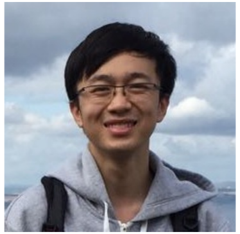
Predicting Inference Latency of Neural Architectures on Mobile Devices



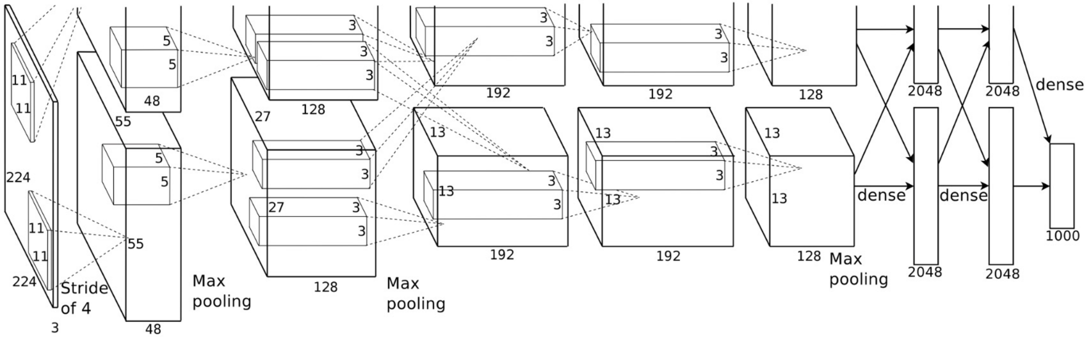
Zhuojin Li **Marco Paolieri** **Leana Golubchik**

qed.usc.edu

USC University of Southern California

ICPE, April 18, 2023

Background: ML Inference on Mobile Devices



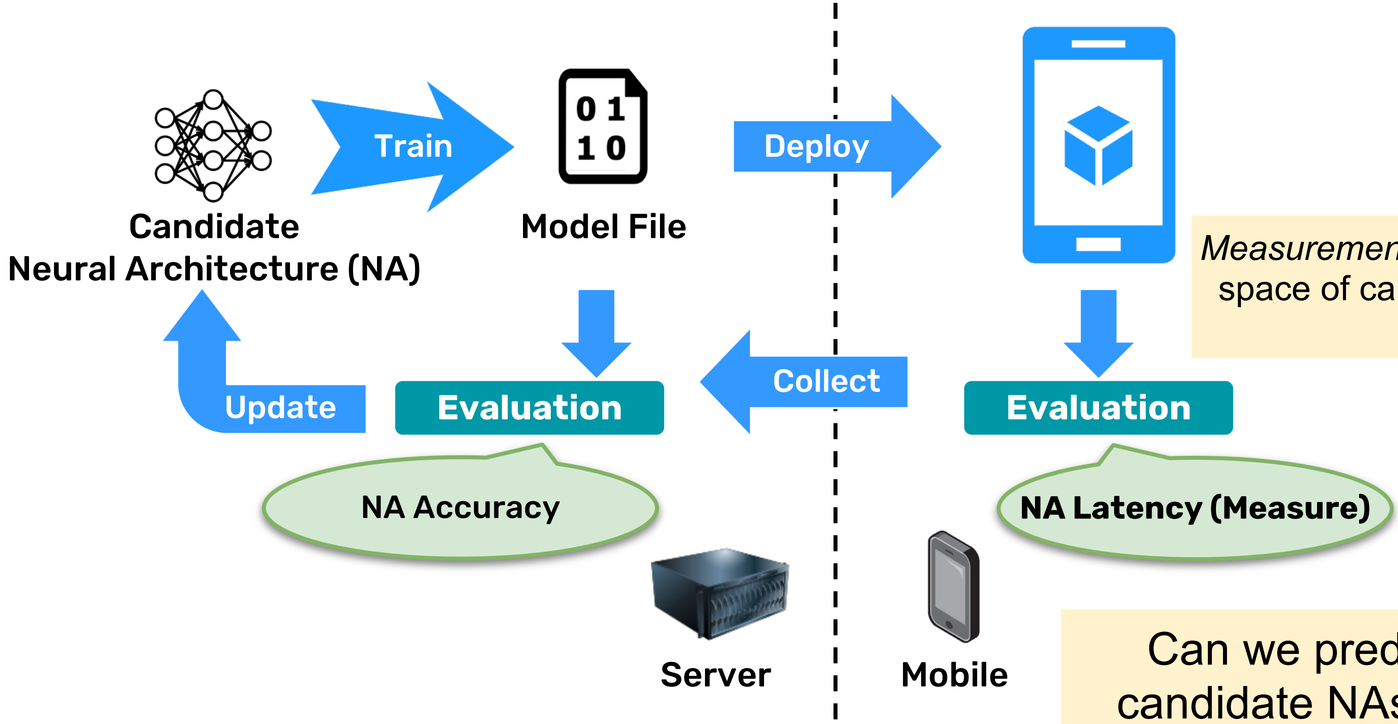
Convolutional Neural Networks
[Krizhevsky et al., 2012]

