Northwest University China qq@stumail.nwu.edu.cn Shaanxi Normal University China renjie@snnu.edu.cn Northwest University China gl@nwu.edu.cn

National University of Defense Technology China j.fang@nudt.edu.cn

Peking University China fengyansong@pku.edu.cn

Lancaster University China z.wang@lancaster.ac.uk

***Abstract*—Abstract comes here...**

¡ccs2012¿ ¡concept¿ ¡concept$_id >$ 10010520.10010553.10010562.10010564 $< /concept_id >< concept_desc > Computersystemsorganization~Embeddedsoftware < /concept_desc >< concept_significance >$ 500 $< /concept_significance >< /concept >< concept >< concept_id >$ 10010147.10010169 $< /concept_id >< concept_desc > Computingmethodologies~Parallelcomputingmethodologies < /concept_desc >< concept_significance >$ 300 $< /concept_significance >< /concept >< /ccs2012 >$

[500]Computer systems organization Embedded software [300]Computing methodologies Parallel computing methodologies

***Keywords*-eep learning, Adaptive computing, Embedded systemseep learning, Adaptive computing, Embedded systemsD**

# Understanding Model Compression for Embedded Deep Inference

Zheng Wang

## I. Introduction

FIX:ZW: I will come back to this later.

## II. Motivation

Compressing a deep learning model into a smaller size typically leads to a smaller model footprint, but it does not always lead to faster inference time and lower energy consumption. As a motivation example, considering applying FIX:four commonly used model compression techniques to FIX:xx influential deep learning models – including FIX:xx CNNs and FIX:RNNs. Our evaluation platform is a NVIDIA Jetson TX2 embedded deep learning platform with Tensorflow Mobile FIX:v.xx (see Section III-A).

Figure ?? summarizes the change of the storage size, memory footprint, inference time, and energy consumption after applying each model compression technique.

1

¿¿¿¿¿¿¿ origin/master

## III. Experimental Setup

### A. Platform and Models

Hardware. Our experimental platform is the NVIDIA Jetson TX2 embedded deep learning platform. The system has a 64 bit dual-core Denver2 and a 64 bit quad-core ARM Cortex-A57 running at 2.0 Ghz, and a 256-core NVIDIA Pascal GPU running at 1.3 Ghz. The board has 8 GB of LPDDR4 RAM and 96 GB of storage (32 GB eMMC plus 64 GB SD card).

System Software. Our evaluation platform runs Ubuntu 16.04 with Linux kernel v4.4.15. We use Tensorflow v.1.0.1, cuDNN (v6.0) and CUDA (v8.0.64).

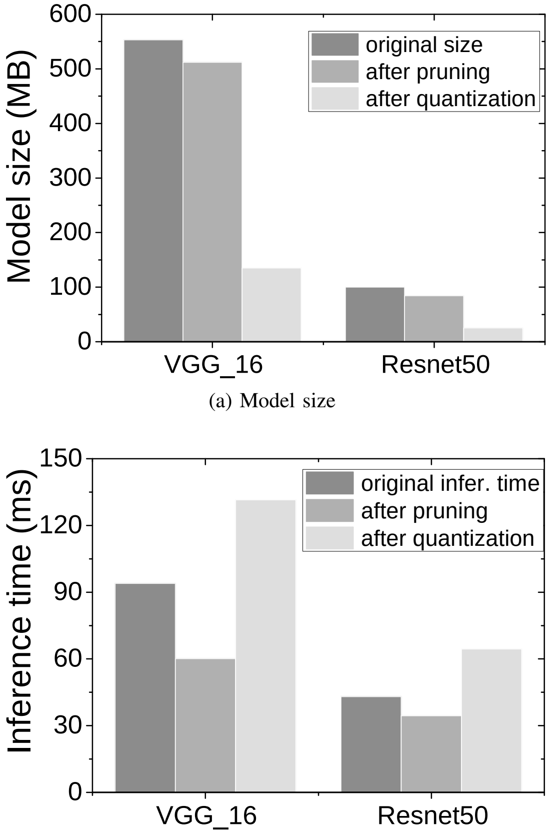
Deep Learning Models. We consider FIX:14 pre-trained CNN models for image recognition from the TensorFlow-Slim library [?]. The models are built using TensorFlow and trained on the ImageNet ILSVRC 2012 training set.

### B. Evaluation Methodology

Evaluation Metrics We consider the following metrics:

- *Inference time (lower is better)*. Wall clock time between a model taking in an input and producing an output, excluding the model load time.
- *Energy consumption (lower is better)*. The energy used by a model for inference. We deduct the static power used by the hardware when the system is idle.
- *Accuracy (higher is better)*. The ratio of correctly labeled images to the total number of testing images.
- *Precision (higher is better)*. The ratio of a correctly predicted images to the total number of images that are predicted to have a specific object. This metric answers e.g., "*Of all the images that are labeled to have a cat, how many actually have a cat?*".
- *Recall (higher is better)*. The ratio of correctly predicted images to the total number of test images that belong to an object class. This metric answers e.g., "*Of all the test images that have a cat, how many are actually labeled to have a cat?*".
- *F1 score (higher is better)*. The weighted average of Precision and Recall, calculated as $2 \times \frac{Recall \times Precision}{Recall + Precision}$. It is useful when the test datasets have an uneven distribution of object classes.

Performance Report. To collect inference time and energy consumption, we run each model on each input repeatedly until the $95\%$ confidence bound per model per input is smaller than $5\%$. In the experiments, we exclude the loading time of the CNN models as they only need to be loaded once in practice. To measure energy consumption, we developed a lightweight runtime to take readings from the on-board energy sensors at a frequency of 1,000 samples per second. We then matched the energy readings against the time stamps of model execution to calculate the energy consumption.



(a) Model size

(b) Inference time

Figure 1: The achieved model size (a) and inference time (b) before and after the compression by quantization and pruning. There is significant room for improvement.

# IV. Experimental Results

## *A. Overall Results*

## *B. Compare to Oracle*

## *C. Analysis*

# V. Related Work

**DNNs have shown astounding successes in various tasks that previously seemed difficult []. Despite the fact that many embedded devices require precise sensing capabilities, adoption of DNN models on such systems has notably slow progress. This mainly due to DNN-based inference being typically a computation intensive task, which inherently runs slowly on embedded devices due to limited resources.**

**FIX:Talk about different compression techniques.**

**There is an extensive body of work on how to accelerate DNN training using xx, xx, and xx. Our work aims to understand how to accelerate deep learning inference by choosing the right model compression technique.**

**As an alternative to on-device inferencing, off-loading computation to the cloud can accelerate DNN model inference [?]. Neurosurgeon [?] identifies when it is beneficial (*e.g.* in terms of energy consumption and end-to-end latency) to offload a DNN layer to be computed on the cloud. The Pervasive CNN [?] generates multiple computation kernels for each layer of a CNN, which are then dynamically selected according to the inputs and user constraints. A similar approach presented in [?] trains a model twice, once on shared data and again on personal data, in an attempt to prevent personal data being sent outside the personal domain. Computation off-loading is not always applicable due to privacy, latency or connectivity issues. The work presented by Ossia ifnextchar.et alet al.partially addresses the issue of privacy-preserving when offloading DNN inference to the cloud [?]. Our work is complementary to prior work on computation off-loading by offering insights to choose the optimal compression technique to best optimize local inference.**

# VI. Conclusions

**This paper has presented**

# References

**[1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.**