Compact Recurrent Neural Network based on Tensor Train for Polyphonic Music Modeling

Andros Tjandra, Sakriani Sakti, Satoshi Nakamura
Graduate School of Information Science
Nara Institute of Science and Technology
{andros.tjandra.ai6, ssakti, s-nakamura}@is.naist.jp

Abstract

This paper introduces a novel compression method for recurrent neural networks (RNNs) based on Tensor Train (TT) format. The objective in this work are to reduce the number of parameters in RNN and maintain their expressive power. The key of our approach is to represent the dense matrices weight parameter in the simple RNN and Gated Recurrent Unit (GRU) RNN architectures as the n-dimensional tensor in TT-format. To evaluate our proposed models, we compare it with uncompressed RNN on polyphonic sequence prediction tasks. Our proposed TT-format RNN are able to preserve the performance while reducing the number of RNN parameters significantly up to 80 times smaller.

1 Introduction

Recurrent neural networks have recently become a popular choice in machine learning for modeling temporal and sequential tasks. Despite it has been studied for about two decades [Elman, 1990, Hochreiter and Schmidhuber, 1997], people interest on RNN has been growing lately thanks to the significant improvement of current computational power and the amount of available data. Many state-of-the-arts in speech recognition [Hannun et al., 2014, Amodei et al., 2015] and machine translation [Wu et al., 2016, Bahdanau et al., 2014, Sutskever et al., 2014] has been achieved by RNNs.

Despite the fact that current RNN has a good expressive power, most RNN models are computationally expensive and have a huge number of parameters. Since RNNs are constructed by multiple linear transformations followed by nonlinear transformations, we need multiple high-dimensional dense matrices as parameters. In time-steps, we need to apply multiple linear transformations between our dense matrix with high-dimensional input and previous hidden states. Especially for state-of-the-art models on speech recognition [Amodei et al., 2015] and machine translation [Wu et al., 2016], such huge models can only be implemented in high-end cluster environments because they need massive computation power and millions of parameters. This limitation prevent us to create efficient RNN models that are fast enough for massive real-time inference or small enough to be implemented in low-end devices like mobile phones [Schuster, 2010] or embedded systems with limited memory.

A number of researchers have done notable work to minimize the accuracy loss and maximize the model efficiency, trying to balance the trade-off between high performance and smaller model. Hinton et al. [2015] and Ba and Caruana [2014] successfully compressed a large deep neural network into a smaller neural network by training the latter on the transformed softmax outputs from the former. Distilling knowledge from larger neural networks has also been successfully applied to recurrent neural network architecture by [Tang et al., 2016]. Denil et al. [2013] utilized low-rank matrix decomposition to represent the weight matrices. A recent study by Novikov et al. [2015] replaced the dense weight matrices with Tensor Train (TT) format [Oseledets, 2011] inside convolutional neural network (CNN) model. With the TT-format, they significantly compress the number of parameters

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

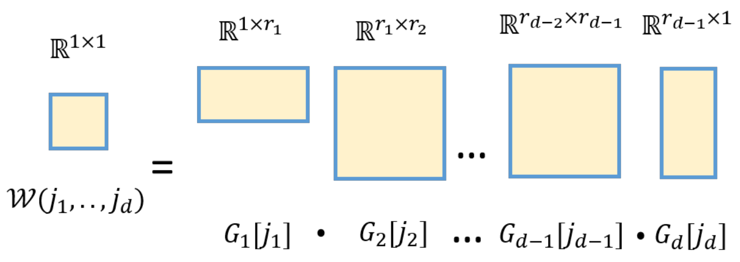
$\mathbb{R}^{1\times 1}$
$\mathbb{R}^{1\times r_1}$
$\mathbb{R}^{r_1\times r_2}$
$\mathbb{R}^{r_{d-2}\times r_{d-1}}$
$\mathbb{R}^{r_{d-1}\times 1}$
$\mathcal{W}(j_1,..,j_d)$ =
$G_1[j_1]$ • $G_2[j_2]$ ... $G_{d-1}[j_{d-1}]$ • $G_d[j_d]$

Figure 1: Illustration of Eq.1: Calculating an element $\mathcal{W}(j_1, .., j_k)$ using set of TT-cores $\{G_k[j_k]\}_{k=1}^{d}$

and kept the model accuracy degradation to a minimum. However, to the best of our knowledge, no study has focused on compressing more complex neural networks such as RNNs with a tensor-based representation.

