On the Effectiveness of Low-Rank Matrix Factorization for LSTM Model Compression

Genta Indra Winata, Andrea Madotto, Jamin Shin, Elham J. Barezi, Pascale Fung

PACLIC 2019, Hakodate

RNN: the performance

Recurrent Neural Networks have become the core of many models for tasks that capture temporal dependency. RNN-based models achieve **impressive performance** in many NLP tasks.

RNN: the performance

Recurrent Neural Networks have become the core of many models for tasks that capture temporal dependency. RNN-based models achieve **impressive performance** in many NLP tasks.

Speed

There is an obvious drawback of scalability that accompanies these excellent performances. **Larger capacity, lower speed!**

RNN: the performance

Recurrent Neural Networks have become the core of many models for tasks that capture temporal dependency. RNN-based models achieve **impressive performance** in many NLP tasks.

Speed

There is an obvious drawback of scalability that accompanies these excellent performances. **Larger capacity, lower speed!**

Larger capacity leads to higher performance?

In general, the larger model performs better.

RNN: the performance

Recurrent Neural Networks have become the core of many models for tasks that capture temporal dependency. RNN-based models achieve **impressive performance** in many NLP tasks.

Speed

There is an obvious drawback of scalability that accompanies these excellent performances. **Larger capacity, lower speed!**

Larger capacity leads to higher performance?

In general, the larger model performs better? **It is NOT entirely true.**

RNNs are indeed overparameterized

Such belief that the LSTM memory capacity is proportional to model size, several recent results have empirically **proven the contrary**, claiming that LSTMs are **over-parameterized** (Denil et al., 2013; James Bradbury and Socher, 2017; Merity et al., 2018; Melis et al., 2018)

Removing components of RNN suffers **little to no performance loss** (Levy et al., 2018).

RNNs are indeed overparameterized

Such belief that the LSTM memory capacity is proportional to model size, several recent results have empirically **proven the contrary**, claiming that LSTMs are **over-parameterized** (Denil et al., 2013; James Bradbury and Socher, 2017; Merity et al., 2018; Melis et al., 2018)

Removing components of RNN suffers **little to no performance loss** (Levy et al., 2018).

All weights are equally important? **Probably not**

Existing methods

Fine-tuning or retraining from scratch are still required!

Distillation based compression techniques (Hinton et al., 2015) and Pruning Techniques (See et al., 2016) often accompany retraining and selective retraining steps to achieve optimal performance.

Existing methods

Fine-tuning or retraining from scratch are still required!

Distillation based compression techniques (Hinton et al., 2015) and Pruning Techniques (See et al., 2016) often accompany retraining and selective retraining steps to achieve optimal performance.

How about pre-trained models?

It does not cover any scenarios involving large pre-trained models, e.g., ELMo (Peters et al., 2018), re-training can be very expensive in terms of time and resources (many GPUs or TPUs $$$).

Smaller, Faster, and even Better

Less computation cost
Less memory
More practical

Compact Representations

We come up with an idea of leveraging **low-rank matrix factorization for LSTM models** to create compact and dense representations of the model to **preserve the important information without losing significant performance**.

Low-rank matrix factorization methods:

- Truncated SVD
- Semi-NMF

Low-rank Matrix Factorization

Low-rankness has a correlation to the sparsity of matrices.

Understanding the characteristics of learned weights in neural networks helps us knowing the effectiveness of model compression techniques.

Finally, we can answer:

- Why RNN-based models such as **LSTM are overparameterized**?
- How compression methods can be applied efficiently **without hurting performance**?
- Which weight is more important? Additive (W_i) or multiplicative recurrence (W_h)?

Singular Value Decomposition

SVD produces a factorization by applying orthogonal constraints on the U and V factors. The objective is to find a combination of basis vectors which restrict to the orthogonal vectors in feature space that minimize reconstruction error. We take top `r` singular values.

$$\mathbf{W} = \mathbf{USV},$$
$$\underset{\mathbf{U},\mathbf{S},\mathbf{V}}{\text{minimize}} \quad ||\mathbf{W} - \mathbf{USV}||_F^2.$$

U = `m x r` dimensions, S = `r x r` dimensions, and V = `r x n` dimensions.

Semi-NMF

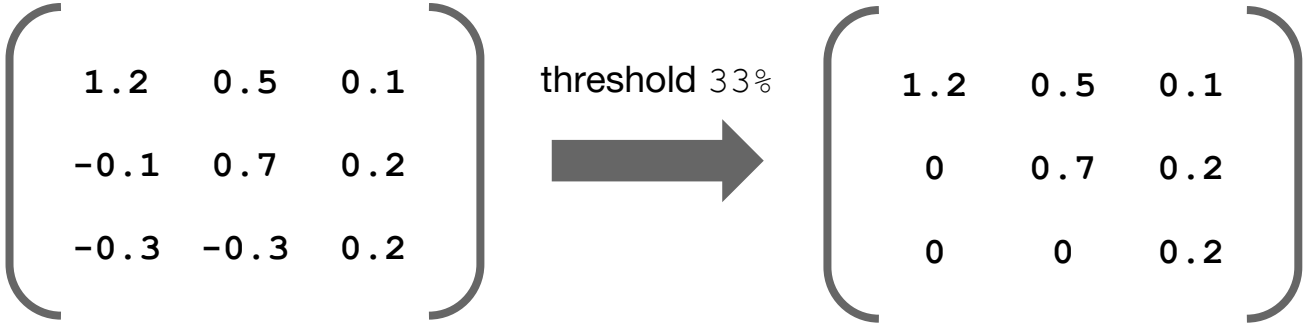
Semi-NMF generalizes Non-negative Matrix Factorization (NMF) by relaxing some of the sign constraints on negative values for U and W. We minimize the objective function:

$$\mathbf{W}_{\pm} \approx \mathbf{U}_{\pm}\mathbf{V}_{+},$$
$$\underset{\mathbf{U},\mathbf{V}}{\text{minimize}} \quad ||\mathbf{W} - \mathbf{UV}||_F^2 \quad \text{s.t. } \mathbf{V} \geq 0.$$

