Exploring the Depths of Recurrent Neural Networks with Stochastic Residual Learning

Sabeek Pradhan
Department of Computer Science
Stanford University
sabeekp@cs.stanford.edu

Shayne Longpre
Department of Computer Science
Stanford University
slongpre@cs.stanford.edu

Abstract

Recent advancements in feed-forward convolutional neural network architecture have unlocked the ability to effectively use ultra-deep neural networks with hundreds of layers. However, with a couple exceptions, these advancements have mostly been confined to the world of feed-forward convolutional neural networks for image recognition, and NLP tasks requiring recurrent networks have largely been left behind. In this paper, we apply two recent innovations in ultra-deep convolutional networks, ResNets and stochastic depth, to RNNs used for sentiment classification. We also add a new innovation, stochastic timesteps, which is similar to stochastic depth but over horizontal timesteps rather than vertical layers. We achieve classification accuracies on the five class, fine-grained Stanford Sentiment Treebank that are very close to state of the art, without using the parse tree information utilized by current SOTA methods. We believe that these results bode well for the potential of ultra-deep networks in recurrent and NLP settings in addition to their existing uses in feed-forward and computer vision ones.

1 Introduction

Recent advancements in feed-forward convolutional neural network architecture have unlocked the ability to effectively use ultra-deep neural networks with hundreds of layers. For many applications, these ultra-deep networks have outperformed shallower networks by remarkable margins, such as Microsofts ResNet [3] in the 2015 ImageNet Competition. Our objective is to apply a selection of these state-of-the-art deep network techniques to recurrent neural networks (RNNs) and to the task of natural language processing (NLP), in particular sentiment analysis. Traditionally, RNNs are shallow across layers, instead obtaining their depth across timesteps. We anticipate our modifications to the traditional RNN architecture will increase the depth capacity of these networks, thereby permitting the training of more complex and potentially higher performing models. Our motivation stems from the potential insights new architectures could yield in the application of deeper recurrent neural networks.

The specific combination of techniques we are applying to the basic Recurrent Neural Network include ResNets [3] and Stochastic Depth networks [4] over network layers as well as over timesteps. We will evaluate our modified networks accuracy and depth capacity against that of (a) traditional stacked RNNs, (b) syntactic tree-structured models, and (c) present state-of-the-art sentiment classification models. The Stanford Sentiment Treebank (SST) dataset has been thoroughly used for benchmarking a variety of high-performance RNN models, and is thus an apt choice.

2 Related Work

Ultra-deep networks are a relatively new innovation. One of the first mainstream ultra-deep architectures was Highway Networks [9]. That architecture used an LSTM-inspired gating mechanism that would allow information to flow mostly unmodified across many layers of a feed-forward convolutional neural network, similar to the way LSTMs allow information to flow mostly unmodified across many timesteps. There have already been attempts to adapt this architecture to recurrent networks, most notably [11], which uses a very similar gating mechanism to train LSTM networks up to 8 layers in depth and achieve state of the art performance on speech recognition tasks.

However, undoubtedly the most famous ultra-deep architecture, and the one from which we take the heaviest inspiration, is the ResNet architecture introduced in [3], which also won the 2015 ILSVRC competition. Rather than use a sophisticated but complex gating mechanism to determine how much of a layers information should be passed on unmodified to the following layers, ResNets use a parameter-free approach. Each layer learns a residual which is added to the layers input and then passed as the input to the next layer. More formally, if the input to a layer (or group of layers) is x and the output of that layer (or group of layers) is $F(x)$, then the input to the next layer (or group of layers) is $x + F(x)$, whereas it would be $F(x)$ in a conventional neural network and $D(x) \times x + (1 - D(x)) \times F(x)$ for some function $D(x)$ in a highway network.

An even more recent innovation on this framework is stochastic depth networks [4], which randomly drops layers from the ResNet architecture while training. This approach allows the test-time network to serve as an ensemble of shallower ResNets, the same way Dropout allows the test-time network to serve as an ensemble of smaller networks more generally. Stochastic depth has yielded both accuracy and training time improvements over ResNets on the CIFAR and SVHN datasets.

