Snowflakes at the Edge: A Study of Variability among NVIDIA Jetson AGX Xavier Boards

Hazem A. Abdelhafez
Hassan Halawa
hazem@ece.ubc.ca
hhalawa@ece.ubc.ca
University of British Columbia
Vancouver, Canada

Karthik Pattabiraman
Matei Ripeanu
karthikp@ece.ubc.ca
matei@ece.ubc.ca
University of British Columbia
Vancouver, Canada

ABSTRACT

While applications deployed at the edge often rely on performance stability (or, at a minimum, on a predictable level of performance), variability at the edge remains a real problem [4]. This study uncovers a surprising source of variability: intrinsic variability (in performance and power consumption) among edge platforms that are nominally identical. We focus on a popular platform designed for edge applications, the NVIDIA Jetson AGX, and aim to answer the following high-level questions through rigorous statistical analysis: *(i)* are the edge devices in our study statistically different from each other in terms of applications' runtime performance and power draw (although they are sold under the same product model and family)?, *(ii)* if the differences between these edge devices are statistically significant, what is the magnitude of these differences?, and *(iii)* do these differences matter from the application's perspective?

CCS CONCEPTS

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

KEYWORDS

Performance and power variation, Edge computing, Jetson AGX.

ACM Reference Format:
Hazem A. Abdelhafez, Hassan Halawa, Karthik Pattabiraman, and Matei Ripeanu. 2021. Snowflakes at the Edge: A Study of Variability among NVIDIA Jetson AGX Xavier Boards . In *4th International Workshop on Edge Systems, Analytics and Networking (EdgeSys'21), April 26, 2021, Online, United Kingdom.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3434770.3459729

1 INTRODUCTION

To maximize the energy efficiency for the applications running on edge devices (some of which are battery powered), recent architectures employ both processing units (PUs) and memory that are configurable at runtime (e.g., by changing the operational frequencies or the number of active processing cores). To the same end, there is an increasing use of heterogeneous architectures where each processing unit is specialized for a certain type of workload (e.g., massively parallel, deep learning, and computer vision accelerators), and offers high energy efficiency. The flexibility of these architectures, however, comes at the cost of increased complexity for software developers who must manually target and tune the specific processing units on which to run their workload, while meeting their application's quality of service (QoS) objectives.

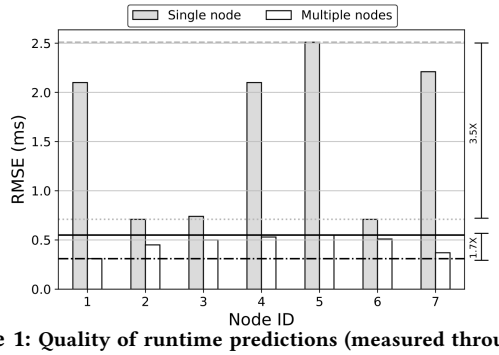


Figure 1: Quality of runtime predictions (measured through root mean square error - RMSE) across multiple NVIDIA Jetson AGX boards that are nominally identical. The models we present are an ideal case for a 'best-fit' model (i.e., lookup table), and compare: *(i)* training on a single node (shown in grey) which does not account for the *intrinsic* inter-node variability, and *(ii)* training over multiple nodes (shown in white) which takes into account inter-node variability. Note that in both cases the nodes we train on and those we test on are distinct. Two aspects are worth noting: *(i)* the average quality of the predictions; and *(ii)* the variability in the quality of the predictions across deployment nodes. All models aim to predict the runtime of a 2D convolution kernel (frequently used by convolutional neural networks) and are evaluated over $\approx 10K$ points (hardware platform configurations, input shapes, and kernel parameters). The x-axis is the ID of the deployment node (i.e., held-out test node).

Our long-term objective is to automate this process. To this end, the first step is to accurately predict the impact of workload placement and hardware tuning (e.g., frequency choice) on application performance and power consumption. This will enable: *(i)* well-informed deployment decisions (e.g., compute and battery capacity provisioning, or cost estimation), and *(ii)* optimizations that meet QoS objectives while minimizing the energy footprint (e.g., QoS-aware dynamic voltage and frequency scaling).

When we started developing such a prediction model, however, we obtained a surprising (to us) result: the quality of the predictions offered by the model varied significantly depending on the hardware platform where it was deployed, even if the platforms were

EdgeSys'21, April 26, 2021, Online,United Kingdom

ACM ISBN 978-1-4503-8291-5/21/04…$15.00
https://doi.org/10.1145/3434770.3459729

nominally identical. The grey bars in Fig. 1 present this experiment. Furthermore, despite trying our best to eliminate all the potential causes of the observed variability (e.g., operating system image, interference with other OS services, thermal throttling, as described in Section 3.6), the results remained unchanged.

