Improving Language Modeling using Densely Connected Recurrent Neural Networks

Fréderic Godin, Joni Dambre and Wesley De Neve

IDLab - ELIS, Ghent University - imec, Ghent, Belgium
firstname.lastname@ugent.be

Abstract

In this paper, we introduce the novel concept of densely connected layers into recurrent neural networks. We evaluate our proposed architecture on the Penn Treebank language modeling task. We show that we can obtain similar perplexity scores with six times fewer parameters compared to a standard stacked 2-layer LSTM model trained with dropout (Zaremba et al., 2014). In contrast with the current usage of skip connections, we show that densely connecting only a few stacked layers with skip connections already yields significant perplexity reductions.

1 Introduction

Language modeling is a key task in Natural Language Processing (NLP), lying at the root of many NLP applications such as syntactic parsing (Ling et al., 2015), machine translation (Sutskever et al., 2014) and speech processing (Irie et al., 2016).

In Mikolov et al. (2010), recurrent neural networks were first introduced for language modeling. Since then, a number of improvements have been proposed. Zaremba et al. (2014) used a stack of Long Short-Term Memory (LSTM) layers trained with dropout applied on the outputs of every layer, while Gal and Ghahramani (2016) and Inan et al. (2017) further improved the perplexity score using variational dropout. Other improvements are more specific to language modeling, such as adding an extra memory component (Merity et al., 2017) or tying the input and output embeddings (Inan et al., 2017; Press and Wolf, 2016).

To be able to train larger stacks of LSTM layers, typically four layers or more (Wu et al., 2016), skip or residual connections are needed. Wu et al. (2016) used residual connections to train a machine translation model with eight LSTM layers, while Van Den Oord et al. (2016) used both residual and skip connections to train a pixel recurrent neural network with twelve LSTM layers. In both cases, a limited amount of skip/residual connections was introduced to improve gradient flow.

In contrast, Huang et al. (2017) showed that densely connecting more than 50 convolutional layers substantially improves the image classification accuracy over regular convolutional and residual neural networks. More specifically, they introduced skip connections between *every* input and *every* output of *every* layer.

Hence, this motivates us to densely connect all layers within a stacked LSTM model using skip connections between every pair of layers.

In this paper, we investigate the usage of skip connections when stacking multiple LSTM layers in the context of language modeling. When every input of every layer is connected with every output of every other layer, we get a densely connected recurrent neural network. In contrast with the current usage of skip connections, we demonstrate that skip connections significantly improve performance when stacking only a *few* layers. Moreover, we show that densely connected LSTMs need fewer parameters than stacked LSTMs to achieve similar perplexity scores in language modeling.

2 Background: Language Modeling

A language model is a function, or an algorithm for learning such a function, that captures the salient statistical characteristics of sequences of words in a natural language. It typically allows one to make probabilistic predictions of the next word, given preceding words (Bengio, 2008). Hence, given a sequence of words $[w_1, ...w_T]$, the goal is to estimate the following joint probability:

$$Pr(w_1, ..., w_T) = \prod_{t=1}^{T} Pr(w_t|w_{t-1}, ..w_1) \quad (1)$$

In practice, we try to minimize the negative log-likelihood of a sequence of words:

$$L_{neg}(\theta) = -\sum_{t=1}^{T} log(Pr(w_t|w_{t-1}, ..w_1)). \quad (2)$$

Finally, perplexity is used to evaluate the performance of the model:

$$Perplexity = \exp\left(\frac{L_{neg}(\theta)}{T}\right) \quad (3)$$

3 Methodology

Language Models (LM) in which a Recurrent Neural Network (RNN) is used are called Recurrent Neural Network Language Models (RNNLMs) (Mikolov et al., 2010). Although there are many types of RNNs, the recurrent step can formally be written as:

$$\boldsymbol{h_t} = f_\theta(\boldsymbol{x_t}, \boldsymbol{h_{t-1}}) \quad (4)$$

in which $\boldsymbol{x_t}$ and $\boldsymbol{h_t}$ are the input and the hidden state at time step t, respectively. The function f_θ can be a basic recurrent cell, a Gated Recurrent Unit (GRU), a Long Short Term Memory (LSTM) cell, or a variant thereof.

