zTT: Learning-Based DVFS with Zero Thermal Throttling for Mobile Devices

Seyeon Kim, Kyungmin Bin, Sangtae Ha, Kyunghan Lee, Song Chong

MobiSys '21: Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services

Presented by Huai-an Su

Published in IEEE Transactions on Wireless Communications

Outline

- Motivation
- zTT : Learning-based DVFS
- Implementation
- Results
- Conclusion

Motivation

More and more high performance tasks are ran on mobile devices

- High quality games
- Multi-task apps
- 5G chipset

Motivation

Main problems on mobile devices

- Overheat
- Power consumption

ANTS LAB

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

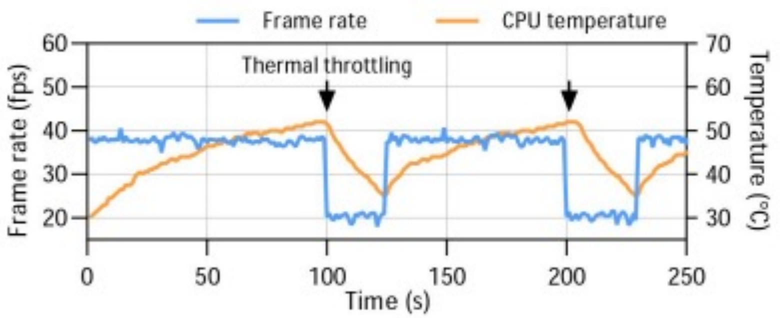
Motivation

Solution to power: DVFS

- Adjusting GPU/CPU voltage-frequency to reduce power consumption

Solution to overheat: thermal throttling

- Lowering the performance when the temperature meets a predefined threshold



Motivation

Problems with DVFS:

- Inefficiency due to independent governor (CPU/GPU)

Problems with thermal throttling:

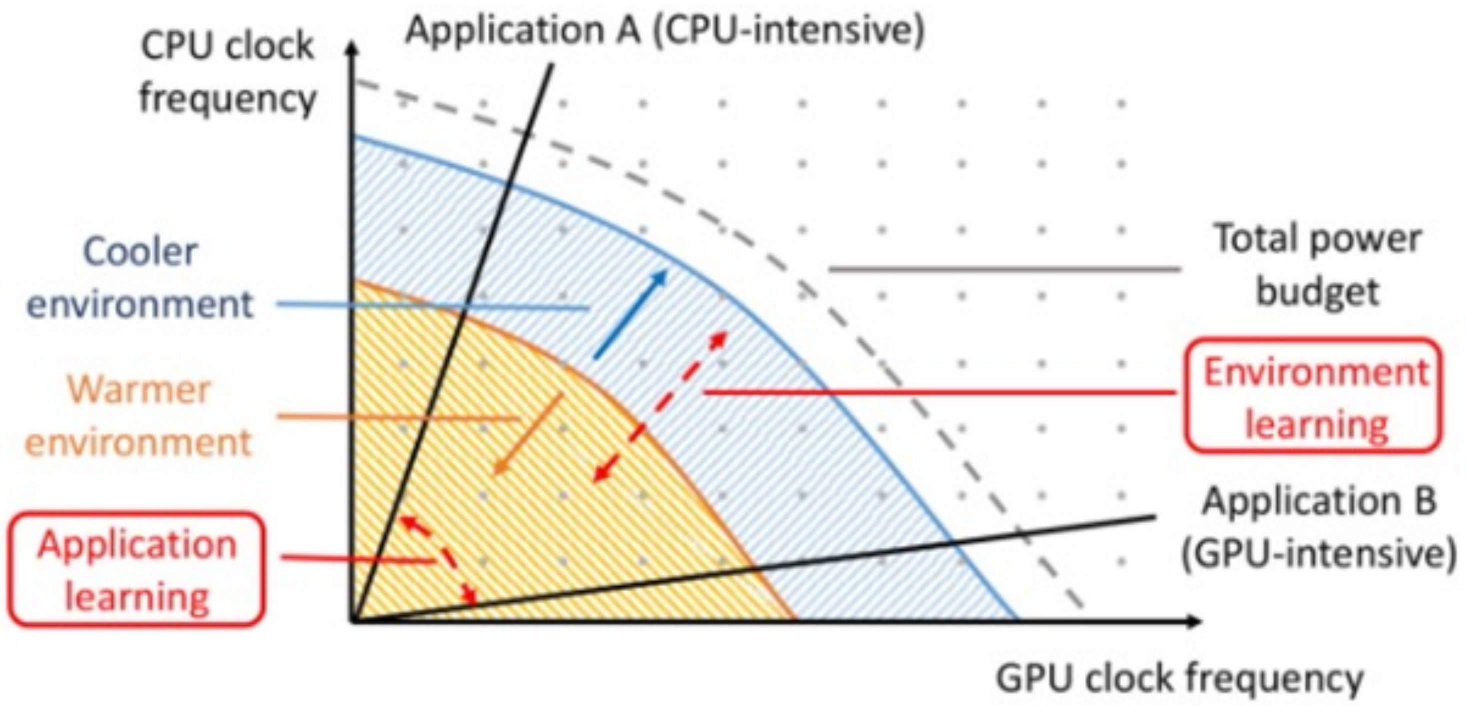
- Changing of outside environment and inside performance

