Pushing the limits of RNN Compression

Urmish Thakker, Igor Fedorov, Jesse Beu, Dibakar Gope, Chu Zhou, Ganesh Dasika , Matthew Mattina

arm Research

Arm ML Research Lab

Motivation & Overview

- RNNs power IoT applications like wake-word detection, human activity recognition (HAR) and predictive maintenance
- IoT devices have small storage capacity (2 KB-32 KB) and smaller caches
- **Target compression factors of 16-30x** *(> 94% reduction in parameters)* to ensure these applications fit on IoT devices.
- State-of-the art techniques [4][5] lead to significant accuracy loss for these compression factors
- ***Kronecker Product (KP) achieves the best task accuracy for 16-30x*** *compression factors while simultaneously improving inference run-time.*

Introduction to Kronecker Products

- Let $A \in R^{mxn}, B \in R^{m1xn1}$ *and* $C \in R^{m2xn2}$ then, the KP between B and C is given by

$$A = B \otimes C$$

$$A = \begin{pmatrix} b_{1,1} \circ C & \cdots & b_{1,n1} \circ C \\ \vdots & \ddots & \vdots \\ b_{m1,1} \circ C & \cdots & b_{m1,n1} \circ C \end{pmatrix}$$

- B and C are referred to as Kronecker factors (KF) of A and m = m1xm2 and n = n1xn2
- *If m = 154, n =164, m1 = 11, n1=41, m2 = 14, n2 = 4, we get 50x compression!*
- We can use more than 2 KFs. Eg -

$$W = W1 \otimes W2 \otimes \ldots . Wn.$$

- More n's will lead to more compression!

Prior Work

- Focused on training stability while increasing the size of the network (Table 3, [1])
- Compressed FC layers in CNNs [2] and did not measure the inference run-time

Key questions that we answer

- Can a RNN with matrix expressed as KP of smaller matrices maintain baseline accuracy at high compression factors
- How many number of KF matrices, n, should we choose?
- What is the impact on inference run-time when RNN matrices are expressed as KP of KF matrices

Kronecker Product Recurrent Neural Network

- Matrix-vector product when matrix is expressed as KP of 2 KF –

$$\boldsymbol{y = Ax = (B \otimes C)x = vec(CXB^T)}, \quad (1)$$

$$where\ X \in R^{n2xn1}$$

- *What happens as n increases?*
  - *Loss in training accuracy* due to vanishing gradient issue
  - *Increased inference runtime* as equation (1) is applied recursively
- Restrict the # of factors to 2!
- Traditional RNN: $h_t = f\ (A \times [x\ \ h_{t\,-\,1}])$
- **KPRNN**: $\boldsymbol{h_t = f\ ((B \otimes C) \times [x\ \ h_{t\,-\,1}])}$
  - B and C learned via back-propagation
  - During inference, use *equation 1* to get speed-up over baseline

Results

- Datasets – Image Recognition (MNIST, USPS), Key-word spotting (Google KWS), HAR (Discovery)
- Inference run-time measured on A53 core of HiKey Board

Benchmark Name	Parameter Measured	Baseline	Compression Technique			
			Small Baseline	Pruning	LMF	KP
MNIST	*Compression Factor*	1x	10x	17x	13x	**18x**
	Accuracy (%)	99.4	87.5	96.5	97.4	**98.4**
	Runtime (ms)	6.3	0.7	0.7	1.8	4.6
HAR1	*Compression Factor*	1x	20x	29x	28x	**30x**
	Accuracy (%)	91.9	88.9	82.9	89.9	**91.2**
	Runtime (ms)	470	30	98	64	157
KWS	*Compression Factor*	1x	16x	24x	21x	**25x**
	Accuracy (%)	92.5	89.7	84.9	89.1	**91.2**
	Runtime (ms)	26.8	1.9	5.9	4.1	17.5
USPS	*Compression Factor*	1x	4x	9x	8x	**16x**
	Accuracy (%)	98.8	91.2	88.5	89.5	**93.2**
	Runtime (ms)	1.17	0.4	0.375	0.28	0.6

- *Runtime Improvement*: KP improves the baseline runtime by 54%, 49% and 66.59% for KWS, USPS and HAR1.
- *Comparison with SB*: Better accuracy by 10.94%, 2.3%, 7.7% and 1.97% for MNIST, HAR1, KWS and USPS
- *Comparison with Pruning*: Better accuracy by 2%, 8.17%, 6.29% and 4.68% for MNIST, HAR1, KWS and USPS
- *Comparison with LMF*: Better accuracy by 1.04%, 1.2%, 2.07% and 3.64% for MNIST, HAR1, KWS and USPS.

Conclusion

We show how to compress RNNs aggressively (16-30x) while simultaneously preserving more accuracy than any other state-of-the-art technique

Full paper available on arxiv, scan this QR code for link

References

[1] Kronecker Recurrent Units

[2] Compression of fully-connected layer in neural network by kronecker product

[3]http://www.mathcs.emory.edu/~nagy/courses/fall10/515/KroneckerIntro.pdf

[4] To prune or not to prune

[5] Speeding up Convolutional Neural Networks with Low Rank Expansions

The 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing @ NeurIPS 2019