Deploy

Inference

Mobile Devices with Limited Resources

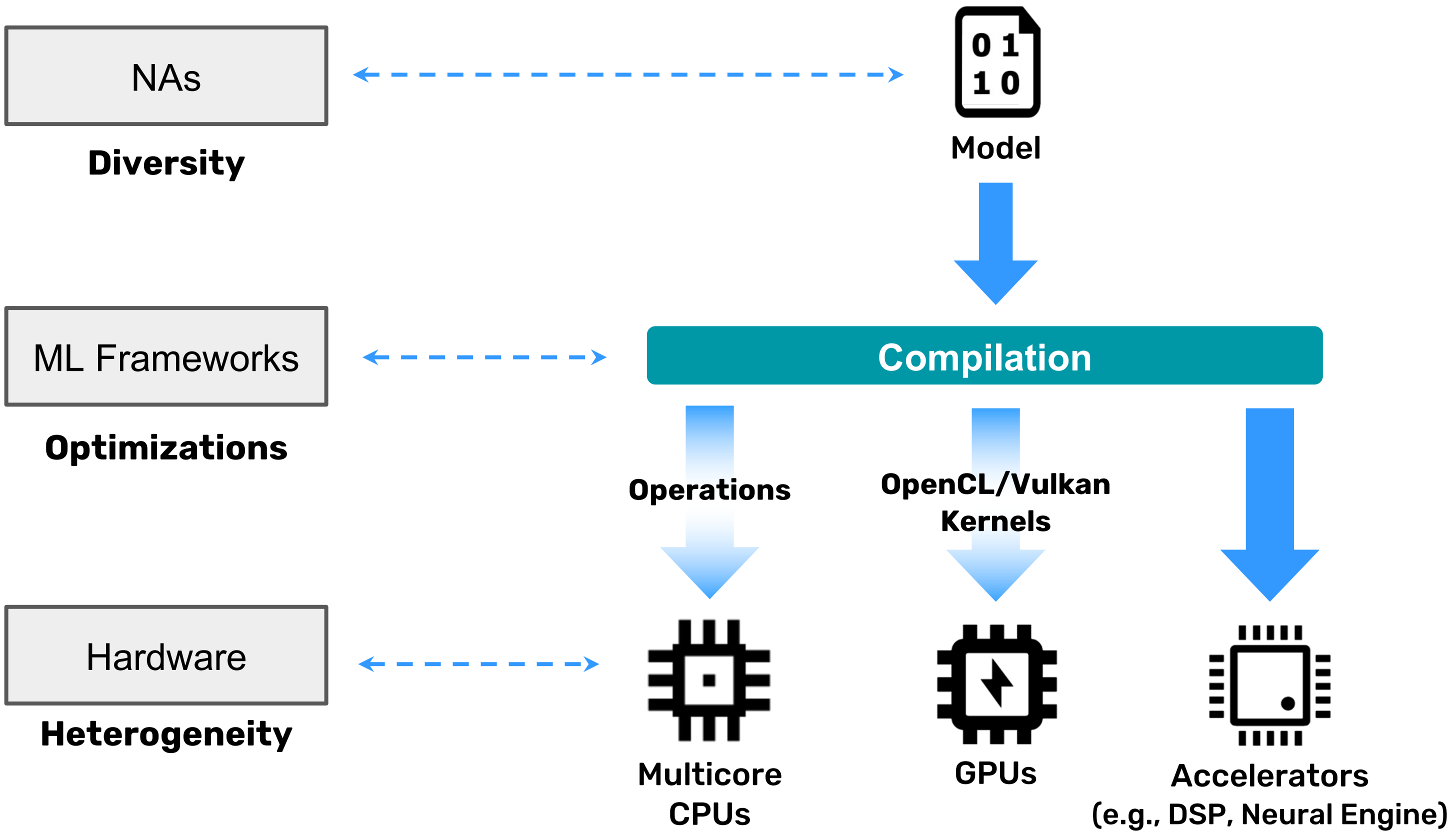
Background: Neural Architecture Search (NAS)