“Semi-NMF is more preferable in application to Neural Networks because of this generic capability of having negative values”

Pruning

As a baseline, instead of reducing the size of matrices, we zero out the values. To elaborate, for each weight matrix W, we mask the low-magnitude weights to zero, according to the compression ratio.



Complexity

We explore the complexity of each algorithm as the following:

$\mathbf{A} \in R^{z \times m}$, $\mathbf{W} \in R^{m \times n}$ $\mathbf{U} \in R^{m \times r}$, $\mathbf{V} \in R^{r \times n}$	Op	Cost	Memory
Uncompressed	$\mathbf{A} \cdot \mathbf{W}$	$O(z \times m \times n)$	$O(z \times m + m \times n)$
Pruning[1]	$\mathbf{A} \cdot (\frac{r(m+n)}{mn}\mathbf{W})$	$\Omega(z \times r(m+n))$	$O(z \times m + r(m+n))$
Matrix Factorization	$(\mathbf{A} \cdot \mathbf{U}) \cdot \mathbf{V}$	$\Theta(z \times r(m+n))$	$O(z \times m + r(m+n))$

Applying Low-Rank Matrix Factorization to LSTM

Factorized LSTM

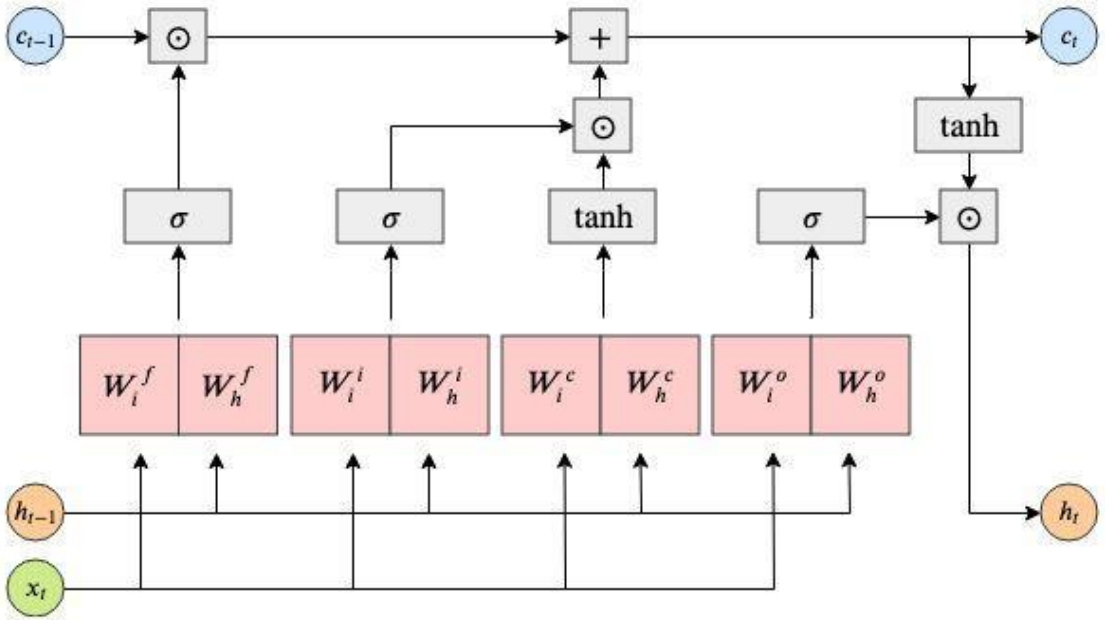
We propose to apply low-rank matrix factorization to the LSTM cell, particularly to the input weight W_i and recurrent weight W_h