This leads us to the central hypothesis of this paper: *there are significant intrinsic differences among the hardware platforms used*, and these are the root cause of the variability we observed. While hardware variability is well known (e.g., stemming from multiple factors such as process variation, and imperfections in the manufacturing process), the common expectation is that under the *same hardware model*, manufacturers offer products that are within a narrow tolerance band. However, we find that this is not the case.

Objective. Our goal is to identify and quantify the intrinsic variability between different edge hardware platforms that are offered as the same model. We use rigorous statistical techniques, and attempt to answer the following high-level research questions (RQs):

- **RQ1:** Are the hardware platforms we study (NVIDIA Jetson AGX) statistically different in terms of the applications' runtime performance and power draw? (Section. 5.1)
- **RQ2:** If the hardware platforms are statistically different, what is the magnitude of these differences? (Section. 5.2)
- **RQ3:** Do these differences among the platforms matter from an application's perspective? (Section. 5.3)

Challenges. There are two main challenges in characterizing the intrinsic variability of edge platforms. First, the configuration space to explore is huge. This stems from two factors: *(i)* a large number of hardware components that can be tuned (e.g., processing units, memory controller), and *(ii)* a wide range of available configurations for each component (e.g., configurable frequencies cover an over 10x frequency range, with over a dozen levels for each component). This makes it difficult to brute-force the search to measure the metrics of interest such as the runtime and the average power consumption for a particular workload. The second challenge is the need for a careful experimental design to exclude, to the extent possible, other potential sources of variability such as the: *(i)* (in)accuracy of measurement tools (e.g., software for runtime, and hardware for power monitoring), *(ii)* differences in software libraries and drivers, *(iii)* interference from background activities run by the operating system, and *(iv)* fan's impact and thermal throttling.

Contribution. The main contribution of this paper is to highlight and characterize hardware variability at the edge. We use the popular NVIDIA Jetson AGX platform as a case study, and investigate the performance and power consumption behavior of 13 identical AGX boards for several machine learning (ML) inference workloads under the same conditions (i.e., input parameters and frequency configurations). We demonstrate the presence of inter-node variability, for both runtime and power consumption. We present the first quantitative analysis of this variability (to our knowledge) for the NVIDIA Jetson AGX platform using rigorous statistical techniques. Finally, we present a generic, platform-neutral, statistical methodology for studying variability on computing platforms.

2 VARIABILITY IN COMPUTING SYSTEMS

Classifying variability. One way to classify variability is by its source: *(i)* software (e.g., cache contention, concurrent applications' interference, OS activities) [3, 6], *(ii)* hardware (e.g., process and manufacturing variations for components - such as processors, memory modules, and storage) [10, 14], and *(iii)* environmental factors such as thermal conditions [11].

Hardware variability. From the users' perspective (e.g., application developers), there is an expectation that any group of units belonging to the same hardware platform and running the same software should exhibit similar performance and power draw behavior. The hardware manufacturers realize this expectation, and, at the same time, understand that inter-node variability arises due to several sources (discussed above), some of which are unavoidable. Hence, they attempt to address this via different methods, such as: *(i)* increasing design guardbands on microchips [14] which increases the cost, *(ii)* grouping nodes with similar properties in classes of nodes labeled differently (e.g., for marketing purposes), and *(iii)* documenting and informing the users about the performance/power deviations through data sheets and technical specifications.

Implications of ignoring variability. Variability, among other factors, imposes challenges on building generic models for accurately predicting the performance and power consumption of a hardware platform, using data collected from a single device to make predictions on any other device of the same type [10].

In the mobile computing domain, Wu et al. [6] show a significant inference performance variability across mobile devices of the same/different model, and draw attention to the risk of poor user experience if deployments are optimized for the *average* observed performance. However, unlike us, they do not focus on variability among devices of the same model, nor do they present a systematic approach to characterize and quantify the variability.

In high-performance computing (HPC) environments, large-scale applications typically work in a lock-step fashion using synchronization barriers. Thus, any significant imbalance due to performance variability amongst machines (assumed to deliver the same performance) is detrimental to application performance [8].

Addressing variability. *The first step towards addressing variability is to identify and quantify it.* For example, Weisbach et al. [8] propose a set of benchmarks based on a lightweight kernel to characterize hardware performance variability in HPC systems. Kocoloski and Lange [5] propose Varbench as a set of benchmarks to precisely measure performance variability's impact on Bulk Synchronous Parallel (BSP) applications. *The second step is to incorporate variability in the design and development process.* One way to achieve this is to build probabilistic models (e.g., for runtime estimation) that take variability into account to achieve better QoS guarantees [3]. Another approach is to dynamically adapt the runtime environment (e.g., scale processors' frequencies), or re-balance workload [5] to handle variability. A more conservative approach is to optimize the system based on the worst case (i.e., deploying less accurate models to meet QoS objectives in 95% of the cases [6]).

