Hot Under the Hood: An Analysis of Ambient Temperature Impact on Heterogeneous Edge Platforms

Amirhossein Ahmadi
amirhmi@ece.ubc.ca
University of British Columbia

Hazem A. Abdelhafez
hazem@ece.ubc.ca
University of British Columbia

Shashwat Jaiswal
shashwatj@iitbhilai.ac.in
Indian Institute of Technology Bhilai

Karthik Pattabiraman
karthikp@ece.ubc.ca
University of British Columbia

Matei Ripeanu
matei@ece.ubc.ca
University of British Columbia

ABSTRACT

Applications deployed at the edge are often subject to critical Quality of Service (QoS) objectives, such as meeting deadlines while optimizing for energy consumption. To design and operate middleware that satisfies these QoS objectives, it is crucial to understand the runtime and power consumption characteristics of the edge platform. However, while edge platforms are frequently deployed in environments where ambient factors cannot be controlled, most characterizations are performed without considering environmental factors. We characterize the impact of *ambient temperature* on the power consumption and runtime of machine learning inference applications running on a popular edge platform, the NVIDIA Jetson TX2. Our rigorous data collection and statistical methodology reveals a sizeable ambient temperature impact on power consumption (about 20% on average, and up to 40% on some workloads) and a moderate impact on runtime (up to 5%).

CCS CONCEPTS

• **Computer systems organization → Embedded systems**.

KEYWORDS

Performance and power variation, Edge computing, Jetson TX2

ACM Reference Format:
Amirhossein Ahmadi, Hazem A. Abdelhafez, Shashwat Jaiswal, Karthik Pattabiraman, and Matei Ripeanu. 2023. Hot Under the Hood: An Analysis of Ambient Temperature Impact on Heterogeneous Edge Platforms. In *6th International Workshop on Edge Systems, Analytics and Networking (EdgeSys '23), May 8, 2023, Rome, Italy*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3578354.3592868

1 INTRODUCTION

Context. Today's heterogeneous edge platforms, such as those from the NVIDIA Jetson family, provide an efficient solution for executing compute- and memory-intensive applications (e.g., machine learning inference) at the edge. These applications are often subject to Quality of Service (QoS) objectives, such as energy limitations or task deadlines, that must be met to avoid serious consequences, such as a drone crashing due to a depleted battery, or an autonomous vehicle having an accident due to missing a deadline while analyzing video captures of its environment.

To meet the QoS objectives of edge applications, it is crucial to characterize the edge platform's runtime and power consumption [12]. Accurate characterization is equally important to support decisions to tune the frequency of the various processing units or the memory controller with the same goal of meeting QoS constraints. Such characterizations, however, typically do not consider the impact of environmental factors such as temperature [11].

Embedded edge devices are often deployed in uncontrolled, harsh environments where they are exposed to a range of ambient temperatures that can range from as low as -40°C in cold storage facilities to as high as 70°C in desert environments when directly exposed to the sun [8]. As the ambient temperature increases, the power consumption of the components on the board can also increase due to a multitude of factors such as increased current leakage, and internal elements' resistance. Additionally, elevated ambient temperatures can also cause a reduction in the reliability and performance of the board (see §2).

Objective. The goal of this paper is to *determine the extent to which ambient temperature impacts the power consumption and application runtime on heterogeneous edge platforms.*

Challenges. There are two main challenges. The first challenge is ensuring that the observed variations in power and runtime can be attributed to temperature changes, rather than to confounding factors e.g., background tasks performed by the operating system, runtime variability caused by differences in the application's memory layout with each execution, or measurement inaccuracies - both in hardware tools used to monitor power consumption and software tools used to measure runtime. To overcome this challenge, we use a rigorous experimental setup and data collection methodology to minimize the effects of factors other than temperature, on runtime and power consumption.

