Use of Synthetic Benchmarks for Machine-Learning-based Performance Auto-tuning

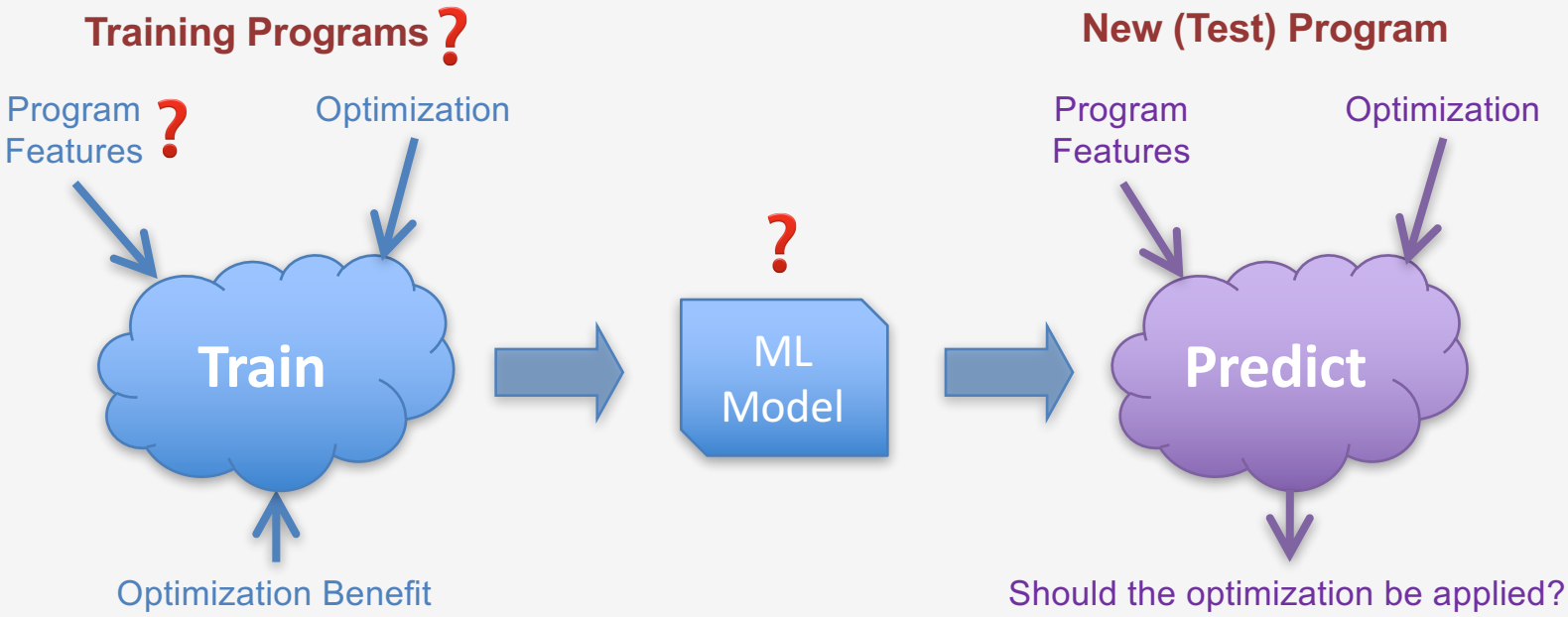
Tianyi David Han and Tarek S. Abdelrahman

The Edward S. Rogers Department of
Electrical and Computer Engineering
University of Toronto

david.han@alumni.utoronto.ca tsa@ece.utoronto.ca

Machine-Learning-based Auto-Tuning

- Increased interest in Machine Learning (ML) based auto-tuning, particularly for GPUs



