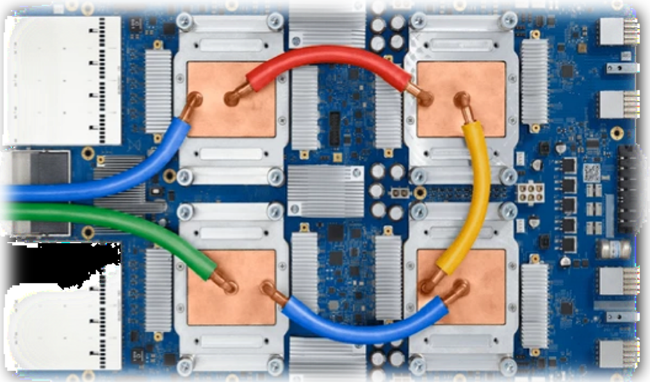
A Comprehensive Benchmark of Deep Learning Libraries on Mobile Devices

Qiyang Zhang, Xiang Li, Xiangying Che, Xiao Ma, Ao Zhou,
Mengwei Xu, Shangguang Wang, Yun Ma, Xuanzhe Liu

DL Inference on Smartphones

- Increasingly popular DL

Object Detection



NN accelerators

- Emerging DL libraries

TensorFlow Lite

MNN Mobile Neural Network

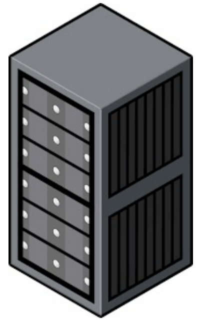
MACE Mobile AI Compute Engine

PyTorch Mobile

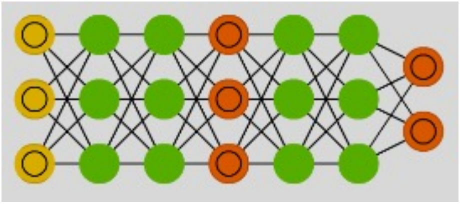
DL libraries are not fully understood

MLPerf DeepBench Fathom …

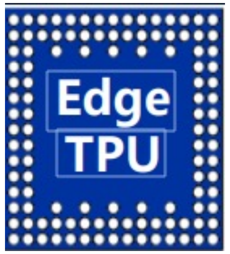
- Existing Benchmarks mainly focus on:



server



NN design



AI chip

Benchmark	Scenario	Benchmark objective	Supported DL libs
MLPerf [48]	T/I@S/E	Hardware	/
DeepBench [16]	T/I@S/E	Hardware	/
DAWNBench [15]	T/I@S	Hardware, algorithm, and DL libs	/
AI Matrix [7]	I@S	Hardware and DL libs	4
AI-Benchmark [34]	I@E	Hardware	1
Fathom [19]	T/I@S	Algorithm	1
gaugeNN [22]	I@E	DL apps and models	1
AIIA [13]	I@E	Hardware	3
This work	I@E	DL libs	6

The comparison of existing benchmarks and ours.

- A comprehensive benchmark for on-device DL inference
  The benchmark triumphs at the aspect of rich support for more various DL libs

Measurement Settings

- Rich support
  - the most popular DL libs: **TFLite, NCNN, MNN, PyTorch Mobile, Mace, SNPE**.
  - **15** models in total, for various tasks and different model precision.

Measurement Settings

Supported DL libs and models in this work.

Models	Tasks	TFLite	ncnn	mnn	MACE	PyTorchMobile	SNPE
mobilenetV1[29]	image classification	$C_{32,8}$-$G_{32,8}$-D_8	$C_{32,8}$-$G_{32,8}$	$C_{32,8}$-$G_{32,8}$	$C_{32,8}$-G_{32}	$C_{32,8}$	$C_{32,8}$-$G_{32,8}$-D_8
mobilenetV2[55]	image classification	$C_{32,8}$-$G_{32,8}$-D_8	[illegible]	[illegible]	[illegible]	[illegible]	$C_{32,8}$-$G_{32,8}$-D_8
inceptionV3 [60]	image classification	$C_{32,8}$-$G_{32,8}$-D_8	[illegible]	[illegible]	[illegible]	[illegible]	$C_{32,8}$-$G_{32,8}$-D_8
inceptionV4 [59]	image classification	$C_{32,8}$-$G_{32,8}$-D_8	[illegible]	[illegible]	[illegible]	[illegible]	$C_{32,8}$-$G_{32,8}$-D_8
vgg16 [57]	image classification	$C_{32,8}$-$G_{32,8}$-D_8	[illegible]	[illegible]	[illegible]	[illegible]	$C_{32,8}$-$G_{32,8}$-D_8
squeezenet [32]	image classification	$C_{32,8}$-$G_{32,8}$-D_8	$C_{32,8}$-$G_{32,8}$	C_{32}-G_{32}	C_{32}-G_{32}	$C_{32,8}$	$C_{32,8}$-$G_{32,8}$-D_8
nasnet_mobile [84]	image classification	C_{32}-G_{32}	-	C_{32}-G_{32}	C_{32}-G_{32}	C_{32}	-
densenet [31]	image classification	C_{32}-G_{32}	-	C_{32}-G_{32}	-	C_{32}	C_{32}-G_{32}
mnasnet [61]	image classification	C_{32}-G_{32}	C_{32}-G_{32}	C_{32}-G_{32}	C_{32}-G_{32}	C_{32}	C_{32}-G_{32}
resnetv2_50 [58]	image classification	C_{32}-G_{32}	C_{32}-G_{32}	C_{32}-G_{32}	C_{32}-G_{32}	C_{32}	C_{32}-G_{32}
deeplabv3 [81]	semantic segmentation	C_{32}-G_{32}	-	C_{32}-G_{32}	C_{32}-G_{32}	-	-
ssd_mobilenetV1 [46]	object detection	C_{32}-G_{32}	C_{32}-G_{32}	C_{32}-G_{32}	C_{32}-G_{32}	C_{32}	-
yolo-fastest [80]	object detection	C_{32}-G_{32}	C_{32}-G_{32}	C_{32}-G_{32}	-	-	-
yolo3 [53]	object detection	C_{32}-G_{32}	C_{32}-G_{32}	C_{32}-G_{32}	-	-	-
albert_tiny [70]	text classification	C_{32}-G_{32}	-	C_{32}-G_{32}	-	-	-