In this work, we introduce our recent work on a novel RNN architecture called as TT-RNN [Tjandra et al., 2017] [1], which is an RNN based on TT-format. We apply TT-format to reformulate two different RNNs: a simple RNN and a GRU RNN. Our proposed RNN architectures are evaluated on polyphonic sequence modeling. In section 2 we describe our proposed TT-RNN. In section 3, we describe the tasks and datasets, followed by the experimental results. Finally, we summarize our result in Section 4.

2 Compressing RNN with TT-format

We can represent d-dimensional array (tensor) $\mathcal{W}$ [2] in TT-format [Oseledets, 2011] if for each $k \in \{1, .., d\}$ and for each possible value of the k-th dimension index $j_k \in \{1, .., n_k\}$ there exists a matrix $G_k[j_k]$ such that all elements of $\mathcal{W}$ can be computed (illustrated in Fig. 1) by the following equation:

$$\mathcal{W}(j_1, j_2, .., j_d) = G_1[j_1] \cdot G_2[j_2] \cdot .. \cdot G_d[j_d]. \tag{1}$$

For all matrices $G_k[j_k]$ related to the same dimension k, they must be represented with size $r_{k-1} \times r_k$, where r_0 and r_d must be equal to 1 to retain the final matrix multiplication result as a scalar. In TT-format, we define a sequence of rank $\{r_k\}_{k=0}^{d}$ and we call them TT-rank from tensor $\mathcal{W}$. The set of matrices $\mathcal{G}_k = \{G_k[j_k]\}_{j_k=1}^{n_k}$ where the matrices are spanned in the same index are called TT-core.

To create an intuitive example on how to represent a tensor using a set of TT-cores, we illustrate how to represent a tensor $\mathcal{W}$ element at $(1, 0, 3)$ with 3 TT-cores in Figure 2.

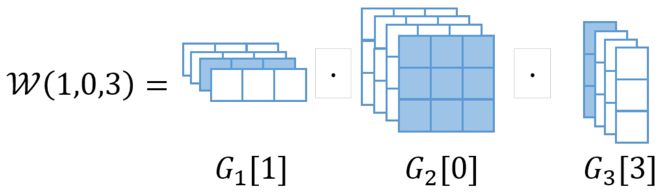

Figure 2: Representing a tensor $\mathcal{W}$ element at $(1, 0, 3)$ using 3 TT-cores $\mathcal{G}_1, \mathcal{G}_2$ and $\mathcal{G}_3$. Blue shaded vectors or matrices are used for chain multiplication

Inside RNN architecture, we focus our attention on compressing two dense weight matrices: W_{xh}, W_{hh}. We define $W_{xh} \in \mathbb{R}^{M\times N}$ as input-to-hidden and $W_{hh} \in \mathbb{R}^{M\times M}$ as hidden-to-hidden weight matrices.