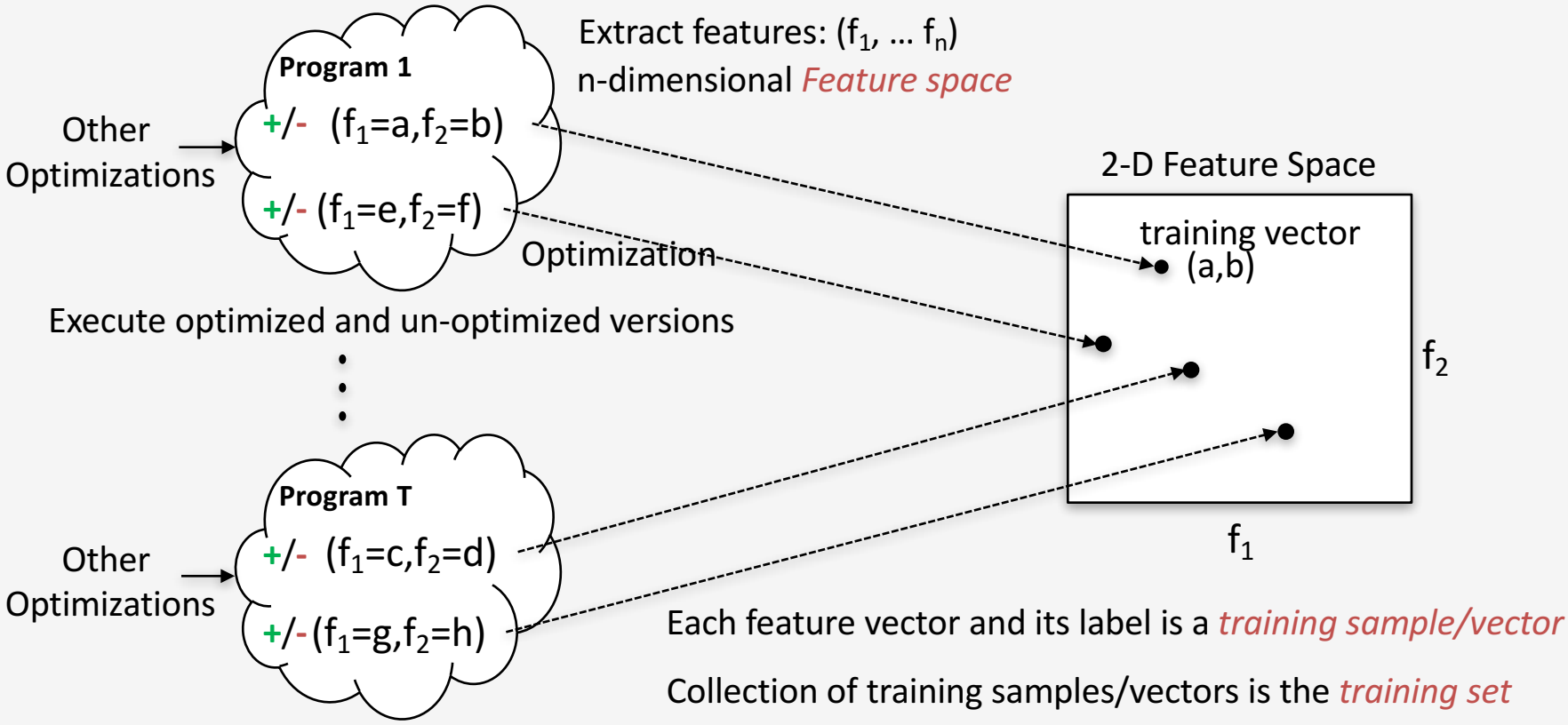
This Work

- Develop the *AVD* metric for the goodness training programs
- Consider the *local memory optimization* problem for GPUs
- Models trained with real programs are poor predictors
- Develop systematic ways for generating synthetic benchmarks
- Show significant improvement in models' ability to predict when trained with synthetic benchmarks, by a factor of 1.5X

Outline

- Motivation
- Goodness of Training Set
- Local Memory Optimization
- Real and Synthetic Benchmarks
- Related work
- Conclusions and Future Work