C/G/D: CPU/GPU/DSP
32/8: FP32/INT8

Measurement Settings

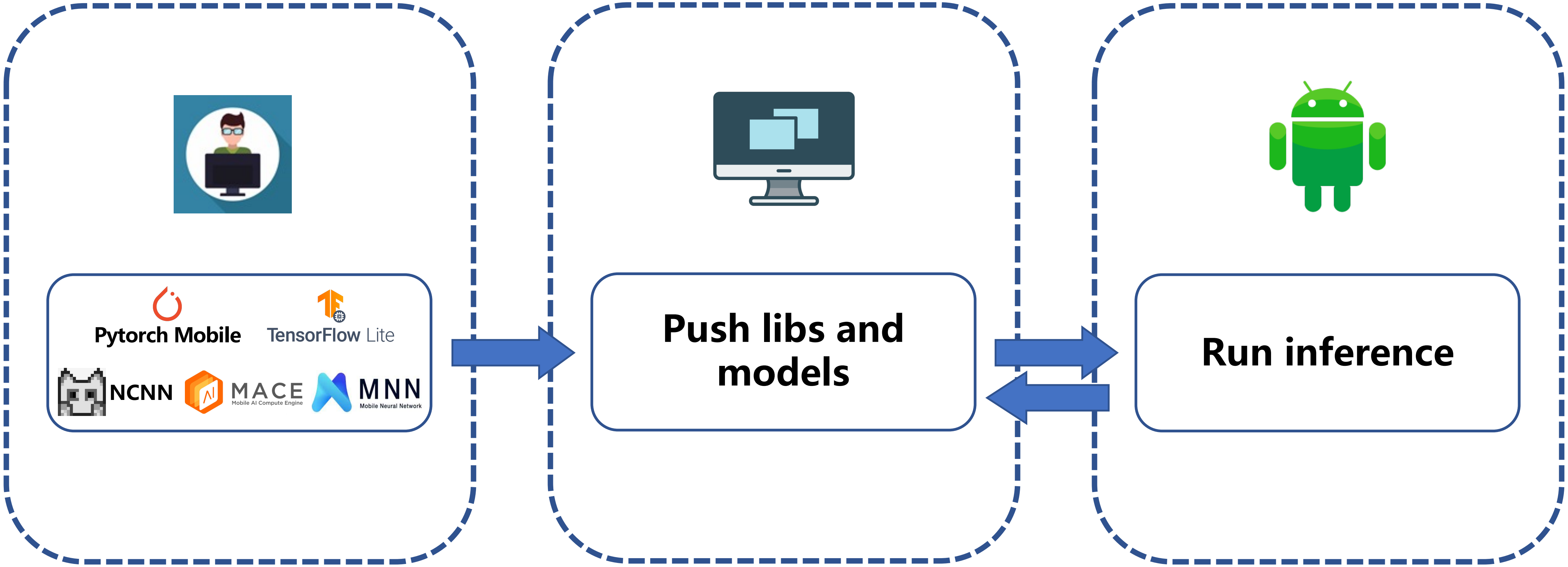
- Rich support
  - The most popular DL libs: **TFLite, NCNN, MNN, PyTorchMobile, Mace, SNPE**.
  - **15** models in total, for various tasks and different model precision.
- Devices
  - **10** different device models with various SoCs and GPUs
- Detailed metrics
  - **The inference time** and **operator-level information.**

Research goal

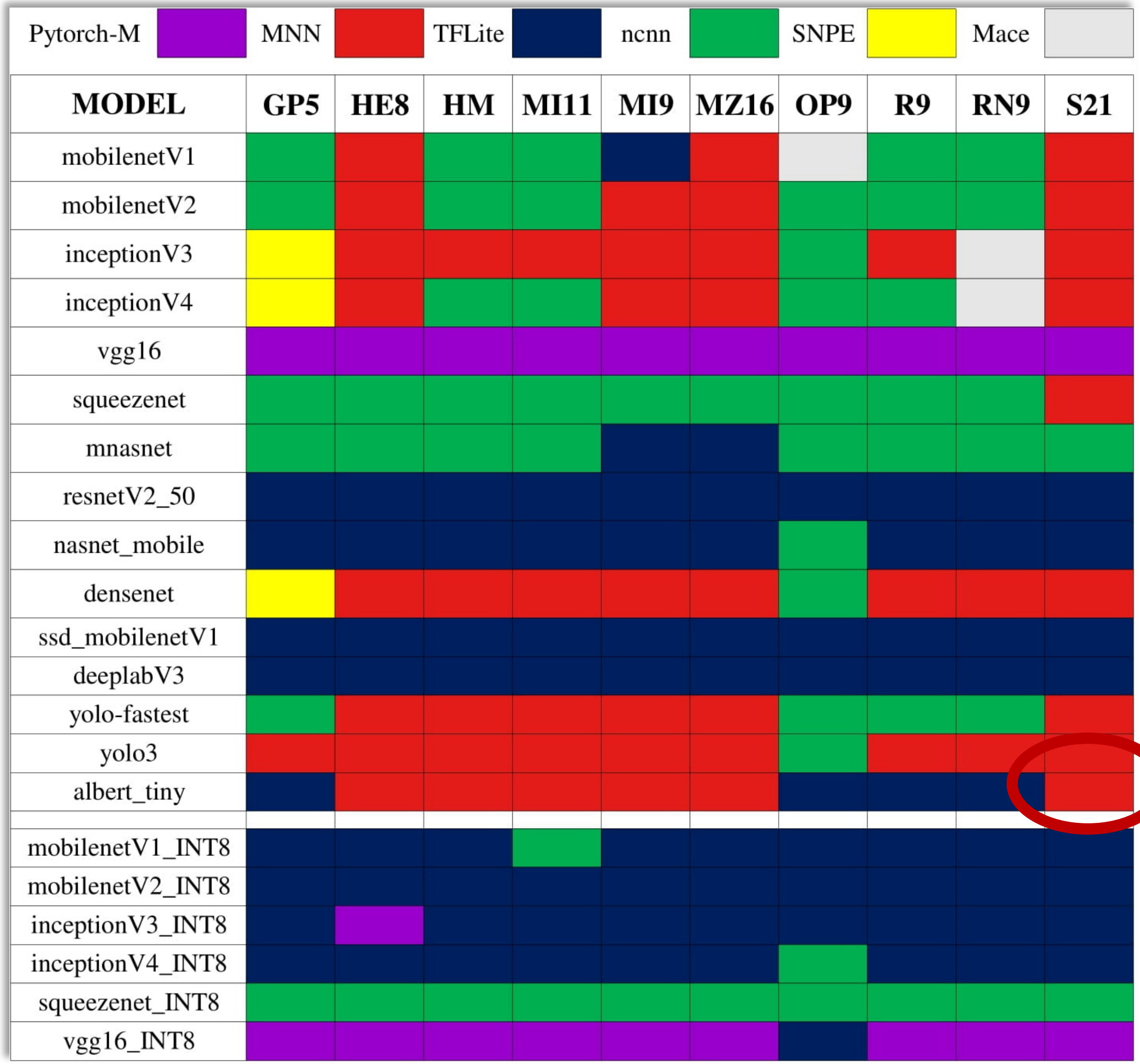
(**The first**) **measurement** to understand how DL libs affect inference.

- **Performance Fragmentation**
- Impacts of **Quantization and Hardware**
- **Operator-level Integration**
- **Cold-start** Inference
- **Longitudinal** Inference