To reduce the number of parameters

$mn \rightarrow r(m+n)$, where $r << m, n$

$$W_{m \times n} \approx U_{m \times r} V_{r \times n}$$

An LSTM cell

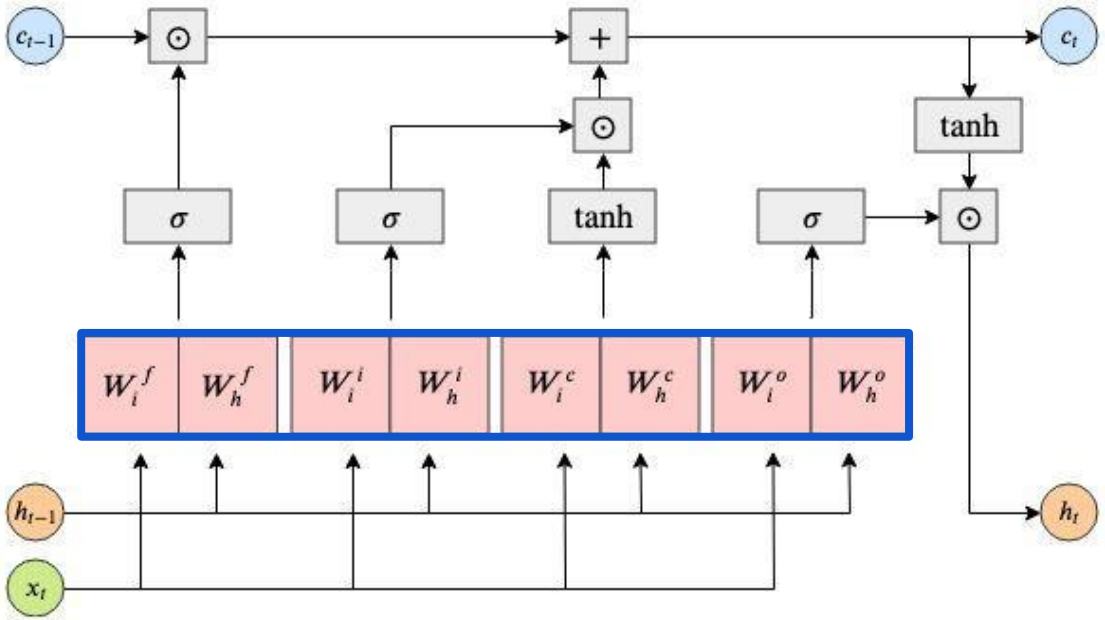


$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \hat{\mathbf{c}}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} (\mathbf{W}_i \quad \mathbf{W}_h) \begin{pmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{pmatrix},$$

$$\mathbf{W}_i = \begin{pmatrix} \mathbf{W}_i^i \\ \mathbf{W}_i^f \\ \mathbf{W}_i^o \\ \mathbf{W}_i^c \end{pmatrix}, \mathbf{W}_h = \begin{pmatrix} \mathbf{W}_h^i \\ \mathbf{W}_h^f \\ \mathbf{W}_h^o \\ \mathbf{W}_h^c \end{pmatrix},$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t,$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t).$$

An LSTM cell



$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \hat{\mathbf{c}}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} (\mathbf{W}_i \quad \mathbf{W}_h) \begin{pmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{pmatrix},$$

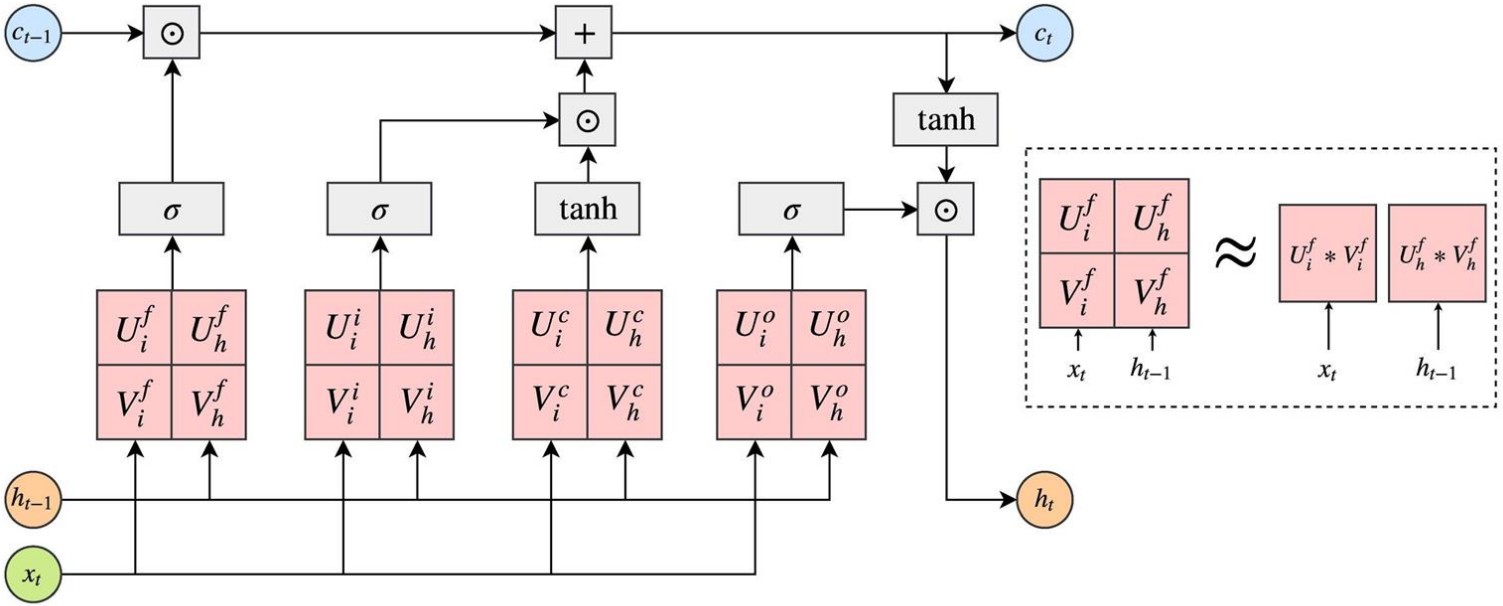
$$\mathbf{W}_i = \begin{pmatrix} \mathbf{W}_i^i \\ \mathbf{W}_i^f \\ \mathbf{W}_i^o \\ \mathbf{W}_i^c \end{pmatrix}, \quad \mathbf{W}_h = \begin{pmatrix} \mathbf{W}_h^i \\ \mathbf{W}_h^f \\ \mathbf{W}_h^o \\ \mathbf{W}_h^c \end{pmatrix},$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t,$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t).$$

Addictive recurrence

Multiplicative recurrence

A **Factorized LSTM** cell



Weight i and **Weight h** are factorized

Experiments

We evaluate using five different publicly available datasets spanning two domains:

1. **Perplexity in two Language Modeling (LM) datasets**: English Penn Tree Bank and WikiText2
2. **Accuracy/F1 in three downstream NLP tasks** (Question Answering, Sentiment Analysis, and Lexical Entailment) using **ELMo**

Language Modeling

We use AWD-LSTM (Merity et al., 2018), following the same training details for both datasets.