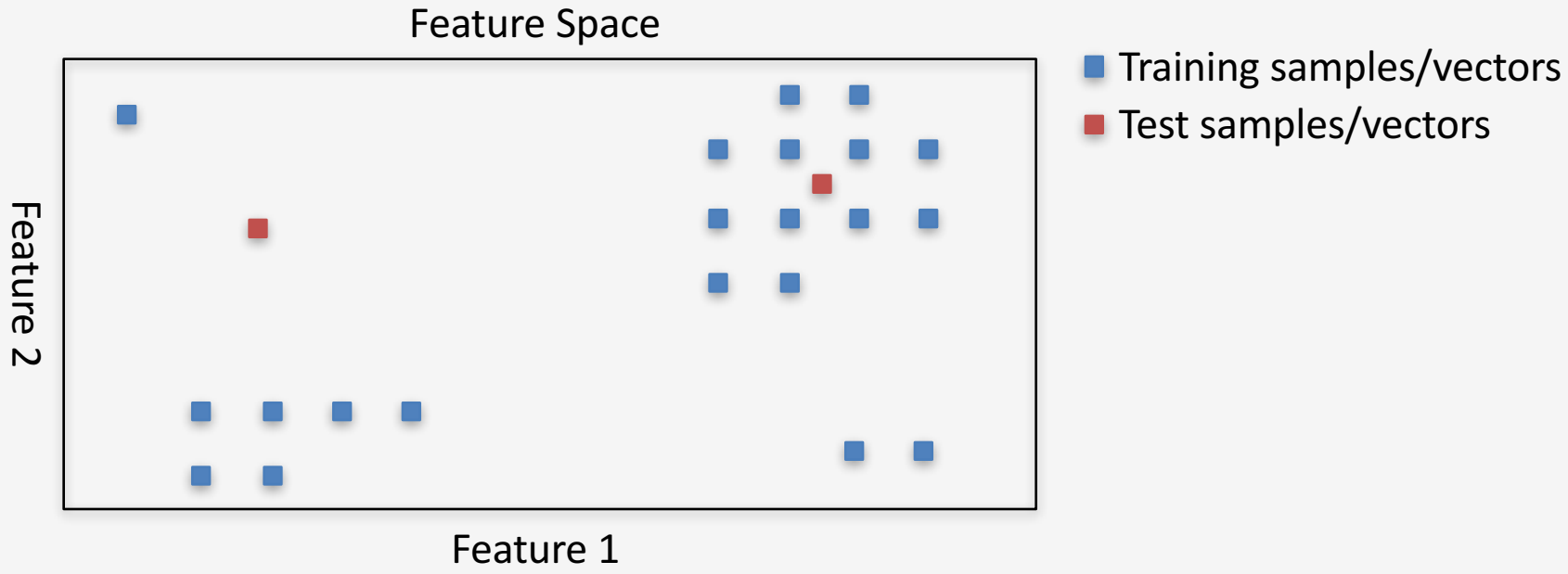
Preliminaries



Each feature vector and its label is a *training sample/vector*

Collection of training samples/vectors is the *training set*

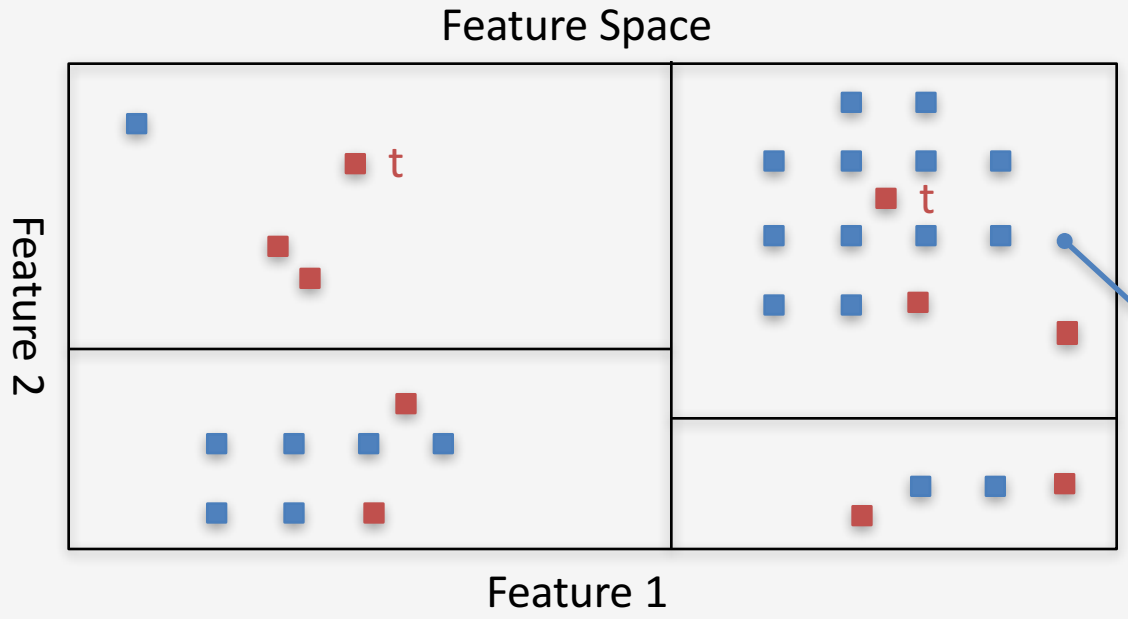
Goodness of Training Set



- How well do training samples "cover" test samples?

?

Coverage – Vicinity Density (VD)



- Training samples/vectors
- Test samples/vectors

Train a regression tree

size(R): # vectors in R

vectors(R): # training samples in R

VD(t): vectors(R_t)/size(R_t)

T: Test set

- **Average Vicinity Density: AVD(T)** = $\sum_{\forall\, t \in T} \mathbf{VD(t) / |T|}$

Outline

- Motivation
- Goodness of Training Set
- Local Memory Optimization
- Real and Synthetic Benchmarks
- Related work
- Conclusions and Future Work

Local Memory Optimization

- User-managed caching of data from the GPU device memory into fast local (shared) memory
  - Exploit data re-use and avoid the penalty of memory non-coalescing

+	−	−
Smaller number of global memory transactions	Overhead of data movement and synchronization	Potential for reduced parallelism

Factors: degree of non-coalescing, degree of reuse, parallelism, contextual accesses, etc.

Hard to predict if the optimization is beneficial [Han and Abdelrahman 2015]

Outline

- Motivation
- Goodness of Training Set
- Local Memory Optimization
- Real and Synthetic Benchmarks
- Related work
- Conclusions and Future Work

Real Benchmarks

Benchmark	Application Domain	Description
transpose	Dense Linear Algebra	Matrix transpose
matrixMul_A		Matrix multiply (cache A)
matrixMul_B		Matrix multiply (cache B)
MVT		Matrix vector multiply
SGEMM		C = alpha * A * B + beta * C
convSep_row	Structured Grid (Stencil)	Separable 2D convolution (row filter)
convSep_col		Separable 2D convolution (column filter)
blur		Blur filter
SAD		Sum of Absolute Difference
SAD_frame		Cache the frame image in SAD
LBM		Lattice Boltzman Machine (struct elements)
STENCIL		3D, 27-point
MRI-GRIDDING	Unstructured Grid	Maps non-uniform 3D input data onto a regular 3-D grid

ML Features

- We use 15 features:
  - Inter-warp (single-access) data reuse
  - Stencil pattern data-reuse
  - Memory non-coalescing (single-access)
  - Total amount of (all) data accessed by a workgroup
  - Data utilization rate in cooperative loading
  - Cooperative loading efficiency
  - Parallelism levels before and after the optimization
  - Grid efficiency
  - Computation length
  - # of contextual accesses
  - Memory non-coalescing in contextual accesses

