Characterizing the Performance of Accelerated Jetson Edge Devices for Training Deep Learning Models

Prashanthi S.K, Sai Anuroop Kesanapalli and Yogesh Simmhan
Indian Institute of Science

Presented in SIGMETRICS @ ACM FCRC 2023, Orlando, FL

Full paper

DISTRIBUTED RESEARCH ON EMERGING APPLICATIONS AND MACHINES
Department of Computational and Data Sciences
Indian Institute of Science, Bangalore

Motivation – Why train on the edge?

- Edge devices such as Raspberry Pi, Google Coral and Intel Movidius typically used for inferencing tasks on DNN models
- GPU accelerated NVIDIA Jetson edge devices getting more powerful, can be used in training too
  - Leading edge AI performance on industry benchmark MLPerf
  - In this work, we consider NVIDIA Jetson edge devices
- From an application perspective, edge training because
  - Growth in data collected from edge devices
  - Bandwidth constraint in moving data (esp. video) to the cloud
  - Attention to privacy and security

Jetson Devices

AGX

NX

Nano

Credit card (for size ref.)

Motivation – Why train on the edge?

Comparable to a recent desktop GPU (3060 Ti)!

Device/ Feature	Nano	Xavier NX	Xavier AGX	Orin AGX	Edge TPU
CPU	4 core ARM Cortex A57	6 core Carmel, ARM v8.2	8 core Carmel, ARM v8.2	12 core, ARM A78AE	4 core, ARM A53
GPU	128 core (Maxwell)	384 core (Volta) 48 tensor cores	512 core (Volta) 64 tensor cores	2048 core (Ampere) 64 tensor cores	GC7000 (graphics only)
Memory	4 GB	8 GB	32 GB	32 GB	1-4GB
Storage	MicroSD*, USB3	MicroSD*, M.2, USB3	eMMC*, M.2, eSATA, MicroSD, USB3	eMMC*, M.2, MicroSD, USB3	eMMC, MicroSD, USB3
Accelerators	None	2 DLA	2 DLA	2 DLA	ML accl
PeakPower(W)	10	15	65	60	2W
Cost (USD)	129	399	999	1999	130
Form factor	69 x 45 mm	103 x 90 mm	105 x 105 mm	110 x 110 mm	88 x 60 mm

Training on the edge is motivated by a convergence of emerging applications and growing hardware capabilities

Why characterize edge training?

Not addressed in existing literature

✔Inference on edge – characterization [4,17], variability [1], energy usage [2,19]
✔Training on server/cloud – characterization [34], caching [26, 34]
? Training on edge

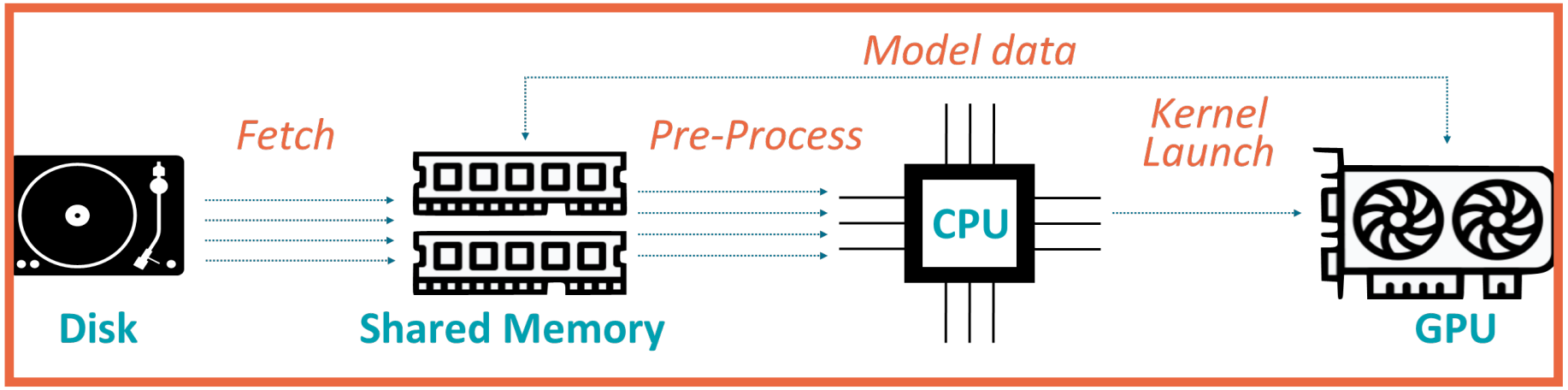
Large parameter space

- Device parameters such as storage media, power modes, DVFS
  - Ex. Xavier AGX has ~29k possible power modes
- Training parameters such as I/O pipelining and parallelism, minibatch size

Heterogeneity and variety in processing elements

- DLA (ASIC) and other custom accelerators apart from standard CPU, GPU
- Shared DRAM- possible memory contention even with a single workload

Deep Learning Training Pipeline on Jetson



Fetch to shared memory	Pre-processing on CPU	Compute on GPU
• Data fetched from storage • Storage media – SSD, HDD, SD card, eMMC	• Resizing, data augmentations	• Forward pass • Backward propagation • Update weights

How do we configure and tune device & framework parameters to train efficiently on the edge?