The second challenge is the non-normal distribution of runtime and power data, which precludes the use of parametric data analysis and summarization techniques, such as mean, standard deviation, and coefficient of variation. To overcome this challenge, we use non-parametric statistical techniques (i.e., techniques that do not make assumptions about the underlying distributions). These techniques

ACM ISBN 979-8-4007-0082-8/23/05…$15.00
https://doi.org/10.1145/3578354.3592868

(i) use non-parametric summarizing metrics: e.g., median and interquartile intervals, and *(ii)* vary the amount of data to collect based on the variability observed in the data during its collection.

Contributions:

- *Methodology (§3).* The primary contribution of this study is a methodology for characterizing the impact of *ambient temperature* on runtime and power consumption on heterogeneous edge platforms. To overcome the above two challenges, we: *(i)* employ a rigorous experimental setup and data collection methodology, and *(ii)* use non-parametric statistical techniques.
- *Characterization (§5).* We use this methodology and a temperature-controlled chamber to characterize the impact of ambient temperature on runtime and power consumption on machine learning (ML) inference workloads running on the NVIDIA Jetson TX2 heterogeneous edge platform. We place the board inside the temperature-controlled chamber and change the ambient temperature from 25 to 70°C in a systematic manner (§4). We make four main observations. First, we observe that power and runtime data are not normally distributed. Second, the impact of the ambient temperature on power consumption is sizeable (about 20% on average, and up to 40% on some workloads and frequency configurations). Third, between 30% and 50% of the increase in power consumption is linked to an increase in dynamic power at higher temperatures. Finally, there is a moderate impact of ambient temperature on runtime (up to 5% depending on workload and frequency configuration).

Implications: There are two implications of this study. First, our findings add to an increasing body of knowledge [4] that indicates that non-parametric statistical techniques should be the default choice when characterizing computing systems' performance and power consumption (i.e., techniques that assume normality should be used only after verifying that the data collected matches a normal distribution).

Secondly, and more importantly, our findings indicate that when modeling for power, energy (and, depending on the desired accuracy, runtime) of edge applications, one should take ambient temperature into account. One area where such models are employed is to drive the choice of the frequency configuration of the board's processing units and memory controller to meet power or runtime constraints or minimization objectives (e.g., NeuOS[1], PredJoule [9], POET [5], Mcdnn [10], MASA [2], MOSAIC [6]). Because these solutions are based on models that are temperature agnostic, they may lead to configurations that either violate power/runtime constraints or are sub-optimal (as our preliminary experiments in §6 suggest).

2 BACKGROUND

Temperature changes can affect integrated circuits' power consumption, reliability, and performance [15, 16].

Power. Integrated circuits consume more power at higher temperatures due to a variety of factors, the main one being the increased resistance of the circuitry. Additionally, as temperature increases, the leakage current in the transistors also increases. Finally, as temperature increases, the capacitance of the circuitry may also increase, which in turn leads to increased power consumption.

Reliability. At higher temperatures, thermal noise also increases, which can cause errors in data transmission and processing. To prevent these errors, integrated circuits may not only need to use more power for error correction, but also their reliability and performance may be impacted. Higher temperatures can also accelerate the diffusion of impurities in materials, which can cause degradation and aging of the integrated circuits over time. Temperature cycling can also lead to thermal fatigue and cracking of materials.

Performance. While we find that temperature has a moderate impact on performance (up to ±5% with fixed clock frequency and without any throttling), there is limited information in the literature on the underlying mechanisms that may create a performance-temperature relationship. Among the possible causes are: *(i)* the impact of more frequent error correction on memory or data transmission needed to recover the more frequent faults at higher temperatures (mentioned above); *(ii)* slight deviations in clock frequency with changes in temperature; and *(iii)* decrease in the mobility of charge carriers in semiconductors as temperature increases, which can affect the speed of integrated circuits.

3 METHODOLOGY

This section presents the key elements of our methodology. We start by describing the measures taken to limit noise in our environment (e.g., possibly introduced by measurement noise, scheduling decisions, and interference with OS tasks or other applications) (§3.1). We then present the results of our preliminary experiments that indicate that the collected data does *not* follow a Gaussian distribution. Therefore, we adopt an approach that leverages non-parametric statistical techniques that do not make any assumptions about the distribution of the underlying data, during data collection (§3.2) and analysis (§5). Key to this technique is the fact that the amount of data collected is itself data-dependent: we collect enough data until the estimate for the median fits in the desired confidence range (e.g., ±1% of the median) with 99% confidence.