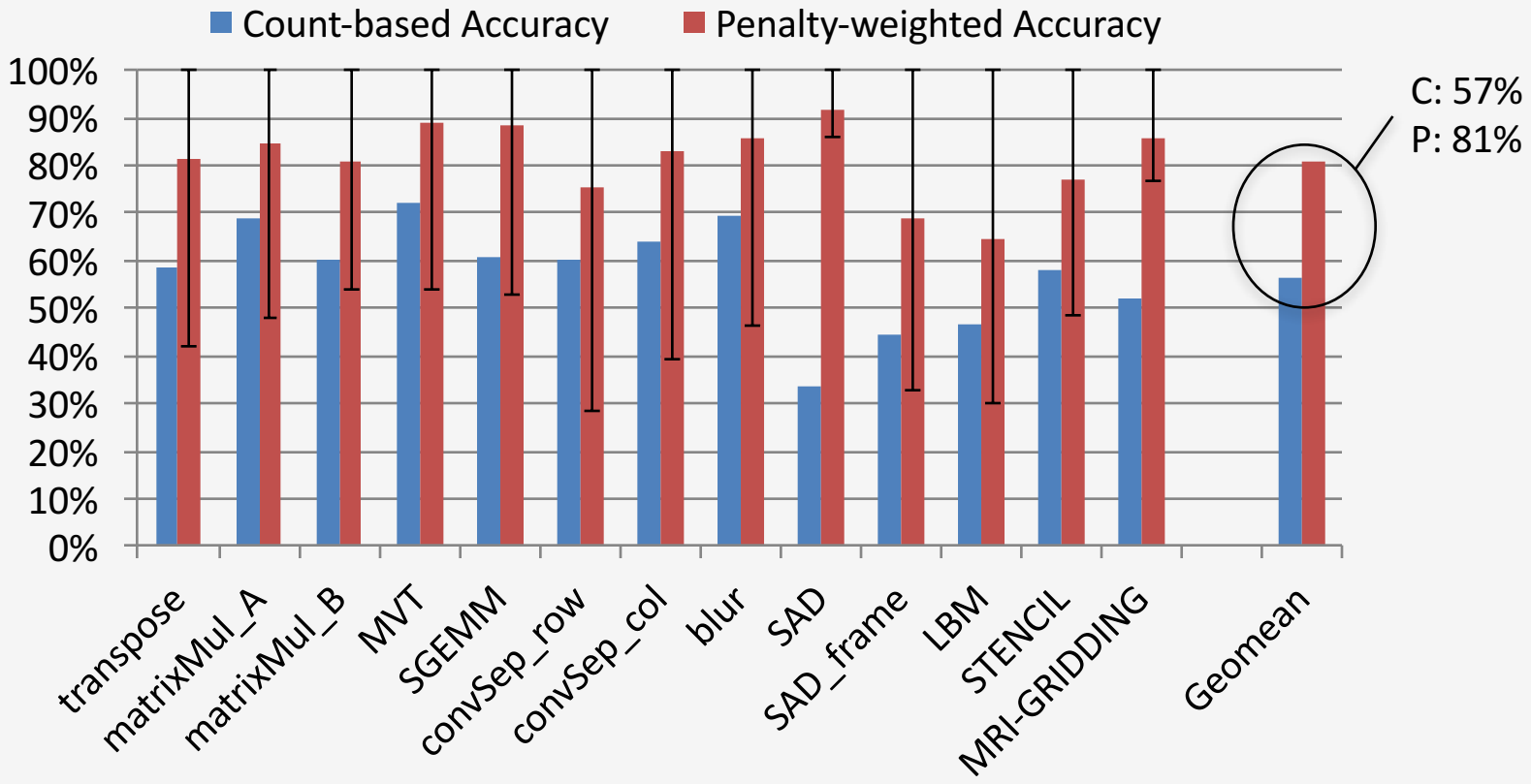
Evaluation Methodology

- Execute every benchmark with and without local memory optimization for all possible launch configurations
  - Nvidia Tesla 2090, CUDA 6.0, Intel Xeon E5-2620 host
  - Calculate the speedup of the optimized version
  - Label each benchmark/launch configuration (beneficial/not beneficial)
- Leave-one-out evaluation
  - Build a model with 12 benchmarks and predict for the 13th
  - Each model a Random Forest with 20 trees, 4 features per tree
  - These are called real models because they are trained with real benchmarks data

Model Accuracy

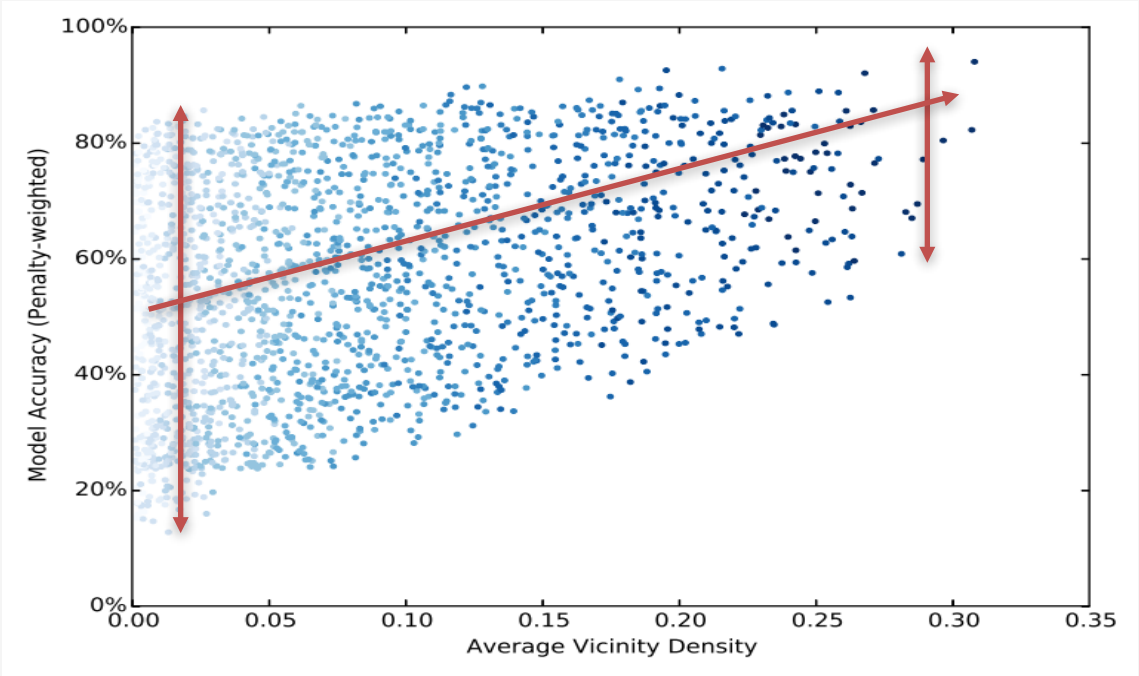
- Use the known labels of the test set to determine the accuracy of the model
  - Averaged for each benchmark over all launch configurations
- **Count-based** prediction accuracy
  - % of test vectors where the prediction is correct
- **Penalty-weighted** prediction accuracy
  - % of performance achieved by the prediction
  - Weigh the misprediction by the degradation in performance it causes

