Workload Prediction for Adaptive Power Scaling Using Deep Learning

Steve Tarsa, Amit Kumar, & HT Kung
Harvard, Intel Labs MRL
May 29, 2014
ICICDT '14

HARVARD
School of Engineering and Applied Sciences

In these slides…

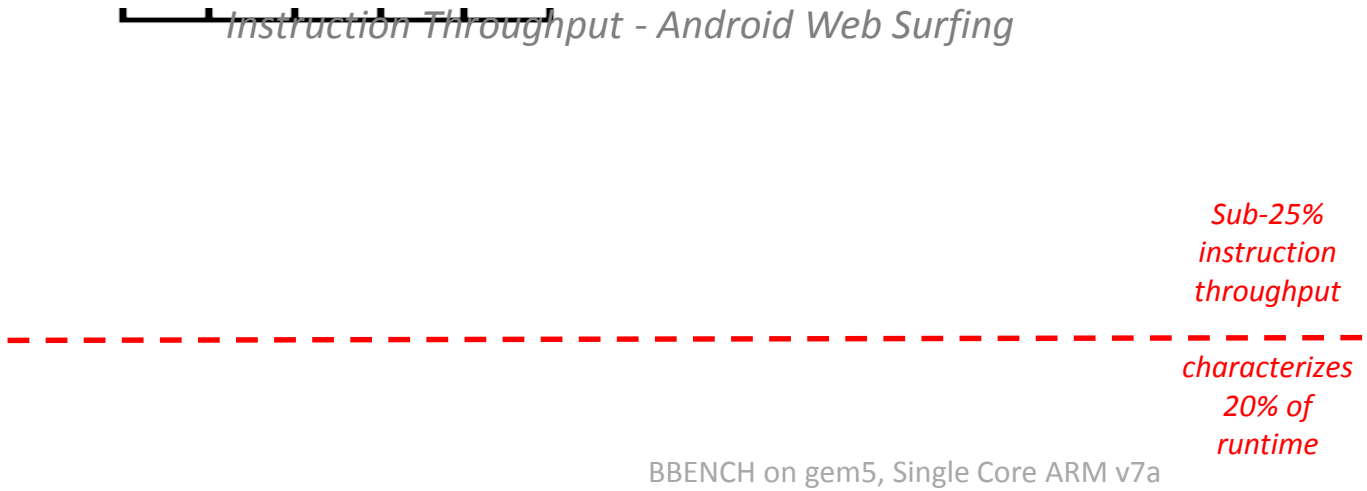
Machine learning (ML) is applied to performance counters in order to model workloads and predictively optimize frequency/voltage

- Deep machine learning (ML) methods are popular due to successes in computer vision, natural language processing, etc.
- We demonstrate that ML improves statistical accuracy over techniques like regression in complicated scenarios, for which accurate models are elusive
- At the architecture level, we use ML to capture hidden structure in counter data that corresponds to cross-layer user/OS/chip interactions

Hierarchical sparse coding improves accuracy and look-ahead range for predicting instruction throughput dips, giving more time for chip adjustment

- Multi-layer (i.e. “deep”) ML models first extract canonical features, and then their interrelationships to find high-dimensional patterns over time on little training data
- Our methods rely on pattern matching, and can be implemented in circuitry with simple low-precision inner product computations
- *We demonstrate 3x improvement in look-ahead range and a 50% power reduction during throughput dips for web surfing on an ARMv7a/Android Gingerbread device*

User-driven workloads, e.g. web surfing, have many opportunities for dynamic power optimization using DVFS, when instruction throughput drops temporarily



But, anticipating dips in CPU activity requires modeling complicated interactions between users, OS/apps, and chip architecture

User & Workload

e.g. Browsing habits, multi-tasking habits

OS & Application

e.g. Process Management, Web Page Caching Policy

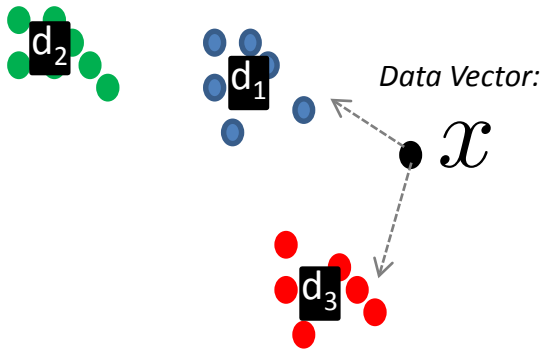
Chip Architecture

e.g. Data or Instruction cache configuration

Instead of modeling by hand, machine learning extracts “hidden” structure from raw data, yielding statistical models with better prediction and training requirements than standard regression methods