3 Models

3.1 Long Short Term Memory Cell

All our models incorporate a standard Long Short Term Memory (LSTM) cell within the given recurrent neural network framework. While we have the opportunity to substitute this element with Gated Recurrent Units (GRU) [1] and Coupled Input-Forget Gate (CIFG) cells [2] we felt that limiting the scope to LSTMs was a suitable choice given (a) its consistent performance across models, and (b) the fact that this decision is unlikely to affect our core investigation.

In figure 1 below we have the standard LSTM representation. The mathematical formulae can be read at [2].

3.2 Vanilla Deep/Stacked LSTM

Our vanilla stacked LSTM acts as a baseline comparable for our own model variations. In Figure 2 we have a visual representation of 2 timesteps and 3 LSTM layers followed by a fully connected projection layer. While a basic and ubiquitously used model, it has proven highly performant with few layers and is thus a high standard for a sentiment analysis baseline. We also use this model at a greater number of layers to compare its performance over layer ranges as compared to our model variants.

3.3 Residual Recurrent Neural Networks

Our Residual Recurrent Neural Network (Res-RNN) is configured with a minor change to how the vanilla stacked LSTM computes an output for the subsequent layer. Only the first layer LSTM (across all timesteps) remains unchanged from the vanilla model. This is because the first layer needs to adapt the dimensionality to the hidden layer size from the input size, as well as learn the initial representation that the later layers will leverage in computing their residuals. All additional layers compute their outputs differently, applying residual learning as in the ResNet. Each layer learns a residual to be added to the previous layer rather than a new representation altogether. This greatly ameliorates the vanishing gradient problem and allows for far deeper networks to be trained.

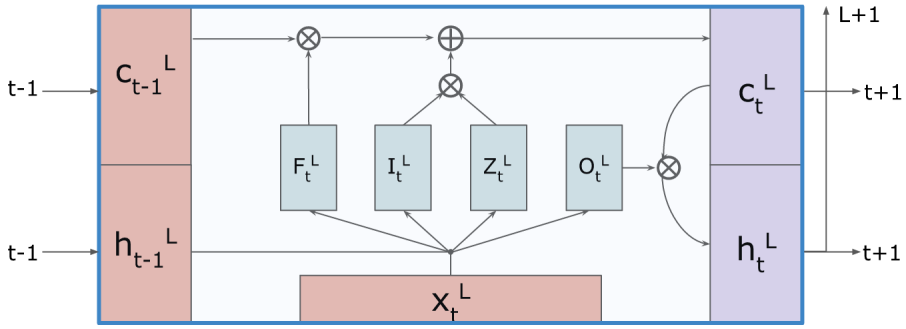

Figure 1: An individual LSTM layer

In particular, if the input to layer l at timestep t is $x_t^{(l)}$, the previous hidden and cell states for layer l from timestep $t-1$ are $h_{t-1}^{(l)}$ and $c_{t-1}^{(l)}$, respectively, then $h_t^{(l)}$ and $c_t^{(l)}$ are calculated using the normal LSTM equations, and the input to layer $l+1$ at timestep t is $x_t^{(l+1)} = x_t^{(l)} + h_t^{(l)}$.

The residual addition is visually represented in the figure 2.

3.4 Stochastic Depth

Stochastic Depth, which uses the ResNet architecture introduced above, randomly drops layers during training. Since our LSTM layers are of equivalent dimensionality we can simply take the output from one layer and feed it as the input not to the immediately succeeding layer, but the one beyond that. For obvious reasons the fully connected layer and first LSTM layer after the input cannot have stochastic depth applied to them.

The advantage of this technique is that it provides a Dropout-style ensemble of shallower networks consisting of the undropped layers. See figure 3 for a visual representation, where the second layer is dropped, as depicted in shaded red.

3.5 Stochastic Timesteps

Stochastic Timesteps is the term we have given to applying stochastic depth over timesteps rather than layers. This simply consists of randomly dropping omitting timesteps (a word in the input sentence) during training. The shaded red zone in figure 3 represent the dropped timestep. For our implementation we eliminated the possibility of ever dropping the first word/timestep. This is a convenient fix to prevent empty inputs when stochastic timesteps happens to trigger on every timestep of a small input sentence.