Contributions

Understand effects of **caching, s/w pipelining & parallelizing data fetch and pre-processing** on the stall time and epoch training time, and the interplay between CPU and GPU performance

Study impact of **storage media and mini-batch sizes** on stalls, GPU compute and end-to-end times, and confirm deterministic training performance of these devices across time and instances

Investigate the consequence of **Dynamic Voltage and Frequency Scaling (DVFS) and power modes** on the training time, energy usage and their trade-off

Predict training time and energy per epoch for a candidate DNN model for any custom power mode of a given device with limited profiling

Insights for practitioners to tune their edge device and DNN framework as well as for systems researchers to **help design systems software** for performant training on the edge

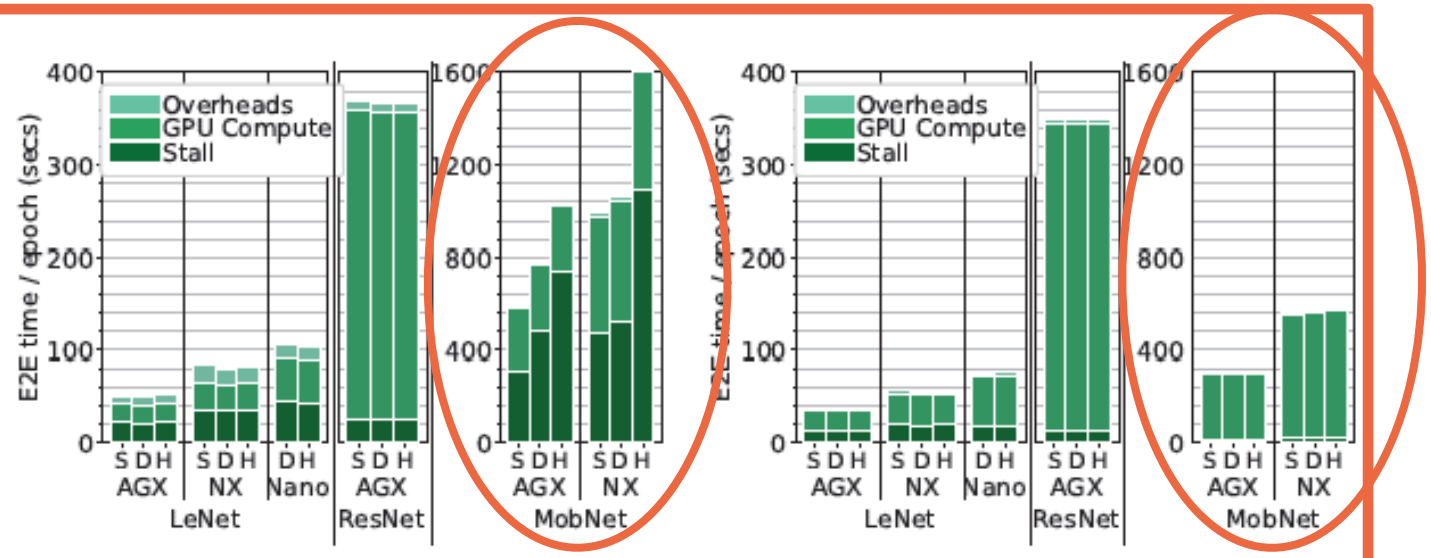
Ours is the first paper to conduct a rigorous and principled performance characterization of training on an accelerated edge device

Experimental Setup

- Model – 3 edge-friendly image classification models
- Framework – PyTorch
- Devices
  - 3 device classes – AGX, NX and Nano
- Storage Media
  - SSD, HDD, SD card
- Default Configuration
  - Fan set to max, MAXN-eq power mode, DVFS off

Model	FLOPs	Dataset	Size	AGX	NX	Nano
ResNet18	1.82G	**CIFAR-10**	150MB	Yes	No	No
MobileNetv3	225.4M	**GLD23k**	2.83GB	Yes	Yes	No
LeNet-5	4.4M	**MNIST**	46MB	Yes	Yes	Yes

Selected Results – Storage Media



(a) Epoch 0 (no caching), $w = 0$ (no pipelining)

