RNN Compression using Hybrid Matrix Decomposition

Urmish Thakker, Ganesh Dasika, Jesse Beu, Dibakar Gope and Matthew Mattina
Machine Learning Group, Arm Research

***Abstract*—RNNs can be large and computationally intensive. Yet, many applications that use RNNs run on edge devices with very limited compute and storage capabilities. These applications often have strict real-time deadlines, so any compression technique that is used must not impact the inference run-time. Hybrid Matrix Decomposition (HMD) is a new compression technique that divides the matrix into two parts - an unconstrained upper half and a lower half composed of rank-1 blocks. This results in output features where the upper sub vector has "richer" features while the lower sub vector has "constrained" features. HMD can compress RNNs by 2× while being 2× faster than pruning and more accurate than a traditional matrix factorization technique, better enabling the deployment of "TinyML" applications.**

I. Hybrid Matrix Decomposition (HMD)

Algorithm 1 Matrix vector product with HMD

Input 1: Matrices A', B, C, D and E
Input 2: Vector I
Output: Matrix O of dimension $M \times 1$
$O_{\text{top\_r\_rows}} \leftarrow A' \times I$
$Temp1 \leftarrow B \circ (C \cdot I_{\text{top\_N/2\_rows}})$
$Temp2 \leftarrow D \circ (E \cdot I_{\text{bottom\_N/2\_rows}})$
$O_{\text{bottom\_M-r\_rows}} \leftarrow Temp1 + Temp2$

HMD breaks a matrix into two parts - a fully parameterized upper half and a constrained lower half. Figure 1 gives a visual representation of this decomposition technique for a single matrix along with the parameters required for storage. This creates a dense matrix representation making it more hardware-friendly than pruning. Additionally, it creates a higher rank matrix than low rank matrix factorization (LMF), giving it more expressibility.

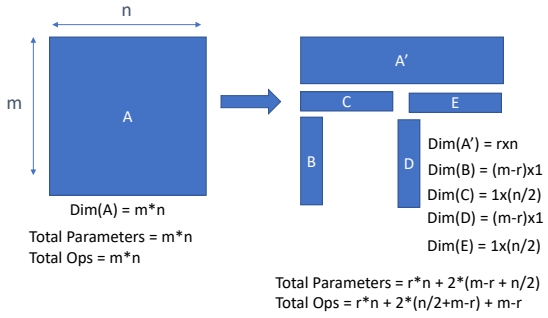

Fig. 1. Representation of a matrix using hybrid decomposition

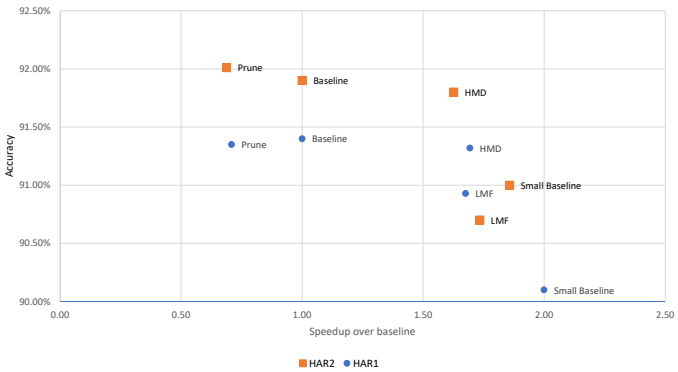

Fig. 2. Accuracy vs Speedup (over baseline) when the LSTM layer is compressed by a factor of 2×. Speedup values < 1 indicate a slowdown

HMD will also lead to reduction in the number of computations of a matrix vector product, through the use of Algorithm 1 which takes advantage of the associative property of matrix multiplication. The ops count for the algorithm are shown in Figure 1. A RNN cell that uses HMD replaces all of its weight matrices with its HMD representation and the matrix vector product computations in these cells are computed using algorithm 1.

In order to test HMD, we compress the human activity recognition network in [1] (HAR1) and [2] (HAR2) by a factor of 2. We measured the accuracy of the baseline, compressed networks and a smaller baseline with same number of parameters as the compressed network. We also measured the runtime of these networks on a Hikey board with 4 Cortex A73 and 2 MB Cache. We use a single threaded version of the eigen library to get the runtime numbers. Figure 2 shows the results of these experiments. Pruning achieves the best accuracy amongst all compression techniques, however at a cost of significant slowdown in runtime. LMF achieves faster runtime, but with poorer accuracy than baseline. HMD is able to achieve similar accuracy as the baseline, while also being 1.7× faster.

References

[1] N. Hammerla, S. Halloran, and T. Ploetz, "Deep, convolutional, and recurrent models for human activity recognition using wearables".
[2] F. Ordez and D. Roggen, "Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition".