A Hybrid Approach to Offloading Mobile Image Classification

Johann Hauswald*, Thomas Manville*, Qi Zheng*, Ronald Dreslinski*, Chaitali Chakrabarti*, Trevor Mudge*
*University of Michigan – Ann Arbor, §Arizona State University

Abstract

- Current mobile devices offload some computation in image classification applications to the cloud to save time and device power.
- We find that two computations (feature extraction and matching play) have the potential to be executed locally.
- We analyze the ability of a mobile platform to execute feature extraction and matching, and prediction workloads under various offloading scenarios.
- The best scenario is to execute feature extraction with an onboard GPU and to offload the rest of the computation (11% faster than the next best scenario).
- Alternatively, compressing and sending the image over the network achieves lowest data transferred (2.5x better) and lowest energy usage (3.7x better) than the next best option.

Predictive Models

- Feature extraction, description + matching have the potential to be accelerated or offloaded to save onboard resources or decrease runtime.
- Onboard image-prediction is currently infeasible because device storage is limited and prediction models require MBs-GBs of storage.

Scenarios for Offloading Image-Classification

Methods

- We use OpenCV C++ implementations of the image-processing algorithms in the pipeline (above figure).
- Training and prediction images are from the Caltech 256 dataset and a Samsung Galaxy S3 mobile phone.
- Final runtime results are based on images from the Galaxy S3 mobile phone. We use larger images from the S3 for the image-size dependent components of the pipeline (feature extraction, description, and matching) to give an accurate estimate of runtime.

Breakdown of total runtime for all configurations. Runtime values based on Samsung Galaxy S3 images. The bold box shows a comparison of runtime, data transferred and energy for each configuration. The values for the recommended configurations are also in bold.

Payload Size

- We use Google Protocol Buffers to measure the size of the data after each stage in the pipeline.
- Based on a 2MB Image:
  - Feature extraction generates 500KB of data
  - Feature description + matching generates 5.7MB of data

Conclusion

- Proposed an efficient solution to maximize runtime and limit onboard resource usage.
- The best configuration for optimal runtime (11% faster) executes feature extraction with a GPU onboard and offloads the rest of the pipeline.
- Alternatively, compressing and sending the image over the network achieves lowest data transferred (2.5x better) and lowest energy usage (3.7x better) than the next best option.