We also fine-tune the compressed model for several epoch to investigate its effectiveness upon our compression technique.

Efficiency Measure

To calculate the compression efficiency, we propose a new efficiency measure:

$$E(r) = \begin{cases} \dfrac{R(M, M^r)}{R(P, P^r)}, & \text{if } R(M, M^r) \geq 0 \\ \dfrac{R(M, M^r)}{1 - R(P, P^r)}, & \text{otherwise} \end{cases}$$

where M represents any evaluation metric (i.e. Accuracy, F1-score, Perplexity), P represents the number of parameters.

Lower E(r) is better. Negative E(r) = positive improvement.

Norm Measures

We investigate norms to understand the characteristics of our models.

- L1 norm: maximum column-sum -> sparsity

$$\mathcal{L}_j = \sum_{i=1}^{m} |w_{ij}|, \qquad \|\mathbf{W}\|_1 = \max_j \mathcal{L}_j,$$

- Std L1 norm

$$\bar{\mathcal{L}} = \sum_{j=1}^{n} \mathcal{L}_j, \qquad \sigma(\|\mathbf{W}\|_1) = \sum_{j=1}^{n} (\bar{\mathcal{L}} - \mathcal{L}_j)^2$$

- Nuclear norm ->

$$\|\mathbf{W}\|_{nuc} = \sqrt{\mathbf{W} * \mathbf{W}} = \sum_{i=1}^{\min m,n} \sigma_i(\mathbf{W})$$

Language Model on English PTB

PTB	Param.	w/o fine-tuning		w/ fine-tuning	
		PPL	E(r)	PPL	E(r)
AWD-LSTM	24M	58.3‡	-	57.3	-
TT-LSTM	12M	168.6	2.92†	-	-
Semi-NMF $\mathbf{W}_h$ (r=10)	9M	78.5	0.72	58.11	-0.02
SVD $\mathbf{W}_h$ (r=10)	9M	78.07	0.32	58.18	-0.02
Pruning $\mathbf{W}_h$ (r=10)	9M	83.62	0.89	57.94	-0.03
Semi-NMF $\mathbf{W}_h$ (r=400)	18M	59.7	0.05	57.84	-0.02
SVD $\mathbf{W}_h$ (r=400)	18M	**59.34**	**0.006**	57.81	-0.02
Pruning $\mathbf{W}_h$ (r=400)	18M	59.47	0.03	**57.19**	**-0.04**
Semi-NMF $\mathbf{W}_i$ (r=10)	15M	485.4	19.81	81.4	1.04
SVD $\mathbf{W}_i$ (r=10)	15M	462.19	6.83	88.12	1.35
Pruning $\mathbf{W}_i$ (r=10)	15M	676.76	28.69	82.23	1.08
Semi-NMF $\mathbf{W}_i$ (r=400)	20M	62.7	0.42	58.47	-0.01
SVD $\mathbf{W}_i$ (r=400)	20M	60.59	0.02	58.04	-0.01
Pruning $\mathbf{W}_i$ (r=400)	20M	59.62	0.10	57.65	-0.02

Even after massively compressing W_h (`r=10`) is relatively better than W_i. (9M < 15M)

Language Model on English PTB

PTB	Param.	w/o fine-tuning		w/ fine-tuning	
		PPL	E(r)	PPL	E(r)
AWD-LSTM	24M	58.3‡	-	57.3	-
TT-LSTM	12M	168.6	2.92†	-	-
Semi-NMF $\mathbf{W}_h$ (r=10)	9M	78.5	0.72	58.11	-0.02
SVD $\mathbf{W}_h$ (r=10)	9M	78.07	0.32	58.18	-0.02
Pruning $\mathbf{W}_h$ (r=10)	9M	83.62	0.89	57.94	-0.03
Semi-NMF $\mathbf{W}_h$ (r=400)	18M	59.7	0.05	57.84	-0.02
SVD $\mathbf{W}_h$ (r=400)	18M	**59.34**	**0.006**	57.81	-0.02
Pruning $\mathbf{W}_h$ (r=400)	18M	59.47	0.03	**57.19**	**-0.04**
Semi-NMF $\mathbf{W}_i$ (r=10)	15M	485.4	19.81	81.4	1.04
SVD $\mathbf{W}_i$ (r=10)	15M	462.19	6.83	88.12	1.35
Pruning $\mathbf{W}_i$ (r=10)	15M	676.76	28.69	82.23	1.08
Semi-NMF $\mathbf{W}_i$ (r=400)	20M	62.7	0.42	58.47	-0.01
SVD $\mathbf{W}_i$ (r=400)	20M	60.59	0.02	58.04	-0.01
Pruning $\mathbf{W}_i$ (r=400)	20M	59.62	0.10	57.65	-0.02

Even after massively compressing W_h (`r=10`) is relatively better than W_i (9M < 15M)

For (`r=10`), we improve 2.13x speedup.