[1]The long version of this paper with title "Compressing Recurrent Neural Network with Tensor Train" has been presented at 30th International Joint Conference on Neural Network (IJCNN) 2017

[2]In this paper, we denote d-dimensions tensor with calligraphic uppercase letter (e.g., $\mathcal{W}$) and matrix with standard uppercase letter (e.g., W)

First, we factorize matrix shape M into $\prod_{k=1}^{d} m_k$ and N into $\prod_{k=1}^{d} n_k$. Next, we determine the TT-rank $\{r_k\}_{k=0}^{d}$ for our model and substitute W_{xh} with tensor $\mathcal{W}_{xh}$ and W_{hh} with tensor $\mathcal{W}_{hh}$. Tensor $\mathcal{W}_{xh}$ is represented by set of TT-cores $\{\mathcal{G}_k^{xh}\}_{k=1}^{d}$ where $\forall k \in \{1,..,d\}$, $\mathcal{G}_k^{xh} \in \mathbb{R}^{m_k \times n_k \times r_{k-1} \times r_k}$, and tensor $\mathcal{W}_{hh}$ is represented by set of TT-cores $\{\mathcal{G}_k^{hh}\}_{k=1}^{d}$ where $\forall k \in \{1,..,d\}$, $\mathcal{G}_k^{hh} \in \mathbb{R}^{m_k \times m_k \times r_{k-1} \times r_k}$. We define bijective functions $\mathbf{f}_i^x$ and $\mathbf{f}_i^h$ to access row p from W_{xh} and W_{hh} in the set of TT-cores. We rewrite our simple RNN formulation to calculate h_t:

$$a_t^{xh}(p) = \sum_{j_1,\dots,j_d} \mathcal{W}_{xh}(\mathbf{f}_i^x(p), [j_1,..,j_d]) \cdot \mathcal{X}_t\,(j_1,..,j_d) \tag{2}$$

$$a_t^{hh}(p) = \sum_{j_1,\dots,j_d} \mathcal{W}_{hh}(\mathbf{f}_i^h(p), [j_1,..,j_d]) \cdot \mathcal{H}_{t-1}\,(j_1,..,j_d) \tag{3}$$

$$a_t^{xh} = \left[a_t^{xh}(1),..,a_t^{xh}(M)\right] \tag{4}$$

$$a_t^{hh} = \left[a_t^{hh}(1),..,a_t^{hh}(M)\right] \tag{5}$$

$$h_t = f(a_t^{xh} + a_t^{hh} + b_h), \tag{6}$$

where $\mathcal{X}$ is the tensor representation of input x_t and $\mathcal{H}_{t-1}$ is the tensor representation of previous hidden states h_{t-1}.

For more complex and powerful RNNs such as GRU-RNN [Chung et al., 2014], instead of two dense weight matrices, we have six dense weight matrices: (W_{xr}, W_{hr}, W_{xz}, W_{hz}, W_{xh}, and W_{hh}). First, we substitute all six dense matrices with tensor ($\mathcal{W}_{xr}$, $\mathcal{W}_{hr}$, $\mathcal{W}_{xz}$, $\mathcal{W}_{hz}$, $\mathcal{W}_{xh}$, $\mathcal{W}_{hh}$). Tensors $\mathcal{W}_{xr}$, $\mathcal{W}_{xz}$, $\mathcal{W}_{xh}$ are represented by a set of TT-cores ($\{\mathcal{G}_k^{xr}\}_{k=1}^{d}$, $\{\mathcal{G}_k^{xz}\}_{k=1}^{d}$, $\{\mathcal{G}_k^{xh}\}_{k=1}^{d}$) where $\forall k \in \{1,..,d\}$, ($\mathcal{G}_k^{xr}, \mathcal{G}_k^{xz}, \mathcal{G}_k^{xh} \in \mathbb{R}^{m_k \times n_k \times r_{k-1} \times r_k}$). Tensor $\mathcal{W}_{hr}$, $\mathcal{W}_{hz}$, $\mathcal{W}_{hh}$ are represented by a set of TT-cores ($\{\mathcal{G}_k^{hr}\}_{k=1}^{d}$, $\{\mathcal{G}_k^{hz}\}_{k=1}^{d}$, $\{\mathcal{G}_k^{hh}\}_{k=1}^{d}$) where $\forall k \in \{1,..,d\}$, ($\mathcal{G}_k^{hr}, \mathcal{G}_k^{hz}, \mathcal{G}_k^{hh} \in \mathbb{R}^{m_k \times m_k \times r_{k-1} \times r_k}$). By using similar formulation in Equations (2) to (6), we calculate the value for reset gate r_t, update gate u_t and candidate hidden state h_t. For more detailed explanation, please refer to the long version of our paper Tjandra et al. [2017][3].