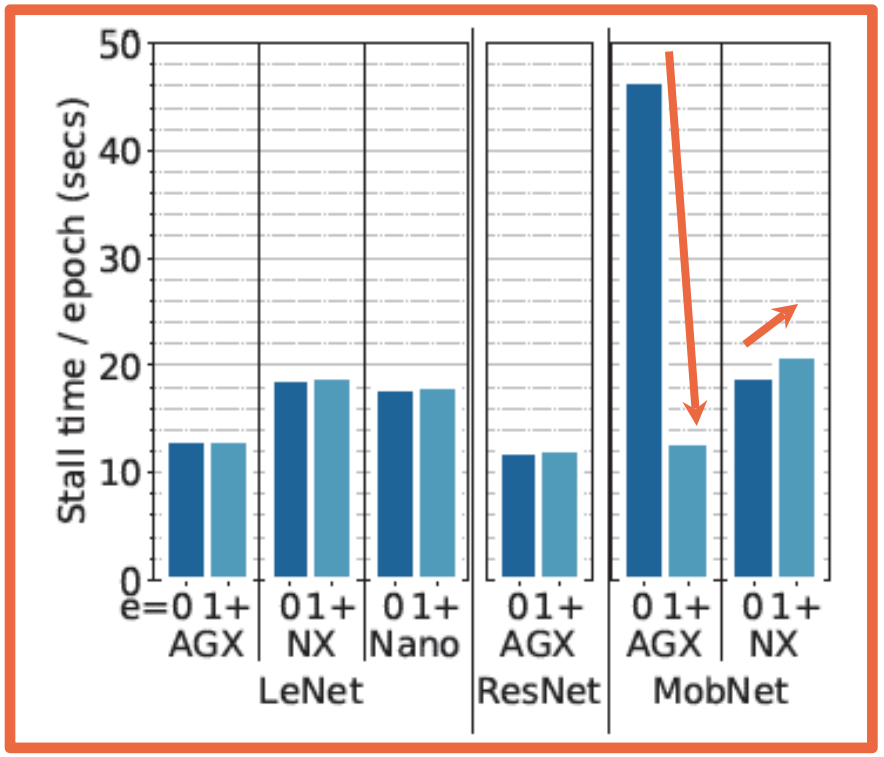
(b) Epochs 1+ with $w = 4$

Impact
Significant savings in capital cost for an edge deployment at scale

Effect of SSD, SD card and HDD storage media on the stall time and the end-to-end time per epoch

- *Caching and pipelining can hide the stall times of a slower storage media, and a faster disk may not offer benefits*
- *Any drop in stall time due to a faster storage media depends on the I/O pressure during fetch*

Selected Results – Page cache

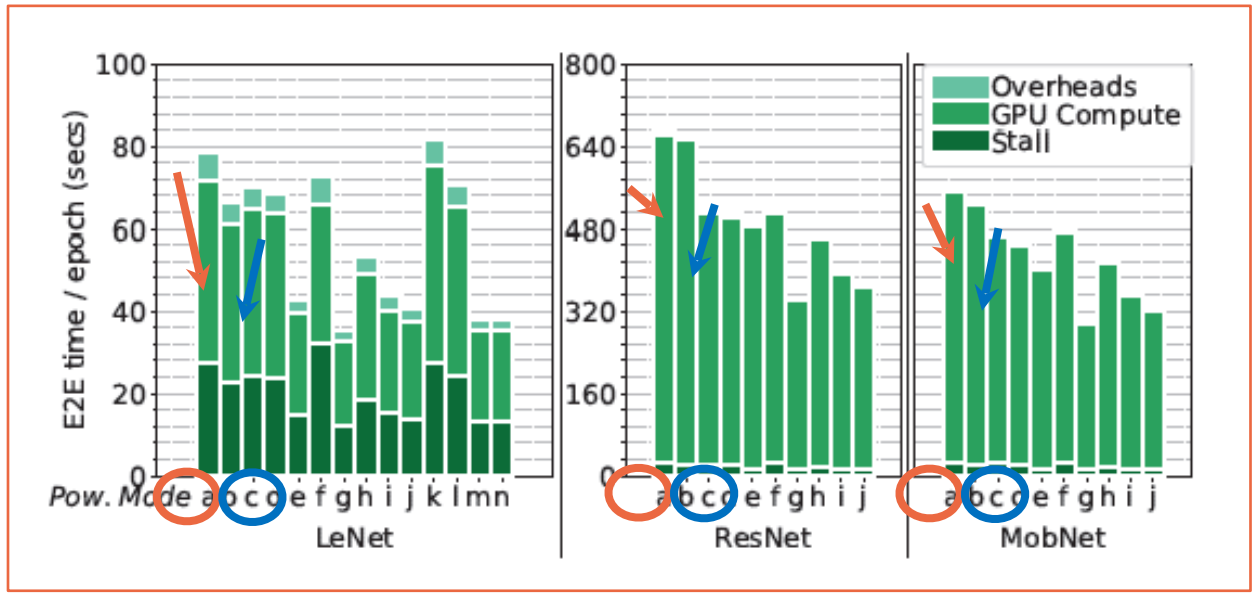


Stall times for Epoch 0 and Epoch 1+

- *A large page cache can reduce the stall time.*
  - Stall time for **MobileNet on AGX** drops from **46.4**s for epoch 0 to **12.9**s for epoch 1+.
  - AGX's 32 GB of RAM is able to fit the MobileNet model and its entire GLD dataset (2.8 GB) at the end of epoch 0.Thereafter, we see 0 IOPS.
  - Stall times for **MobileNet on NX** are **18.9**s for epoch 0 and a comparable **20.9**s for epoch 1+
  - NX only has 8 GB of RAM and less than 6% of data is cached as per vmtouch

Impact
Design choice of RAM size depending on both model and dataset

Selected Results – Power Modes



E2E time per epoch for different power modes of AGX

Label	CPU Cores	CPU MHz	GPU MHz	RAM MHz
d	8	1200	900	**1600**
e	8	**2100**	900	1600
f	2	2100	900	1600
g (MAXN)	**8**	**2265**	**1377**	**2133**
h	8	2265	1377	**1066**