Language Model on English PTB

PTB	Param.	w/o fine-tuning		w/ fine-tuning	
		PPL	E(r)	PPL	E(r)
AWD-LSTM	24M	58.3‡	-	57.3	-
TT-LSTM	12M	168.6	2.92†	-	-
Semi-NMF $\mathbf{W}_h$ (r=10)	9M	78.5	0.72	58.11	-0.02
SVD $\mathbf{W}_h$ (r=10)	9M	78.07	0.32	58.18	-0.02
Pruning $\mathbf{W}_h$ (r=10)	9M	83.62	0.89	57.94	-0.03
Semi-NMF $\mathbf{W}_h$ (r=400)	18M	59.7	0.05	57.84	-0.02
SVD $\mathbf{W}_h$ (r=400)	18M	**59.34**	**0.006**	57.81	-0.02
Pruning $\mathbf{W}_h$ (r=400)	18M	59.47	0.03	**57.19**	**-0.04**
Semi-NMF $\mathbf{W}_i$ (r=10)	15M	485.4	19.81	81.4	1.04
SVD $\mathbf{W}_i$ (r=10)	15M	462.19	6.83	88.12	1.35
Pruning $\mathbf{W}_i$ (r=10)	15M	676.76	28.69	82.23	1.08
Semi-NMF $\mathbf{W}_i$ (r=400)	20M	62.7	0.42	58.47	-0.01
SVD $\mathbf{W}_i$ (r=400)	20M	60.59	0.02	58.04	-0.01
Pruning $\mathbf{W}_i$ (r=400)	20M	59.62	0.10	57.65	-0.02

Even after massively compressing W_h (`r=10`) is relatively better than W_i (9M < 15M).

For (`r=10`), we improve 2.13x speedup.

Indeed, fine-tuning improves the performance.

Language Model on WikiText2

WT-2	Params	PPL	E(r)
AWD-LSTM	24M	65.67‡	-
Semi-NMF $\mathbf{W}_h$ (r=10)	9M	102.17	65.14
SVD $\mathbf{W}_h$ (r=10)	9M	99.92	62.49
Pruning $\mathbf{W}_h$ (r=10)	9M	109.16	72.64
Semi-NMF $\mathbf{W}_h$ (r=400)	18M	66.5	4.33
SVD $\mathbf{W}_h$ (r=400)	18M	**66.1**	**2.28**
Pruning $\mathbf{W}_h$ (r=400)	18M	66.23	2.94
Semi-NMF $\mathbf{W}_i$ (r=10)	15M	481.61	197.57
SVD $\mathbf{W}_i$ (r=10)	15M	443.49	194.89
Pruning $\mathbf{W}_i$ (r=10)	15M	856.87	211.23
Semi-NMF $\mathbf{W}_i$ (r=400)	20M	68.41	22.68
SVD $\mathbf{W}_i$ (r=400)	20M	67.11	12.18
Pruning $\mathbf{W}_i$ (r=400)	20M	66.37	5.97

Similar trend to PTB

W_h (`r=10`) is STILL relatively better than W_i (9M < 15M)

ELMo: SST-5

SST-5	r=10		r=400		Best	
	Acc.	E(r)	Acc.	E(r)	Acc. (avg)	E(r) (avg)
BCN	-	-	-	-	53.7‡	-
BCN + ELMo	-	-	-	-	54.5‡	-
Semi-NMF $\mathbf{W}_h$	50.18	0.29	53.93	0.21	54.16 (52.93)	0.09 (0.17)
SVD $\mathbf{W}_h$	50.4	0.27	54.11	0.13	54.11 (52.84)	0.12 (0.17)
Pruning $\mathbf{W}_h$	**50.81**	**0.25**	**54.66**	**-0.03**	**54.88 (53.59)**	**-0.07 (0.06)**
Semi-NMF $\mathbf{W}_i$	38.23	1.1	54.11	0.15	54.11 (50.56)	0.12 (0.34)
SVD $\mathbf{W}_i$	40.58	0.94	54.34	0.05	54.38 (51.19)	0.02 (0.26)
Pruning $\mathbf{W}_i$	34.57	1.35	54.61	-0.01	54.66 (50.01)	-0.02 (0.33)

Pruning achieves slightly higher result than MF.

ELMo: SNLI and SQuAD

SNLI	r=10		r=400		Best	
	Acc.	E(r)	Acc.	E(r)	Acc. (avg)	E(r) (avg)
ESIM	-	-	-	-	88.6	-
ESIM + ELMo	-	-	-	-	88.5‡	-
Semi-NMF $\mathbf{W}_h$	87.24	0.04	88.45	0.01	88.47 (88.18)	0.003 (0.01)
SVD $\mathbf{W}_h$	87.27	0.04	88.46	0.005	88.46 (88.18)	0.003 (0.01)
Pruning $\mathbf{W}_h$	**87.51**	**0.03**	**88.53**	**-0.003**	**88.53 (88.23)**	**-0.003 (0.01)**
Semi-NMF $\mathbf{W}_i$	77.08	0.39	88.44	0.01	88.44 (86.59)	0.01 (0.07)
SVD $\mathbf{W}_i$	78.15	0.35	88.48	0.002	88.48 (86.77)	0.007 (0.06)
Pruning $\mathbf{W}_i$	73.67	0.5	88.48	0.005	88.5 (85.8)	0.001 (0.09)

Pruning achieves slightly higher result than MF in SNLI.