Measurement is unscalable as the search space of candidate NAs becomes huge [Cai et al., 2019]

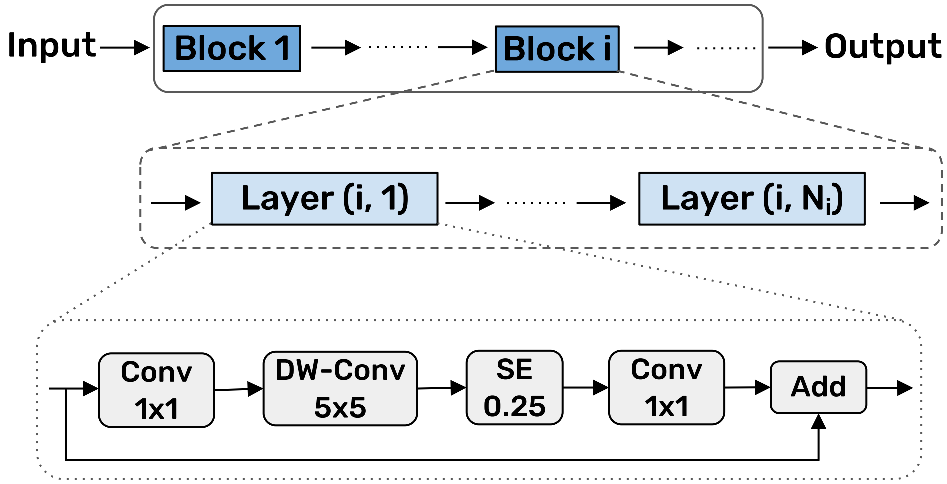
Can we predict the **latency** of candidate NAs without deploying them on actual device?

Challenges of Prediction Models



Challenge (1/3): NA Diversity

Candidate NAs

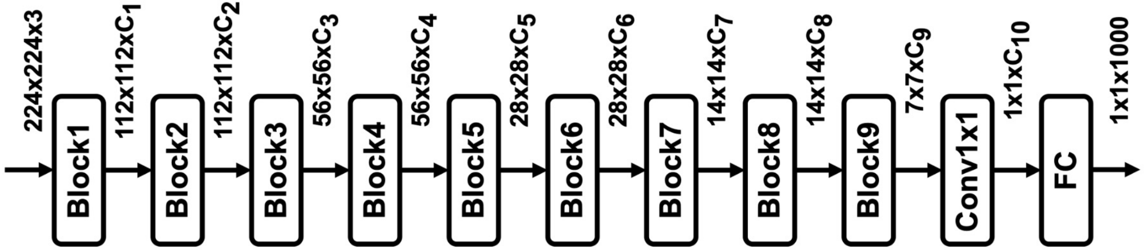


Example of Search Space

Conv:
Input shape,
Output shape,
Kernel size,
Stride,
Group size...

