SIMPLIFYING LONG SHORT-TERM MEMORY ACOUSTIC MODELS FOR FAST TRAINING AND DECODING

Yajie Miao[1*], *Jinyu Li*[2], *Yongqiang Wang*[2], *Shi-Xiong Zhang*[2], *Yifan Gong*[2]

[1]Carnegie Mellon University, Pittsburgh, PA 15213
[2]Microsoft Corporation, One Microsoft Way, Redmond, WA 98052

ABSTRACT

On acoustic modeling, recurrent neural networks (RNNs) using Long Short-Term Memory (LSTM) units have recently been shown to outperform deep neural networks (DNNs) models. This paper focuses on resolving two challenges faced by LSTM models: high model complexity and poor decoding efficiency. Motivated by our analysis of the gates activation and function, we present two LSTM simplifications: deriving input gates from forget gates, and removing recurrent inputs from output gates. To accelerate decoding of LSTMs, we propose to apply frame skipping during training, and frame skipping and posterior copying (FSPC) during decoding. In the experiments, model simplifications reduce the size of LSTM models by 26%, resulting in a simpler model structure. Meanwhile, the application of FSPC speeds up model computation by 2 times during LSTM decoding. All these improvements are achieved at the cost of 1% WER degradation.

Index Terms— Long Short-Term Memory, recurrent neural network, model simplification, decoding efficiency

1. INTRODUCTION

The application of deep neural networks (DNNs) has achieved tremendous success for large vocabulary continuous speech recognition (LVCSR) [1, 2, 3]. As an alternative to DNNs, recurrent neural networks (RNNs) have also been studied extensively for the task of acoustic modeling [4]. RNNs are characterized by recurrent connections on the hidden layers which allow temporal information to be propagated through many time steps. More recently, deep RNNs using Long Short-Term Memory (LSTM) units (which we will consistently refer to as LSTMs) have been shown to outperform both DNNs and the conventional RNNs [5, 6, 7, 8]. As a type of RNNs, LSTMs are effective in modeling long-term temporal dependency without suffering from the optimization hurdles that plague RNNs [9]. Although performing promisingly, LSTMs have the drawback of a highly complex model structure. In addition to the memory cells which store the temporal states, LSTMs also contain multiplicative gates to control the information flow into and out of the memory cells. The model complexity increases the cost of both training and decoding of LSTM models. Another challenge for LSTMs (and also for the conventional RNNs) involves the difficulty in accelerating decoding. Decoding efficiency is a critical consideration when acoustic models are deployed in real-world applications. For DNNs, previous studies [10] have successfully applied *frame skipping and posterior copying* (FSPC) to speed up decoding. However, this FSPC approach is not applicable to LSTMs, since frame skipping breaks the original temporal dependency between neighbouring frames.

This paper focuses on tackling these two challenges. We present two simplifications to the LSTM structure. First, by visualizing statistics of the gates activation, the input and forget gates are observed to display a negative correlation. This motivates us to derive the activation of the input gates from the activation of the forget gates. As a result, the computation and model parameters attached to the input gates can be eliminated. Second, we exclude the recurrent inputs from the computation of the output gates. In the experiments, these two simplifications, when applied separately, reduce the model size by 16% and 10% respectively, with no word error rate (WER) loss incurred. They can be combined readily for further model size reduction. Furthermore, to accelerate decoding of LSTMs, we propose to apply frame skipping during training, and *frame skipping and posterior copying* (FSPC) during decoding. With this strategy, WER degradation caused by FSPC becomes negligible when every 1 out of 2 frame is skipped. Our final LSTM model, which integrates the proposed techniques, contains 26% less parameters than the baseline LSTM. Meanwhile, applying FSPC speeds up model computation by 2 times during decoding. All these improvements are achieved at the cost of 1% WER degradation.

2. VANILLA LSTMS

Fig. 1 depicts the vanilla LSTM that has been most commonly used in literature. The central component in the architecture is the memory cell. Three types of multiplicative gates (input, output, forget) are added to control the flow of information.

[*]Research conducted as an intern at Microsoft.