zTT : Learning-based DVFS

Approach

- Learning from the environment and application performance
- Predicting future temperature or future application performances
- Optimizing power consumption while preventing the device from reaching thermal threshold

zTT : Learning-based DVFS



zTT : Learning-based DVFS

Design

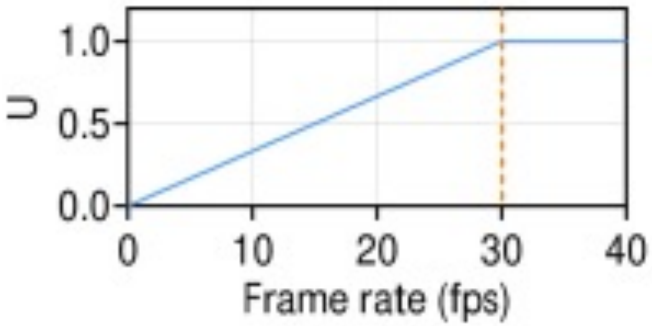
- Reward $r(t) = U(t) + \frac{\beta}{P(t)} + W(t)$
- U(t) as utility function (User QoE)
- P(t) as total power consumption
- W(t) as thermal constraints
- β as trade-off term

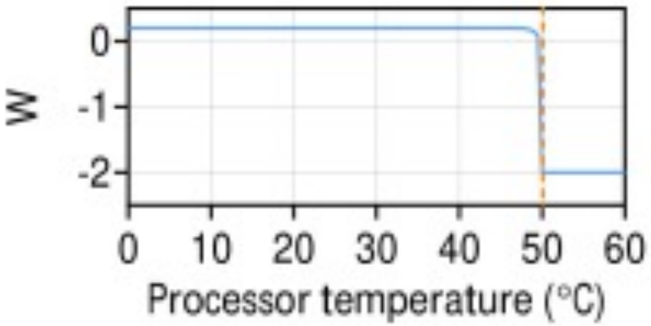
zTT : Learning-based DVFS

Design

$$U(t) = \begin{cases} 1, & if\ x(t) \geq X_t \\ \frac{x(t)}{X_t}, & otherwise \end{cases}$$

$$W_C(t) = \begin{cases} \lambda \cdot tanh(T_{C,th} - T_C(t)), & if\ T_C(t) < T_{C,th} \\ -10 \cdot \lambda, & otherwise \end{cases}$$ [4]





zTT : Learning-based DVFS

Design

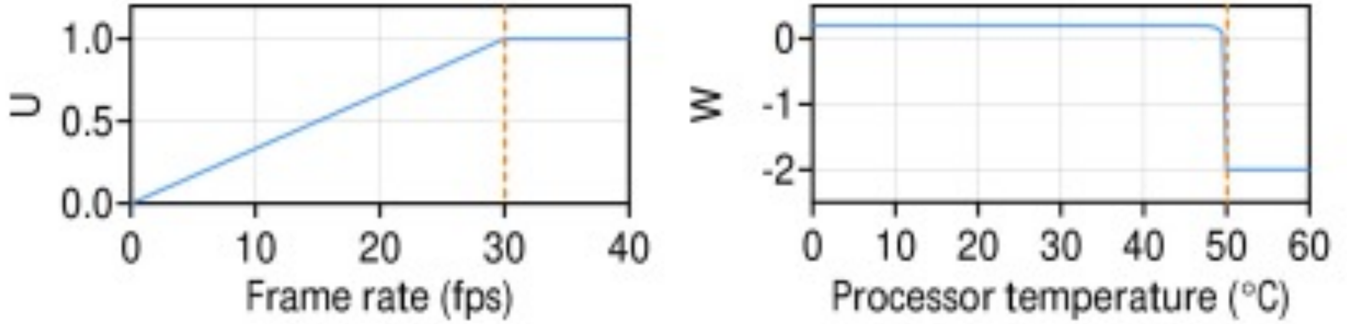
- Markov Decision Process

CPU/GPU Temperature Frame rate

- State s(t) = (fc(t), fg(t), Tc(t), Tg(t), Pc(t), Pg(t), x(t))