SQuAD	r=10		r=400		Best	
	F1	E(r)	F1	E(r)	F1 (avg)	E(r) (avg)
BiDAF	-	-	-	-	77.3‡	-
BiDAF + ELMo	-	-	-	-	81.75‡	-
Semi-NMF $\mathbf{W}_h$	76.59	**0.21**	81.55	0.04	81.55 (80.32)	0.03 (0.07)
SVD $\mathbf{W}_h$	**76.72**	**0.21**	81.62	0.027	81.62 (80.47)	0.02 (0.06)
Pruning $\mathbf{W}_h$	52.02	0.49	81.73	0.006	81.65 (**80.6**)	0.006 (**0.05**)
Semi-NMF $\mathbf{W}_i$	60.69	0.88	**81.78**	**-0.0003**	**81.78** (77.93)	**-0.0003** (0.17)
SVD $\mathbf{W}_i$	57.14	1.03	**81.78**	**-0.0003**	**81.78** (77.69)	**-0.0003** (0.17)
Pruning $\mathbf{W}_i$	52.02	1.24	81.73	0.004	81.73 (76.06)	0.004 (0.25)

SQUAD has similar trend as LM.

Summing up the results

W_i (additive) versus W_h (multiplicative)

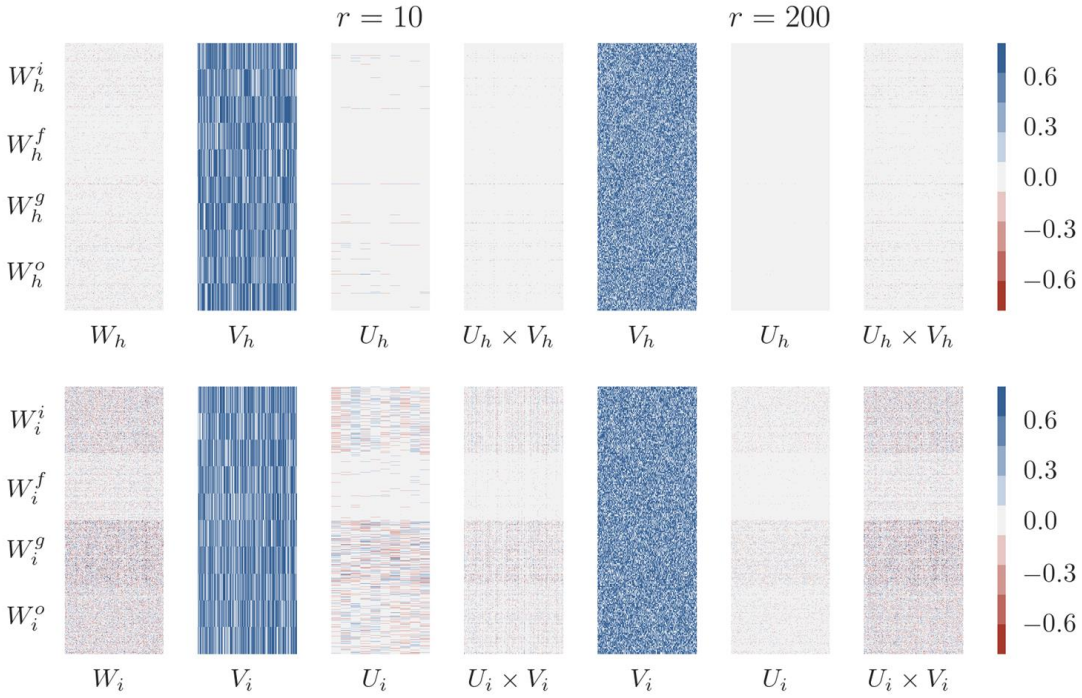
We can see that compressing W_h is in general more efficient and better performing than W_i except for SVD in SST-5 especially for very low `r`.

Improvements!

Compressing ELMo (`r=400`) improves the performance on SNLI, SQuAD, and SST-5.

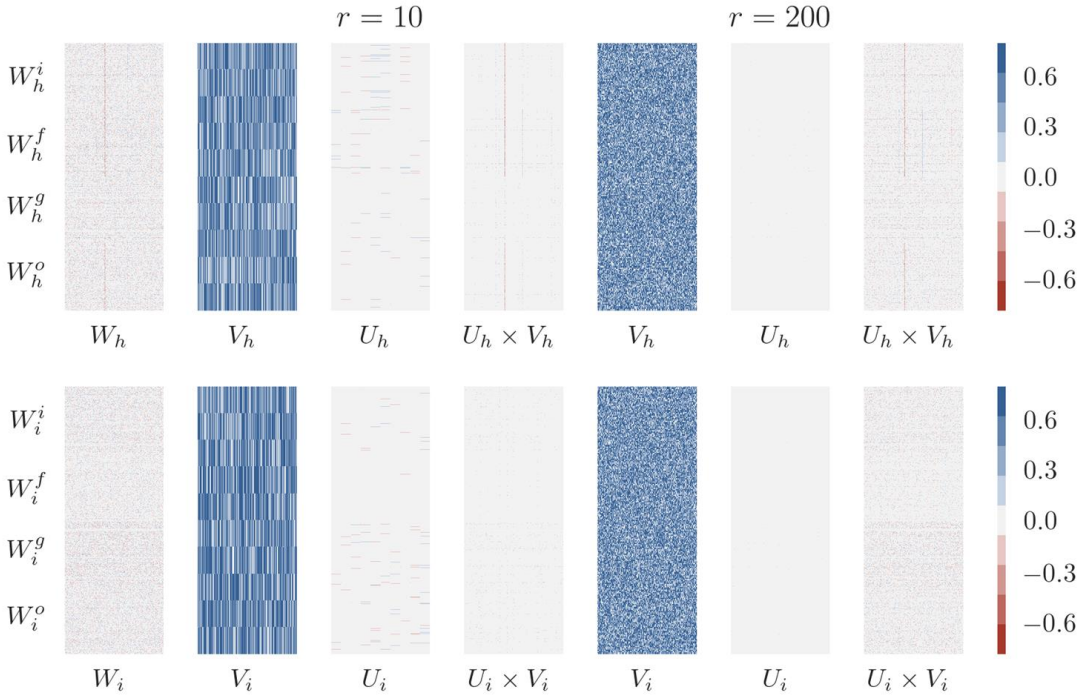
Further Analysis

Heatmap Analysis (PTB)



