Improving accuracy of neural networks compressed using fixed structures via doping

Urmish Thakker, Ganesh Dasika, Paul Whatmough, Matthew Mattina, Jesse Beu

Arm ML Research Lab

Summary

- Efficient structures (circular, block-circular, Kronecker products (KP) etc) have shown remarkable results in compressing NNs
  - *KP can compress LSTMs in IoT workloads by 15-38*x, outperforming traditional compression techniques
- However, these techniques are hard to scale to larger networks
- We introduce doping - a sparse matrix overlay that provides additional degrees of freedom to matrix expressed using efficient structures
  - To use doping, we need to overcome co-matrix adaption using co-matrix row regularization

" Using Doped KP, we can compress a large language model with LSTM layers of size 25 MB by 25x with 1.4% loss in perplexity score outperforming other traditional compression techniques (pruning, LMF etc) by a large margin"

Introduction of 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 *50*x *compression!*
- We can use more than 2 KFs. Eg -

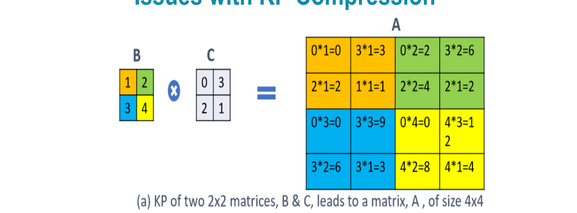
$$W = W1 \otimes W2 \otimes \ldots . Wn.$$

- Prior work [1] showed –
  - How many number of KF matrices, n, should we choose
  - The impact on inference run-time when RNN are expressed as KP of 2 matrices
  - We can compress IoT Workloads by 15-38x
- However, this work does not scale to large networks

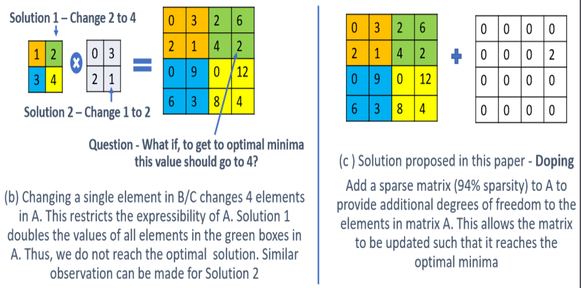
Applying KP to large Language Model

- Applied KP to medium Language Model from [2].
  - LSTM layers with matrices of size 2600x1300 and layers total size of 25 MB.
  - Baseline Perplexity of 82.07
- Result of KP Compression –
  - *Compression by a factor of 338*x
  - *Perplexity of 104.03, perplexity loss of 38%!*
- [1] suggests using Hybrid KP to recover lost accuracy
  - Able to recover the lost accuracy, but compression reduces to a factor of 5x
- ***Questions***
  " Why does KP Compression lead to loss in accuracy for large models?"
  " How can you scale KP to larger networks without sacrificing accuracy or giving up on their significant compression benefits"

Issues with KP Compression



(a) KP of two 2x2 matrices, B & C, leads to a matrix, A , of size 4x4



(b) Changing a single element in B/C changes 4 elements in A. This restricts the expressibility of A. Solution 1 doubles the values of all elements in the green boxes in A. Thus, we do not reach the optimal solution. Similar observation can be made for Solution 2

(c) Solution proposed in this paper - **Doping**

Add a sparse matrix (94% sparsity) to A to provide additional degrees of freedom to the elements in matrix A. This allows the matrix to be updated such that it reaches the optimal minima

Doped Kronecker Products

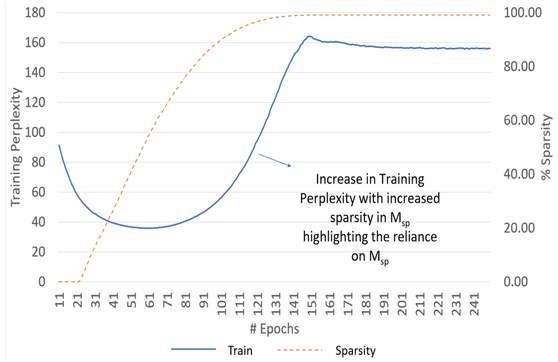
- $W = Mkp + Msp, where\ Mkp = B \otimes C \ -(1)$
- Sparsity of M_{sp} determines the amount of compression
  - Eg – If W is of size 100x100, B and C are of size 10x10 each, then 95% sparsity in M_{sp} will lead to 14x compression
- During initial phase of training, M_{sp} is dense, as training progresses, required sparsity levels are reached

Co-matrix Adaptation (CMA)

- Applying DKP as in equation 1 does not work. Table 1 shows the result of applying DKP as shown in equation 1

Baseline Perplexity	82.04	
Compression Factor	338x	100x
Sparsity of M_{sp}	100%	99.93%
DKP Perplexity	104	138.3

- DKP as shown in equation 1 does not work (perplexity increases!) because of CMA



- Possible methods to overcome CMA:

$$W = B \otimes C + \beta * Msp, \min||\beta|| - (2a)$$

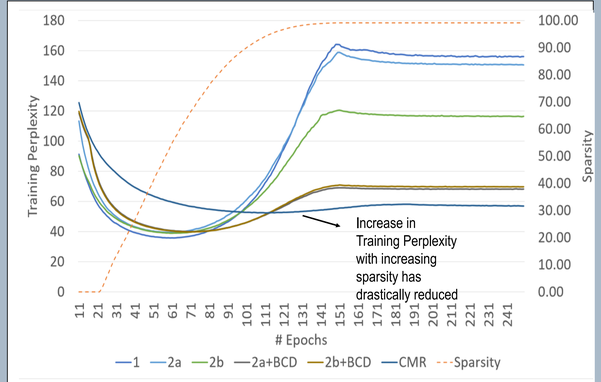
$$W = \alpha * (B \otimes C) + \beta * Msp, \min\left|\left|\beta + \frac{1}{\alpha}\right|\right| - (2b)$$

- Equation 2a and 2b can be further trained using Block Coordinate Descent (BCD)

Co-matrix row dropout regularization (CMR)

- Equation 2a and 2b help manage CMA
  - But, can we do better?
- Hypothesis: During CMA, incoming neurons from the M_{kp} matrix and M_{sp} matrix learn to co-adapt leading to lost capacity
- Introduce stochastic behavior where either M_{kp} neuron or M_{sp} neuron are not available to drive the output neuron
- CMR:

$$W = (B \otimes C) \odot b_1 + M_{sp} \odot b_2$$
$$where, b_1, b_2 \sim Bernoulli\ (p)$$



Results

	Compression Factor	Test Perplexity
Baseline	1	82.04
4-bit quantization [3]	8	83.84
3-bit quantization [4]	10.67	83.14
Tensor Train Decomposition [5]	1.67	168.64
Weight Distortion with Pruning [6]	10	84.64
Low-rank matrix factorization	20	114.29
HMD [7]	20	105.43
HKD [1]	20	99.882
Magnitude Pruning	20	85.14
This work (DKP+CMR)	**25**	**83.24**

Read our paper for more details

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

This work

Prior work on KP Compression

References

[1] Compressing RNNs for IoT devices by 15-38x using Kronecker Products
[2] Recurrent Neural Network Regularization
[3] Weighted-Entropy-based Quantization for Deep Neural Networks
[4] Retraining-Based Iterative Weight Quantization for Deep Neural Networks
[5] Compression of Recurrent Neural Networks for Efficient Language Modeling
[6] DeepTwist: Learning Model Compression via Occasional Weight Distortion
[7] Run-Time Efficient RNN Compression for Inference on Edge Devices