The final prediction $Pr(w_t|w_{t-1}, ..w_1)$ is done using a simple fully connected layer with a softmax activation function:

$$\boldsymbol{y_t} = softmax(\boldsymbol{W}\boldsymbol{h_t} + \boldsymbol{b}). \quad (5)$$

Stacking multiple RNN layers To improve performance, it is common to stack multiple recurrent layers. To that end, the hidden state of a a layer l is used as an input for the next layer $l + 1$. Hence, the hidden state $\boldsymbol{h_{l,t}}$ at time step t of layer l is calculated as:

$$\boldsymbol{x_{l,t}} = \boldsymbol{h_{l-1,t}}, \quad (6)$$

$$\boldsymbol{h_{l,t}} = f_{\theta_l}(\boldsymbol{x_{l,t}}, \boldsymbol{h_{l,t-1}}). \quad (7)$$

An example of a two-layer stacked recurrent neural network is illustrated in Figure 1a.

However, stacking too many layers obstructs fluent backpropagation. Therefore, skip connections or residual connections are often added. The latter is in most cases a way to avoid increasing the size of the input of a certain layer (i.e., the inputs are summed instead of concatenated).

A skip connection can be defined as:

$$\boldsymbol{x_{l,t}} = [\boldsymbol{h_{l-1,t}}; \boldsymbol{h_{l-2,t}}] \quad (8)$$

while a residual connection is defined as:

$$\boldsymbol{x_{l,t}} = \boldsymbol{h_{l-1,t}} + \boldsymbol{h_{l-2,t}}. \quad (9)$$

Here, $\boldsymbol{x_{l,t}}$ is the input to the current layer as defined in Equation 7.

Densely connecting multiple RNN layers In analogy with DenseNets (Huang et al., 2017), a densely connected set of layers has skip connections from every layer to every other layer. Hence, the input to RNN layer l contains the hidden states of all lower layers at the same time step t, including the output of the embedding layer e_t:

$$\boldsymbol{x_{l,t}} = [\boldsymbol{h_{l-1,t}}; ...; \boldsymbol{h_{1,t}}; \boldsymbol{e_t}]. \quad (10)$$

Due to the limited number of RNN layers, there is no need for compression layers, as introduced for convolutional neural networks (Huang et al., 2017). Moreover, allowing the final classification layer to have direct access to the embedding layer showed to be an important advantage. Hence, the final classification layer is defined as:

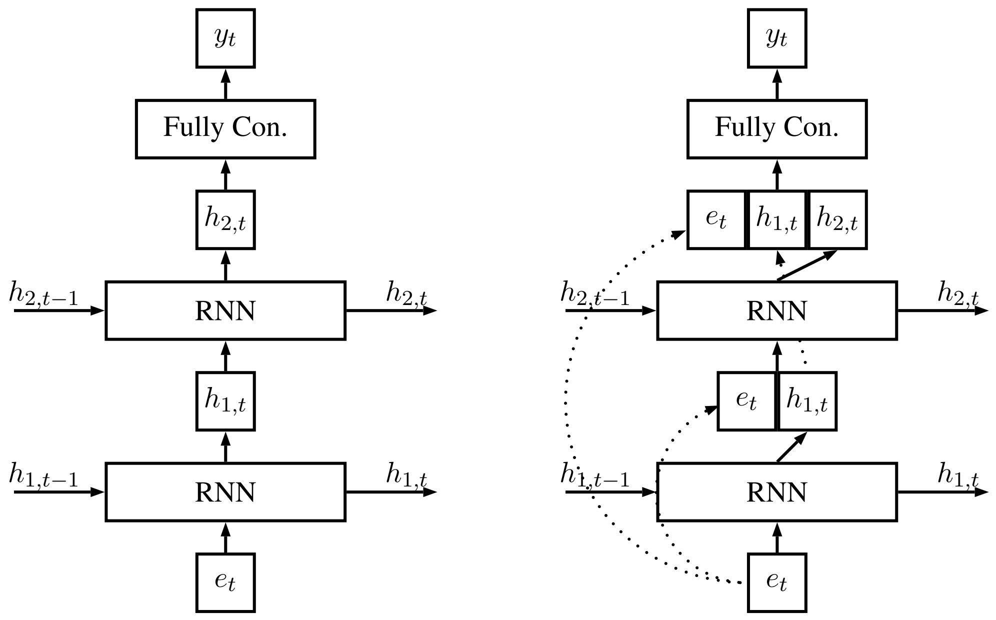
$$\boldsymbol{y_t} = softmax(\boldsymbol{W}[\boldsymbol{h_{L,t}}; ...; \boldsymbol{h_{1,t}}; \boldsymbol{e_t}] + \boldsymbol{b}). \quad (11)$$

An example of a two-layer densely connected recurrent neural network is illustrated in Figure 1b.