Analysis Workflow



Performance Fragmentation

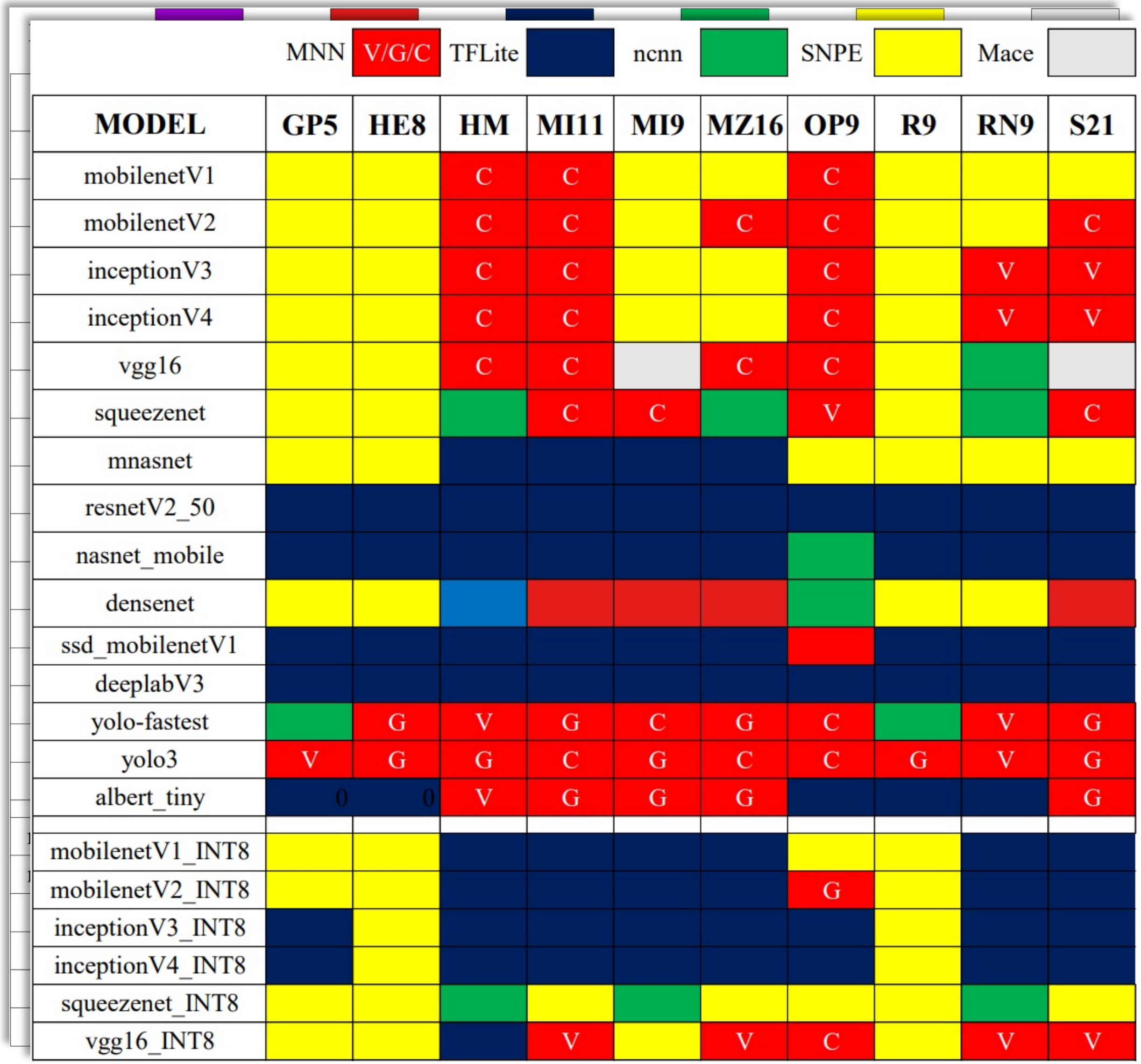


Best-performing lib on CPU

DL lib with the smallest inference time in 6 DL libs

The DL lib with fastest speed to run "albert_tiny" on "Samsung S21" is "**MNN**".

Performance Fragmentation



MNN V/G/C | TFLite | ncnn | SNPE | Mace

MODEL	GP5	HE8	HM	MI11	MI9	MZ16	OP9	R9	RN9	S21
mobilenetV1			C	C			C			
mobilenetV2			C	C		C	C			C
inceptionV3			C	C			C		V	V
inceptionV4			C	C			C		V	V
vgg16			C	C		C	C			
squeezenet				C	C		V			C
mnasnet										
resnetV2_50										
nasnet_mobile										
densenet										
ssd_mobilenetV1										
deeplabV3										
yolo-fastest		G	V	G	C	G	C		V	G
yolo3	V	G	G	C	G	C	C	G	V	G
albert_tiny	0	0	V	G	G	G				G
mobilenetV1_INT8										
mobilenetV2_INT8							G			
inceptionV3_INT8										
inceptionV4_INT8										
squeezenet_INT8										
vgg16_INT8				V		V	C		V	V

Best-performing lib on CPU/GPU

There is no one-size-fit-all DL lib for optimal performance across models and hardware.

Performance Fragmentation

Performance gap of DL libs can be large.

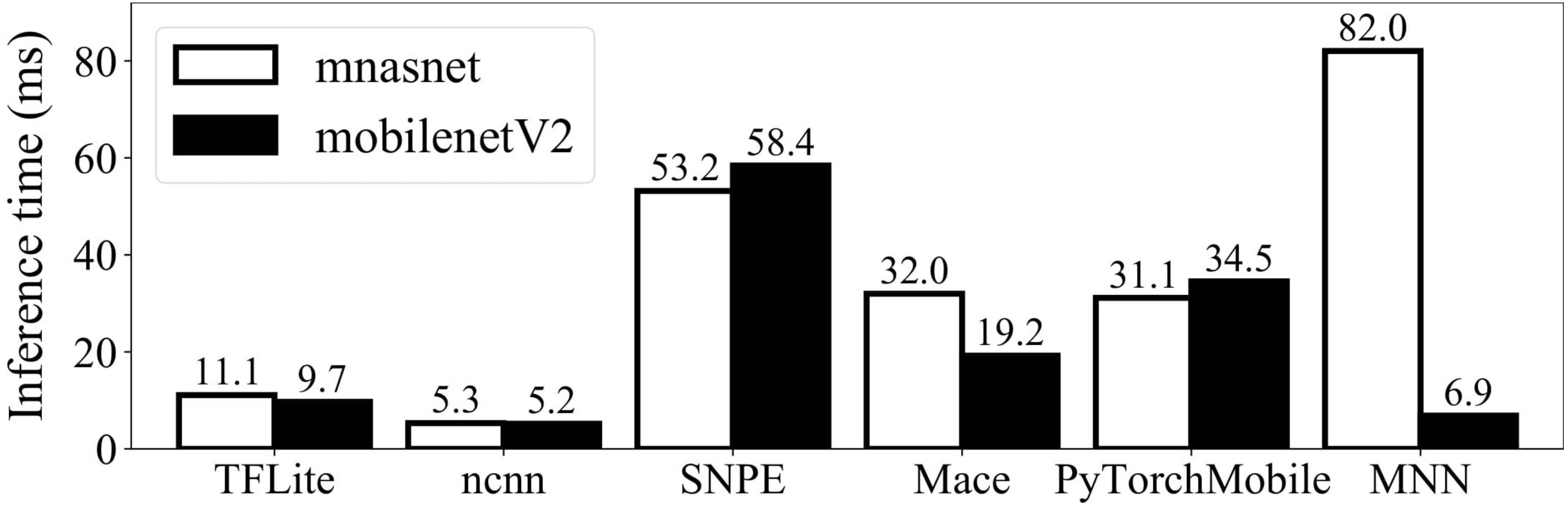
Models	Best vs. Worst		Best vs. 2nd Best	
	CPU (×)	GPU (×)	CPU (×)	GPU (×)
mobilenetV1	4.0~15.4 (8.7)	1.7~14. [illegible]	[illegible] .5)	1.0~4.0 (1.9)
mobilenetV2	5.6~18.8 (11.2)	2.9~15. [illegible]	[illegible] .5)	1.0~2.9 (1.6)
inceptionV3	2.6~5.6 (3.8)	3.0~13.4 ([illegible])	1.1~2.4 (1.7)	1.0~4.0 (2.1)
inceptionV4	2.0~5.4 (3.2)	2.4~11.0 (5.8)	1.1~2.0 (1.5)	1.0~3.6 (2.0)
vgg16	7.1~54.3 (16.2)	4.4~7.0 (5.5)	1.3~4.2 (2.4)	1.1~2.2 (1.5)
squeezenet	4.6~19.9 (9.1)	1.9~12.6 (5.9)	1.0~5.9 (2.5)	1.1~2.5 (1.6)
average	8.7	6.0	1.9	1.8