3.1 Environment setup

To collect the benchmarks' runtime and power consumption data, we use a data collection methodology that reduces measurement bias (i.e., noise) such as interference with the OS or background applications. To this end, we do the following:

- *Core Isolation and Thread Affinity.* For any benchmark, we isolate four out of the six CPU cores on the Jetson TX2 to run benchmark. We also schedule the benchmark threads to the four cores and prevent the OS from migrating them during the characterization experiments. This ensures that the benchmarks run on dedicated CPU cores, without significant interference from the OS scheduler, or other applications.
- *Concurrent User-Space Applications.* We disable the GUI along with several non-essential services (e.g., containers service, calendar, system upgrade) to reduce the impact of other software components on the collected data.
- *I/O Operations and Remote Access.* We prevent remote access into the board during the characterization runs, and schedule the IO operations (e.g., loading data, saving results, progress notifier) to occur after the characterization data is collected. This ensures

that no network traffic or disk I/O operations will affect the benchmark results.

- *PyTorch-specific Measures.* We turn off PyTorch gradient computations (as we focus on the inference task) and turn on JIT (Just-in-Time) optimizations. This enables performance optimizations (e.g., improved runtime, and lower memory usage) for the target ML inference benchmarks.

3.2 Data Collection and Terminology

To collect power consumption and runtime data we:

- Reboot the board and run the target benchmark five times to avoid application-level and board-level cold start effects.
- We repeatedly perform the sequence of *(i)* sending the input from the CPU to the GPU, *(ii)* running the benchmark on the GPU, and *(iii)* sending the output result from the GPU to the CPU. We refer to this sequence as a *block*. We create blocks to eliminate the overhead associated with the CUDA timer (start and stop functions) and synchronization. We determine the size of each block based on two criteria: First, it must have at least 100 iterations, and, second, the CUDA runtime and synchronization overhead should be less than 0.1% of the block runtime. We determine the block size after the warm-up iterations.

Terminology: We define a *runtime observation* as the average benchmark runtime during a block, and a *power observation* as the power reading obtained from the onboard power monitors during the block execution. A collection of observations is a *sample*.

Testing for Normality of the Data. We use the two most powerful Normality tests [7]: Shapiro-Wilk, and Anderson-Darling. In both tests, the *null hypothesis* is that the input data sample belongs to the Gaussian distribution. We apply both tests to the runtime and power samples we collect.

Table 1 summarizes the results: at the 99% confidence level, both tests reject the *null hypothesis* for the majority of the data samples for both runtime and power consumption. Hence, there is no sufficient evidence to consider the collected data as Normally distributed, which leads us to *use non-parametric statistical techniques to analyze the data and draw accurate conclusions.*

Non-Parametric Statistical Technique to Estimate the Median. A common practice in empirical-based data analysis is to assume that the data belongs to a specific parametric distribution (usually Gaussian). This assumption makes it possible to use parametric data analysis and summarization metrics (e.g., mean, standard deviation, coefficient of variation, t-test) and limits the number of observations needed to estimate these metrics for a desired confidence. However, since our data is not normally distributed, we use non-parametric statistical analysis metrics (e.g., median, inter-quartile range, quartile-based coefficient of variation) and determine when to stop collecting data based on the values observed during collection (similar to [4]).

We define two stop criteria for collecting observations for a *sample*: *(i)* the sample must contain at least 50 observations, and *(ii)* the Relative Margin of Error (RME) for the median estimation is ≤ 0.5% with a confidence level of 99%.