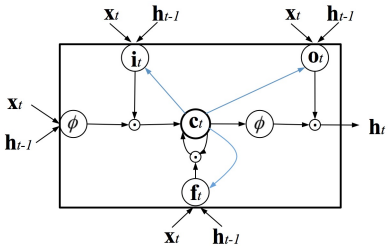

Fig. 1. A memory block of the vanilla LSTM.

Furthermore, the LSTM is enriched with peephole connections [11] that link the memory cells to the gates to learn precise timing of the outputs. The LSTM outputs are recurrently fed as the inputs. At the time step t, the vector formulas of the computation can be described as:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{p}_i\mathbf{c}_{t-1} + \mathbf{b}_i) \quad (1a)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{p}_f\mathbf{c}_{t-1} + \mathbf{b}_f) \quad (1b)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \phi(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) \quad (1c)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{p}_o\mathbf{c}_t + \mathbf{b}_o) \quad (1d)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \phi(\mathbf{c}_t) \quad (1e)$$

where $\mathbf{x}_t$ is the input vector. The vectors $\mathbf{i}_t$, $\mathbf{o}_t$, $\mathbf{f}_t$ are the activation of the input, output, and forget gates respectively. The $\mathbf{W}_{.x}$ terms are the weight matrices for the inputs $\mathbf{x}_t$, the $\mathbf{W}_{.h}$ are the weight matrices for the recurrent inputs $\mathbf{h}_{t-1}$, and the $\mathbf{b}_.$ terms are the bias vectors. The $\mathbf{p}_i$, $\mathbf{p}_o$, $\mathbf{p}_f$ are parameter vectors associated with peephole connections. The functions σ and ϕ are the logistic sigmoid and hyperbolic tangent nonlinearity respectively. The operation $\odot$ represents element-wise multiplication of vectors.

A criticism of LSTMs lies in their highly complicated model structure. The large number of components have complex interactions where redundancy may exist. Therefore, the vanilla LSTM may not be the optimal architecture for particular tasks, e.g., acoustic modeling. Attempts [12, 13, 14] have been made to simplify the LSTMs, e.g., the Gated Recurrent Unites (GRU) [12]. However, these proposals are primarily based on extensive architecture search, and justifying the identified architecture remains difficult.

3. SIMPLIFIED LSTMS

In this section, we present two simplifications to the vanilla LSTMs, specifically on the task of large-scale acoustic modeling. These simplifications are based on our analyzing the activation of the gates and identifying potential redundancy in the vanilla LSTM structure.

3.1. Deriving Input Gates from Forget Gates

The first simplification is motivated by visualization of the gates activation. On our experiment setup (Section 5), we train a vanilla LSTM model as reviewed in Section 2. The trained model is then applied to some development data for a forward pass. We dump the activation of the gates in the highest LSTM layer, and visualize their statistics by following the method in [15]. Specifically, a gate is defined to be right or left-saturated if its activation is greater than 0.9 or smaller than 0.1 respectively. For each gate, we compute two statistics: the fraction of the data points on which this gate is right-saturated and the faction on which it is left-saturated. These two statistics serve as the coordinates to position the gate on a 2-dimensional plane. In Fig. 2, the 1024 input gates are depicted on the left and the 1024 forget gates on the right.

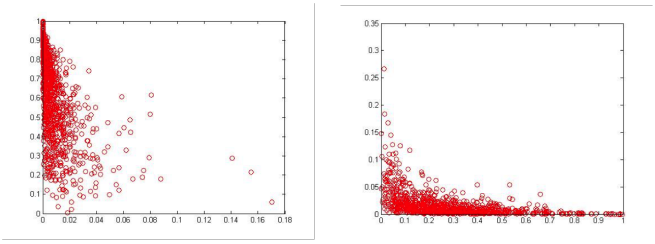

Fig. 2. **Left:** Saturation plot for 1024 input gates. **Right:** Saturation plot for 1024 forget gates. Each gate is represented by a red circle.

These visualizations reveal the distributions of the gates activation in the saturation regime. We can see that the activations of the input and forget gates are not entirely independent. Instead, they roughly show a negative correlation. The input gate activations distribute along the vertical axis, and the forget gate activations along the horizontal axis. This negative correlation also complies with the design purpose of the gates in the vanilla LSTMs: the input gates control how much of *new information* is absorbed from the inputs to update the memory cell states, whereas the forget gates control how much of the *old information* is kept in the memory cell.