the longer one divided by the shorter one

The performance gaps of different DL libs

Performance Fragmentation

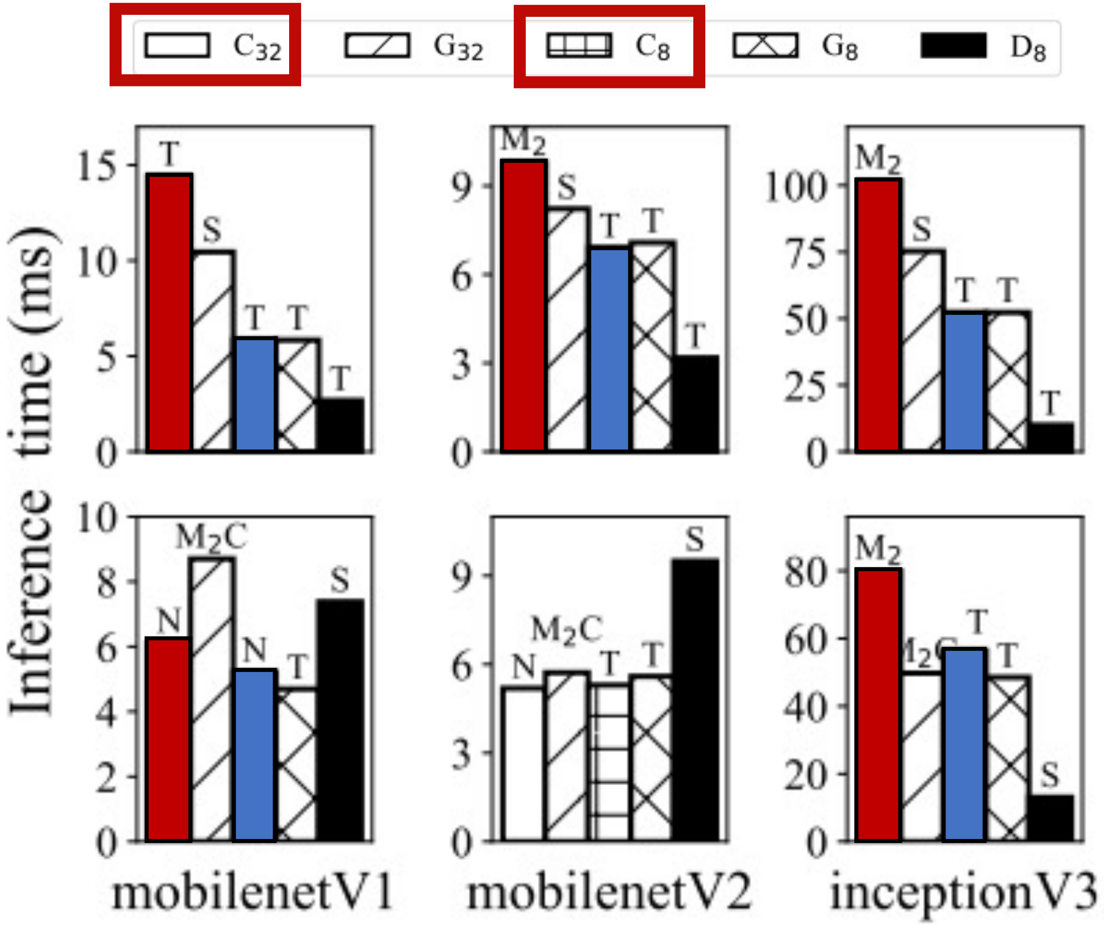
With software heterogeneity, the model structure is not the sole factor that determines relative performance.



Implication: To pursue the optimal performance, the developers need to incorporate different DL libs and dynamically load one based on the current model and hardware platform.

Performance Fragmentation

Benefit brought by INT8 quantization is under expectation.

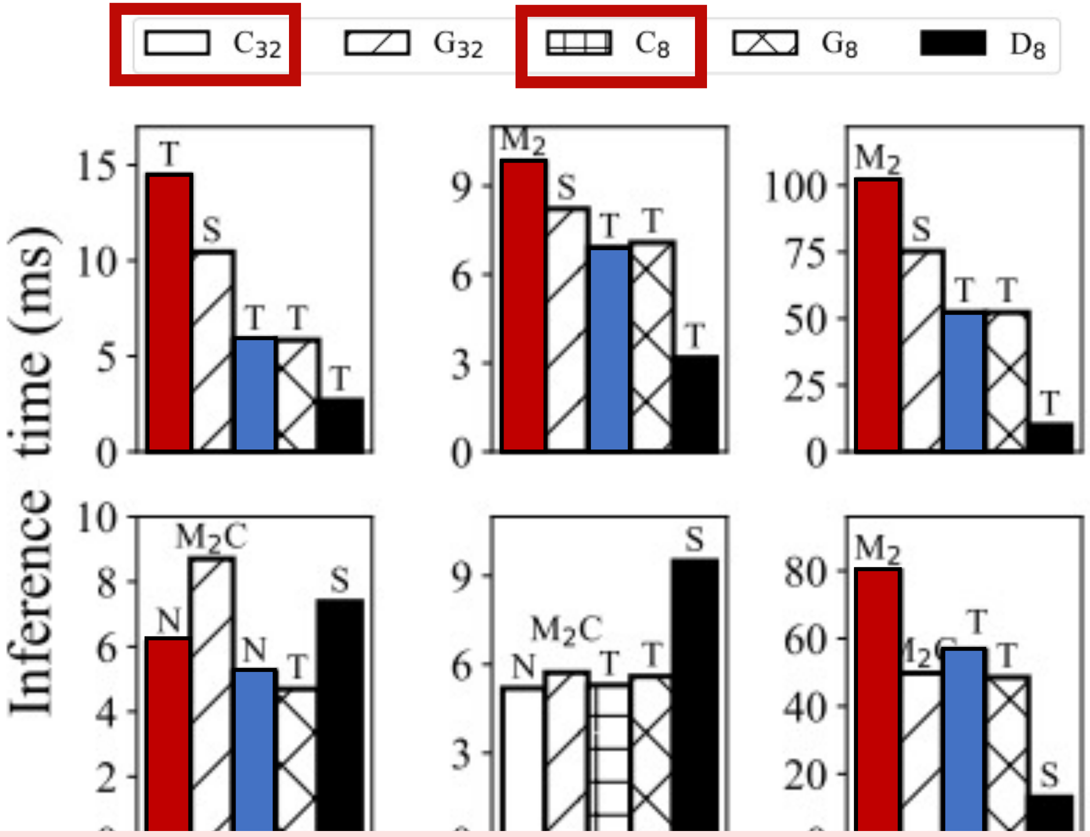


0.8×–3.0× faster than FLOAT

Best inference speed across all DL libs

Performance Fragmentation

Benefit brought by INT8 quantization is under expectation.