- *For lightweight models (LeNet), CPU frequency plays a key role in GPU compute time*
  - Mode d to e, CPU frq increases from 1.2 to 2.1GHz, GPU compute time drops by ~37%
  - Kernel launch overheads dominate
- *GPU compute time and stall times are affected by the memory frequency*
  - Mem frequency doubles from mode *h* to *g*
  - GPU compute time drops by ~33% for LeNet, ~26% for ResNet and 28% for MobileNet
  - Stall times reduce by ~34%, ~28% and ~28%
  - Consequence of shared RAM

Impact
Workload-aware power mode selection

Selected Results – Power Modes

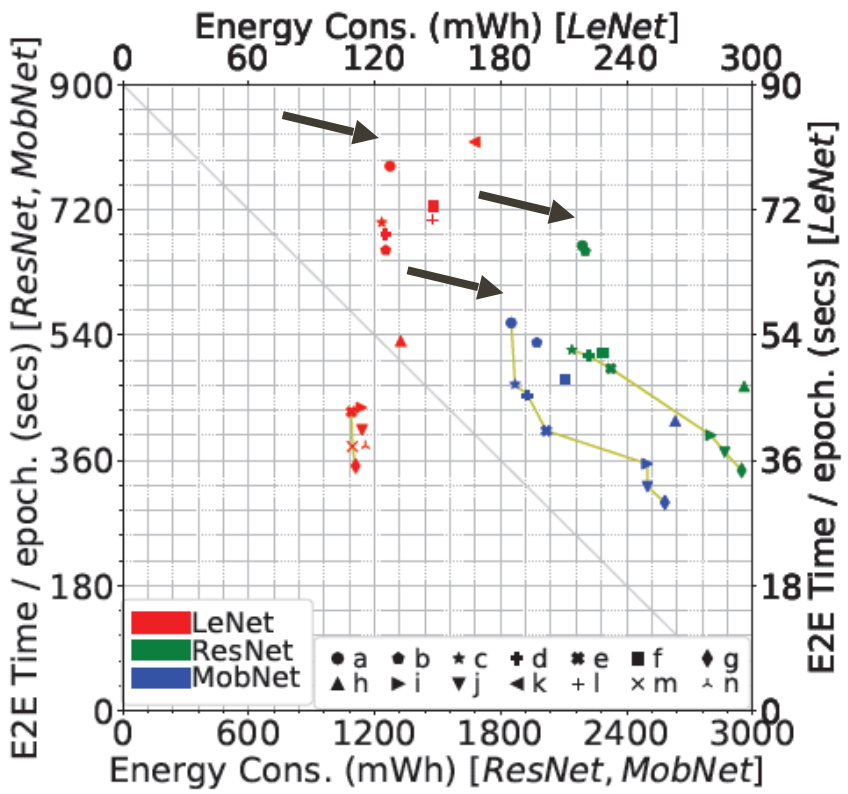


Fig. 8. Scatter plot of *E2E time vs. Energy consumed per epoch* (1+), for power modes *a–n* of AGX for LeNet (secondary axes) and ResNet/MobileNet (primary axes). The yellow lines indicate the *Pareto front* per DNN.

- *System default power mode may not be Pareto optimal*
  - Default power mode *a* is not on Pareto front for LeNet/ResNet, sub-optimal tradeoffs!

Impact
Pareto optimal time-energy tradeoffs

End-user might have a constraint on time, energy or power.

Which power mode to choose?

Performance and energy modeling is needed

Empirical Modeling

Overview

- Predict time and energy per epoch required to train a candidate DNN model for any custom power mode of a given device.
- We use simple regression techniques and minimal profiling over a limited number of epochs(3) and power modes(4).
  - This takes ~2h even for the costliest DNN we evaluate

Prediction model

- Time model: We identify 4 representative power modes and fit a linear regression model for each DNN
- Training time: $Ti = a \cdot x1 + b \cdot x2 + c \cdot x3 + d \cdot x4 + e$
  - CPU frequency($x1$), CPU cores ($x2$), GPU frequency ($x3$) and memory frequency ($x4$)
- Energy model is developed similarly, takes predicted training time as input

