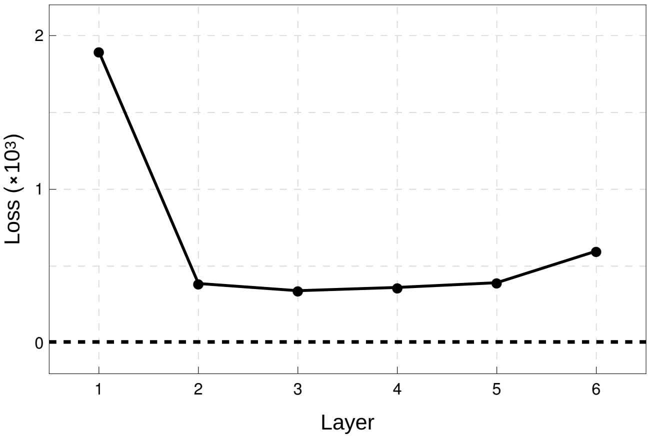


Figure 6: Changes in loss when the recurrence layers are biased towards carry behavior (effectively removed), one layer at a time.