Thus far, collecting one sample corresponds to one reboot of the node. This works well for power. However, similar to Mytkowicz et al. [13], we observe that there are differences in runtime between different boot states. These are due to differences in the memory layouts between different re-boots [3, 13]. Since characterizing a single boot state is not enough, we aim to estimate the median runtime across different reboot states and use the same non-parametric technique as above. Therefore, for runtime, after collecting a sample, we use its median as an observation corresponding to one boot state[1]. We then continue to gather such observations (i.e., one per reboot) and repeat this process until we can accurately estimate the median with RME ≤ 0.5% and a confidence level of 99%. This way, the task of gathering runtime data is time-consuming as a large number of reboots and board warm-ups are needed. On average, for one workload in one frequency setting (i.e., fixed temperature and frequency configuration), we collect a total of 6800 runtime observations within 91 board reboots.

Table 1: Shapiro-Wilk and Anderson-Darling normality tests' results. Each row depicts the percentage of samples for a benchmark, for which the null hypothesis (i.e., that the data is normally distributed) can be rejected at 99% confidence level. We run the tests on two boards: NVIDIA Jetson TX2 and AGX boards. For TX2, for each workload, each cell aggregates experiments at medium and high frequencies each at 25 and 70°C (see §4 for details). For AGX, each cell aggregates 455 unique frequency configurations combinations for the CPU, GPU, and memory controller all at ambient temperature for a total of 6370 samples.

Benchmark	Runtime		Power	
	Shapiro-Wilk 99%	Anderson-Darling 99%	Shapiro-Wilk 99%	Anderson-Darling 99%
Squeezenet	100	100	100	100
Mobilenetv2	100	50	100	100
Alexnet	50	50	100	100
Googlenet	100	100	100	100
Inception3	100	100	100	100
Resnet	100	100	100	100
Shufflenetv2	100	25	100	100
GeoMean: TX2	**91.94**	**67.29**	**100**	**100**
Squeezenet	78.60	73.72	89.34	99.98
Mobilenetv2	88.07	85.15	88.78	100
Alexnet	86.72	80.66	89.36	100
Googlenet	72.75	66.48	90.53	100
Inception3	68.93	62.10	92.97	100
Resnet	87.60	84.00	91.24	100
Shufflenetv2	89.45	85.98	84.62	100
GeoMean: AGX	**82.60**	**76.33**	**89.51**	**100**

4 EXPERIMENTAL SETUP

This section presents the main elements of our experimental setup: workload (§4.1), experimental platform (§4.2), and the temperature-controlled chamber (§4.3).

4.1 Workload

We use pre-trained Convolutional Neural Networks (CNNs) loaded via Torchvision v0.8 – a widely-used machine-vision package that is compatible with the PyTorch project and provides a variety of popular datasets, models, and image transformations for visual data processing. To ensure compatibility with the networks, we generate a shape-compatible input tensor for each network using randomly generated numbers, and use this input during the inference task. We study a total of seven classification networks: Alexnet, Googlenet, Resnet, Mobilenetv2, Squeezenet, Shufflenetv2, and Inception3.

[1]We verify that these observations are not normally distributed either.

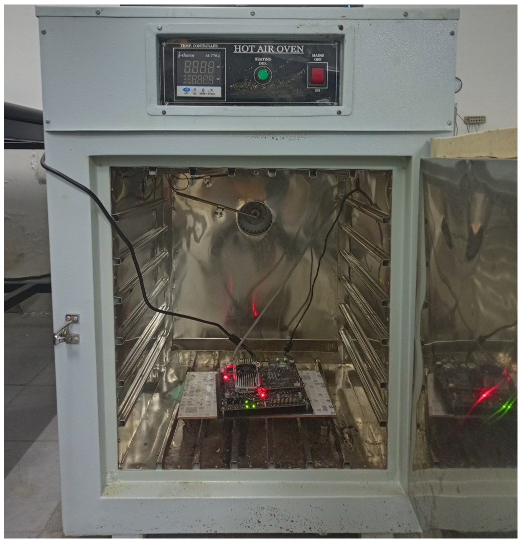

Figure 1: Temperature-controlled chamber used.

4.2 Hardware Platform

Hardware Platform. We use the NVIDIA Jetson TX2, a heterogeneous edge computing platform specifically designed to accelerate ML applications (Table 2). Its architecture comprises a heterogeneous combination of low-powered ARM CPU cores and the NVIDIA Pascal GPU, which share a unified memory module.