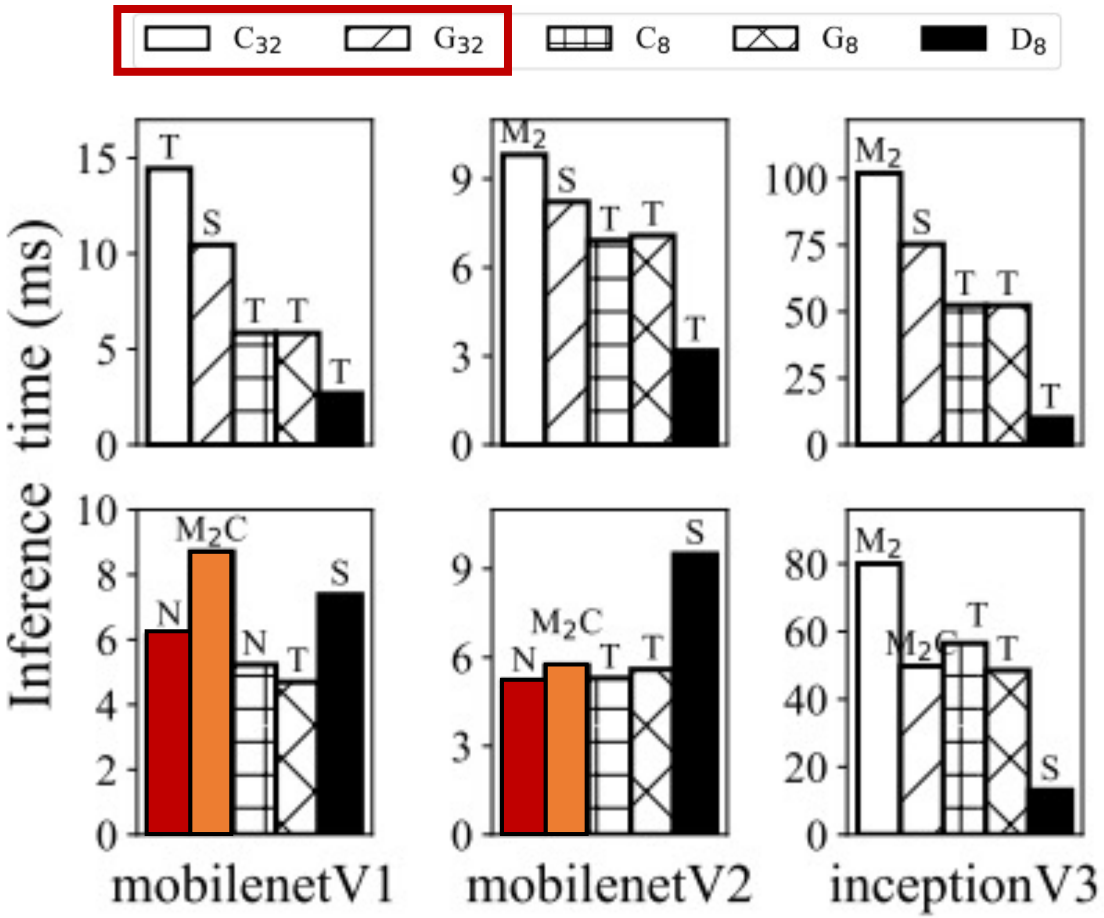


0.8×–3.0× faster than FLOAT

Implication: There exists great potential at software level to accelerate the inference of quantized models.

Impacts of Hardware

GPU can not always accelerate DL inference.

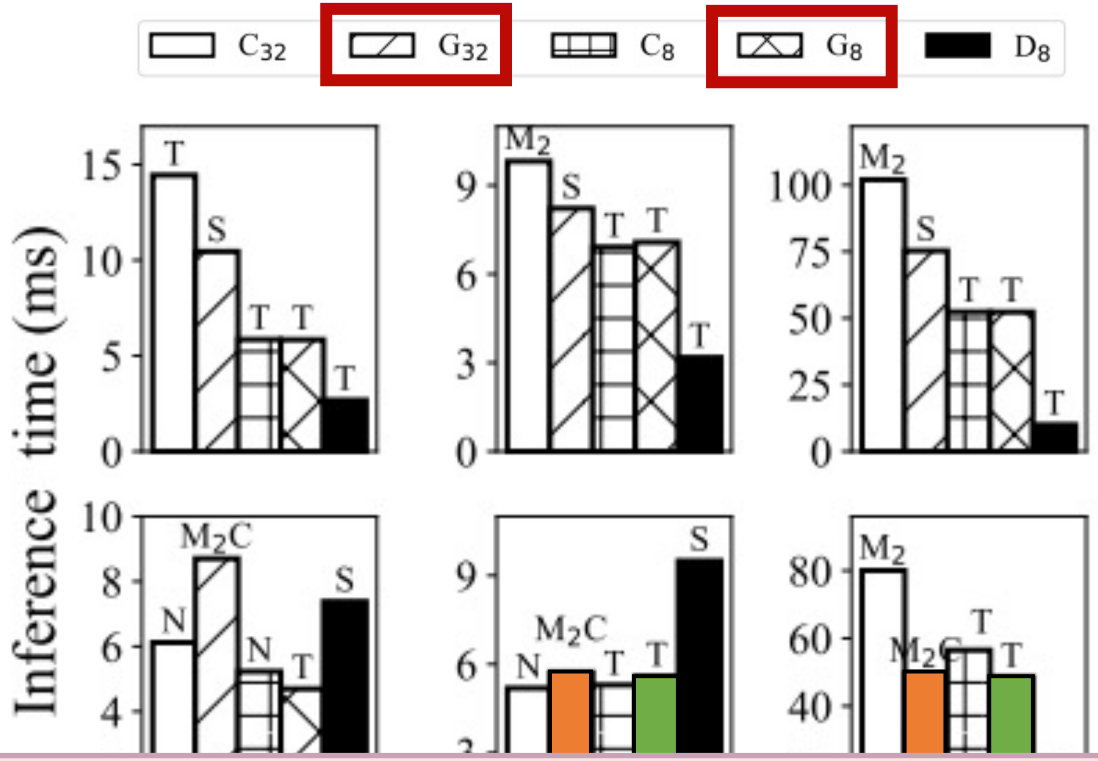


speedup 1.4×–1.9× compared to CPU

Best inference speed across all DL libs

Impacts of Hardware

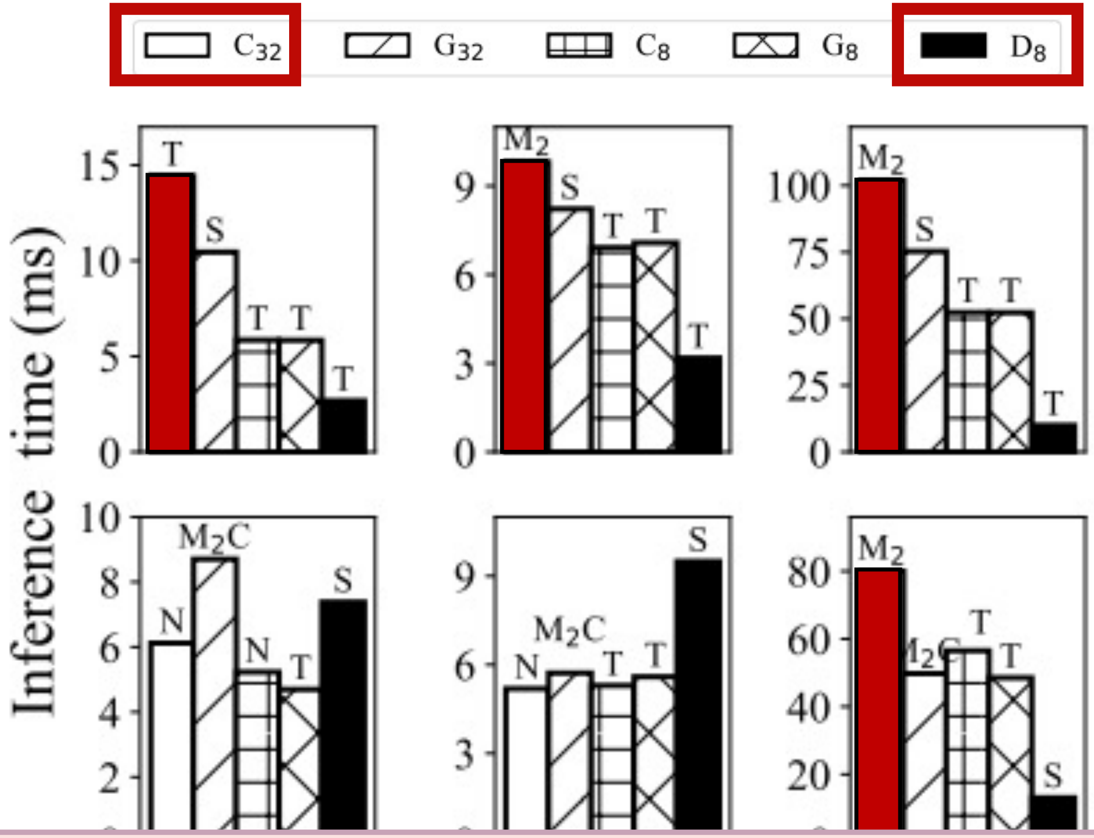
On INT8-based models, GPU can hardly bring any benefit.



