# Character-Level Language Modeling with Recurrent Highway Hypernetworks

Joseph Suarez, Stanford University


## Abstract

We explore, expand upon, and combine recurrent highway networks (RHNs) and recurrent hypernetworks. We confirm that RHNs benefit from far better gradient flow than LSTMs in addition to improved task accuracy. We likewise provide theoretical treatment of hypernetworks complementary to the experimental results of the original work. By combining RHNs and hypernetworks, we make a significant improvement over either individually on character-level language modeling on Penn Treebank while relying on much simpler regularization.


## Recurrent Highway Networks

We first consider the GRU:

$$\begin{aligned}[h, t] =& x_i U + s_{i-1} W \quad r = \tanh(x_i \hat{U} + (s_{i-1} \circ h)\hat{W}) \\ h, t =& \sigma(h), \sigma(t) \qquad s_i = (1-t) \circ r + t \circ s_{i-1}\end{aligned} \tag{1}$$

where $x \in \mathbb{R}^d$, $h, t, r, s_i \in \mathbb{R}^n$, and $U, \hat{U} \in \mathbb{R}^{d \times 2n}, W, \hat{W} \in \mathbb{R}^{n \times 2n}$ are weight matrices where $d, n$ are the input and hidden dimensions. $\sigma$ is the sigmoid nonlinearity, and $\circ$ is the Hadamard (elementwise) product. An RHN is a simplified GRU variant:

$$\begin{aligned}&RHNCell(x_i, s_{i-1}, l) : \\ &[h, t] = \mathbb{1}\left[l = 0\right] x_i U + s_{i-1} W \; c, t = 1 - t, dropout(t) \\ &h, t = \tanh(h), \sigma(t) \qquad\qquad s_i = c \circ s_{i-1} + t \circ h\end{aligned} \tag{2}$$

where $l$ is the layer index, which is used as an indicator.

## Hypernetworks

A hypernetwork for a recurrent model is a secondary network that directly modifies the weights of the main architecture. For ease of combination with RHNs, we define a *hypervector* $z$ as a linear upscaling projection applied to the hypernetwork outputs.

$$z(a) = W_p a \tag{3}$$

where $a \in \mathbb{R}^h$ is the activation vector output by an arbitrary recurrent architecture, $W_p \in \mathbb{R}^{n \times h}$ is an upscaling projection from dimension $h$ to $n$, and $h \ll n$. The hypervector scales the weights of the main recurrent network by:

$$W(z(a)) = z(a) \circ W \tag{4}$$

where $\circ$ is the element-wise product across columns: each element of $z$ scales one column of $W$.

## Recurrent Highway Hypernetworks

We adapt hypernetworks to RHNs by directly modifying the RHN cell using (4):

$$\begin{aligned}&HyperCell(x_i, s_{i-1}, l, z) : \\ &[h, t] = \mathbb{1}\left[l = 0\right] x_i U(z) + s_{i-1} W(z) \; c, t = 1 - t, dropout(t) \\ &h, t = \tanh(h), \sigma(t) \qquad\qquad s_i = c \circ s_{i-1} + t \circ h\end{aligned} \tag{5}$$

As $RHNCell$ and $HyperCell$ have different state sizes by design, we must upscale between the networks. Our final architecture at each timestep for layer $l$ is:

$$s_h = RHNCell(s_h, l) \; z = [M_{pl} s_h, M_{pl} s_h] \; s_n = HyperCell(s_n, l, z) \tag{6}$$

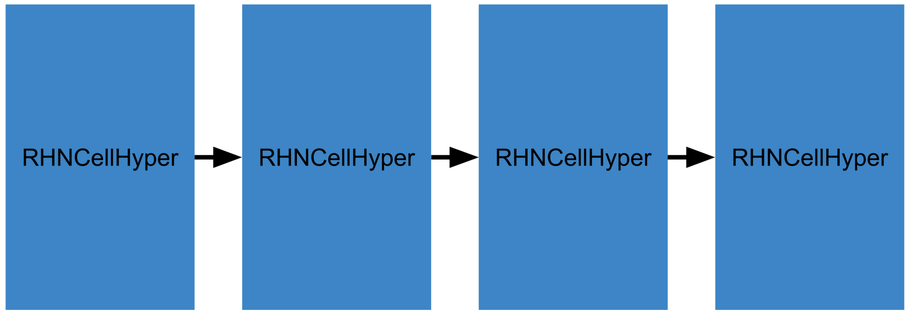
where $M_{pl} \in \mathbb{R}^{h \times n}$ is the upscale projection matrix for layer $l$ and $z$ is the concatenation of $M_{pl} s_h$ with itself.

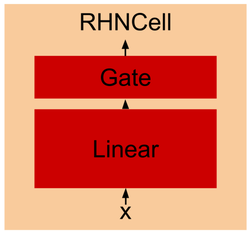
## References

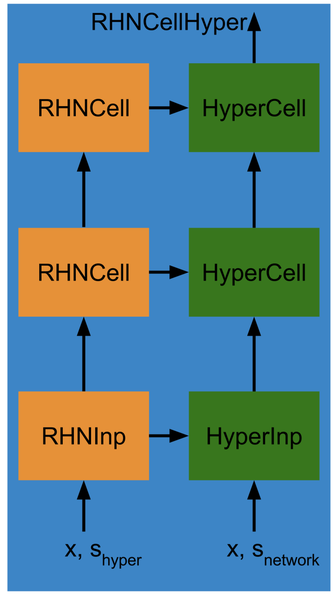

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutnik, and Jurgen Schmidhuber. Recurrent highway networks. arXiv preprint arXiv:1607.03474, 2016.