Such an observation motivates us to couple the input and forget gates, rather than compute them independently. Meanwhile, the forget gates have been previously proved to be indispensable for the LSTMs to function [16]. We thus propose to derive the input gates $\mathbf{i}_t$ from the forget gates $\mathbf{f}_t$. The computation of the input gates is simplified from Eq. (1a) to:

$$\mathbf{i}_t = 1 - \mathbf{f}_t \quad (2)$$

This simplification eliminates the model parameters (and the computation) of the input gates in the vanilla LSTMs. The direct mapping from the forget to the input gates, however, is likely to be too arbitrary. To allow for more flexibility, we add a weight vector $\mathbf{w}_{if}$ to $1 - \mathbf{f}_t$, which gives us another variant of the simplification:

$$\mathbf{i}_t = \mathbf{w}_{if} \odot (1 - \mathbf{f}_t) \quad (3)$$

where $\odot$ is element-wise multiplication. The weight vector $\mathbf{w}_{if}$ is learned together with other model parameters during

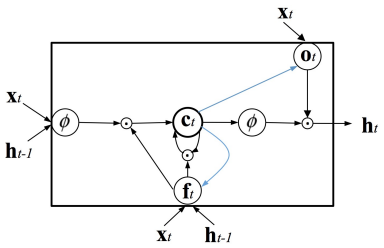

Fig. 3. The structure of the simplified LSTM (sLSTM).

training. The increase of model parameters coming from the introduction of $\mathbf{w}_{if}$ is negligible. For simplicity of formulation, we refer to these two variants as *iFromf* and *iFromf_w* respectively.

3.2. Removing the Recurrent Inputs from Output Gates

The second simplification involves the computation of the output gates. In the vanilla LSTMs, the output gates, as with the input and forget gates, rely on both the original inputs $\mathbf{x}_t$ and the recurrent inputs $\mathbf{h}_{t-1}$. However, we argue that it is redundant to include $\mathbf{h}_{t-1}$ in the output-gate computation. This is because the output gates are placed at the end of the LSTM computation, simply acting to regulate the memory cell states into the final LSTM outputs. Since the history information has been sufficiently modeled by the memory cells, including $\mathbf{h}_{t-1}$ in the output gates brings no additional benefits. As a result, we remove $\mathbf{h}_{t-1}$ from the output gates, and thereby simplify the computation to Eq. (4). We refer this variant as NoOH.

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{p}_o\mathbf{c}_t + \mathbf{b}_o) \tag{4}$$

The two aforementioned simplifications can be integrated into the final *simplified LSTM* (sLSTM) model whose structure is depicted in Fig. 3.

4. FAST DECODING OF LSTMS

Decoding efficiency is a critical challenge for real-world deployment of LSTMs. For DNNs, an effective strategy to accelerate decoding is the *frame skipping and posterior copying* (FSPC) technique [10]. In [17], this frame sub-sampling idea is employed for fast training of connectionist temporal classification (CTC) models with CD phones. Fig. 4 shows how FSPC is ported to LSTMs. During decoding, we are not performing LSTM computation on every frame. Instead, we compute the model on selected frames (one frame out of every two in Fig. 4) and skip the remaining frames. The senone posteriors of a skipped frame are copied from the posteriors of the nearest non-skipped frame before it. Note that skipping a frame means skipping the computation of the entire model, not just the computation of the softmax classification layer.

However, direct application of FSPC to LSTM decoding results in severe degradation. Empirical evidence can

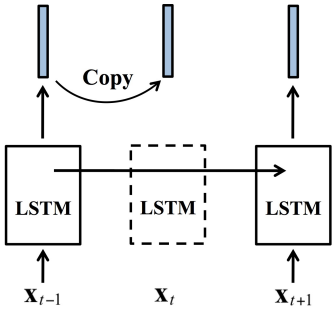

Fig. 4. Application of FSPC to decoding of LSTMs. The blue bar represents the senone posteriors at each frame.

be found in Section 5.2. This is because frame skipping breaks the original temporal dependency between neighbouring frames ($\mathbf{x}_{t-1}$ and $\mathbf{x}_t$, $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$), and artificially establishes dependency between $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t+1}$. This dependency pattern is mismatched with the dependency modeled by the LSTM during training. To mitigate this mismatch, we propose to train the LSTMs also with frame skipping. On the training data, frame skipping is carried out to split each training utterance into multiple shorter utterances. The senone labels of the frames are selected accordingly. The configuration (the number of frames to be skipped) of frame skipping on the training data has to be consistent with that on the decoding data. For instance, with 1 frame skipped, the odd and even frames of every training utterance are selected into two separate utterances. This doubles the number of training utterances, but keeps the amount of training data unchanged.