Implication: Observations motivate developers to focus on GPU optimization. It also motivates researchers to design models suitable for GPU.

Impacts of Hardware

DSP can significantly accelerate INT8 model in most cases.



reduce inference time of INT8 model by 2.0×–12.9×

Implication: The current DL libs can not fully exploit the capacity of each hardware.

Operator-level Integration

How about integrating the best-performing operator from DL libs?

Oracle lib that combines the fastest operator from those DL libs

Models	Mace	tflite	SNPE	ncnn	Oracle time
mobilenetV1				14.4	13.5 ($\downarrow$6.1%)
mobilenetV2				14.4	10.6 ($\downarrow$26.3%)
inceptionV3				123	86.3 ($\downarrow$29.9%)
inceptionV4				74.9	180.3 ($\downarrow$7.6%)
vgg16	180.3	73.1	341.7	409.0	73.1 ($\downarrow$0%)

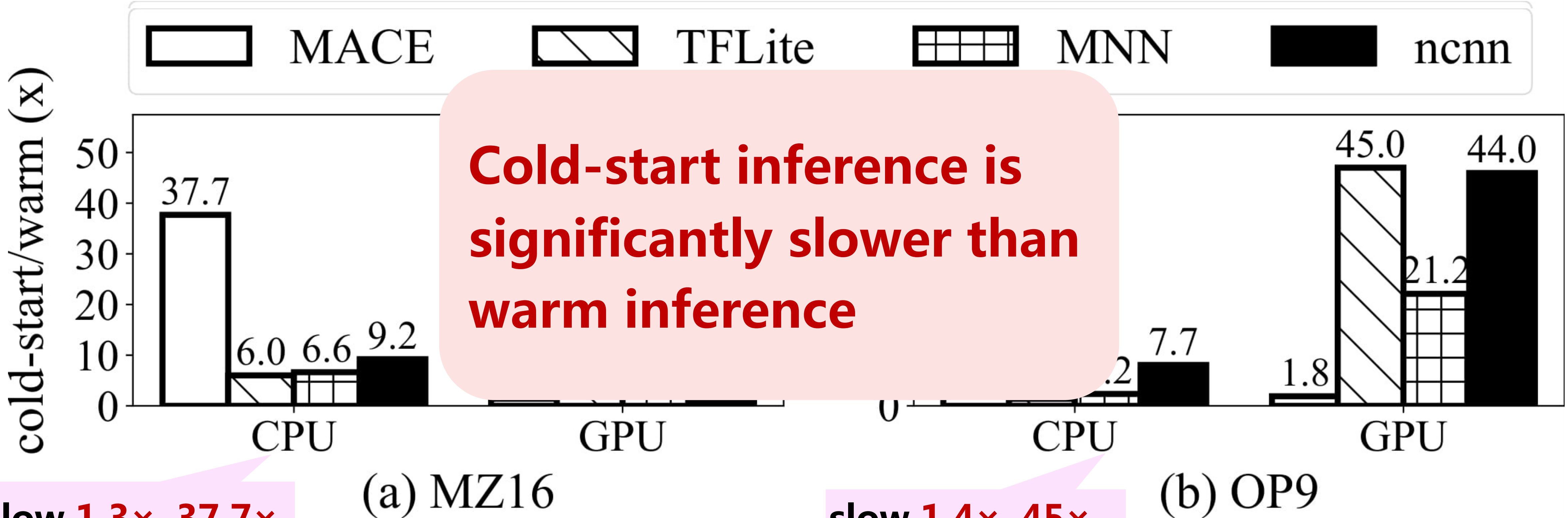
Oracle time brings inference time reduction

The benefits that integrate the wisdom of DL libs （ms）

Implication: Those diversities need to be unified before the operator implementation can be combined.