Real Models Accuracies



Model Accuracy vs. AVD

- For each real benchmark as a test set, evaluate the model accuracy and AVD for all possible training sets



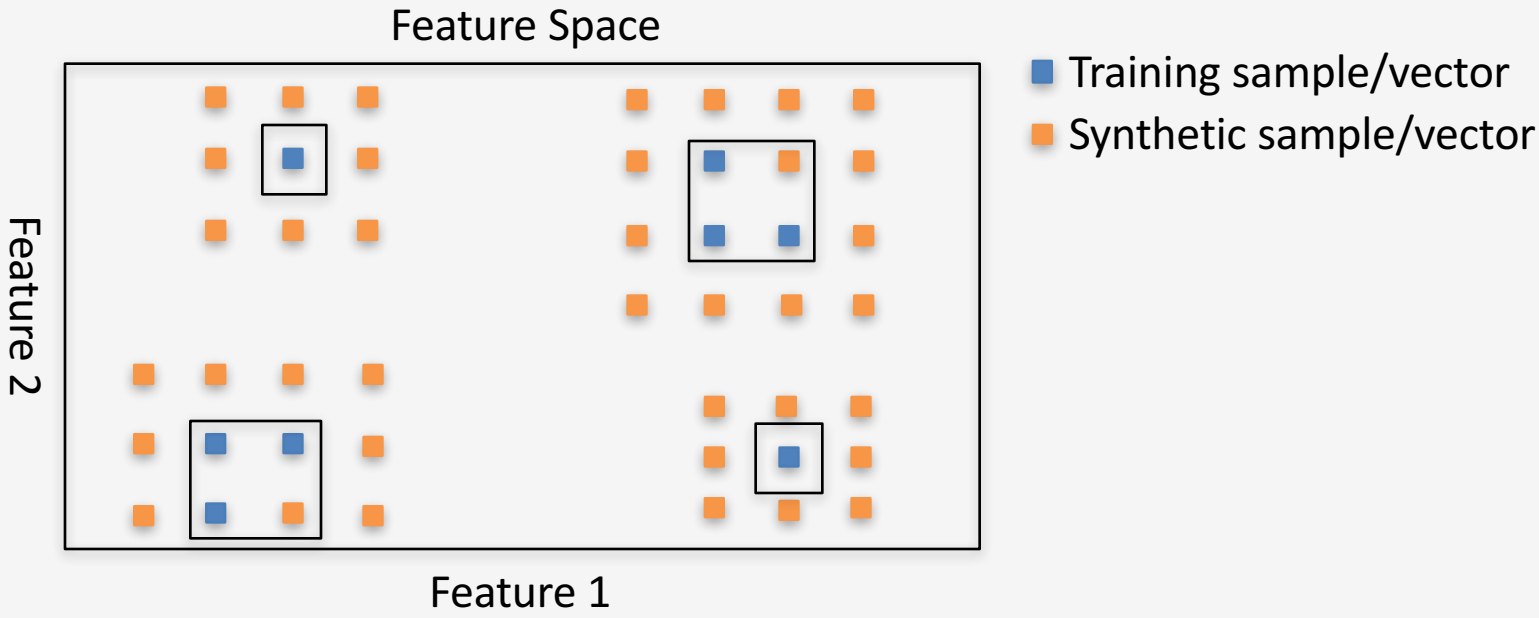
$2^{12} - 1 = 4095$ possible training sets per test set