5. EXPERIMENTS

The proposed model simplifications and decoding acceleration are evaluated using a Microsoft internal Windows Phone short message dictation task. The training data consist of 375 hours of US-English speech, and the test set contains 125k words. The input features to LSTMs are 29-dimension log-filter-bank features with their first and second-order derivatives. The system has 6k senones which have been determined by the GMM/HMM system. Training of LSTMs is carried out with the CNTK toolkit [18]. Our baseline LSTM model contains 4 LSTM layers which are followed by a fully-connected layer for classification. With a projection layer inserted [6], every LSTM layer has 1024 memory cells and 512 output units (the dimension of the projection layer). The baseline LSTM model amounts to 20 million parameters and achieves the WER of 15.35% on the test set. A standard 5-layer DNN with 2048 nodes in each hidden layer has 17.13% WER.

5.1. Experiments on Model Simplifications

The comparisons of the various LSTM variants are shown in Table 1. It is worth noting that the first simplification

(Section 3.1) is applied only to the higher 3 LSTM hidden layers. In the lowest LSTM layer, the input gates are always independent of the forget gates. This allows the temporal dependency from the raw acoustic features to be modeled more reliably. The NoOH simplification is applied to all the LSTM layers. We observe that the two types of simplifications, when used separately, cause no WER degradation. Meanwhile, they bring 16% and 10% model size reduction respectively, compared to the baseline model with the vanilla LSTM. The two variants of the first simplification perform comparably in terms of WERs. However, the training of iFromf_w is found to converge faster than the training of iFromf, partly because the weight vector $\mathbf{w}_{if}$ in Eq. (3) enables the gates coupling to be learned more easily. The simplified model sLSTM, which combines iFromf_w and NoOH, gets a 26% model size reduction and negligible WER loss (from 15.32% to 15.42%).

For more complete evaluations, we also show the results of three comparison models. Instead of coupling the input and forget gates, *NoI* removes the input gates entirely. That is, the activation of the input gates is constantly set to 1. The WER in this case rises to 16.03%. In contrast, our iFromf model is more effective due to the compensation of the input gates from the forget gates. Another comparison model consists of 3 (instead of 4) vanilla LSTM layers and reduces the size of the baseline model by 23%. However, this model gives a worse WER (16.04%) than the sLSTM. This indicates that the simplifications proposed in this work simplify LSTMs from a structural perspective, instead of purely reducing model parameters to prevent overfitting in the training stage. We finally compared the sLSTM model with the GRU [12]. The GRU network contains 4 GRU layers each of which has 700 cells. Table 1 shows that applying GPUs gets the comparable model size reduction to the sLSTM. However, the WER deteriorates to 17.57%, a 14% degradation from the baseline.

Table 1. *Comparisons of the various LSTM variants. MSR refers to the percentage of model size reduction compared to the baseline LSTM model.*

LSTM Variants	WER%	MSR
Baseline (4 vanilla LSTM layers)	15.35	0%
sLSTM(iFromf)	15.32	16%
sLSTM(iFromf_w)	15.29	16%
sLSTM(NoOH)	15.29	10%
sLSTM(iFromf_w + NoOH)	15.42	**26%**
sLSTM(NoI)	16.03	16%
3 vanilla LSTM layers	16.04	23%
GRU	17.57	27%

5.2. Experiments on Fast Decoding with FSPC

In this subsection, we employ the vanilla LSTMs without any simplifications to examine the effectiveness of FSPC in speeding up decoding. The results are presented in Table 2. Applying FSPC directly to the original LSTM model degrades the WER dramatically (up to 60% with 1 frames skipped), which is consistent with our postulation in Section 3. In contrast, when the LSTM model is trained with frame skipping, FSPC starts to perform reasonably well. In particular, with 1 frame skipped, FSPC can accelerate the stage of model computation by 2 times during decoding, while degrading the WER by 1% (15.54% vs 15.35%).

