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

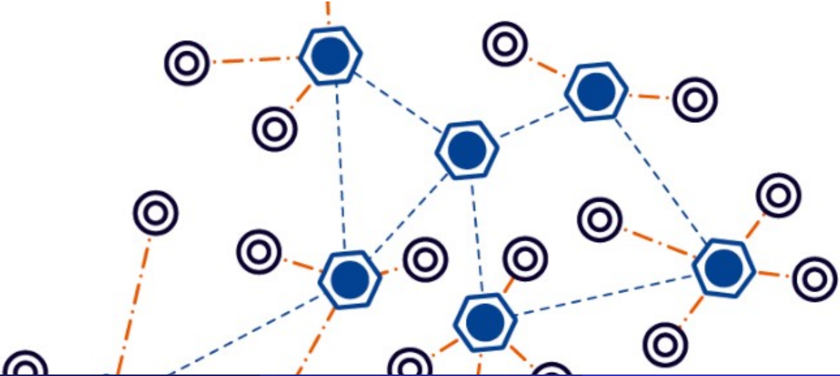
ec3bd@virginia.edu

https://qdata.github.io/deep2Read

Overview

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

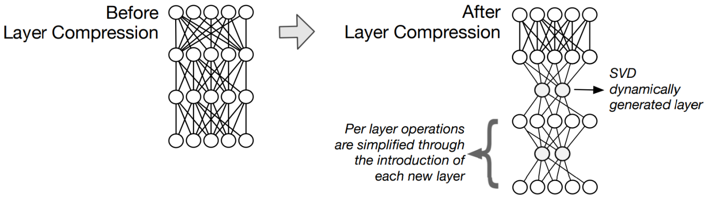
$$W^L_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n}$$

approximated by:

$$\hat{W}^L_{m\times n} = U_{m\times c}\Sigma_{c\times c}V^T_{c\times n}$$

$$\hat{W}^L_{m\times n} = U_{m\times c}N^T_{c\times n}$$

Results in $(m+n)\times c$ necessary weights instead of mn, $c << m, n$



Redundancy Estimation

Reconstruction error determined:

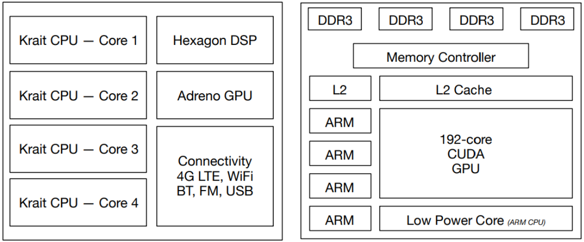
$$\varepsilon(W^L_{m\times n}, \hat{W}^L_{m\times n}) = \sqrt{\frac{\sum_{i=1}^{m} ||w_i - \hat{w}_i||_2^2}{m}}$$

- Sum the ε 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 ε

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



(a) Snapdragon 800 (b) Tegra K1

Large-Picture Decomposition Algorithm

Algorithm 1 Decomposition Search

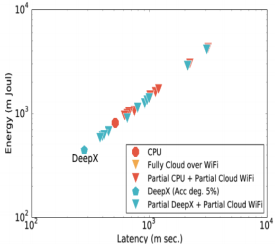
Input: (i) Model with n layers, (ii) $\mathcal{E}_{TH}$ (Allowed level of overall approximation error), and (iii) $e_1, e_2, \ldots, e_k$ (Energy footprint of all available processors).
for all layer$_i$ $\in$ Model **do**
    layerType = getLayerType(layer$_i$) ▷ Identifying layer type based on operations
    if layerType == convolution or pooling **then**
        BlockSize = extractFilteringBlocks()
    else ▷ Fully connected layers
        BlockSize = extractFeedForwardBlocks()
    for $j = 1$ to P **do** ▷ Extracting parameters for all processors
        E_j, B_j =getProcessorParameters(BlockSize, e_j)
    if layerType == Feed-forward **then**
        for k=90,-10,10 **do** ▷ Linear searching parameter space
            $\mathcal{E}$ =CompressSVD($W^{layer_i}_{m \times n}$, k) ▷ Estimating Reconstruction Error
            if $\mathcal{E} < \mathcal{E}_{TH}$ **then**
                Save $U_{m \times c}$ and $N^T_{c \times n}$
            else
                break ▷ Stop parameter searching
        updateLayer($layer_i$, $U_{m \times c}$, $N^T_{c \times n}$)
    applyOptimization(BlockSize, $\{E\}^k_{j=1}, \{B\}^k_{j=1}$**)** ▷ using Equation 5a
Assign blocks to processors as identified by the optimizations

Recomposition

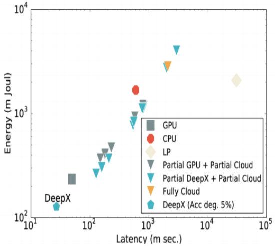
$$
\begin{aligned}
\text{min.} \quad & \alpha \sum_{i=1}^{P} E_i B_i + \beta \max_{i \in \mathcal{P}} \{T_i B_i\} \\
\text{s.t.} \quad & \sum_{i=1}^{P} B_i = N \\
& B_i \leq L_i, \forall i \in \mathcal{P}, \\
& B_i \geq 0, B_i \in \mathcal{Z}, \forall i \in \mathcal{P},
\end{aligned}
$$

- $\mathcal{P} = \{1, 2, ...P\}$ 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

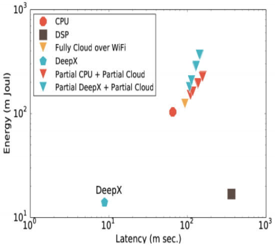
Results



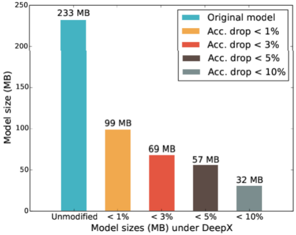
(a) AlexNet – Snapdragon



(b) AlexNet – Tegra



(c) SpeakerID – Snapdragon



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,"
arXiv preprint arXiv:1412.6115, 2014.

T. He, et al.,
"Reshaping deep neural network for fast decoding by nodepruning,"
ICASSP '14

The End