Results – Empirical Modeling

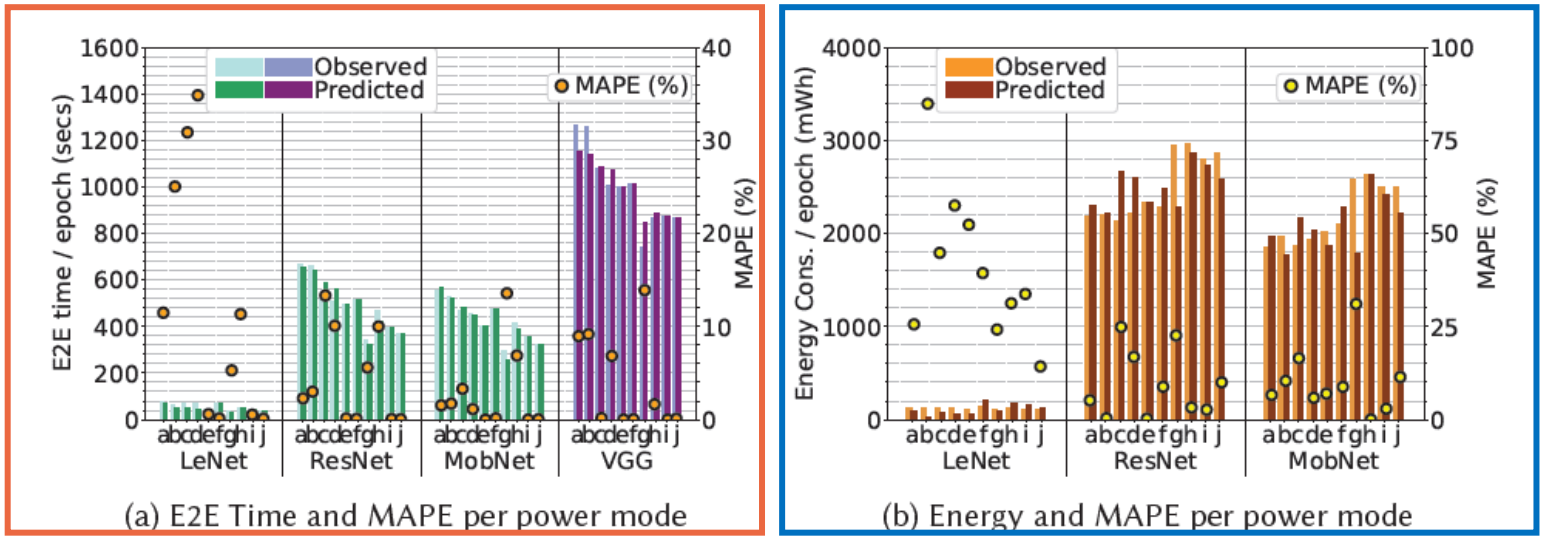


Fig. 10. Predicting end-to-end (E2E) training time and energy use per epoch for power modes.

Impact
Device selection for federated learning

- Training time model
  - MAPE within 10% for 31 out of 40, within 15% for all but 3
  - Further, evaluating on a new model (VGG) gives <10% error for 9 out of 10 modes
- Energy model
  - Single model across all DNNs
  - ResNet and MobileNet <25% MAPE
  - But Lenet and VGG see poor results
  - Needs more investigation

Summary of Experiments

- **Pipelining/parallel fetch and pre-process** – vary number of Dataloader workers
- **Storage media** – SSD, SD card, HDD
- **Page caching** – with and without cache drop at the end of every epoch
- **Variability study** – training time and energy across epochs and device instances
- **Minbatch size** – vary minibatch size and analyze consequences
- **DVFS** – impact of DVFS on time and energy
- **Baseload** – plug load measured using external power monitor under various idle states
- **Power modes** – models trained and analyzed on 10-12 power modes

Each experiment replicated on 3 device classes, run thrice to validate reproducibility

Additional Takeaways

- Unlike inferencing studies, we do not see much variability in training time across device instances or over time
  - Significant variability seen with different OS / PyTorch / CUDA versions
- Baseload accounts for a large part of training energy
  - Wake on LAN can be used to reduce energy footprint for periodic workloads like federated learning
- DVFS has negligible effect on E2E time and energy consumption

A total of **24** claims in the paper

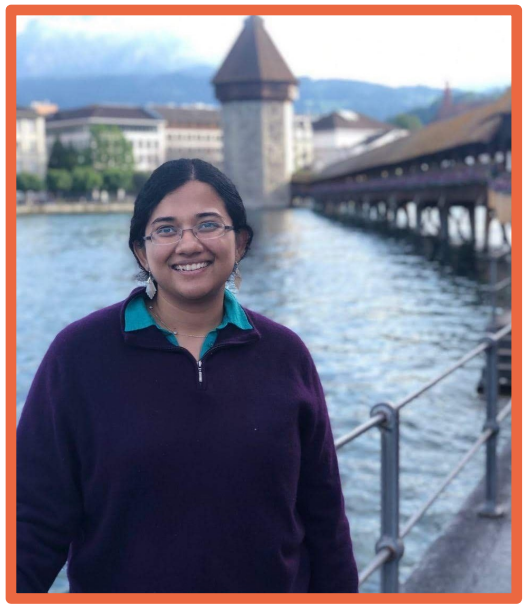
Conclusions

- Confirms conventional wisdom backed up with quantifiable metrics
- Counter-intuitive results help rethink system design and tuning for DNN workloads.

Future Work

- More accurate performance and energy modeling
- Optimize concurrent inference and training on the edge

Thank you! Questions?