3 METHODOLOGY

Our objective is to identify and quantify variability in performance and power consumption between several NVIDIA Jetson AGX boards (that are fully identical: hardware and software wise). This section first presents the key statistical background, and our design choices to answer the research questions we highlight in the introduction (Sections 3.1, 3.2, 3.3). Then it presents the workload we

use (Section 3.4), our solution to deal with the large configuration space (Section 3.5), and finally, our measures to eliminate other sources of variability (Section 3.6).

Terminology: A *sample* is a set of *observations* (i.e., measurements for runtime / power at one configuration point for one board).

3.1 RQ1: Are the boards statistically different?

To answer this question, we leverage *statistical significance tests*. These tests are generally used in two contexts: *(i)* to determine if a sample from a certain population belongs to a specific parametric distribution [13] (i.e., goodness-of-fit test), or *(ii)* to determine whether samples drawn from different populations belong to the same distribution. We are interested in the latter goal: for a chosen configuration point, we extract a sample (runtime or power) on each board and use the statistical significance tests to understand whether these samples come from the same distribution.

Statistical significance tests often calculate the maximum distance between the cumulative distribution functions of the samples provided. Then, based on this distance, the tests return a specific value from which a significance level (the *P-value*) is calculated. For each test, there is a *null hypothesis* (e.g., that the provided samples belong to the same distribution). If the test returns a P-value that is lower than a specific threshold, typically set to 1% (5%), then we can reject the null hypothesis with high confidence, i.e., 99% (95%).

We use two statistical significance tests: *(i)* K-samples Anderson Darling (AD) [7], and *(ii)* two-samples Kolmogorov-Smirnov (KS) [16]. First, the K-samples Anderson-Darling test [7, 13], allows testing on multiple nodes at a time (i.e., the null hypothesis in this case is that *all K* samples drawn from the K nodes belong to the same distribution). Second, the two-samples KS test [16] allows testing for only two samples (i.e., do samples drawn from nodes A and B belong to the same distribution?).

3.2 RQ2: How significant is the difference between boards?

Using the statistical significance tests we can answer whether, for a specific configuration point, boards are statistically different in terms of performance or power (i.e., a 'yes/no' question). However, statistical significance tests do not indicate the magnitude of the difference (if any); i.e., they do not indicate the *effect size* [17].

There are several techniques to estimate the effect size. One of the most popular is the *Cohen's d* [1] test. It measures the effect size as the difference between two samples' means divided by their pooled standard deviations. This technique, however, makes two assumptions about the underlying distributions as follows: *(i)* both samples are drawn from normal distributions, and *(ii)* these distributions have the same variance. However, because we do not know if these assumptions hold for our case, we use the *scaled robust d (d_r)* test [9], a modification of *Cohen's d* that is shown to be robust under violations of these assumptions [17]. Similar to *Cohen's d*, the *scaled robust d (d_r)* compares the differences between the means of two samples relative to their pooled standard deviation. More intuitively, a large d_r value indicates that the means of the two sets are far apart relative to the observed variation.

We later use this estimate of the distance between distributions in two ways: first, in a relative way, we use it to characterize differences between all the nodes pairs (at various configuration points) and show there is a large diversity of differences between nodes. Second, we use it in an absolute way, to study how many node pairs are significantly dissimilar - by choosing a threshold for d_r.

3.3 RQ3: Does the observed variability matter from an application's perspective?

While different boards may have statistically different performance (or power draw) characteristics, and we can quantify these differences by comparing them with the variability we see in the measurement samples, we do not know if these differences are important from an application perspective. While this question can only be unequivocally answered in an application context, we provide two data points to build some intuition about the impact of these differences. First, we train a runtime predictor that takes inter-node variability into account, and demonstrate that the quality of its predictions is better and more uniform than that of a predictor trained on a single node (Fig.1 and Section 5.3). Second, we present finer granularity statistics of experimental data that we collected. These statistics allow an application developer to form a preliminary opinion on whether the observed variability would have an impact in their application context (Section 5.3).

3.4 The Workload

We focus on ML *inference* models based on convolutional neural networks (CNNs). CNNs are one of the most widely used ML models. They have applications in image and speech analysis, and are primarily applied to computer vision related tasks such as object recognition and classification. Such applications are often deployed at the edge [6, 12], thus inference models based on CNNs are a good application-specific choice to benchmark the variability of edge platforms. Section 4 presents more details about the CNNs we use.

3.5 Dealing with the large configuration space

On the AGX board, there are $\approx 3.6K$ unique combinations (i.e., configurations) of CPU, GPU, and memory controller frequencies (Section 4 provides more details). We use sampling to reduce the space by almost one order of magnitude: we sort each component's (e.g., CPU, memory controller) frequency range, and sample every other frequency. This sampling scheme reduces the space to 525 configurations, yet covers most of the frequency range of each component.