Frequency Configurations. The Jetson TX2, in contrast to recent Intel and AMD microarchitectures where power is the predominant resource and frequency is dynamically adjusted to fit within the power cap, features a manual governor (i.e., userspace mode) that grants greater control over the system's frequency configuration. To ensure frequency consistency throughout the experiment, we use this userspace governor mode to set the frequency configuration of the underlying components and ensure that it remains unchanged.

We use two frequency configuration settings dubbed *(i) high frequency* with CPU at 2.03GHz, GPU at 1.3GHz, and memory at 1.86GHz, and *(ii) medium frequency* with CPU at 1.11GHz, GPU at 0.73GHz, and memory at 0.66GHz. We exclude the low frequency settings, as they result in unacceptably long runtimes for most applications.

Software Stack. We use NVIDIA Jetpack v5.0.2, which has the latest Jetson Linux (v34.1.1) and CUDA v11.4. Our deep learning framework is PyTorch v1.6.

Table 2: Jetson TX2's hardware specifications.

Jetson TX2	
CPU	Dual-Core NVIDIA Denver 2 64-Bit CPU Quad-Core ARM® Cortex®-A57 MPCore
GPU	256-core NVIDIA Pascal™ GPU architecture with 256 NVIDIA CUDA cores
Memory Controller	8GB 128-bit LPDDR4 Memory 1866 MHx - 59.7 GB/s
Storage	32GB eMMC 5.1

Power, Runtime, and Temperature Monitors. To monitor the runtime, we use CUDA event timers, specifically the *CudaTimer* which is based on high-resolution onboard GPU counters. These counters are preferable for measuring GPU operations over CPU counters as they have lower latency and higher (sub-microsecond) resolution.

To measure power consumption, we used the onboard Power Measurement Unit (INA3221), which has an error of ±5%. We read power data with a frequency of 2Hz during the experiment. The PMU has different rails for reading the board's components' power. We use the aggregation of the first four power rails as the board's power consumption: CPU, GPU, SOC, and memory. The PMU is designed to operate correctly in the temperature range of our experiments.

To monitor the CPU and GPU temperature, we use the onboard sensors. The TX2 has eight thermal 'zones', each of which provides sensors with millidegrees Celsius sensitivity for a different component. We monitor the temperatures of the CPU and GPU (thermal zones 1 and 2) to ensure that their temperature did not exceed the thermal throttling point.

4.3 Temperature-Controlled Chamber

We use a temperature-controlled chamber (Figure 1) that can maintain a fixed temperature (±1°C error) within a temperature range of 20 to 100°C. A temperature-controlled chamber is a specialized environment that is used to test a product by replicating the temperature and humidity conditions experienced by it during its operation. The chamber uses forced air convection, to circulate air within the testing area to maintain a stable ambient temperature.

We evaluate the performance and power consumption of the board under varying ambient temperatures. To this end, we place the board within the chamber and establish the desired ambient temperature before initiating the characterization. We wait until two conditions are met: *(i)* the chamber's temperature sensor indicates that the temperature has become within ±1°C of the target temperature, and *(ii)* the board's temperature as reported by its internal sensors is within ±3°C of the ambient temperature.

Ambient Temperature. We use two ambient temperatures that are significantly different: 25 and 70°C. According to the Jetson manual, thermal throttling on the Jetson TX2 begins at 82°C. The fan runs at maximum speed throughout the experiments to keep the board's temperature close to the ambient temperature. We ensure that neither software nor hardware throttling occurs during the experiment by monitoring the onboard temperature sensors.

5 CHARACTERIZATION RESULTS

We characterize the impact of ambient temperature variation on power consumption and runtime.