Prashanthi S.K

Sai Anuroop Kesanapalli

Yogesh Simmhan

Full paper

Characterizing the Performance of Accelerated Jetson Edge Devices for Training Deep Learning Models

Prashanthi S.K, Sai Anuroop Kesanapalli and Yogesh Simmhan
Indian Institute of Science

Presented in SIGMETRICS @ ACM FCRC 2023, Orlando, FL

Full paper

DISTRIBUTED RESEARCH ON EMERGING APPLICATIONS AND MACHINES
Department of Computational and Data Sciences
Indian Institute of Science, Bangalore

Additional Slides

Detailed Specifications - Jetson Devices

Table 1. Specifications of NVIDIA Jetson Devices Evaluated

Feature	Nano [38]	Xavier NX [39]	AGX Xavier [39]	AGX Orin [40]
ARM CPU Architecture	A57	Carmel	Carmel	A78AE
CPU Cores†	4	6	8	12
CPU Frequency (MHz)†	1479	1900	2265	2200
GPU Architecture	Maxwell	Volta	Volta	Ampere
CUDA/Tensor Cores	128/–	384/48	512/64	2048/64
GPU Frequency (MHz)†	921	1100	1377	1300
RAM (GB)	4	8	32	32
Storage Interfaces	μSD, USB	μSD, NVMe, USB	μSD, eMMC, eSATA, NVMe, USB	μSD, eMMC, NVMe, USB
Memory Frequency (MHz)†	1600	1600	2133	3200
Power (W)†	10	15*	65#	60
Price (USD)	$129	$399	$999	$1999

† This is the maximum possible value across all power modes. Actual value depends on the power mode used (Table 3). * This peak power is for Jetpack release $v4.5.1$ and earlier. # The data sheet does not list the power for the MAXN peak power mode. We report the power adapter rating of 65W.

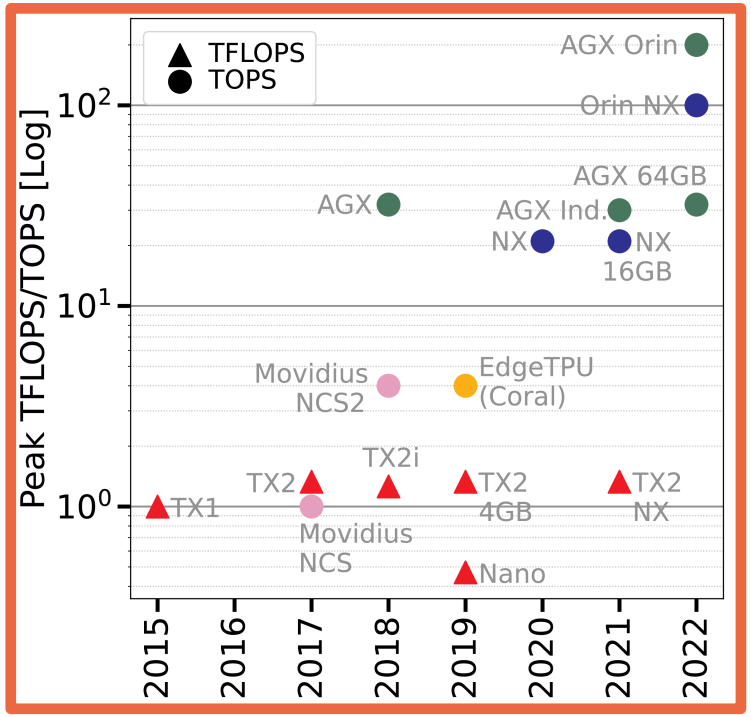
Model and Dataset Details

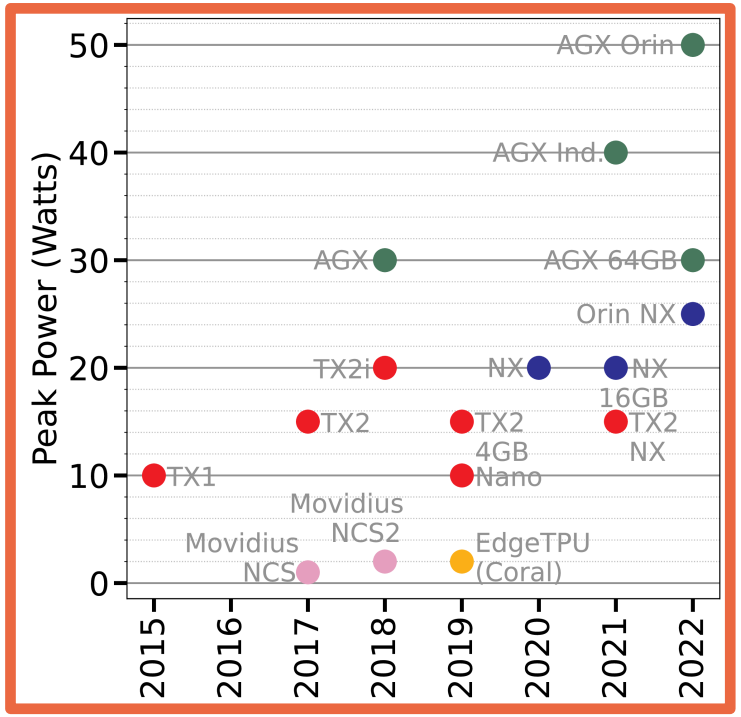
Table 2. DNN Models, Training Datasets and Device Trained On