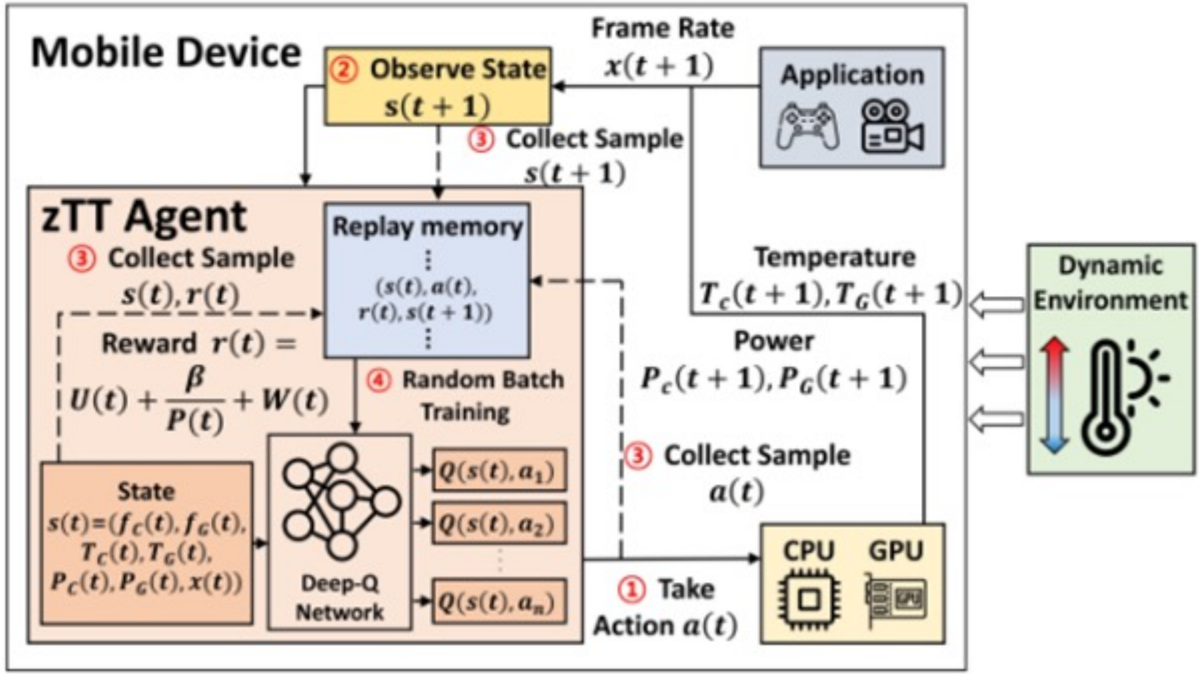
CPU/GPU Frequency CPU/GPU Power

- Action a(t) = (fc(t), fg(t))
- Reward r(t)