Cold-start Inference

The first inference beginning from model loading

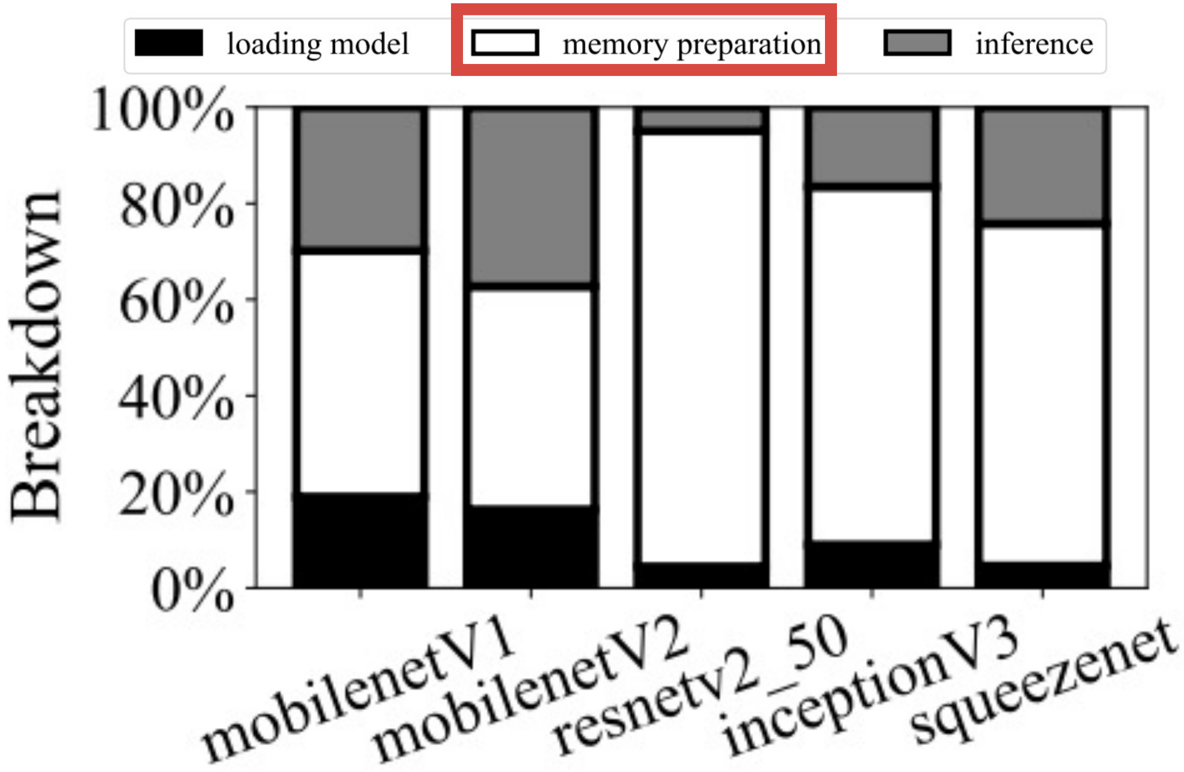


slow **1.3×–37.7×** on **CPU** than warm inference

The ratio of cold-start inference

slow **1.4×–45×** on **GPU** than warm inference

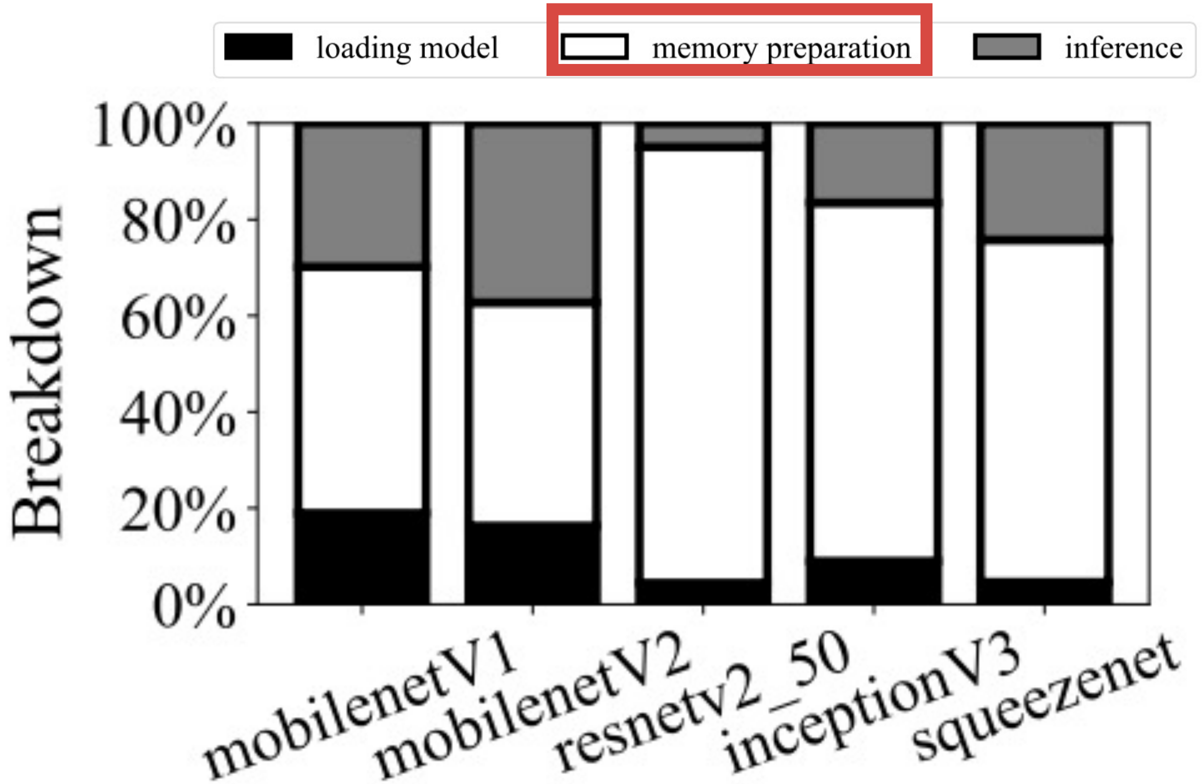
Breakdown of Cold-start Inference



The breakdown of cold-start inference

Memory preparation contributes to the largest overhead in cold-start inference.

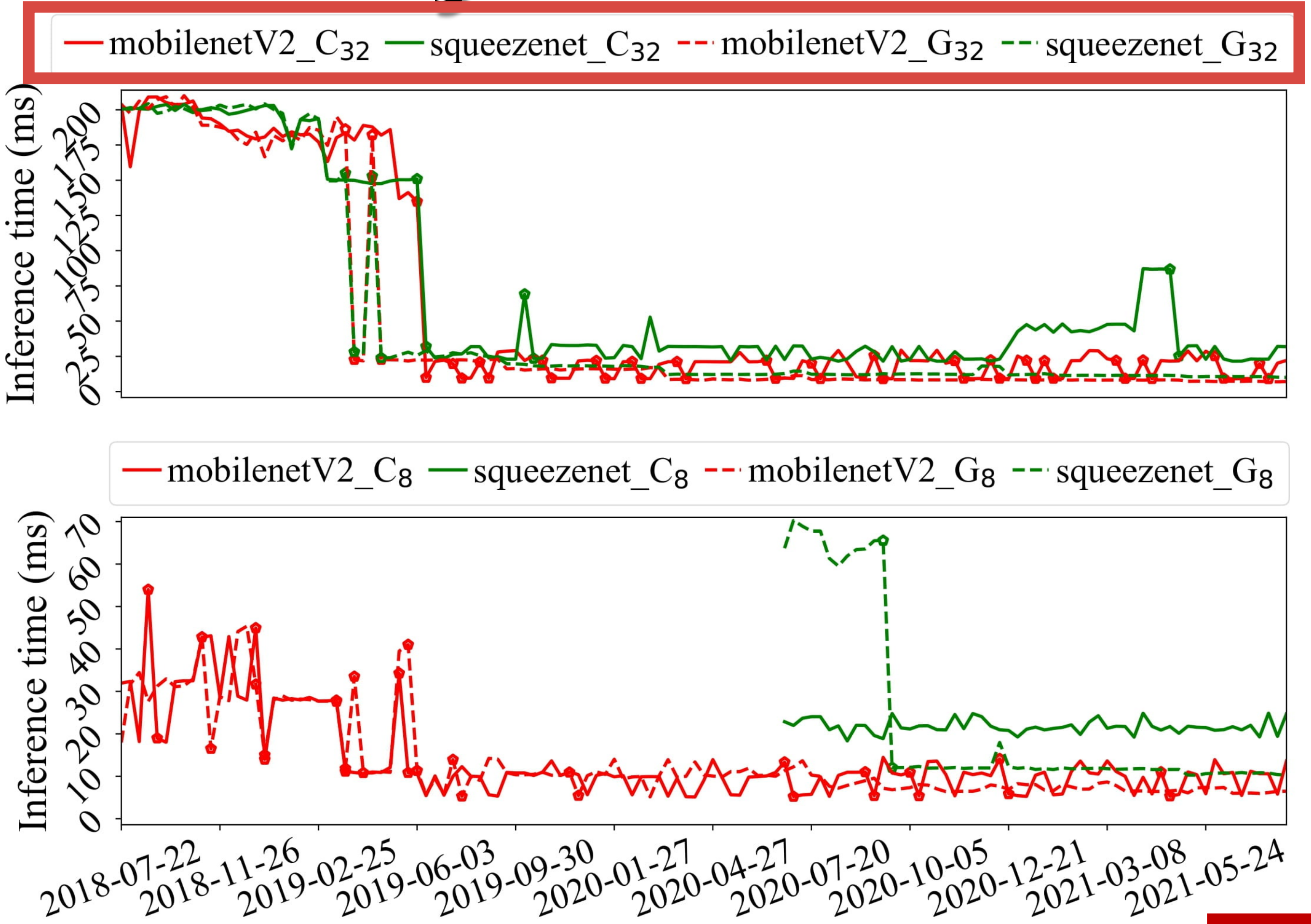
Breakdown of Cold-start Inference



The breakdown of cold-start inference

Implication: *Potential solutions include speeding up memory preparation using multiple threads and generating pipeline to run model loading memory preparation and inference simultaneously.*