4095*13 = ~53K models

Increasing the AVD

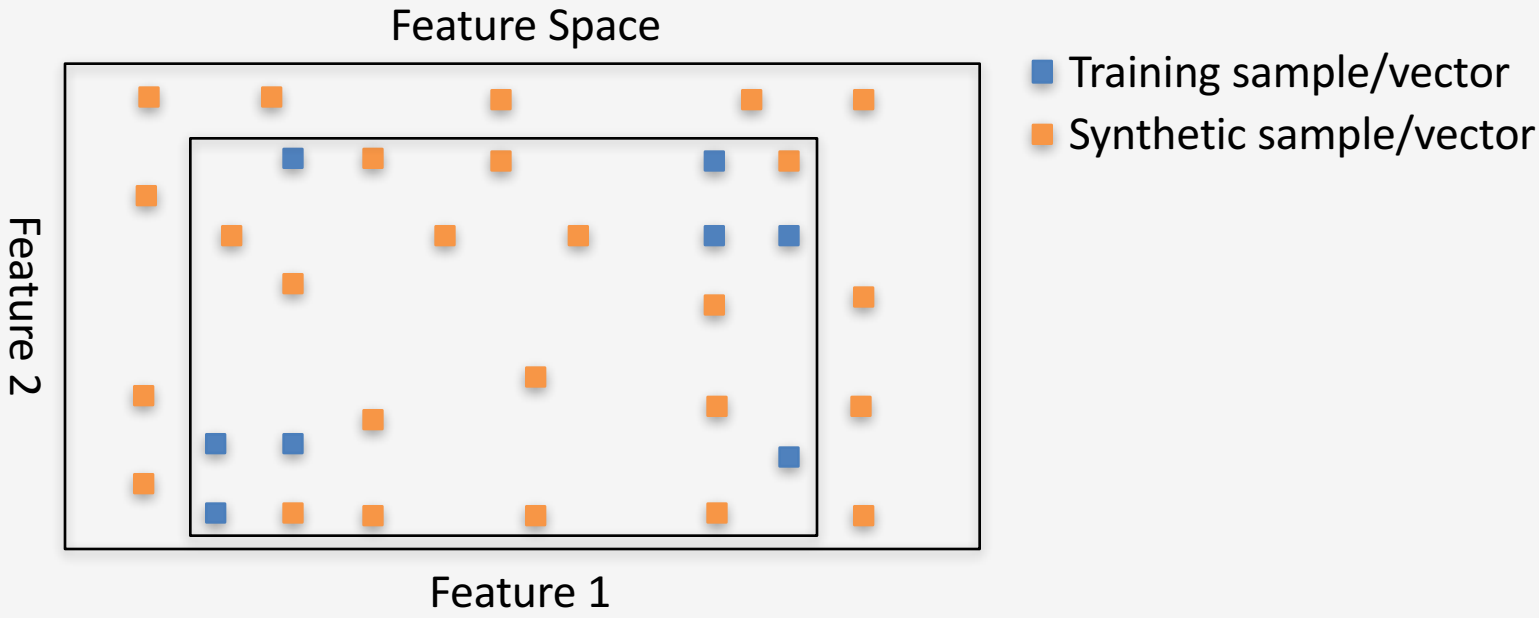
- Generate “synthetic” feature vectors that increase the AVD for test sets
  - Challenge: no a priori knowledge of the test data
- Use a synthetic benchmark template to generate code instances that have these feature vectors
- Repeat the previous experiment, training with synthetic (and real) benchmark samples
  - The models are called synthetic models because they are trained with synthetic benchmarks

Synthetic Vectors: Bounding Box (*bb-k*)



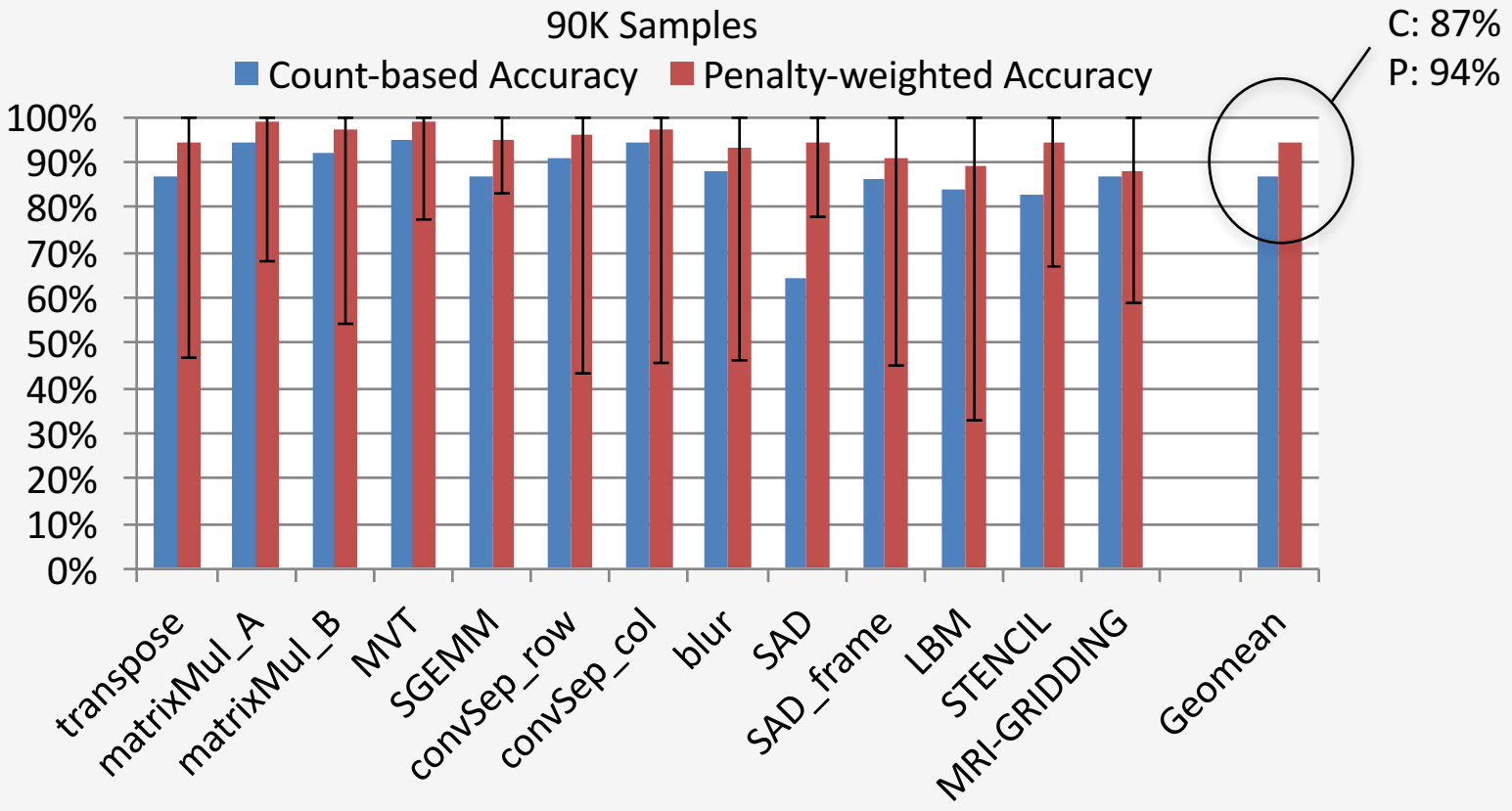
- Uniformly sample in expanded bounding box of training data of each benchmark