A latency *lookup-table* is **infeasible** when the search space is huge
e.g., the search space size in *Once-for-all* [Cai et al., 2019] is over 10^{19}

Solution: Comprehensive NAS Dataset



Search Space Design for Synthetic NAs

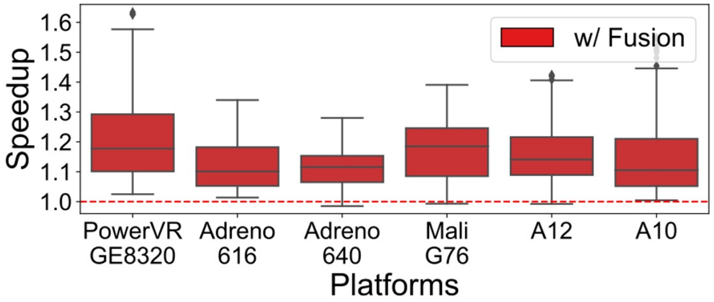
- ❖ **Building Blocks from SOTA Literature**
  - ➢ 1) Convolution
  - ➢ 2) Depthwise Separable Convolution [Howard et al., 2017]
  - ➢ 3) Linear Bottleneck [Sandler et al, 2018]
    - ■ w/ Squeeze-and-Excite [Howard et al., 2017]
  - ➢ 4) Average/Max Pooling
  - ➢ 5) Split and Concatenation

Challenge (2/3): ML Framework Optimizations

Hardware-dependent!

- **Kernel Fusion**

Conv | Add | ReLU → **GPU** → Conv2D_Add_ReLU

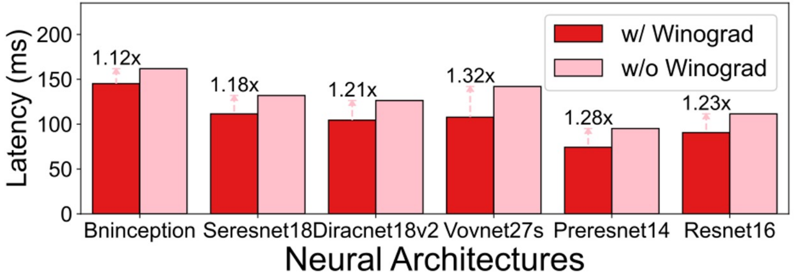


Effects of Kernel Fusion

- **Kernel Selection**

Conv → **PowerVR** → Winograd

Conv → **Adreno** → Conv2D

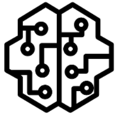


Effects of Applying Winograd Kernels (PowerVR GE8320)

Important to identify how many and which specific kernels are executed on mobile GPUs

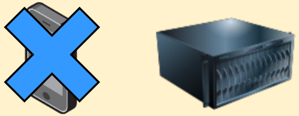
Solution: Characterization of Kernel Selection/Fusion

Model File (e.g., .tflite)



Computational Graph

No Need to Access Mobile Devices



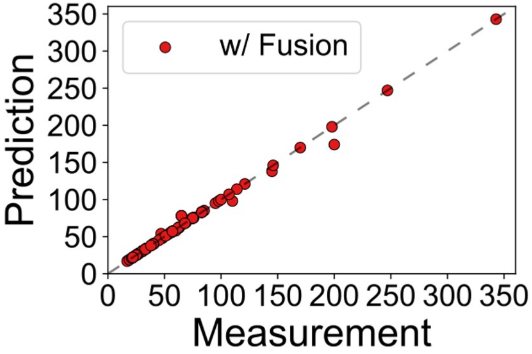
Mobile **Server**

Kernel Selection

Kernel Fusion

Principles extracted from open-source ML frameworks (e.g., TFLite)

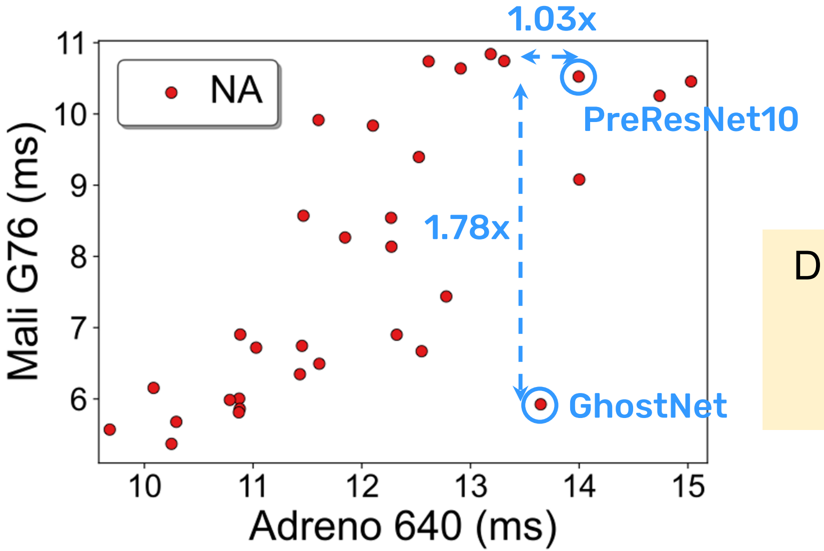
Kernels Predictor **Latency**



# of Kernels

Challenge (3/3): Hardware Heterogeneity (GPU)