Implementation

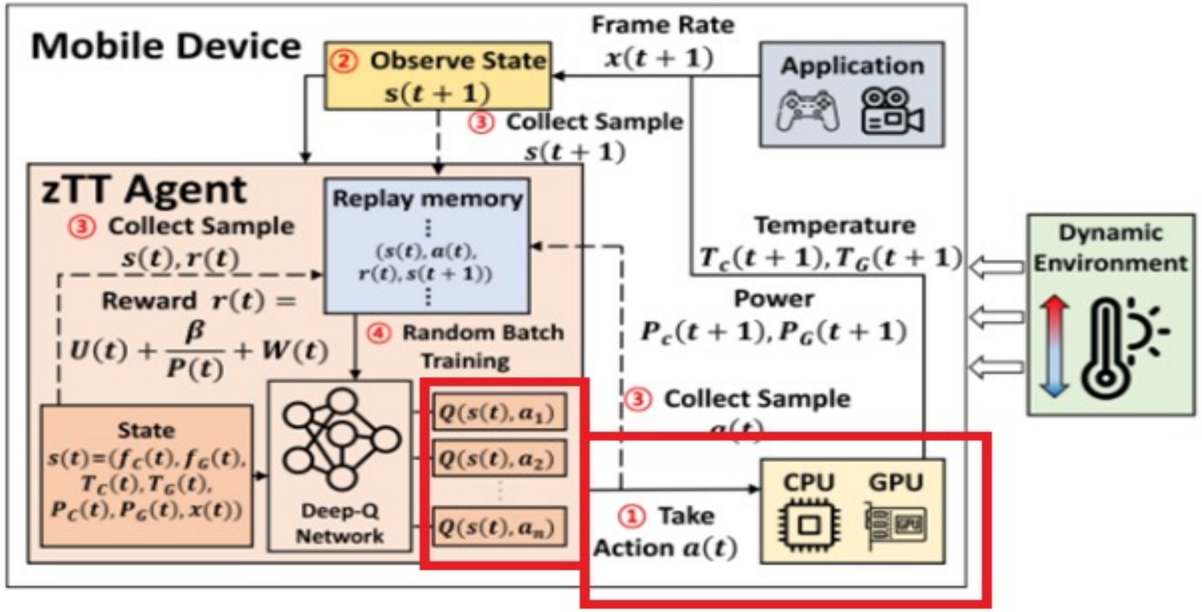
Overview



Implementation

Step 1: take action

- Greedy method (exploration, exploitation)
- Cool--down



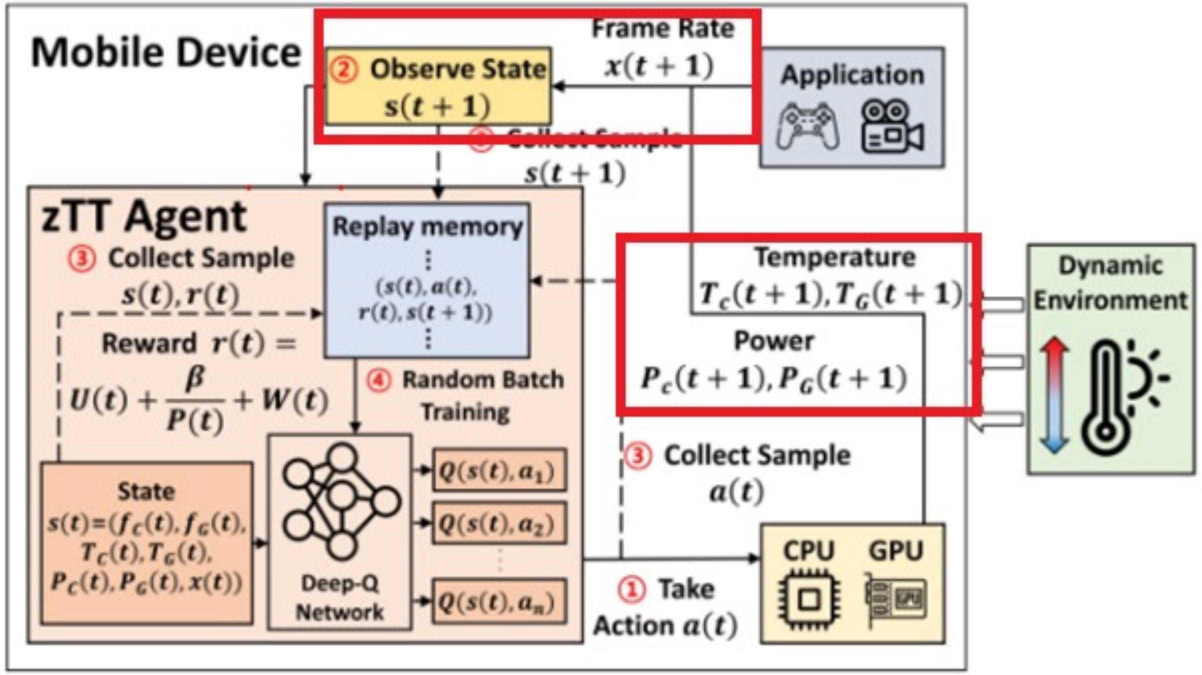
ANTS LAB

UNIVERSITY of HOUSTON
CULLEN COLLEGE of ENGINEERING

Implementation

Step 2: observe state

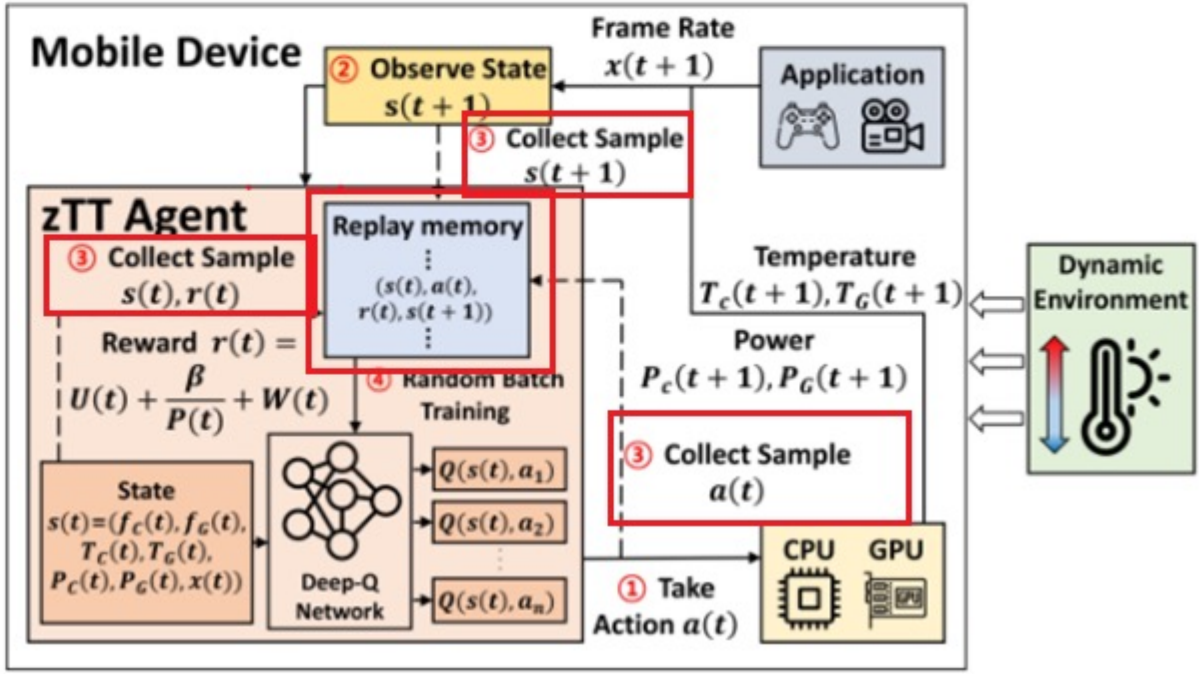
- Observe s(t+1) after a(t)