We believe the advantage of this technique is that it helps train a model more robust to incomplete or superfluous language.

4 Technical Approach

4.1 Dataset

The Stanford Sentiment Treebank dataset [8] has been heavily curated, and recently been used for benchmarking high performance sentiment analysis models, such as in [5], [6], [10], and others. The dataset contains 11855 sentences labelled for sentiment between 1.0 and 0.0 inclusively. We chose to focus our attention to the 5 class sentiment labels, which classified sentences as very negative (0.0-0.2), negative (0.2-0.4), neutral (0.4-0.6), positive (0.6-0.8), or very positive (0.8-1.0). The provided sentences are also split into labelled sub-phrases to make them conducive for syntactic tree models. We trained our model on all train sentences and sub-phrases derived from these train sentences, totalling 166,743 training phrase examples. The validation and test sets, however, contain

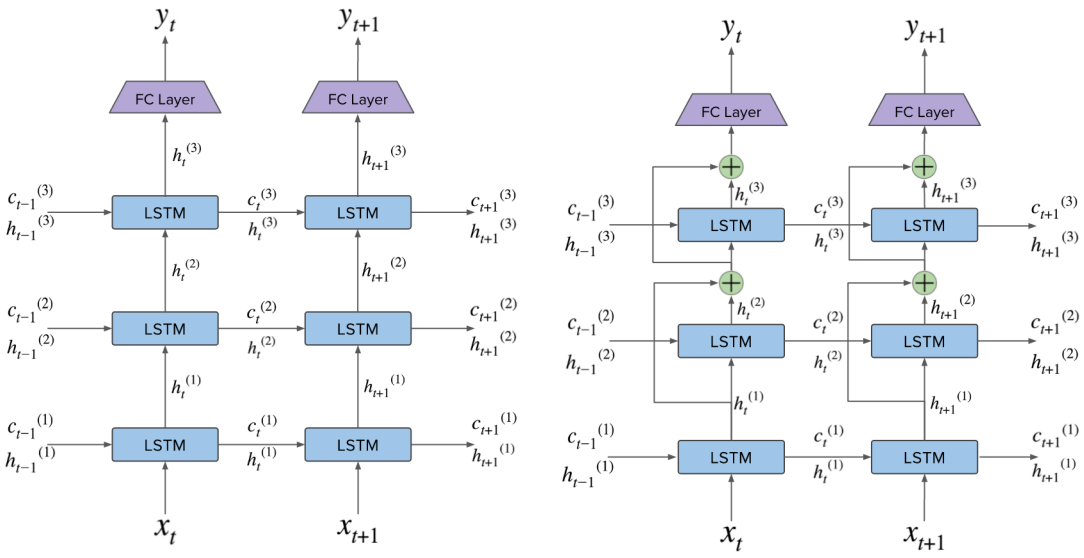

Figure 2: The normal stacked LSTM (left) and Res-RNN (right) architectures

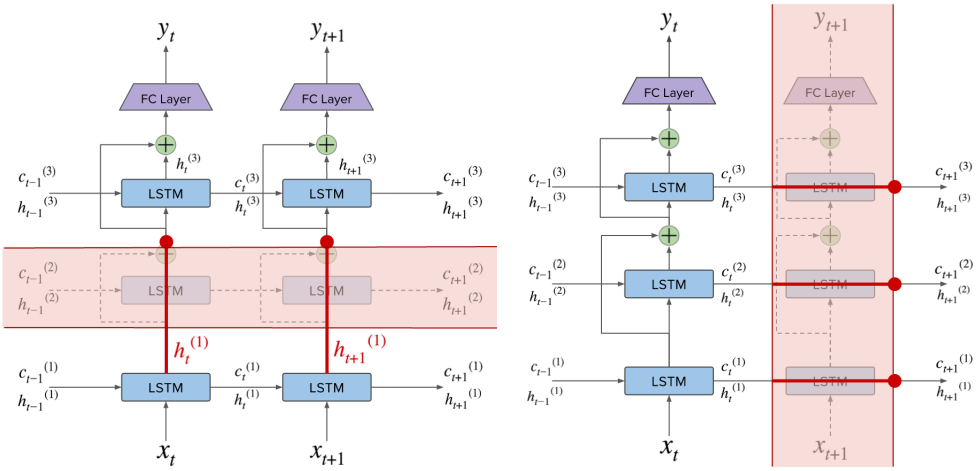

Figure 3: Res-RNN with stochastic depth (left) and stochastic timesteps (right)