Impact on Power Consumption. Figure 2 presents power consumption for each workload in the high (top plot) and medium (bottom plot) frequency settings at both low (left, grey boxplots) and high (right) temperatures. On average, we observe a **18.49%** increase in power consumption at 70°C compared to 25°C in the high-frequency configuration. The Shufflenetv2 workload shows the highest increase of **27.1%**. In the medium frequency setting, the average increase in power consumption is even higher at **21.60%**

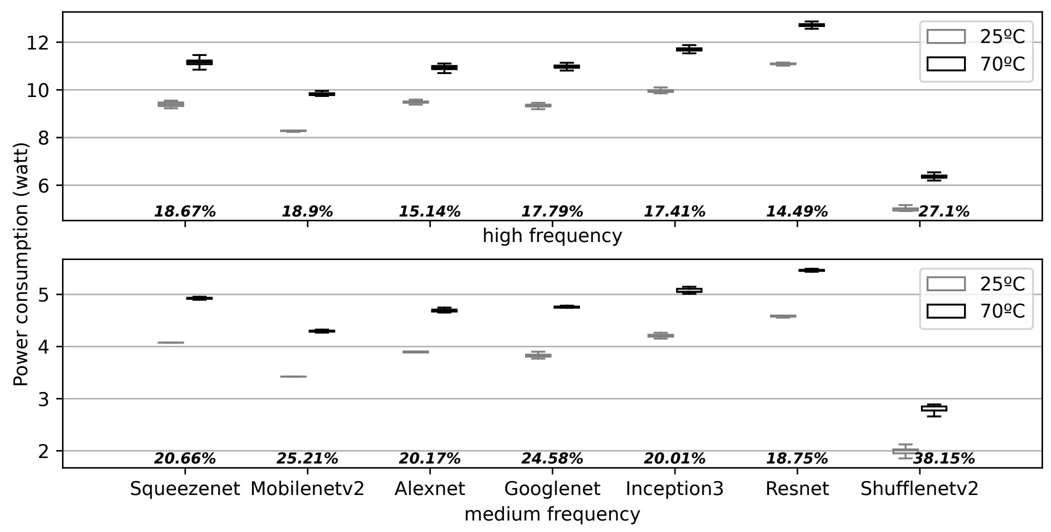

Figure 2: Boxplots present the distribution of measured power while executing different CNN inference workloads at 25 (left, grey boxplots) and 70°C (right) in the high (top plot) and medium (bottom) frequency configurations. Each boxplot presents the median, the top and bottom quartiles, and 5th/95th percentiles are shown by whiskers. Outliers were omitted for clarity. The differences (as %-tiles) between the medians for each model are shown on the x-axis. Lower values are better.

on average, with the Shufflenetv2 workload again showing the maximum increase of **38.15%**.

Impact on Runtime. The data collection process for runtime is time-consuming (see §3.2). Given the considerable time investment (i.e., several days) required to gather data, particularly in the medium frequency configuration where runs take longer, we have opted to focus our attention on the high frequency only[2]. Figure 3 shows that there is a noticeable increase in median runtime for all models in the high frequency setting at 70°C compared to 25°C. We observe an average increase of **2.59%** in median runtime, with a maximum of **5.15%** for Alexnet in the high temperature. We omit Inception3 (which had a runtime exceeding 55ms) from the plot for visual clarity, but our analysis of this model also showed a similar increase (1.7%) in runtime.

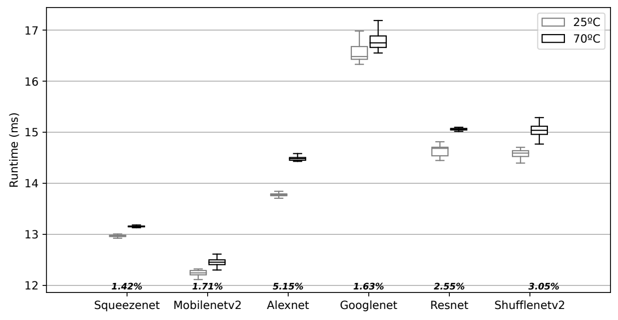

Figure 3: Boxplots present the distribution of the medians of the collected runtime samples for different CNN inference workloads at 25 (left, grey boxplots) and 70°C (right) in the high frequency configurations. Boxplots have the same meaning as in Figure 2. The differences (%) between the medians for each model are shown on the x-axis. Lower values are better.