❖ **Heterogeneous Mobile GPU Architectures**

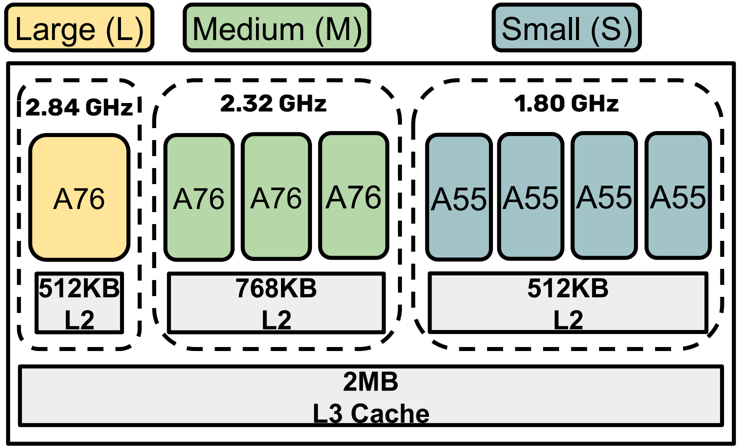


End-to-end Latency Comparisons

Distinct performance across hardware devices
E.g., 1.03x on Adreno 640, while 1.78x on Mali G76

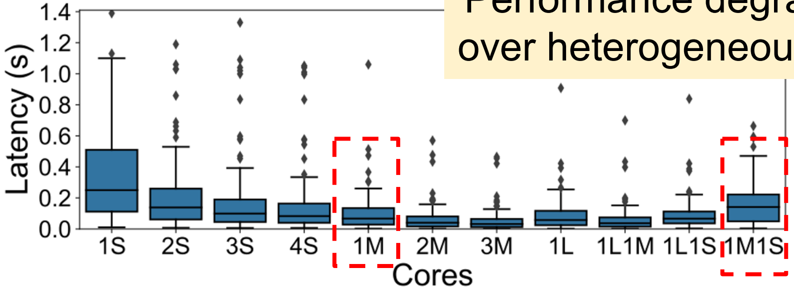
Challenge (3/3): Hardware Heterogeneity (CPU)

- ❖ **Heterogeneous Multi-core CPUs**

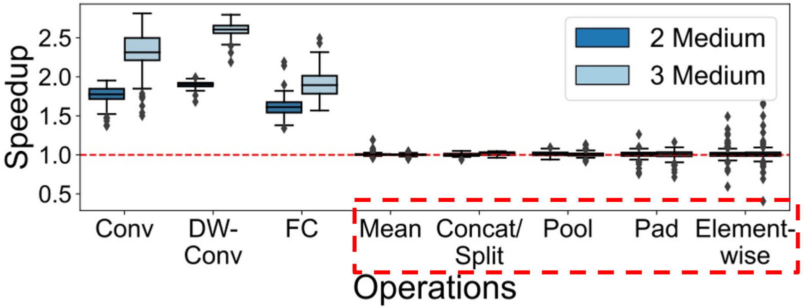


Snapdragon 855 Processor (Arm Big.LITTLE Architecture)

Performance degradation over heterogeneous cores



Effects on End-to-end Latency



Effects on Operation-wise Latency (Speedup over 1 M Core)

Different speedup across operations

Solution: Component-based Predictors

❖ **Build separate ML predictor for each (Platform, Configuration, Operation Type)**

Operation Features → **Predictor (Platform, Configuration, Operation Type)** → **Operation Latency**

(E.g., Snapdragon 855, two medium cores, quantization, Depthwise-Conv2D)

Operations	Features
Conv2D, Depthwise-Conv2D	Input height (width), input channel, output height (width), stride, kernel height (width), filters, group size, input size, output size, kernel size, FLOPs
Fully-Connected	Input channel, filters, parameter size, FLOPs
Max/Average Pooling	Input height (width), input channel, output height (width), stride, kernel height (width), input size, output size, FLOPs
...	...