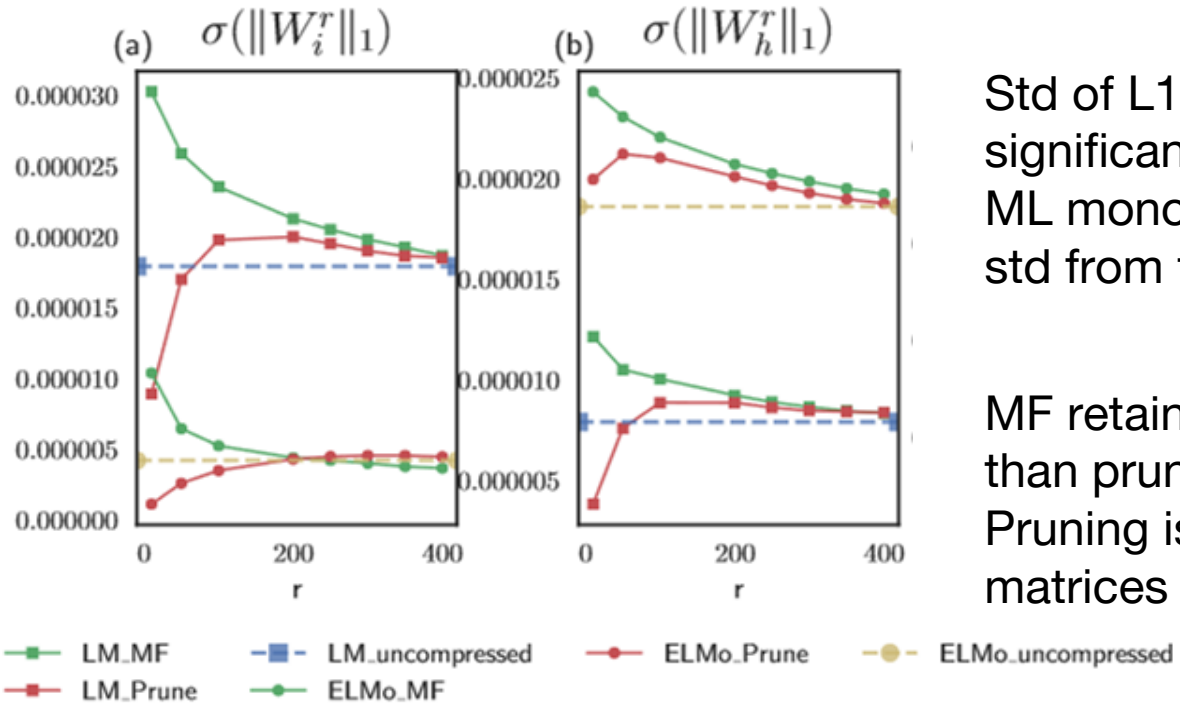
More clear salient red lines after factorization (U_h x V_h)

Heatmap Analysis (ELMo)



More clear salient red lines after factorization (U_h x V_h)