6 SUMMARY AND DISCUSSION

Summary. Our results highlight the importance of taking ambient temperature into account when characterizing heterogeneous edge platforms. Our study indicates that raising the ambient temperature from office-like conditions (25°C) to direct sun exposure in a hot climate (70°C) has a significant impact on power consumption: about 20% on average across our machine learning inference benchmarks and frequency configurations and up to 38% in some cases. Under the same conditions, we also find a moderate (yet surprising) impact on runtime (up to 5%).

Discussion. The rest of this section continues by discussing a number of interrelated topics:

[D1] *What are the mechanisms that generate the observed impact of ambient temperature on power consumption?* This is a known phenomenon (see §2). The novelty our characterization brings is that it precisely quantifies the impact and demonstrates that it is sizeable.

We performed an additional experiment to shed some light on the contribution *static* and *dynamic* power share of the increased power consumption when increasing temperature. To this end, we determine the amount of power consumption increase with temperature with the board *idle* state (no load). We use the methodology presented in §3. Figure 4 shows a **62%** increase in idle power at 70°C compared to 25°C in the medium frequency (0.60 watts), while the increase in high frequency is **36.9%** (0.84 watts). On average, this increase accounts for **70%** of the total increase at full load in the medium frequency and (**53%** in the high frequency). Since most

[2]The data we collected on only a few of the benchmarks in medium frequency supports the conclusions we present here.

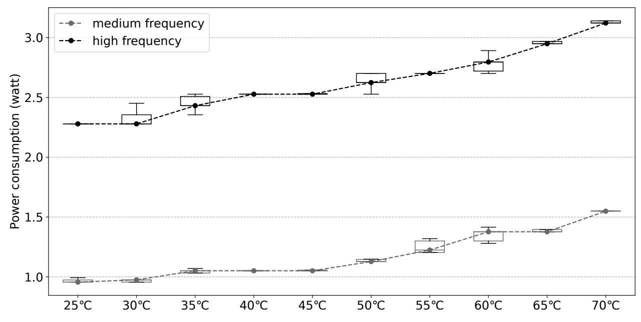

Figure 4: Boxplots present the distributions of the measured idle power consumption at different temperatures in the medium and high frequencies. Median values represented by circle markers are connected by the lines.

power directed to memory is background power (i.e., load independent) [14] we can infer that 30-50% of the power increase with temperature is due to *dynamic* power increase in the CPU and GPU. The rest is a combination of *static* power increase in CPU/GPU and power increase in the memory.

[D2] *What are the mechanisms that generate the observed impact of ambient temperature on performance?* We have attempted to understand whether a specific processing unit or the memory is the main culprit by using microbenchmarks (compute-intensive on CPU/GPU only, and memory intensive). The only instance where we observed a notable impact was a memory intensive benchmark with small buffer sizes (2% slowdown at 70°C).

[D3] *Do these differences matter from an application perspective?* While differences of 20-40% (for power) and even 2-5% (runtime) are generally seen as significant, temperature effects on power and runtime are often neglected when designing edge applications. To further demonstrate the significance of these effects, we focus on one scenario: *optimization frameworks* designed to meet the QoS requirements of edge applications by adjusting the frequency levels of edge platform's components [1, 2, 5, 6, 9, 10]. For example, for a deadline-constrained application, an optimization framework will aim to select a frequency configuration that fulfills the task deadlines with minimum energy consumption.