3 Experiment

In this section, we evaluate our proposed RNN model with TT-formats (TT-SRNN and TT-GRU) and compare them to baseline RNNs (a simple RNN and GRU). We conducted the experiments on sequence prediction tasks, where we predicted the next time-step based on previous information [Graves et al., 2012]. We used polyphonic music datasets for the sequence prediction task. We train our model using BPTT and update our weight parameters with Adam optimizer [Kingma and Ba, 2014].

3.1 Polyphonic Music Dataset

We used four polyphonic music datasets [Boulanger et al., 2012]: Piano-midi.de, Nottingham, MuseData, and JSB Chorales. All of these datasets have 88 binary values per time-step, and each consists of at least seven hours of polyphonic music. Before we fed our input into the RNN, we projected them using hidden layer with 256 hidden units. In the polyphonic modeling task, we measured two different metrics: negative log-likelihood (NLL) and accuracy (ACC). We calculate the accuracy with following equation: $ACC = TP/(TP + FP + FN)$. We only used true positive (TP), false positive (FP), false negative (FN) and ignored the true negative (TN) because most of the notes were turned off in the dataset.

3.2 Model Description

Our baseline models consist of a simple RNN with 512 hidden units and a GRU RNN with 512 hidden units. Our proposed models are TT-SRNN and TT-GRU with $8 \times 4 \times 4 \times 4$ (total 512 hidden units) and $8 \times 4 \times 8 \times 4$ (total 1024 hidden units) output shapes and the TT-ranks (3, 5). For our experiment reports, we simplified the model description as follows: RNN-H★ where ★ denotes the

[3]`https://arxiv.org/abs/1705.08052`

number of hidden units (e.g., RNN-H512 means RNN with 512 hidden units) and TT-SRNN-H★-R♦ where ♦ denotes the TT-rank (e.g., TT-SRNN-H8x4x4x4-R3 means TT-SRNN with hidden units reshaped into 8x4x4x4 in TT-format and TT-rank 3). We used a grid search to determine the best number of hidden layer units for both models and the shape of TT-format based on the validation set performance.

3.3 Result

Table 1 lists all of the results of our experiments on the baseline and proposed models. We repeat all experiments five times with different weight parameters initialization.

The table shows that all of these models have similar performances based on the negative log-likelihood and the accuracy in the test sets. Our proposed models were able to reduce the number of parameters with significant compression ratio and preserved the performance at the same time.

Table 1: Compression rate, negative log-likelihood (NLL) and accuracy (ACC) for all polyphonic music test sets