Solution (cont.): Explore different ML approaches

- Linear
  - Lasso w/ non-negative weights [Tibshirani, 1996]
- Non-linear
  - Random Forest (RF) [Ho, 1995]
  - Gradient-Boosted Decision Tree (GBDT) [Friedman, 2001]
  - Multi-Layer Perceptron (MLP) [Haykin, 1994]

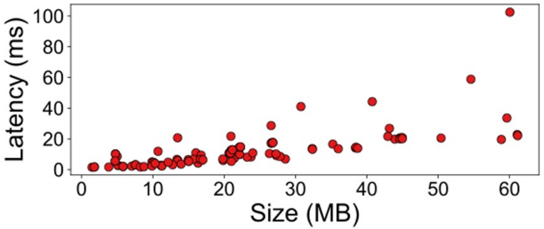
Summary of Key Ideas

To accurately predict the latency of NAs on mobile devices

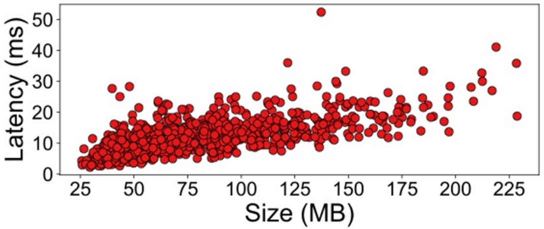
- ❖ **Comprehensive NAS Dataset** → NAs — **Diversity**
- ❖ **Characterization of Kernel Selection/Fusion** → ML Frameworks — **Optimizations**
- ❖ **Separate Predictors for Each (Kernel, Platform, Configuration)** → Hardware — **Heterogeneity**

Experimental setup

- ML workloads
  - 102 Real-world NAs
  - 1000 Synthetic NAs
- 6 hardware platforms
  - Android + iOS
- ML framework: TFLite v2.10



102 Real-world NAs



1000 Synthetic NAs

Device	Platform	CPU	GPU
Google Pixel 4	Snapdragon 855	1x Large (2.84 GHz) 3x Medium (2.32 GHz) 4x Small (1.80GHz)	Adreno 640
Xiaomi Mi 8 SE	Snapdragon 710	2x Large (2.20 GHz) 6x Small (1.70 GHz)	Adreno 616
Samsung Galaxy S10	Exynos 9820	2x Large (2.73 GHz) 2x Medium (2.31 GHz) 4x Small (1.95 GHz)	Mali G76
Samsung Galaxy A03s	Helio P35	4x Large (2.30 GHz) 4x Small (1.80 GHz)	PowerVR GE8320
Apple iPhone XS	A12 Bionic	2x Large (2.49 GHz) 4x Small (1.52 GHz)	Apple-designed G11P
Apple iPhone 7	A10 Fusion	2x Large (2.34 GHz) 2x Small (1.05 GHz)	PowerVR GT7600 Plus (Custom)

Results (1/6): Default Setting

- Evaluate candidate NAs during search across six platforms
  - Training: 900 synthetic NAs
  - Test: 100 synthetic NAs
  - Metric: Mean Average Percentage Error (MAPE)

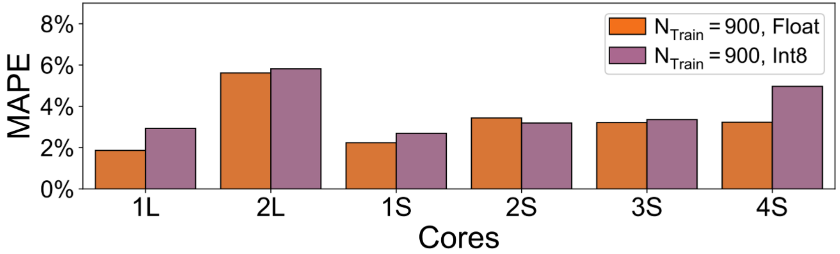
Method	MAPE (CPU)	MAPE (GPU)
Lasso	11.2%	9.4%
RF	2.8%	5.5%
GBDT	2.4%	5.2%
MLP	2.8%	5.1%