Table 2. *%WER of the baseline LSTM model when FSPC is applied for decoding acceleration.*

Configuration	Skip 1 Frame	Skip 2 Frame
FSPC in Decoding	59.79	92.71
Frame Skipping in Training & FSPC in Decoding	15.54	16.03

5.3. Combining Simplified LSTMs and FSPC

We finally integrate model simplifications and FSPC into a unified setup. The sLSTM(iFromf_w + NoOH) model is trained with frame skipping, and decoded using the FSPC technique. This final model, as stated in Section 5.1, contains 26% less parameters than the baseline LSTM. This roughly saves 26% runtime computational cost. Meanwhile, when 1 frame is skipped, FSPC further speeds up model computation by 2 times during decoding. Table 3 shows that all these benefits are achieved at the cost of 1% WER degradation.

Table 3. *Comparisons between the baseline LSTM and our final sLSTM model. The sLSTM is trained with frame skipping and decoded with FSPC.*

LSTM Variants	FSPC Setting	WER%
Baseline LSTM	Skip 0 Frame	15.35
sLSTM(iFromf_w + NoOH)	Skip 1 Frame	15.52

6. CONCLUSIONS AND FUTURE WORK

This paper proposes two simplifications to the highly complex LSTM acoustic models. These simplifications reduce the size of LSTM models significantly, while at the same time causing no performance degradation. Also, to accelerate decoding, we present a simple yet effective strategy: train the LSTM models with frame skipping, and decode them with FSPC. By adopting these techniques, our final acoustic model enjoys a simpler structure and faster decoding speed. Currently, we have successfully applied singular value decomposition (SVD) of weight matrices, as proposed in [19], to LSTMs. It is straightforward to combine SVD together with the proposed simplification for further model size reduction. Also, we are interested to apply the simplified LSTMs as acoustic models in end-to-end ASR pipelines [20].

7. REFERENCES

[1] George E Dahl, Dong Yu, Li Deng, and Alex Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 30–42, 2012.

[2] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, 2012.

[3] Yajie Miao, Hao Zhang, and Florian Metze, "Speaker adaptive training of deep neural network acoustic models using i-vectors," *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, vol. 23, no. 11, pp. 1938–1949, 2015.

[4] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 6645–6649.

[5] Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*. IEEE, 2013, pp. 273–278.

[6] Hasim Sak, Andrew Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2014.

[7] Xiangang Li and Xihong Wu, "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015.

[8] Yajie Miao and Florian Metze, "On speaker adaptation of long short-term memory recurrent neural networks," in *Sixteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2015.

[9] Yoshua Bengio, Patrice Simard, and Paolo Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[10] Vincent Vanhoucke, Matthieu Devin, and Georg Heigold, "Multiframe deep neural networks for acoustic modeling," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 7582–7585.

[11] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber, "Learning precise timing with LSTM recurrent networks," *The Journal of Machine Learning Research*, vol. 3, pp. 115–143, 2003.

[12] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[13] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber, "LSTM: A search space odyssey," *arXiv preprint arXiv:1503.04069*, 2015.

[14] Thomas M Breuel, "Benchmarking of LSTM networks," *arXiv preprint arXiv:1508.02774*, 2015.

[15] Andrej Karpathy, Justin Johnson, and Fei-Fei Li, "Visualizing and understanding recurrent networks," *arXiv preprint arXiv:1506.02078*, 2015.

[16] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever, "An empirical exploration of recurrent network architectures," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015, pp. 2342–2350.

[17] Haşim Sak, Andrew Senior, Kanishka Rao, and Françoise Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," in *Sixteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2015.

[18] Dong Yu, Adam Eversole, Mike Seltzer, Kaisheng Yao, Zhiheng Huang, et al., "An introduction to computational networks and the computational network toolkit," Tech. Rep., Tech. Rep. MSR, Microsoft Research, 2014, https://cntk.codeplex.com/, 2014.

[19] Jian Xue, Jinyu Li, and Yifan Gong, "Restructuring of deep neural network acoustic models with singular value decomposition.," in *Fourteenth Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2013.

[20] Yajie Miao, Mohammad Gowayyed, and Florian Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on*. IEEE, 2015.