Model	Params	Compr.	Nottingham		PianoMidi		MuseData		JSB Chorales	
			NLL	ACC	NLL	ACC	NLL	ACC	NLL	ACC
RNN-H512	393728	1	3.52±0.03	69.9±0.1	7.67±0.02	26.6±0.4	7.37±0.02	35.1±0.6	8.39±0.01	29.3±0.2
TT-SRNN H8x4x4x4-R3	1472	267.48	3.78±0.05	68.4±0.3	7.73±0.03	27.3±0.3	7.68±0.01	32.5±0.4	8.51±0.03	28.5±0.2
TT-SRNN H8x4x4x4-R5	2752	143.07	3.65±0.04	69.0±0.2	7.72±0.01	27.3±0.3	7.63±0.07	33.1±0.7	8.49±0.03	28.8±0.3
GRU-H512	1181184	1	3.39±0.02	71.1±0.2	7.56±0.01	25.7±0.2	7.10±0.02	36.7±0.4	8.23±0.01	30.3±0.1
TT-GRU H8x4x4x4-R3	4416	267.48	3.77±0.06	68.4±0.2	7.61±0.01	27.1±0.5	7.61±0.01	32.1±0.4	8.45±0.02	28.8±0.3
TT-GRU H8x4x4x4-R5	8256	143.07	3.64±0.08	69.2±0.4	7.59±0.01	27.4±0.3	7.50±0.09	33.2±1.0	8.47±0.01	28.5±0.4
RNN-H1024	1311744	1	3.39±0.01	71.0±0.2	7.68±0.01	27.1±0.6	7.23±0.01	36.1±0.3	8.45±0.02	28.6±0.1
TT-SRNN H8x4x8x4-R3	2560	512.4	3.59±0.03	69.5±0.3	7.72±0.04	27.8±0.4	7.69±0.02	32.9±0.4	8.56±0.05	28.8±0.3
TT-SRNN H8x4x8x4-R5	4864	269.68	3.54±0.01	69.7±0.2	7.68±0.03	27.5±0.4	7.57±0.1	33.4±0.9	8.55±0.03	28.6±0.5
GRU-H1024	3935232	1	3.37±0.02	71.1±0.1	7.54±0.01	26.1±0.3	7.00±0.01	37.1±0.3	8.21±0.02	30.7±0.3
TT-GRU H8x4x8x4-R3	7680	512.4	3.52±0.04	69.9±0.3	7.61±0.01	26.8±0.4	7.51±0.1	33.1±0.5	8.50±0.04	28.6±0.3
TT-GRU H8x4x8x4-R3	14592	269.68	3.48±0.04	70.4±0.3	7.59±0.01	27.5±0.2	7.44±0.15	35.0±1.0	8.48±0.02	28.5±0.3
RNN [4]	-	-	4.46	62.93	8.37	19.33	8.13	23.25	8.71	28.46
RNN-RBM [3]	-	-	2.39	75.40	7.09	28.92	6.01	34.02	6.27	33.12

4 Conclusion

In this paper, we presented an efficient and compact RNN model using a TT-format representation. Using a TT-format, we represented dense weight matrices inside the RNN layer with multiple low-rank tensors. Our proposed TT-SRNN and TT-GRU significantly compressed the number of parameters while simultaneously retaining the model performance and accuracy. We evaluated our model with sequence prediction tasks. On the sequence prediction task, we evaluated our model with multiple music datasets, and our proposed RNNs reduced the RNN parameters up to 80 times smaller while preserving the performance.

Acknowledgments

Part of this work was supported by JSPS KAKENHI Grant Numbers JP17H06101 and JP17K00237.

References

D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, et al. Deep speech 2: End-to-end speech recognition in English and Mandarin. *arXiv preprint arXiv:1512.02595*, 2015.

[4]Experiment result from Boulanger et al. [2012]

J. Ba and R. Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

N. Boulanger, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In J. Langford and J. Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1159–1166, New York, NY, USA, 2012. ACM. URL `http://icml.cc/2012/papers/590.pdf`.

J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

M. Denil, B. Shakibi, L. Dinh, N. de Freitas, et al. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, pages 2148–2156, 2013.

J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

A. Graves et al. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.

A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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

D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov. Tensorizing neural networks. In *Advances in Neural Information Processing Systems*, pages 442–450, 2015.

I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011. doi: 10.1137/090752286. URL `http://dx.doi.org/10.1137/090752286`.

M. Schuster. Speech recognition for mobile devices at Google. In *Pacific Rim International Conference on Artificial Intelligence*, pages 8–10. Springer, 2010.

I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

Z. Tang, D. Wang, and Z. Zhang. Recurrent neural network training with dark knowledge transfer. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5900–5904. IEEE, 2016.

A. Tjandra, S. Sakti, and S. Nakamura. Compressing recurrent neural network with tensor train. *arXiv preprint arXiv:1705.08052*, 2017.

Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.