Implementation

Step 3: collect sample

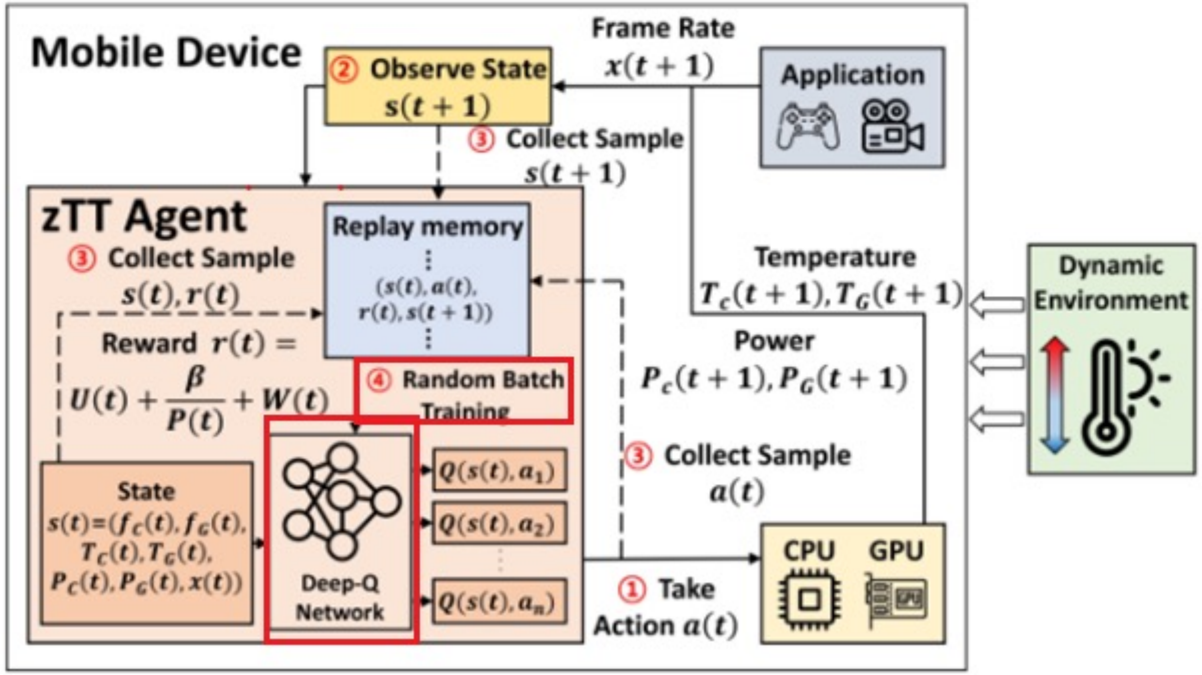
- Collect s(t), r(t), s(t+1), a(t)



Implementation

Step 4: random batch training

- Using deep-Q-learning



Implementation

Specifications of JETSON TX2 and Pixel 3a

Device	JETSON TX2	Pixel 3a
CPU	NVIDIA Denver2 + ARM Cortex-A57	ARM Cortex-A55(LITTLE) + ARM Cortex-A75(big)
GPU	NVIDIA Pascal GPU	Adreno 615
Memory	8GB DDR4	4GB LPDDR4X
OS	Ubuntu 16.04	Android 9.0 Pie