Synthetic Vectors: Bounding Box (*bb-all*)

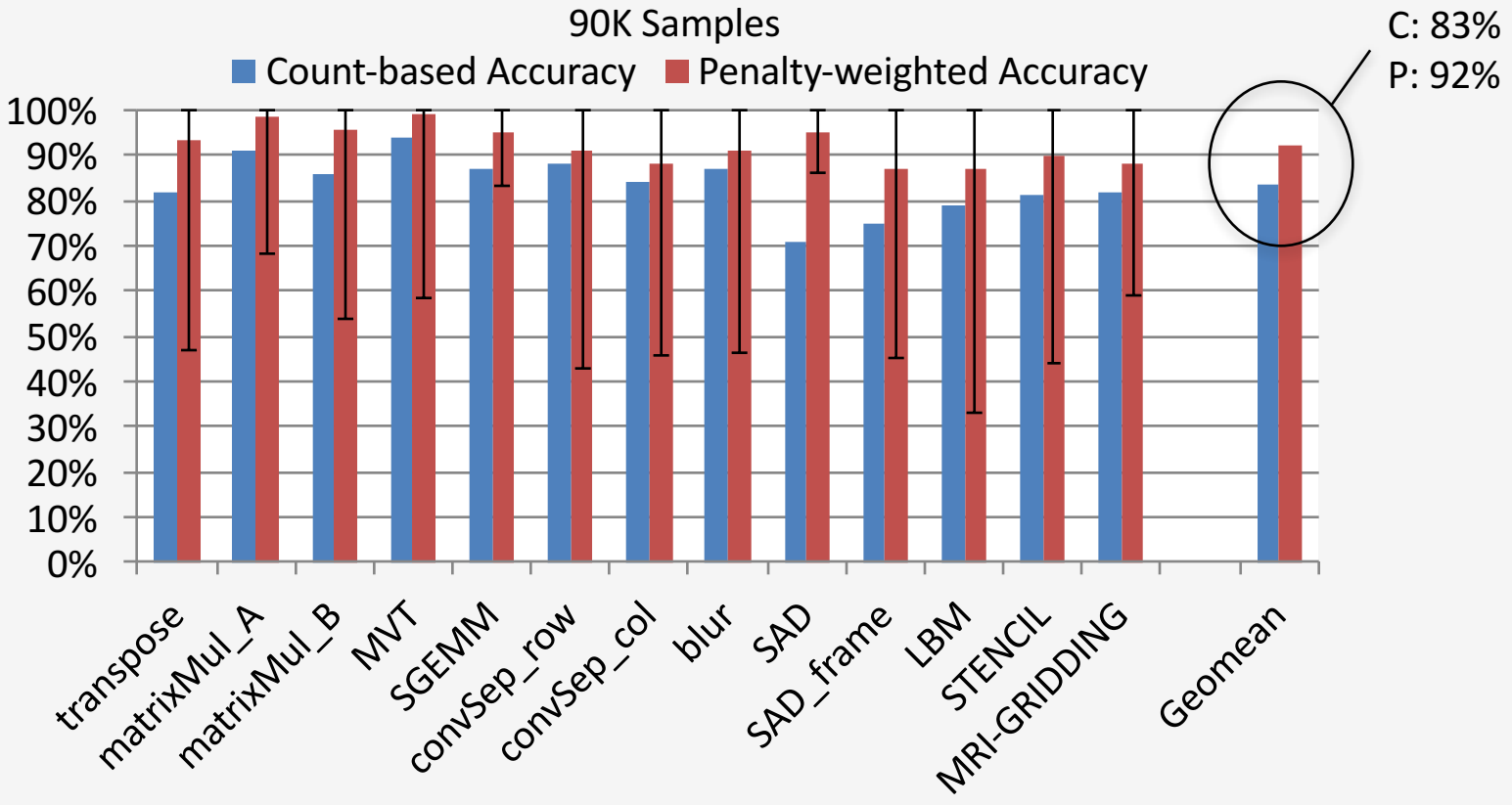


- Uniformly sample in expanded bounding box of all training data of benchmarks

Synthetic Models Accuracies – *bb-k*



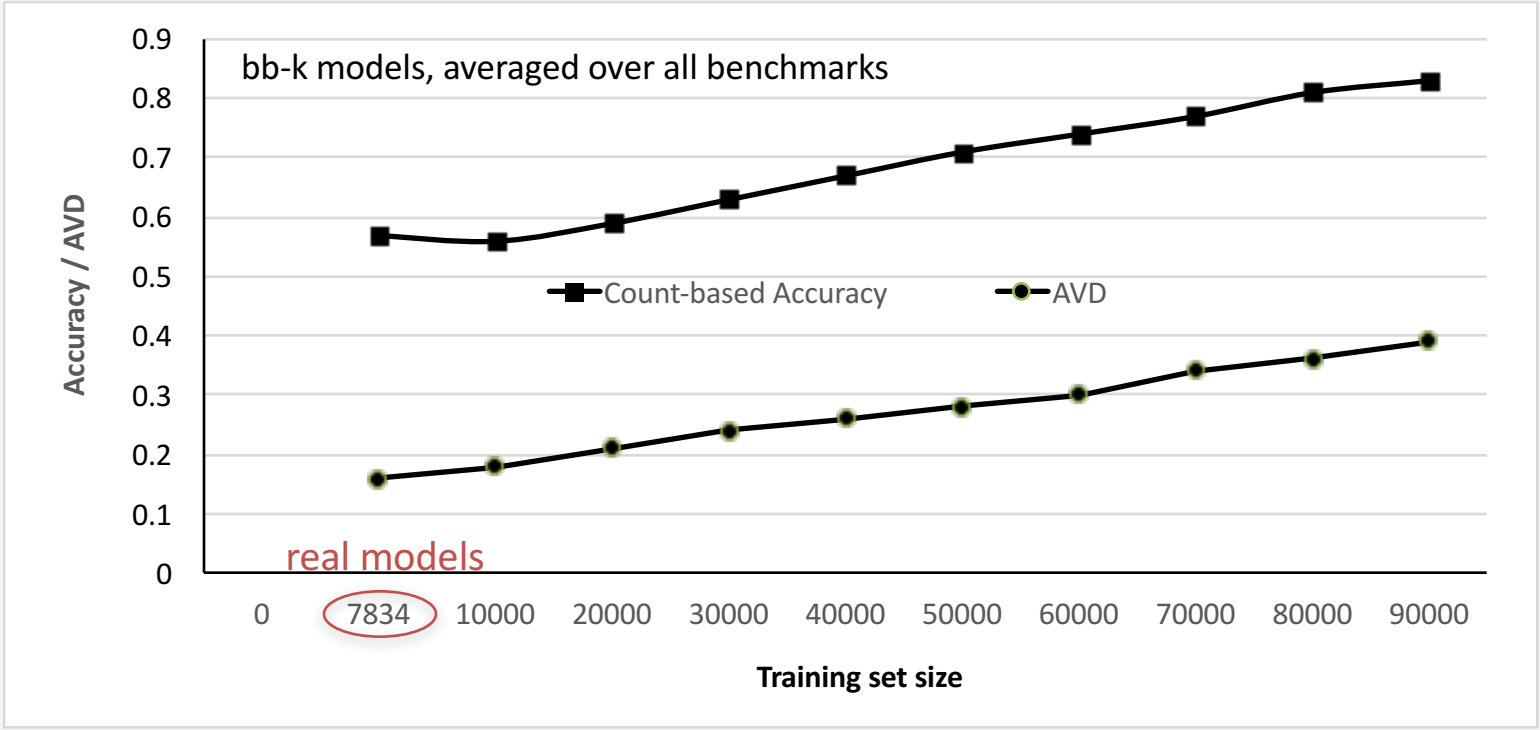
Synthetic Models Accuracies – *bb-all*



Model Accuracy

- We can significantly increase the model accuracy by using synthetic benchmarks, derived to increase the AVD
  - 90,000 synthetic samples versus 7834 real ones (11.5X)
  - 1.52X/1.16X count-based/penalty-weighted prediction accuracies
  - 2.4X AVD

Impact of Training Set Size



Outline

- Motivation
- Goodness of Training Set
- Local Memory Optimization
- Real and Synthetic Benchmarks
- Related work
- Conclusions and Future Work

Related Work

- Considerable work on building machine learning models for performance auto-tuning [Grewe et al. 2013], [Magni et al. 2014], [Agakov et al. 2006], [Cavazos et al. 2007], etc.
  - The use of a small set (10's) of real programs
  - We use a large number of synthetic benchmarks for training
- Use of synthetic benchmarks [Han & Abdelrahman 2015], [Garvey & Abdelrahman 2015], [Cummins et al. 2016]
  - No systematic way for generating synthetic benchmarks
  - This work reasons why there is benefit and systemizes generation

Related Work – Cont'd

- Deep Learning for generating synthetic benchmarks by mining code repositories [Cummins et al. 2017]
  - Focus is on synthetically correct/human readable code
  - Our focus is complementary
- Data sets for training and testing, e.g. [Borovicka et al. 2012]
  - How to select a good subset of the data for training
  - Oversampling to balance data sets, e.g., SMOTE 2002
  - Data already exists and no focus on programs

Conclusions

- Advocated the use of synthetic benchmarks for training machine learning model
  - A metric for the quality of a training set with respect to a test set
  - The metrics show that models trained with a small number of real benchmarks have poor performance because of poor training data
- Proposed methods for generating synthetic benchmarks for training a ML model
  - Shown a significant improvement in model performance
- The use of synthetic benchmarks is effective and useful

Future Work

- This work is an initial step towards showing the effectiveness of synthetic benchmarks
  - Determining a good number of synthetic benchmarks
  - Other approaches for determining feature vectors of synthetic samples, e.g., a hybrid of bb-all and bb-k
  - More efficient ways of generating code from desired feature vectors
  - Other GPU optimizations
  - The use of deeper models for prediction, enabled by the abundance of synthetic data

Questions?