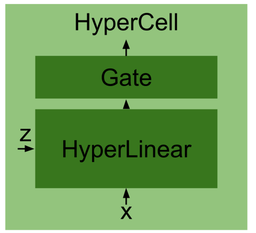
David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. arXiv preprint arXiv:1609.09106, 2016.

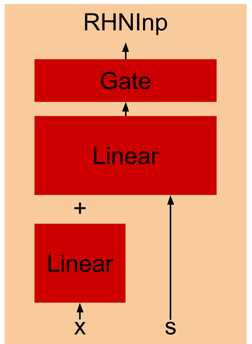

We cite only the two most central papers to our work; refer to the paper for a complete list of references,

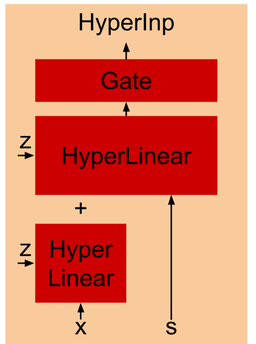


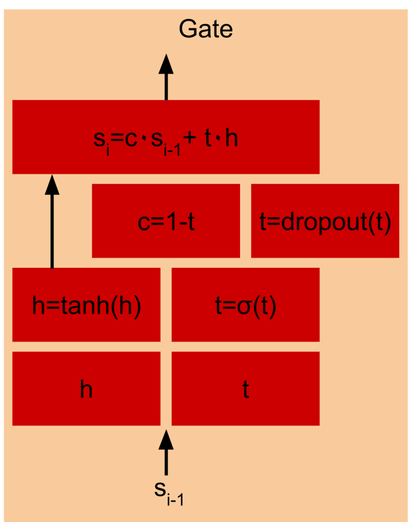



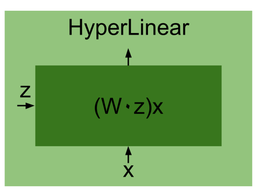




## Model Architecture

- Top level: maintains a single state between timesteps.
- RHNCellHyper: a stack of HyperCells and a stack of RHNCells.
- RHNCell, HyperCell: gated linear layers + hypervector scaling.
- Gate: nonlinearity optimized for gradient flow.

## Experiments and Results

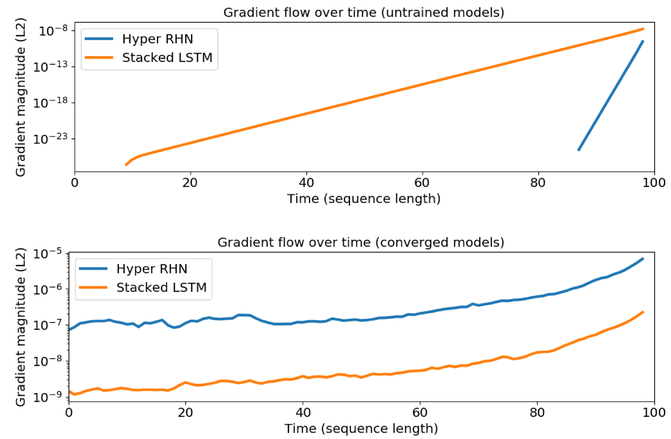
We train a 7-layer HRHN with 1000 neurons per layer on Penn Treebank, outperforming similarly sized Recurrent Highway Networks and LSTM-based Hypernetworks.

Table 1: Comparison of bits per character (BPC) test errors on PTB

| Model | Test | Val | Params (M) |
|---|---|---|---|
| 2-Layer LSTM | 1.28 | 1.31 | 12.2 |
| 2-Layer LSTM (1125 hidden, ours) | – | 1.29 | 15.6 |
| HyperLSTM | 1.26 | 1.30 | 4.9 |
| Layer Norm LSTM | 1.27 | 1.30 | 4.3 |
| Layer Norm HyperLSTM | 1.25 | 1.28 | 4.9 |
| Layer Norm HyperLSTM (large embed) | 1.23 | 1.26 | 5.1 |
| 2-Layer Norm HyperLSTM, 1000 units | 1.22 | 1.24 | 14.4 |
| Recurrent Highway Network (ours) | - | 1.24 | 14.0 |
| HyperRHN (ours) | 1.19 | 1.21 | 15.5 |

## Discussion of Gradient Flow

One natural perspective is that a network with poor initial gradient flow will never converge. We found that gradient flow actually shifts during training in favor of our architecture.




## Conclusions

- We argue for RHNs as a drop-in replacement for stacked LSTMs.
- We argue for HyperNetworks as a de-facto recurrent augmentation.
- We demonstrate that gradient flow shifts during training and should be visualized at convergence.
- We suggest multiple expansions upon hypernetworks for future work that could lead to further improvement upon our architecture
- We present streamlined mathematical and programatic formulations of RHNs and Hypernetworks.
- Our architecture obtains state-of-the-art on PTB with minimal regularization and tuning which normally compromise results on small datasets.
- We open source a clean implementation of our model and baselines at github.com/jsuarez5341/Recurrent-Highway-Hypernetworks-NIPS