Implementation

Experimented applications and devices

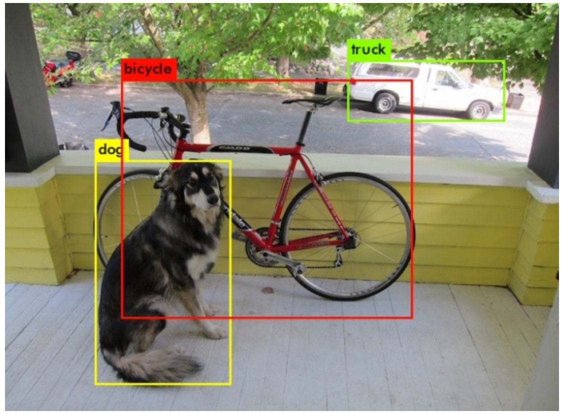
Application	Description	Device
Aquarium [15]	WebGL-based 3D object rendering	JETSON TX2
YOLO [35]	Deep learning-based object detection	JETSON TX2
Video rendering	Rendering a video with OPENCV2	JETSON TX2
Showroom VR [23]	WebGL-based 3D object rendering	Pixel 3a
Skype [43]	Video call	Pixel 3a
Call of duty 4 [1]	3D Mobile game	Pixel 3a

Implementation

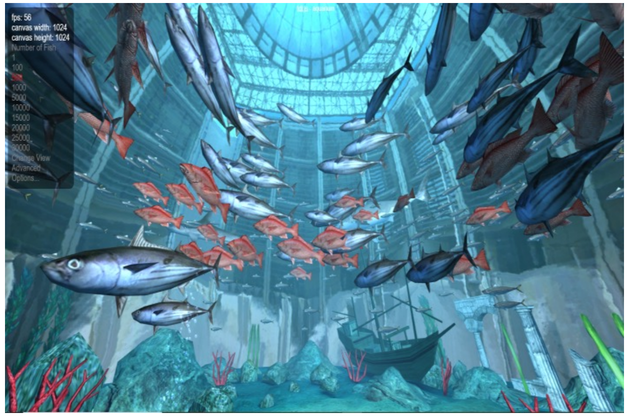
Experimented applications with Jetson Tx2

Video rendering

YOLO



Aquarium



Implementation

Experimented applications with Pixel 3a

Showroom VR

Skype

Call of Duty

Implementation

- Comparison

Governor	Description	Device
Maestro	Proportional-integral control-based DVFS	CPU/GPU
Interactive	CPU utilization-based DVFS	JETSON TX2 CPU
Simple_ondemand	GPU utilization-based DVFS	JETSON TX2 GPU
Schedutil with EAS	CPU load-based DVFS	Pixel3a CPU
msm_adreno_tz	GPU utilization-based DVFS	Pixel3a GPU

Maestro

Default(Jetson Tx2)

Default(Pixel 3a)

Implementation

Learning Application QoE

- Rule out environmental effects
- Conducting experiments in a heavily-cooled environment

Implementation

Learning Static Environments (Pixel 3a)

- With four different environmental settings
- Normal, fan, pocket, protective case

Learning Static Environments (video rendering)

