DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices

Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev Claudio Forlivesi, Lei Jiao, Lorena Qendro, and Fahim Kawsar

Presenter: Eamon Collins

ec3bd@virginia.edu

https://qdata.github.io/deep2Read
Overview

1 Motivation

2 Previous Work

3 Novel Ideas
   - Runtime Layer Compression
   - Deep Architecture Decomposition

4 Results
Motivation

- Edge-computing becoming more valuable
  - More data being gathered by sensor networks
  - Communication is expensive in time and power
- Graph Applications?
  - Point-cloud LIDAR
  - Inference on own network
Previous Work

- SVD well-studied and widely used compression technique
- Existing approaches either require retraining or at least using test data to measure and limit accuracy degradation
- No existing solution includes runtime compression or flexible decomposition into multiple heterogeneous processors
Novel Ideas

- Runtime Layer Compression
  - SVD-based layer compression
  - Redundancy Estimation
- Deep Architecture Decomposition
  - Decomposition Search
  - Recomposition Inference
Runtime Layer Compression

SVD

\[ W_{m \times n}^L = U_{m \times m} \Sigma_{m \times n} V_{n \times n}^T \]

approximated by:

\[ \hat{W}_{m \times n}^L = U_{m \times c} \Sigma_{c \times c} V_{c \times n}^T \]

\[ \hat{W}_{m \times n}^L = U_{m \times c} N_{c \times n}^T \]

Results in \((m + n) \times c\) necessary weights instead of \(mn\), \(c \ll m, n\)
Reconstruction error determined:

\[ \varepsilon(W_{m\times n}, \hat{W}_{m\times n}) = \sqrt{\sum_{i=1}^{m} ||w_i - \hat{w}_i||_2^2} \]

- Sum the \( \varepsilon \) from each compressed layer to get overall error
- Error over multiple layers doesn’t linearly correspond to inference accuracy error, but generally small reconstruction error means small accuracy degradation
- User specifies either maximum acceptable error or maximum acceptable error degradation, both controlled by not allowing over certain total \( \varepsilon \)
Deep Architecture Decomposition

Main Idea: Large complex models are decomposed into unit-blocks that are tailored to the available processors. e.g. Convolution layers may be allocated to onboard GPU, and some of the fully connected layers compressed and allocated to the CPU.

- Split into a search for the best decomposition plan and the assigning to processors
- Constraints can be specified as performance goals for one or more of the metrics: energy, inference time, model error
**Algorithm 1 Decomposition Search**

1: **Input:** (i) Model with \( n \) layers, (ii) \( E_{TH} \) (Allowed level of overall approximation error), and (iii) \( e_1, e_2, \ldots, e_k \) (Energy footprint of all available processors).

2: **for** all layer\(_i\) \( \in \) Model **do**

3: \hspace{1em} `layerType = getLayerType(layer\(_i\))` \( \triangleright \) Identifying layer type based on operations

4: \hspace{2em} **if** layerType == convolution or pooling **then**

5: \hspace{3em} `BlockSize = extractFilteringBlocks()` \( \triangleright \) Fully connected layers

6: \hspace{2em} **else**

7: \hspace{3em} `BlockSize = extractFeedForwardBlocks()`

8: \hspace{2em} **for** \( j = 1 \) to \( P \) **do** \( \triangleright \) Extracting parameters for all processors

9: \hspace{3em} \( E_j, B_j = \text{getProcessorParameters}(\text{BlockSize}, e_j) \)

10: \hspace{2em} **if** layerType == Feed-forward **then**

11: \hspace{3em} **for** \( k = 90, 10, 10 \) **do** \( \triangleright \) Linear searching parameter space

12: \hspace{4em} \( E = \text{CompressSVD}(W^{layer\(_i\)}, k) \) \( \triangleright \) Estimating Reconstruction Error

13: \hspace{3em} **if** \( E < E_{TH} \) **then**

14: \hspace{4em} `Save U_{m\times c} and N_{c\times n}^T` \( \triangleright \) Stop parameter searching

15: \hspace{3em} **else**

16: \hspace{4em} **break**

17: \hspace{4em} `updateLayer(layer\(_i\), U_{m\times c}, N_{c\times n}^T)`

18: \hspace{2em} `applyOptimization(\text{BlockSize}, \{E\}_j^{k=1}, \{B\}_j^{k=1})` \( \triangleright \) using Equation 5a

19: **Assign** blocks to processors as identified by the optimizations.
Recomposition

\[
\begin{align*}
\min \quad & \alpha \sum_{i=1}^{P} E_i B_i + \beta \max_{i \in \mathcal{P}} \{ T_i B_i \} \\
\text{s.t.} \quad & \sum_{i=1}^{P} B_i = N \\
& B_i \leq L_i, \forall i \in \mathcal{P}, \\
& B_i \geq 0, B_i \in \mathbb{Z}, \forall i \in \mathcal{P},
\end{align*}
\]

- \( \mathcal{P} = \{1, 2, ... P\} \) the set of processors available
- \( B_i \) number of blocks assigned to processor \( i \)
- \( L_i \) load limit of processor \( i \)
- \( E_i \) and \( T_i \) are the energy and time respectively it takes for processor \( i \) to compute a single unit-block
Results

(a) AlexNet – Snapdragon
(b) AlexNet – Tegra
(c) SpeakerID – Snapdragon

Energy (m Joule) vs. Latency (m sec.)

- CPU
- Fully Cloud over WiFi
- Partial CPU + Partial Cloud WiFi
- DeepX (Acc deg. 5%)
- Partial DeepX + Partial Cloud WiFi

Model size (MB) under DeepX

- Original model: 233 MB
- Acc. drop < 1%: 99 MB
- Acc. drop < 3%: 69 MB
- Acc. drop < 5%: 57 MB
- Acc. drop < 10%: 32 MB

https://qdata.github.io/deep2Read
Limitations

- Not optimal for all network types, variable improvement even among DNN and CNN
- Resource need estimator
  - Predicting resource usage of a block primitive
  - No attempt made at predicting resource availability
  - Impact of changes in resource availability not measured
References

Y. Gong, et al., 2014
“Compressing deep convolutional networks using vector quantization,”

T. He, et al.,
“Reshaping deep neural network for fast decoding by nodepruning,”
*ICASSP ’14*
The End