Model	# Layers	# Params	Mem. Used [8]	FLOPs	Dataset	# Train. Samples	Size on Disk	Batch Size	AGX	NX	Nano
LeNet-5	7 [28]	60*k* [28]	0.35*MB*	4.4*M* [13]	MNIST	60, 000 [28]	46*MB*	16	✓	✓	✓
MobileNetv3	20 [20]	5.48*M* [43]	124.73*MB*	225.4*M* [43]	GLD23k	23, 080 [52]	2827*MB*	16	✓	✓	
ResNet-18	18 [18]	11.68*M* [51]	53.89*MB*	1.82*G* [51]	CIFAR-10	50, 000 [24]	150*MB*	16	✓		
VGG-11	11 [50]	132.86*M* [51]	509.78*MB*	7.63*G* [51]	CIFAR-10	—"—	—"—	—"—	✓		

Edge Hardware Trends

- From a hardware perspective, edge devices are getting more powerful by the day





Compute and power trends for edge devices over the past 7 years

Related Work – Edge devices

- Characterization
  - DeepEdgeBench [4] compares the inference time and power consumption for various edge devices
  - Hassan et al. [5] GPU matrix multiplication workload characterization on Jetson devices
- Variability
  - Snowflakes [6] latency and power variability across Jetson AGX Xaviers for inferencing
- Energy usage
  - Holly et al. [7] correlate CPU and GPU frequencies and the number of CPU cores with the latency, power and energy for inferening on the Jetson Nano
  - Hazem et al. [8] effect of CPU and GPU frequencies on power consumption for stream processing workloads.
- Benchmark
  - MLPerf [11] does not measure low-level system metrics and lacks a suite for edge training

None of these use training workloads!

Related Work – Cloud / Server Training

- Mohan et. al [9] characterize the training data pipeline and how it affects training time on GPUs. Also propose a modified caching mechanism that minimises I/O
- Quiver [10] proposes a caching strategy based on substitutability without interfering with training requirements
- Gandiva [30] time-slices GPUs efficiently across multiple DNN training jobs
- Antman [31] is a scheduler that uses spare resources to execute multiple jobs on a shared GPU while minimizing interference between the jobs

Learnings not applicable to edge devices directly

Related Work – Summary and Gaps

Why can't learnings from server training be applied directly?

- Different CPU architecture
  - ARM
- Non-traditional storage media
  - Not just HDD and SSD
  - eMMC, SD card
- Memory
  - Shared between GPU and CPU as opposed to discrete GPU memory
  - Slower, low power (LPDDR) as opposed to faster GDDR
- Different power modes available
  - Different CPU, GPU and memory frequencies
  - Lower power budgets, deployment condition effects

Systems research, characterization and performance optimization

✔ Inference on edge

✔ Training on server/cloud

? Training on edge

This opens up interesting system research problems

Power modes

Table 3. Power Modes Evaluated

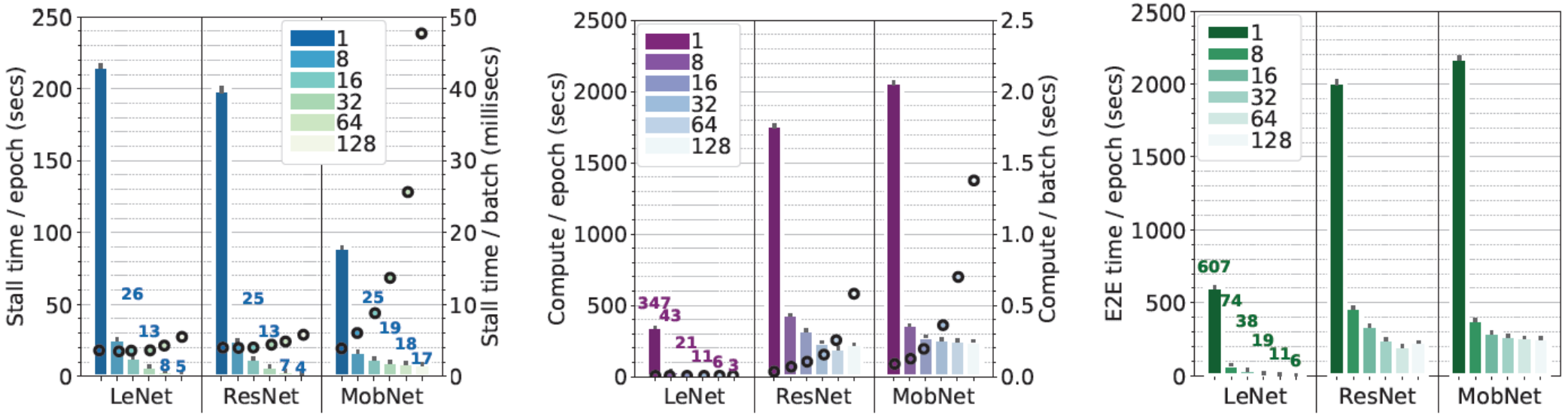
Device	Label	CPU Cores	CPU MHz	GPU MHz	RAM MHz
AGX	a	4	1200	670	1333
	b	**8**	1200	670	1333
	c	8	1200	**900**	1333
	d	8	1200	900	**1600**
	e	8	**2100**	900	1600
	f	**2**	2100	900	1600
	g (MAXN)	**8**	**2265**	**1377**	**2133**
	h	8	2265	1377	**1066**
	i	8	2265	1377	**1333**
	j	8	2265	1377	**1600**
	k	**4**	**1036**	**420**	**2133**
	l	**8**	1036	420	2133
	m	8	**2265**	420	2133
	n	8	2265	**900**	2133
NX	15W	6	1400	1100	1600
Nano	MAXN	4	1479	921	1600
Orin	MAXN	12	2200	1300	3200

Cells in bold indicate a value change from the cell in the previous row.
Since most of these are custom power modes, they do not have a preset power budget.