This is made possible by the wide range of frequencies supported by current edge platforms, leading to a range of 20-50x for performance/power consumption for the platform. The number of possible frequency configurations, however, is large (e.g., 1,768 for TX2 and 3,600 for AGX), and the optimal configuration cannot be found by a brute-force search at execution time. A number of *model-based* frameworks have been proposed to solve this problem [1, 2, 5, 6, 9, 10], but they are all temperature-oblivious even though their effectiveness depends on the accuracy of the models used (to estimate the impact of frequency configuration choice on the target workload). Our experiments with one of these temperature-oblivious frameworks, NeuOS [1] demonstrate this point: the framework is susceptible to multiple QoS violations in case of ambient temperature variations (see Figure 5 and its caption). Such violations in edge applications can have severe consequences

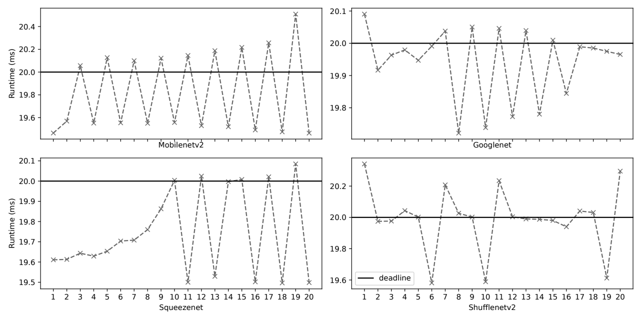

Figure 5: Plots present the application runtime of 20 consecutive inference tasks of different ML models at 70°C while using NeuOS with pre-populated 25°C characterization data as the frequency optimization framework. The QoS objective for all workloads is set to 50 FPS (i.e., 20 ms deadline). Plots present several instances of deadline violations on all workloads which are attributed to the temperature-oblivious nature of this framework.

such as an autonomous system failure, potentially leading to safety violations.

Acknowledgements: This work was partially supported by grants from Huawei and the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

[1] S. Bateni and C. Liu. 2020. Neuos: A latency-predictable multi-dimensional optimization framework for dnn-driven autonomous systems. In *USENIX ATC*. 371–385.
[2] B. Cox and J. Galjaard et al. 2021. Masa: Responsive Multi-DNN Inference on the Edge. In *IEEE PerCom*.
[3] C. Curtsinger and E. Berger. 2013. STABILIZER: Statistically Sound Performance Evaluation. *SIGARCH Comp. Arch. News*, 219–228.
[4] A. Maricq et al. 2018. Taming Performance Variability. In *USENIX OSDI*. 409–425.
[5] C. Imes et al. 2015. POET: a portable approach to minimizing energy under soft real-time constraints. In *IEEE RTAS*. 75–86.
[6] M. Han et al. 2019. MOSAIC: Heterogeneity-, Communication-, and Constraint-Aware Model Slicing and Execution for Accurate and Efficient Inference. In *IEEE PACT*. 165–177.
[7] N. Razali et al. 2011. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 21–33.
[8] P. Chestovich et al. 2022. Temperature Profiles Of Sunlight-Exposed Surfaces In A Desert Climate: Determining The Risk For Pavement Burns. *Journal of Burn Care & Research: Official Publication of the American Burn Association*, irac136–irac136.
[9] S. Bateni et al. 2018. PredJoule: A Timing-Predictable Energy Optimization Framework for Deep Neural Networks. In *IEEE RTSS*. 107–118.
[10] S. Han et al. 2016. Mcdnn: An approximation-based execution framework for deep stream processing under resource constraints. In *Mobile Systems, Applications, and Services Conference*. 123–136.
[11] S. Maity et al. 2021. Thermal-aware adaptive platform management for heterogeneous embedded systems. *ACM TECS*, 1–28.
[12] T. Baruah et al. 2018. Airavat: Improving energy efficiency of heterogeneous applications. In *DATE Conference*. 731–736.
[13] T. Mytkowicz et al. 2009. Producing wrong data without doing anything obviously wrong! *ACM Sigplan Notices*, 265–276.
[14] A. Karyakin and K. Salem. 2017. An analysis of memory power consumption in database systems. In *Workshop on Data Management on New Hardware*. 1–9.
[15] V. Lakshminarayanan and N. Sriraam. 2014. The effect of temperature on the reliability of electronic components. In *IEEE CONECCT*. 1–6.
[16] Y. Lee. 2021. Thermal-Aware Design and Management of Embedded Real-Time Systems. In *DATE Conference*. 1252–1255.