From hardware-counter time series data, we extract common patterns using a clustering algorithm; clusters become atoms in a feature dictionary

Counter Name	Description
cpu.committedInsts	# Committed Instructions
cpu.num_fp_register_reads	# times fp registers read
cpu.dtb.read_accesses	DTB Read accesses
cpu.dtb.read_hits	DTB Read hits
cpu.dtb.read_misses	DTB Read misses
cpu.dtb.flush_entries	# entries flushed from DTB
	⋮

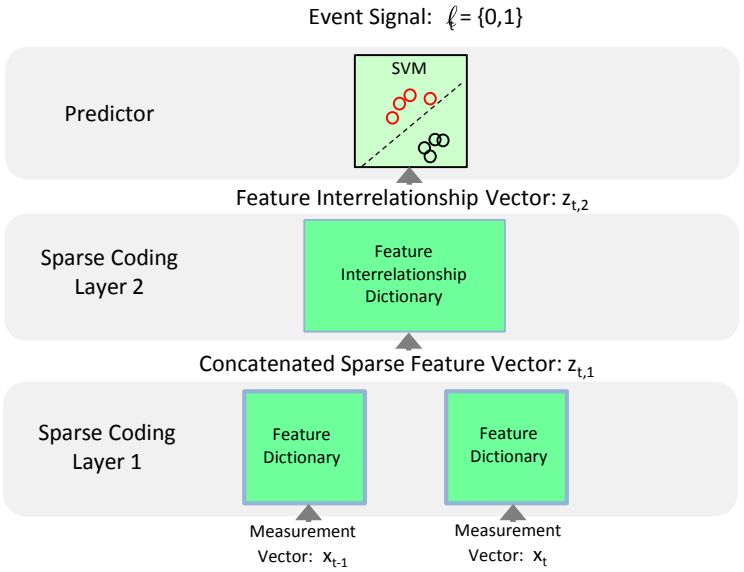


Sparse Code:

$$z = [1, 0, 1]^T$$

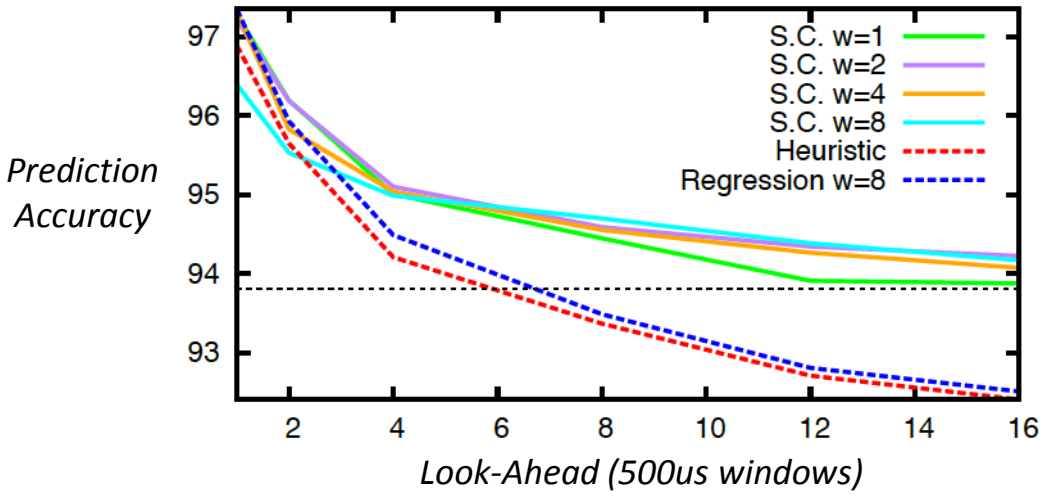
Expressing raw data in terms of a few prominent features removes noise, and generalizes a few training examples for good statistical accuracy under variation

Deep architectures use multiple layers to first find simple features within short windows, and then find feature interrelationships over larger time scales