- Learning through thermal headroom

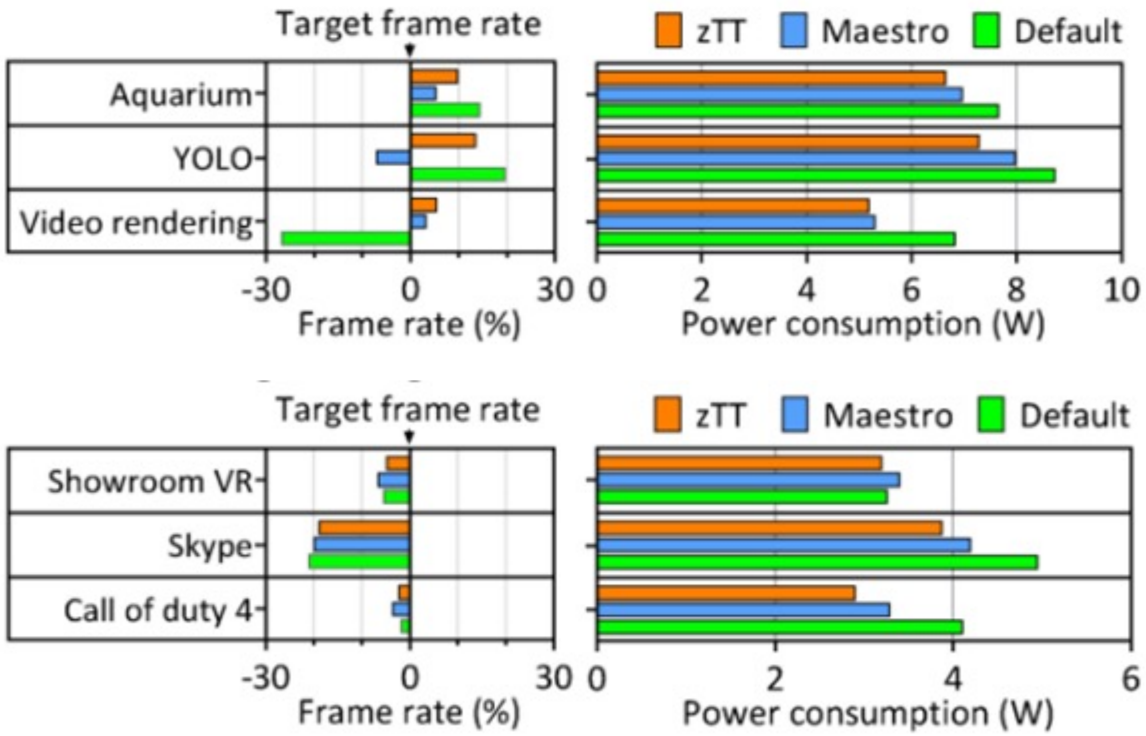
Implementation

Learning Changing Environments (Jetson Tx2)

- Transfer learning for boosting adaptation speed
- Using sample copies for faster convergence
- Environmental changes

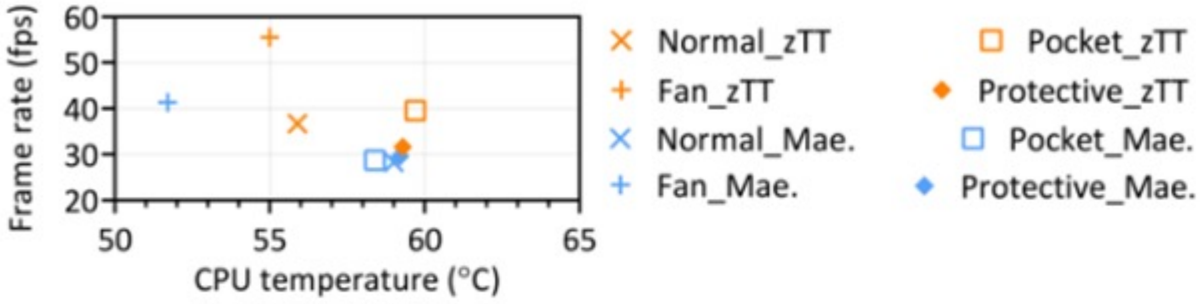
Results

Frame rate and power consumption comparison on different applications (Learning Application QoE)

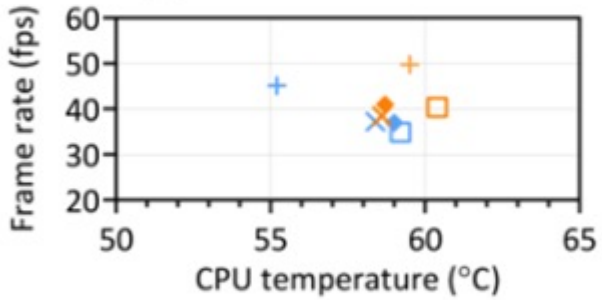


Results

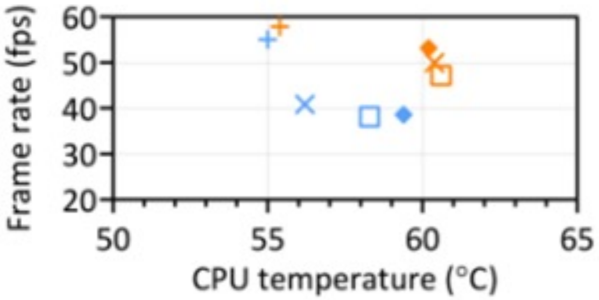
Different environment (Learning Static Environments for Pixel 3a)