4 Experimental Setup

We evaluate our proposed architecture on the Penn Treebank (PTB) corpus. We adopt the standard train, validation and test splits as described in Mikolov and Zweig (2012), containing 930k training, 74k validation, and 82k test words. The vocabulary is limited to 10,000 words. Out-of-vocabulary words are replaced with an UNK token.

Our baseline is a stacked Long Short-Term Memory (LSTM) network, trained with regular dropout as introduced by Zaremba et al. (2014). Both the stacked and densely connected LSTM models consist of an embedding layer followed by a variable number of LSTM layers and a single fully connected output layer. While Zaremba et al. (2014) only report results for two stacked



(a) Stacked RNN

(b) Densely connected RNN

Figure 1: Example architecture for a model with two recurrent layers.

LSTM layers, we also evaluate a model with three stacked LSTM layers, and experiment with two, three, four and five densely connected LSTM layers. The hidden state size of the densely connected LSTM layers is either 200 or 650. The size of the embedding layer is always 200.

We applied standard dropout on the output of every layer. We used a dropout probability of 0.6 for models with size 200 and 0.75 for models with hidden state size 650 to avoid overfitting. Additionally, we also experimented with Variational Dropout (VD) as implemented in Inan et al. (2017). We initialized all our weights uniformly in the interval [-0.05;0.05]. In addition, we used a batch size of 20 and a sequence length of 35 during training. We trained the weights using standard Stochastic Gradient Descent (SGD) with the following learning rate scheme: training for six epochs with a learning rate of one and then applying a decay factor of 0.95 every epoch. We constrained the norm of the gradient to three. We trained for 100 epochs and used early stopping. The evaluation metric reported is perplexity as defined in Equation 3. The number of parameters reported is calculated as the sum of the total amount of weights that reside in every layer.

Note that apart from the exact values of some hyperparameters, the experimental setup is identical to Zaremba et al. (2014).

5 Discussion

The results of our experiments are depicted in Table 1. The first three results, marked with *stacked LSTM (Zaremba et al., 2014)*, follow the setup of Zaremba et al. (2014) while the other results are obtained following the setup described in the previous section.

The smallest densely connected model which only uses two LSTM layers and a hidden state size of 200, already reduces the perplexity with 20% compared to a two-layer stacked LSTM model with a hidden state size of 200. Moreover, increasing the hidden state size to 350 to match the amount of parameters the two-layer densely connected LSTM model contains, does not result in a similar perplexity. The small densely connected model still realizes a 9% perplexity reduction with an equal amount of parameters.

When comparing with Zaremba et al. (2014), the smallest densely connected model outperforms the stacked LSTM model with a hidden state size of 650. Moreover, adding one additional layer is enough to obtain the same perplexity as the best model used in Zaremba et al. (2014) with a hidden state size of 1500. However, our densely connected LSTM model only uses 11M parameters while the stacked LSTM model needs six times more parameters, namely 66M. Adding a fourth layer further reduces the perplexity to 76.8.

Table 1: Evaluation of densely connected recurrent neural networks for the PTB language modeling task.

Name	Hidden state size	# Layers	# Parameters	Valid	Test
Stacked LSTM (Zaremba et al., 2014)[1]	200	2	5M	104.5	100.4
	650	2	20M	86.2	82.7
	1500	2	66M	82.2	78.4
Stacked LSTM	200	2	5M	105	100.9
	200	3	5M	113	108.8
	350	2	9M	91.5	87.9
Densely Connected LSTM	200	2	9M	83.4	80.4
	200	3	11M	81.5	78.5
	200	4	14M	79.2	76.8
	200	5	17M	79.7	76.9
Densely Connected LSTM	650	2	23M	81.5	78.9
Dens. Con. LSTM + Var. Dropout	650	2	23M	81.3	78.3

Increasing the hidden state size is less beneficial compared to adding an additional layer, in terms of parameters used. Moreover, a dropout probability of 0.75 was needed to reach similar perplexity scores. Using variational dropout with a probability of 0.5 allowed us to slightly improve the perplexity score, but did not yield significantly better perplexity scores, as it does in the case of stacked LSTMs (Inan et al., 2017).