Scale of experiments

- **102** experiments
  - 6 epochs per experiment, results averaged over epochs 1 to 5
  - Longer run to demonstrate generalizability
- Total number of epochs run = **5170**
- Number of devices = **16**
  - 4 device classes (Nano, NX, AGX, Orin AGX)
- Each experiment run thrice to validate reproducibility

Selected Results – Mini-batch Size



(a) Stall Time (b) GPU Compute Time (c) E2E time

Fig. 4. Performance on AGX with *batch-size (bs)* changing from 1 to 128. The *time per epoch* is on the bars on the left Y axis, while the *time per mini-batch* is on the markers on the right Y axis.

- Increasing the mini-batch size reduces the training time per epoch until the parallelism of the GPU cores saturate.
- Increasing the mini-batch size increases the stall time per mini-batch but reduces the overall stall time per epoch.

Base-load and Incremental Training Energy

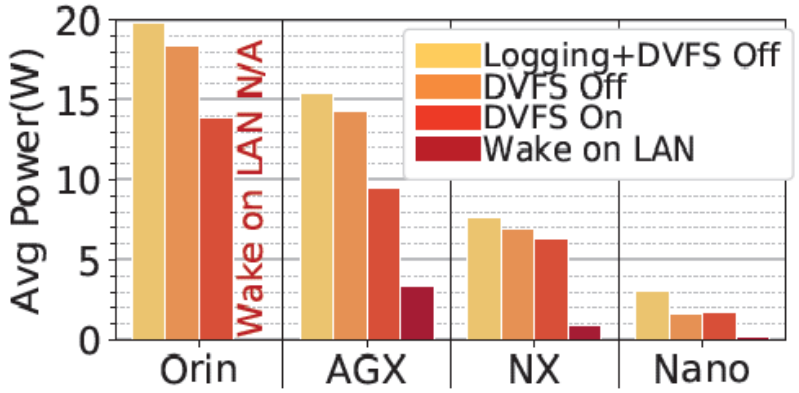
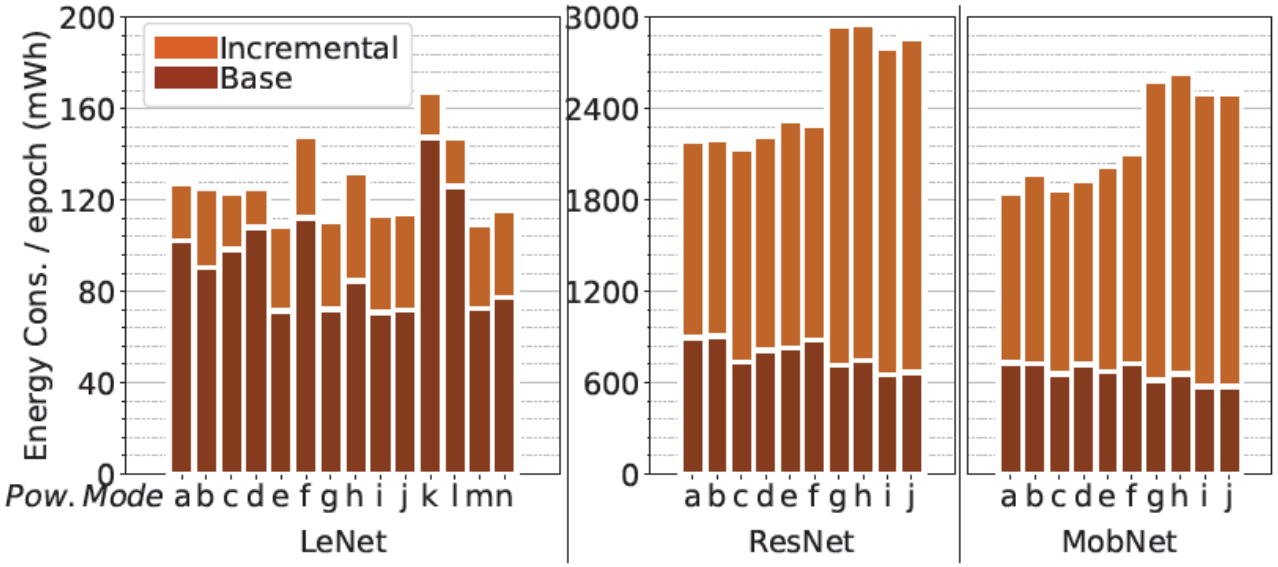


Fig. 7. *Average socket power load (W)* for various idle states on all devices.



(b) Energy