Longitudinal Analysis



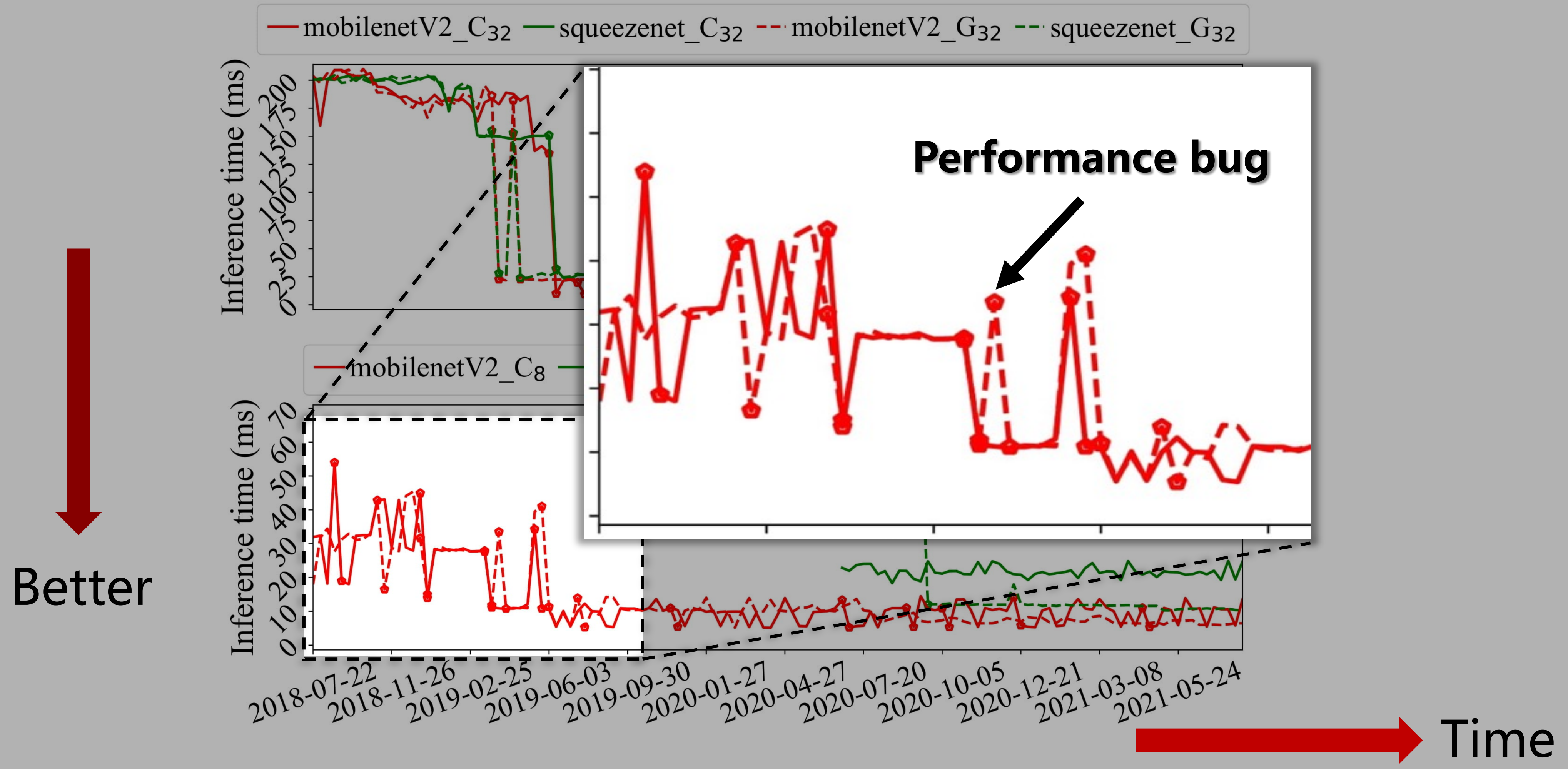
Better

Time

The inference performance evolvement across time of TFLite

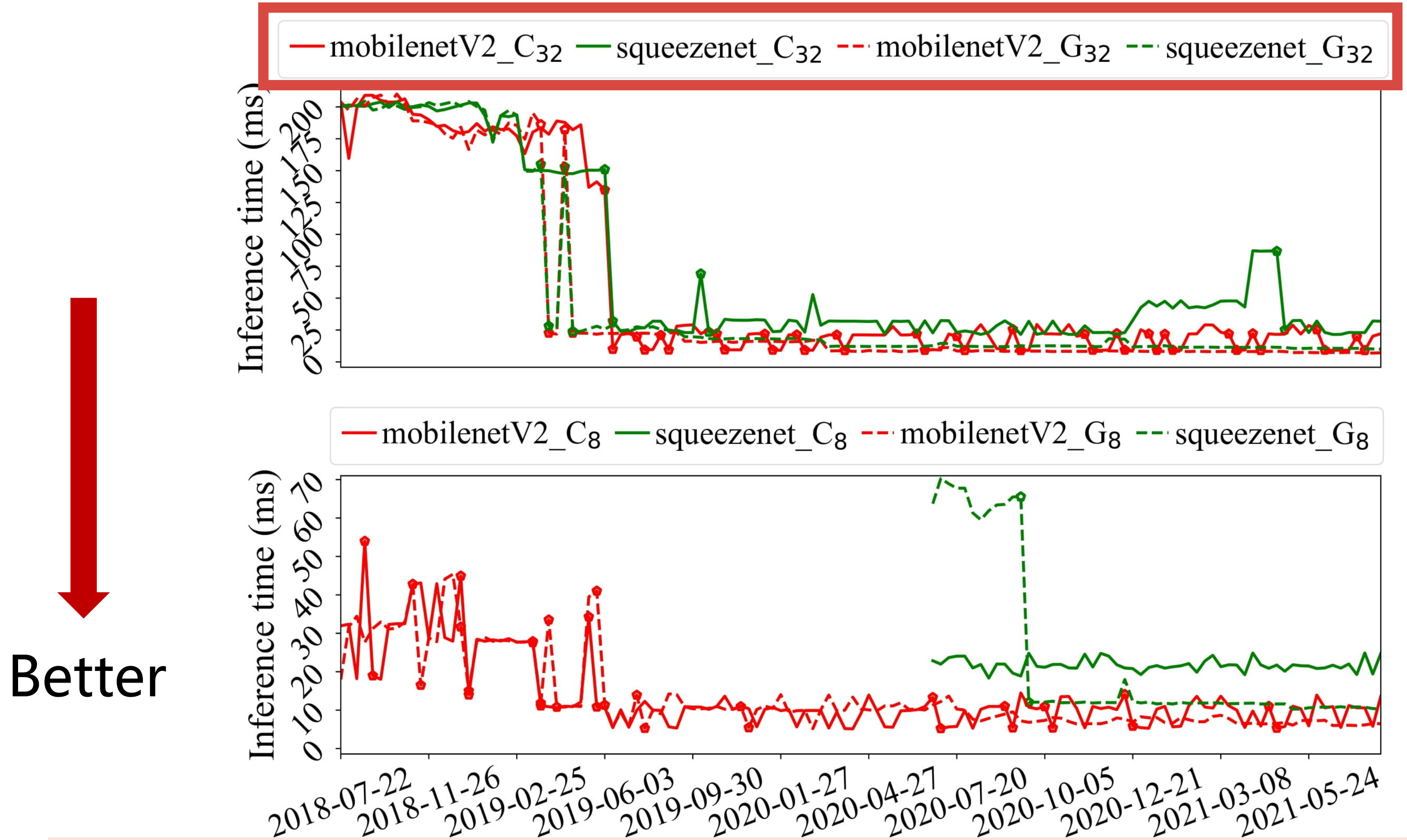
The performance of DL libs are continuously improving in early years, but becomes relatively stable since 2020.

Longitudinal Analysis



The inference performance evolvement across time of TFLite

Longitudinal Analysis



Implication: The current open-source ecosystems is possibly due to a comprehensive benchmarking tool available for developers to test commits.

Summary

- A comprehensive benchmark to quantitatively understand inference performance of DL libs.
- Lead to insightful implications for complete landscape of DL libs ecosystem.

Please check benchmark at

- **https://github.com/UbiquitousLearning/MobileDLFrameworksBenchmark**

Thanks for your attention!