(a) Showroom VR



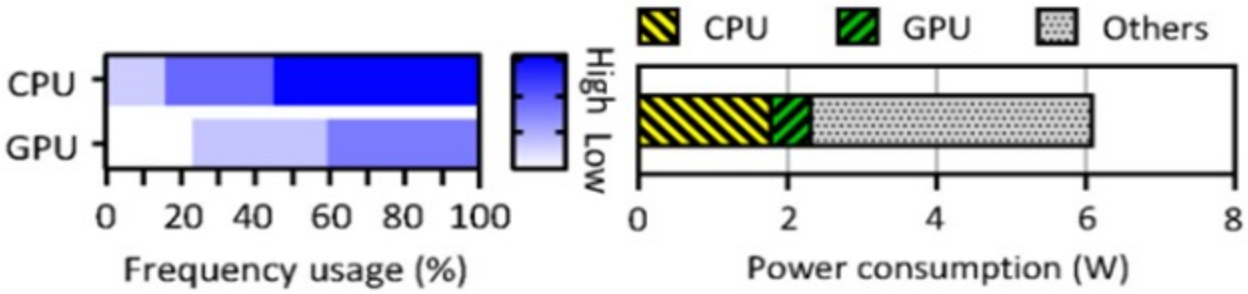
(b) Skype



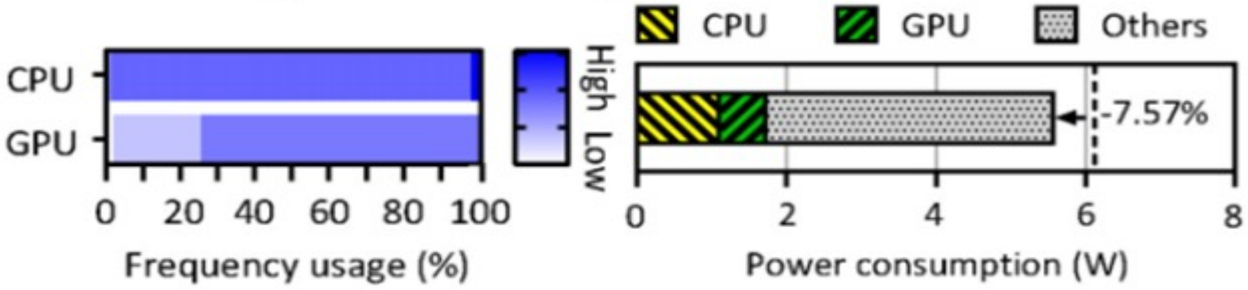
(c) Call of duty 4

Results

The contributions to learning (Learning Static Environments for video rendering)



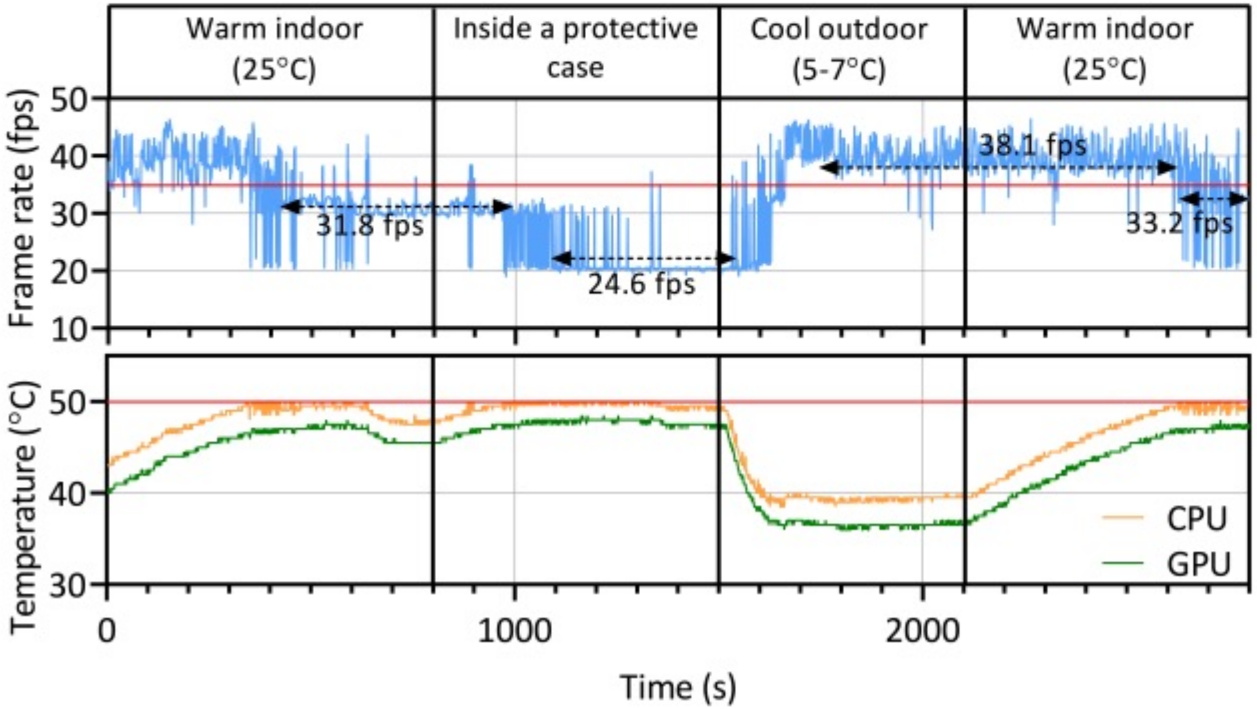
(a) Before learning thermal headroom



(b) After learning thermal headroom

Results

The changing of frame rate (Learning Changing Environments for video rendering)



Conclusion

- Mobile devices facing power and thermal problems
- Traditional DFVS and Thermal throttling have limitations
- zTT DVFS introduces environment and application learning and predicting
- zTT DVFS provides lower power consumption, better performance trade-offs, and reduced issues from temperature.


THANK YOU
UNIVERSITY of HOUSTON | ENGINEERING