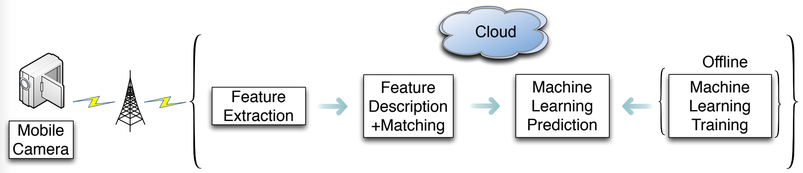
A Hybrid Approach to Offloading Mobile Image Classification

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

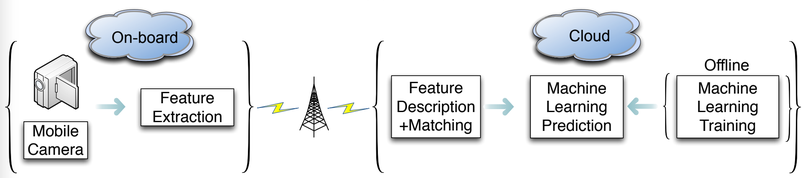
Abstract

- 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.

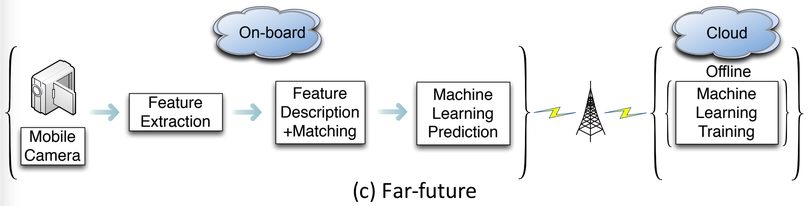
Mobile Image-Classification Pipeline



(a) Current



(b) Near-future



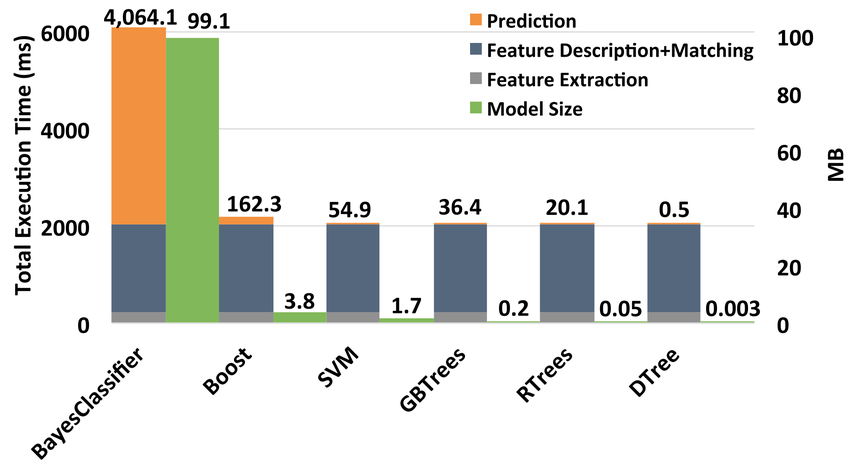
(c) Far-future

As mobile platforms become more energy efficient, more of this pipeline will be able to run locally.

Methodology

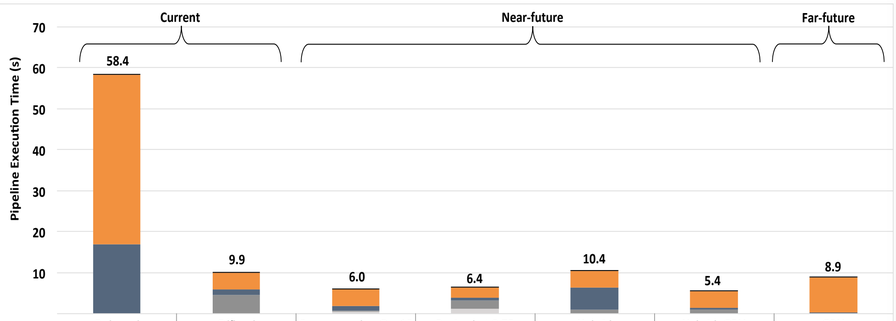
- We use OpenCV C++ implementations of the image-processing algorithms in the pipeline (above figure).
- Training and prediction images are from the Caltech 256 dataset and a Samsung Galaxy S3 mobile phone.
- Final runtime results are based on images from the Galaxy S3 mobile phone. We use larger images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

Runtime Breakdown of the Classification Pipeline



- 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.

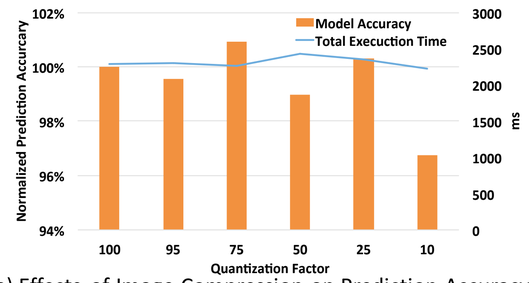
Scenarios for Offloading Image-Classification



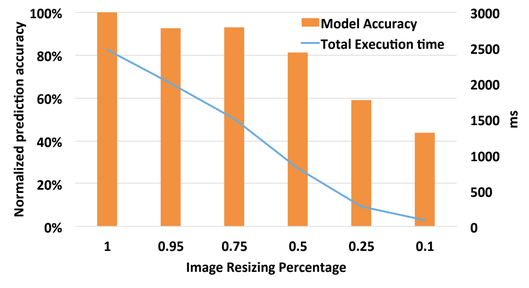
	Onboard	Offload	Compressed at Q = 10	Resized at 0.75	Hybrid	Hybrid+GPU	GPU
Prediction (ms)	56	14	14	14	14	14	56
Feature Description+Matching (ms)	41,317	4,067	4,067	2,434	4,067	4,067	8,464
Feature Extraction (ms)	17,022	1,138	1,138	635	5,367	371	371
Upload (ms)	-	4,670	358	2,174	989	989	-
Compression/ Resize Time (ms)	-	-	437	1,130	-	-	-
Total Time (s)	58.4	9.9	**6.0**	6.4	10.4	**5.4**	8.9
Data Transferred (MB)	-	2.3	**0.2**	1.1	0.5	**0.5**	-
Total Energy (J)	93.4	3.5	**0.6**	2.2	8.7	**5.0**	116.5

- 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.
- For best runtime, we recommend a Hybrid+GPU configuration.
- For lowest data transferred and energy consumption, we recommend compressing the image onboard and offloading computation.

Effects of Image Preprocessing



(a) Effects of Image Compression on Prediction Accuracy



(b) Effects of Image Resizing on Prediction Accuracy

Since JPEG compression does not significantly impact prediction accuracy (compared to image resizing), we propose it as a technique to reduce the amount of data to transfer.

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 (2.5× better) and lowest energy usage (3.7× better) than the next best option.