Method	Accuracy
Res-RNN	50.6%
Res-RNN Stochastic Timesteps	50.5%
Res-RNN Stochastic Depth	49.5%
Vanilla Stacked LSTM	49.1%
Benchmarks from Tai, Socher and Manning 2015	
Constituency Tree-LSTM	51.0%
Deep Recursive Net	49.8%
Paragraph Vector	48.7%
Dependency Tree-LSTM	48.4%

Table 1: Model Comparables

Model Type	Num Layers	Hidden Units	Accuracy
Res-RNN	2	80	50.6%
Res-RNN ($ST = 0.5$)	4	160	50.5%
Res-RNN	12	160	49.7%
Res-RNN ($SD = 0.75$)	12	120	49.5%
Res-RNN ($SD = 0.75$)	4	160	49.5%
Res-RNN ($SD = 0.75$)	4	120	49.5%
Res-RNN	6	120	49.3%
Res-RNN ($SD = 0.75$)	2	80	49.2%
Res-RNN	6	160	49.1%
Vanilla Stacked LSTM	4	120	49.1%

Table 2: Ranked Results

solely full sentences, consisting of 1101 and 2210 examples respectively. In evaluating our models over 5 class sentiment and only on the full, complex sentences (excluding their sub-phrases), we are maximizing the difficulty of the task at hand.

4.2 Hyperparameters and Training Details

We kept a number of hyperparameters consistent between the models, which we mostly got [10]. We used word vectors of dimensionality 300. Where possible, we used Glove vectors pretrained on 840 billion tokens, and for words in the dataset that did not appear in the Glove vocabulary, we randomly initialized every element of the word vector using a random uniform distribution from -0.05 to 0.05. Our optimizer was Adagrad with a learning rate of 0.1 for the word vectors and 0.05 for all other weights. To defend against overfitting we used L2-regularization at 1e-4 and dropout at 0.5. Lastly, we used a batch size of 128 (in contrast to [10], who used batches of 25), and our training stop condition was a maximum of 10 epochs or 2 consecutive epochs without improvement in validation loss.

For preliminary model optimization we ran grid search over the model types, number of layers (2, 4, 6, 12), hidden units (80, 120, 160), and for Res-RNN models, stochastic timesteps (0.5, 0.75, 0.9). For our stochastic depth model, we kept the probability of keeping a layer at 0.75.

5 Results

Our experiments consisted of running grid search as described in the Hyperparameters and Training Details section. We ranked our best models based on the SST 5 class sentiment classification test accuracy. Table 1 shows our best models against comparable near-SOTA models.

Table 2 comprises our top 10 ranked test accuracies on the SST 5 class sentiment classification. Note that SD denotes Stochastic Depth and ST denotes Stochastic Timesteps.

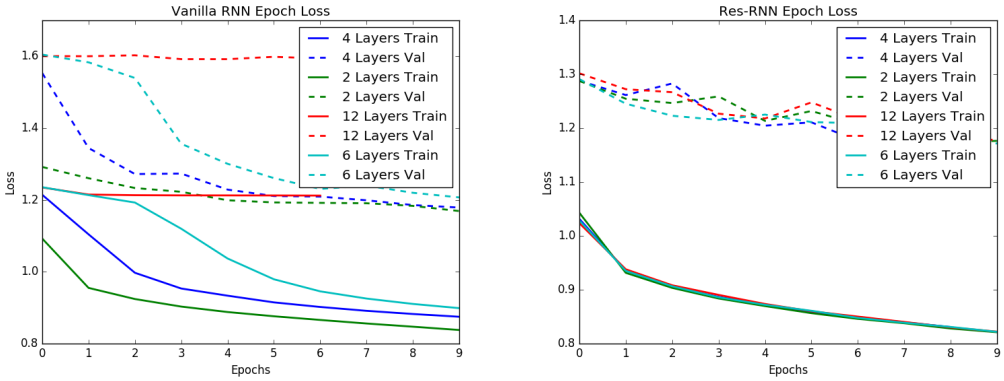

Figure 4: Training and val losses for vanilla (left) and Res-RNN (right) networks for 2, 4, 6, and 12 layers