Non-linear ML methods achieve accurate end-to-end latency predictions on both CPUs and GPUs

Results (2/6): Hardware Heterogeneity

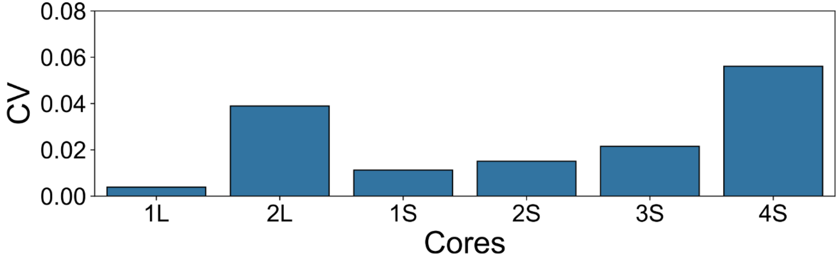
- Device: Apple A12 Bionic
- ML method: GBDT

Training: 900 synthetic NAs
Test: 100 synthetic NAs



Predictions of GBDT on Multicore CPUs

Maximum MAPE of 10.5% across all configurations on six devices



Coefficient of Variation (CV) for Latency Measurements

Measurement variance can affect prediction accuracy

Results (3/6): NA Diversity

- Different distributions between training and test datasets
  - Training: 900 synthetic NAs
  - Test: 102 real-world NAs

Method	MAPE (CPU)	MAPE (GPU)
Lasso	5.4%	7.9%
RF	6.1%	6.8%
GBDT	6.0%	7.0%
MLP	11.6%	8.7%

Accurate predictions under dataset shift between training and test data

Results (4/6): ML Framework Optimizations

❖ **Device: PowerVR GE8320 GPU**

Kernel Fusion

Method	MAPE (w/ Fusion)	MAPE (w/o Fusion)
Lasso	**6.1%**	13.6%
RF	**6.1%**	16.5%
GBDT	**6.2%**	17.1%
MLP	**8.2%**	21.0%

Kernel Selection

Method	MAPE* (w/ Select)	MAPE* (w/o Select)
Lasso	**2.1%**	14.0%
RF	**3.2%**	8.9%
GBDT	**2.0%**	9.1%
MLP	**5.2%**	7.9%

* for real-world NAs that support Winograd kernels

Substantial error reduction across all ML approaches
by accurately characterizing kernel fusion and selection

Results (5/6): Limited Training Data

- ML method: GBDT
  - Training: 30/100/900 synthetic NAs
  - Test: 100 synthetic NAs

Training Size	MAPE (CPU)	MAPE (GPU)
30	**8.1%**	**8.6%**
100	5.1%	6.9%
900	2.4%	5.2%

Sufficiently accurate predictions with only 30 NAs

The cost of profiling 30 NAs for training is **negligible** compared to measuring thousands of candidate NAs

Results (6/6): Comparison to State-of-the-art

Training Dataset	MAPE (CPU)	MAPE (GPU)
NATSBench	56.2%	57.7%
Ours (Synthetic)	3.3%	5.1%

Comparison with Dataset: NATSBench [Dong et al., 2021]
(**Training**: 1000 NAs; **Test**: 44 real-world NAs w/o DW-Conv; **Predictor**: GBDT)

	Test: Synthetic NAs		Test: Real-world NAs	
Predictors	MAPE (CPU)	MAPE (GPU)	MAPE (CPU)	MAPE (GPU)
NN-Meter	14.1%	24.9%	8.5%	41.5%
Ours (GBDT)	2.4%	6.3%	6.3%	7.6%

Comparison with Predictor: NN-Meter [Zhang et al., 2021]
(**Training**: 1000 synthetic NAs)

Summary of Contributions

- Identified aspects of NAs, HW and ML frameworks that substantially affect latency

- Developed a synthetic dataset that provides broader coverage than SOTA
  - Latency measurements under 90 scenarios across 6 mainstream mobile platforms

- Developed a framework for accurately predicting end-to-end latency without deploying and compiling on actual devices with negligible profiling time