In general, adding more parameters by increasing the hidden state and performing subsequent regularization, did not improve the perplexity score. While regularization techniques such as variational dropout help improving the information flow through the layers, densely connected models solve this by adding skip connections. Indeed, the higher LSTM layers and the final classification layer all have direct access to the current input word and corresponding embedding. When simply stacking layers, this embedding information needs to flow through all stacked layers. This poses the risk that embedding information will get lost. Increasing the hidden state size of every layer improves information flow. By densely connecting all layers, this issue is mitigated. Outputs of lower layers are directly connected with higher layers, effectuating efficient information flow.

Comparison to other models In Table 2, we list a number of closely related models. A densely connected LSTM model with an equal number of parameters outperforms a combination of RNN, LDA and Kneser Ney (Mikolov and Zweig, 2012). Applying Variational Dropout (VD) (Inan et al., 2017) instead of regular dropout (Zaremba et al., 2014) can further reduce the perplexity score of stacked LSTMs, but does not yield satisfactory results for our densely connected LSTMs. However, a densely connected LSTM with four layers still outperforms a medium-sized VD-LSTM while using fewer parameters. Inan et al. (2017) also tie the input and output embedding together (cf. model VD-LSTM+REAL). This is, however, not possible in densely connected recurrent neural networks, given that the input and output embedding layer have different sizes.

[1]There are no results reported in Zaremba et al. (2014) for a small network with dropout. These are our own results, following the exact same setup as for the medium-sized architecture.

6 Conclusions

In this paper, we demonstrated that, by simply adding skip connections between *all* layer pairs of a neural network, we are able to achieve similar perplexity scores as a large stacked LSTM model (Zaremba et al., 2014), with six times fewer parameters for the task of language modeling. The simplicity of the skip connections allows them to act as an easy add-on for many stacked recurrent neural network architectures, significantly reducing the number of parameters. Increasing the size of the hidden states and variational dropout did not yield better results over small hidden states and regular dropout. In future research, we would like to investigate how to properly regularize larger models to achieve similar perplexity reductions.

Table 2: Comparison to other language models evaluated on the PTB corpus.

Model	**# Parameters**	**Perplexity Test**
RNN (Mikolov and Zweig, 2012)	6M	124.7
RNN+LDA+KN-5+Cache (Mikolov and Zweig, 2012)	9M	92.0
Medium stacked LSTM (Zaremba et al., 2014)	20M	82.7
Densely Connected LSTM (small - 2 layers)	**9M**	**80.4**
Char-CNN (Kim et al., 2016)	19M	78.9
Densely Connected LSTM (small - 3 layers)	**11M**	**78.5**
Large stacked LSTM (Zaremba et al., 2014)	66M	78.4
Medium stacked VD-LSTM (Inan et al., 2017)	20M	77.7
Densely Connected LSTM (small - 4 layers)	**14M**	**76.8**
Large stacked VD-LSTM (Inan et al., 2017)	66M	72.5
Large stacked VD-LSTM + REAL (Inan et al., 2017)	51M	68.5

Acknowledgments

The research activities as described in this paper were funded by Ghent University, imec, Flanders Innovation & Entrepreneurship (VLAIO), the Fund for Scientific Research-Flanders (FWO-Flanders), and the EU.

References

Yoshua Bengio. 2008. Neural net language models. *Scholarpedia* 3(1):3881. revision #91566.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*.

Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Hakan Inan, Khashayar Khosravi, and Richard Socher. 2017. Tying word vectors and word classifiers: A loss framework for language modeling. In *Proceedings of the International Conference on Learning Representations*.

Kazuki Irie, Zoltán Tüske, Tamer Alkhouli, Ralf Schlüter, and Hermann Ney. 2016. Lstm, gru, highway and a bit of attention: an empirical overview for language modeling in speech recognition. In *Proceedings of Interspeech*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Lus Marujo, and Tiago Lus. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the International Conference on Empirical Methods in Natural Language Processing*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *Proceedings of the International Conference on Learning Representations*.

Tomas Mikolov, Martin Karafit, Luks Burget, Jan Cernock, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of Interspeech*.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *Proceedings of the IEEE Workshop on Spoken Language Technology*.

Ofir Press and Lior Wolf. 2016. Using the output embedding to improve language models. *CoRR* abs/1608.05859. http://arxiv.org/abs/1608.05859.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the International Conference on Neural Information Processing Systems*.

Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. 2016. Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*.

Yonghui Wu, Mike Schuster, and Zhifeng Chen et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR* abs/1609.08144. http://arxiv.org/abs/1609.08144.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR* abs/1409.2329. http://arxiv.org/abs/1409.2329.