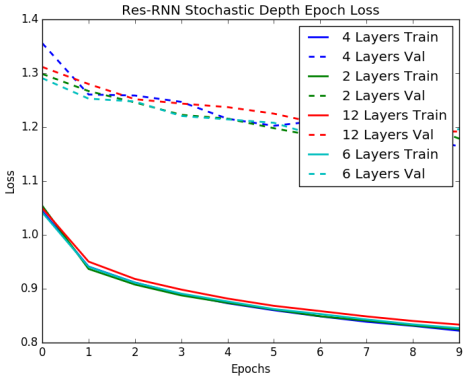

Figure 5: Training and val losses for stochastic depth (right) networks for 2, 4, 6, and 12 layers

6 Conclusion

As shown in our results tables, both ResRNN and stochastic depth over layers allowed for large improvements in test accuracy compared to the best Vanilla RNN models. Indeed, our two best Res-RNN models got within 0.5% of state of the art, which is especially notable considering that the SOTA Constituency Tree-LSTM method takes into account the explicit parse tree of the sentence while the ResRNN was only given the raw sequence of words. In addition, as Figures 4 and 5 show, while Vanilla networks were unable to reduce even their training loss as the number of layers rose, both ResRNN and stochastic depth were, confirming a key result from [3].

There are a couple puzzles which we intend to explore in greater depth. First, test accuracy for ResRNN and stochastic depth models does not appear to be a consistent function of model depth; indeed, our best performing model was a 2 layer ResRNN, which would seem too shallow to fully reap the rewards of the residual architecture. Another is the high performance of stochastic timesteps when dropping half the words while training, which seems too likely to drop important words. And finally, [3] and [4] both found that deeper ResNets or stochastic depth networks worked best when narrower (i.e. with fewer hidden units), which is the not the case with our data. Further testing will hopefully shed some light into these issues.

6.1 Future Work

One potential extension of our work would be to try stochastic depth training using Highway Networks (first pioneered by [9] and then applied to LSTMs by [11]). Currently, the recurrence in our LSTM is not modified to reflect the fact that each layer learns a residual rather than an altogether new representation; another extension might thus be to modify the recurrence relation in a manner similar to Zhang et al. 2015. Since stochastic depth should itself be a form of regularization, one could investigate in more detail whether they overfit less than normal ResRNNs do, and whether they can avoid overfitting even without L2-regularization or dropout. Finally, one could test this architecture on other tasks, such as speech recognition or music modeling, to ensure that the results are generalizable to other areas that RNNs and LSTMs are commonly used for.

References

[1] Chung, Junyoung, et al. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[2] Greff, Klaus, et al. LSTM: A search space odyssey. *arXiv preprint arXiv:1503.04069* (2015).

[3] He, Kaiming, et al. Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385* (2015).

[4] Huang, Gao, et al. Deep networks with stochastic depth. *arXiv preprint arXiv:1603.09382* (2016).

[5] Kim, Yoon. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[6] Le, Quoc V., and Tomas Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053* (2014).

[7] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) **ImageNet Large Scale Visual Recognition Challenge.** *IJCV,* 2015.

[8] Socher, Richard, et al. Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of the conference on empirical methods in natural language processing (EMNLP).* Vol. 1631. 2013.

[9] Srivastava, Rupesh K., Klaus Greff, and Jrgen Schmidhuber. Training very deep networks. *Advances in Neural Information Processing Systems.* 2015.

[10] Tai, Kai Sheng, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* (2015).

[11] Zhang, Yu, et al. Highway Long Short-Term Memory RNNs for Distant Speech Recognition. *arXiv preprint arXiv:1510.08983* (2015).