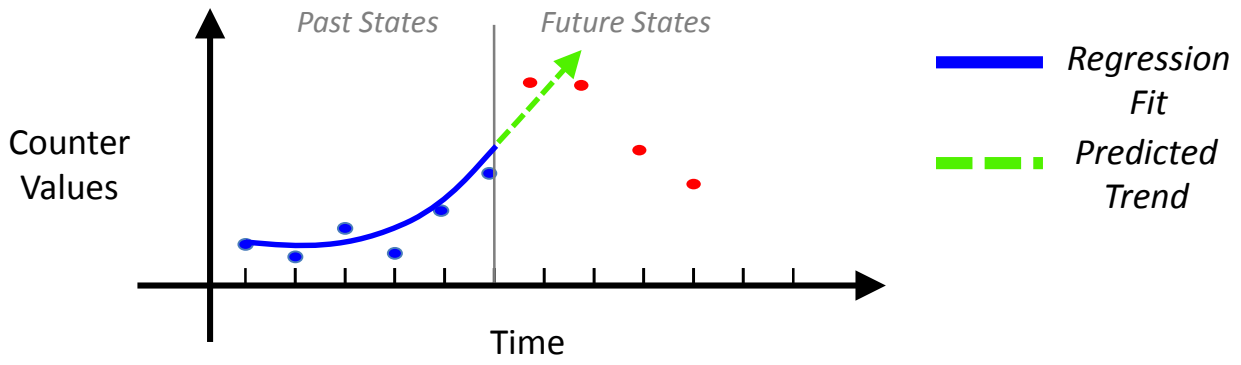
Our prediction method, hierarchical sparse coding + linear SVM classification, relies on pattern matching, and can be built into circuitry with low-precision inner-product computations

Compared to predictions based on regression modeling or heuristics, learned feature-space signatures yield useful predictions with 3x longer look-ahead, giving more time for chip adjustment



Signatures captured over the longest time scales give stable long term predictions, with up to 8ms heads-up.

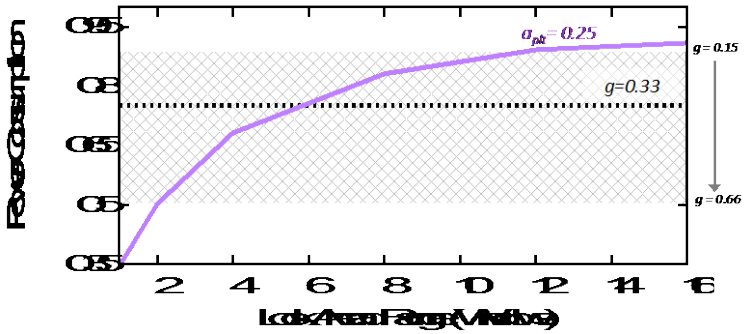
Absent a system model, regression extrapolates observed data to predict future states based on the assumption that counter values change smoothly over time



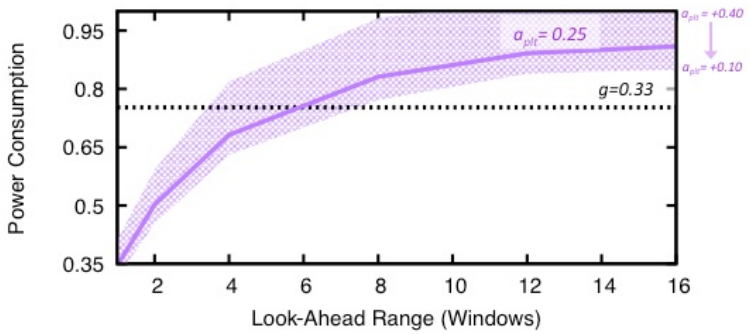
This assumption only holds over small time scales and at high sampling rates, meaning that regression-extrapolated predictions are only useful for short ranges

Power savings are subject to a predictor's false alarms, so we model P_{dyn} relative to baseline power (i.e. gating efficiency) and the cost of false positive recovery

Baseline Power Consumption as Gating Efficiency Increases



Power Consumption with DVFS, as False Positive Recovery Cost Decreases



For a 0.33 gating-efficient design, with a recovery cost of +0.25 additional switching activity, predictive DVFS reduces P_{Dyn} by 50% with 1 ms heads up for chip adjustment

Summary and next steps…

Online deep learning holds promise for chip optimizations, though implementation will come in parts…

- *Offline* learning may yield good static rules that capture much of low-hanging fruit
- Architectures for low-power dictionary learning are being explored
- "Small data" deep learning must be better explored, to optimize accuracy under time-biased training data

Instruction throughput prediction for DVFS is a first-step application, and we will explore others that may lead to larger gains

- Past successes: wireless link prediction
- Past failures: branch prediction, cache prefetching (*scenarios are easy-enough that standard tools perform just as well as ML!)*
- *Others…?*