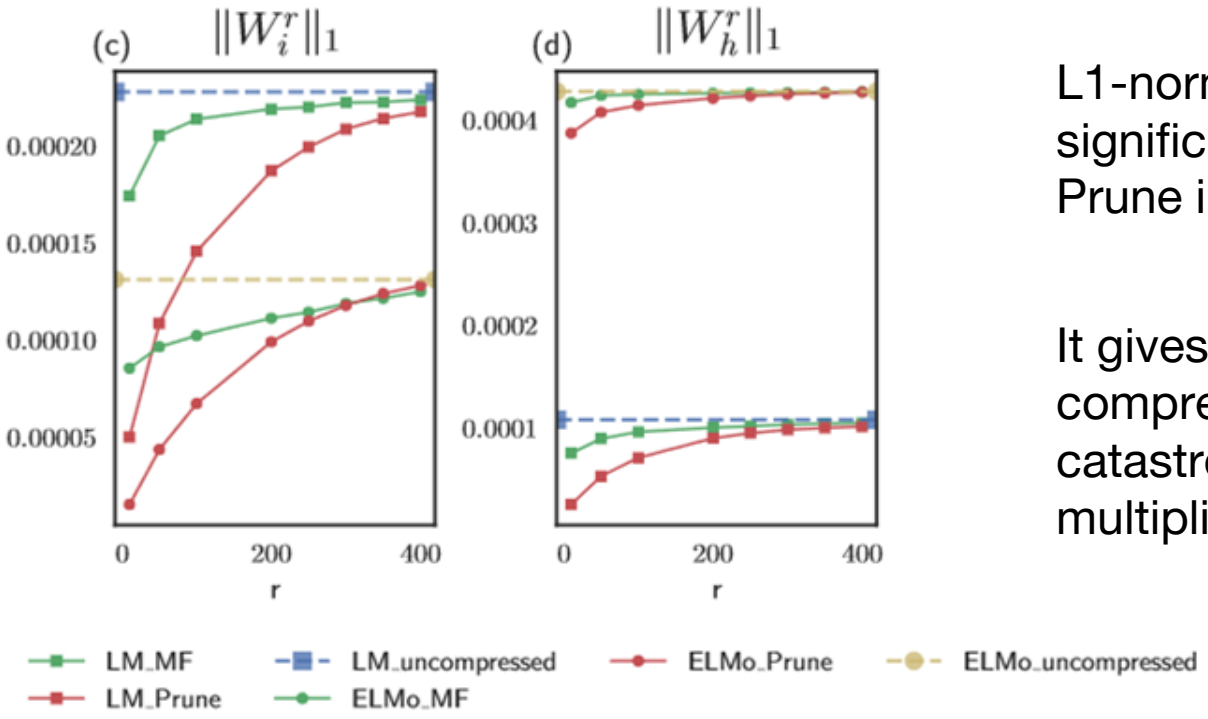
Norm Analysis



Std of L1 norm of pruning drops significantly for lower ranks, while ML monotonically increases the std from the baseline.

MF retains more salient features than pruning for low ranks. Pruning is better used in Sparse matrices (such as ELMo).

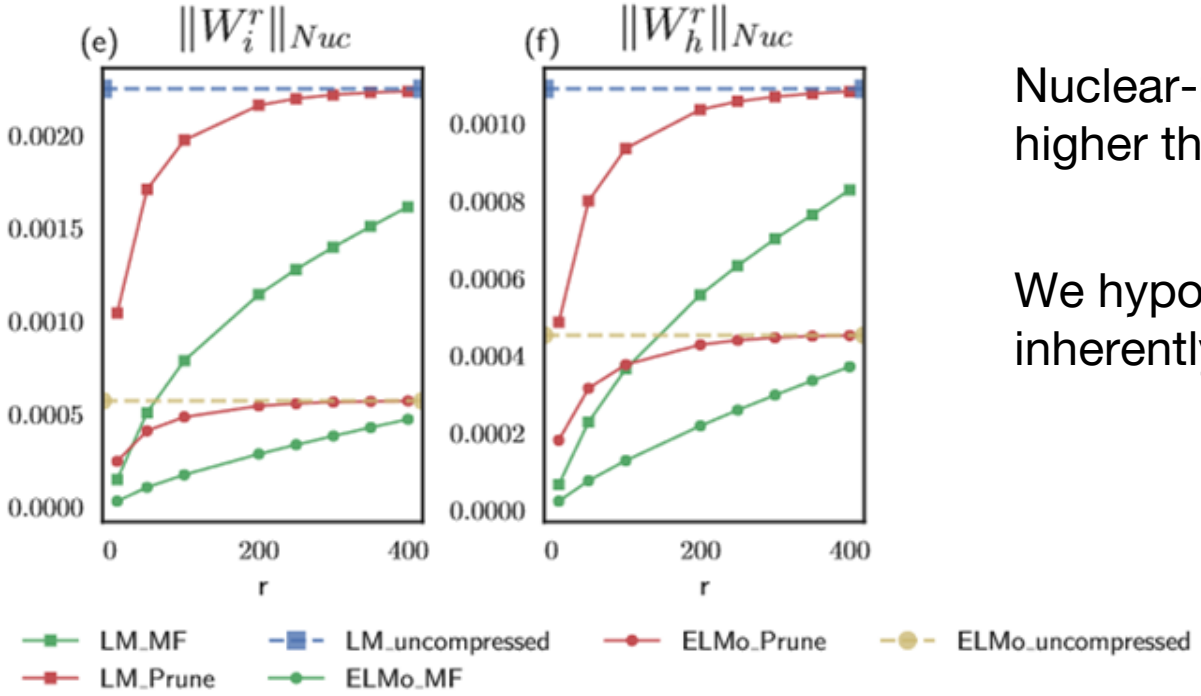
Norm Analysis



L1-norm decreases more significantly for both MF and Prune in W_i than W_h in lower `r`.

It gives us hints that compressing addictive is very catastrophic. Not for multiplicative.

Norm Analysis



Nuclear-norm of W_i is twice higher than W_h.

We hypothesize that W_h is inherently low-rank than W_i.

Conclusion

We empirically verified the limits of compressing LSTM gates using low-rank matrix factorization and pruning in four different NLP tasks.

Conclusion

We empirically verified the limits of compressing LSTM gates using low-rank matrix factorization and pruning in four different NLP tasks.

Compression is useful and effective to reduce over-parameterized networks such as LSTM language models.

Conclusion

We empirically verified the limits of compressing LSTM gates using low-rank matrix factorization and pruning in four different NLP tasks.

Compression is useful and effective to reduce over-parameterized networks such as LSTM language models.

Low-Rank Matrix Factorization works better in general than pruning, except for particularly sparse matrices.

Conclusion

We empirically verified the limits of compressing LSTM gates using low-rank matrix factorization and pruning in four different NLP tasks.

Compression is useful and effective to reduce over-parameterized networks such as LSTM language models.

Low-Rank Matrix Factorization works better in general than pruning, except for particularly sparse matrices.

Compressing multiplicative recurrence works better than compressing additive recurrence.

Thank you

ありがとうございました

Terima kasih

谢谢

All questions are welcome