3.6 Eliminating other sources of variability

Timing measurements. We use the CUDA Events API [20] for precise timing measurements of the networks' inference time (0.5µs resolution). For each network, on a per configuration basis, we collect timing measurements of 105 iterations and discard the first five iterations (i.e., warm up iterations). The remaining 100 measurements are the array of 100 observations that make up a sample. For selected configurations we confirmed that the results do not change if samples are much larger and include 10,000 observations.

Power measurements. We use the two on-board INA3221 [19] (0.5% error) Power Monitoring Units (PMU) that can be read via an exposed virtual file system (sysfs). PMUs have six power 'rails' (for CPU, GPU, memory module, computer vision (CV) accelerator, auxiliary on-chip components, system IO) which measure the power each component draws. We discard the CV accelerator and system IO rails as they are not relevant to our experiments. We sample

the power sensors with $0.5Hz$ sampling rate. We run five warm up runs before measuring power, and discard the first two power measurements (equivalent to one second). After this warm up, we run each network for 30 seconds (to collect 60 observations). We also note that the boards are placed within the same physical rack, hence, they are all in an environment with the same ambient temperature.

Other steps to reduce variability. We create a power profile using the NVIDIA *nvpmodel* tool that is supplied with the Jetson AGX software stack. This profile fixes the frequencies of the other unused board components (e.g., CV and deep-learning accelerators), ensures that all CPU cores and GPU Texture Processing Clusters (TPC) are *on* all the time, and disables the power gating mechanism that turns some cores *off* when idle. We also disabled the system's default DVFS functionality and set all components to the *userspace* governor (which allows us to set the frequencies manually). Moreover, we disable all non-essential OS services running on the boards and we prevent remote access. Finally, we use the same software stack across all nodes as detailed in Section 4.

Thermal throttling. According to the latest NVIDIA Jetson AGX thermal design guide [18], the maximum operating temperature limits (to operate without performance reduction) for the CPU, GPU, and other components are 86, 88, and 82°C. Above these temperatures software or hardware throttling will reduce runtime frequencies to avoid overheating, thus reducing the board's performance, and impacting the reliability of results. One way to eliminate the performance variability due to throttling is to operate the fan at the maximum speed all the time (it usually starts operating automatically at ≈50°C). We also monitor the on-board temperature sensors to make sure the board does not enter any thermal throttling zones (i.e., the temperature is always below 50°C during the experiments).

4 EXPERIMENTAL SETUP

The hardware platform: In this paper, we focus on the NVIDIA Jetson AGX, a popular edge platform. It combines NVIDIA's state-of-the-art GPU technology with low-powered ARM CPU cores in a shared-memory architecture to deliver massive compute power and high energy efficiency in a tiny physical footprint.

The most important feature of the AGX in our context is that it provides a wide (10x) range of frequencies for the main processing elements and the memory controller. Each CPU core's operating frequency can range from 115.2MHz to 2.265GHz at a fine granularity (with 29 supported CPU frequency levels). The Volta-based GPU has 512 CUDA cores, and 64 Tensor cores. The CUDA cores support dynamic frequency scaling between 114.75MHz and 1.377GHz at a fine granularity (with 14 supported core frequency levels). Finally, the AGX allows dynamic frequency scaling for the External Memory Controller (EMC) - between 204MHz and 2133MHz (with 9 supported memory frequency levels). The frequency state space of these three components has a total of $\approx 3.6K$ combinations.

Software stack: NVIDIA JetPack. We use the NVIDIA JetPack SDK version 4.4 across all boards. It includes the latest Linux OS for Tegra and driver package (L4T v32.4.6) for the Jetson platform. It also incorporates a full set of optimized software libraries (e.g., CUDA v10.2) to build applications targeting the various on-board PUs (e.g., GPU, deep learning accelerators, computer vision).

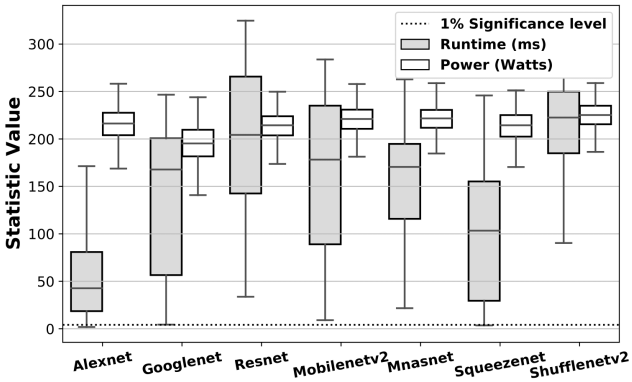


Figure 2: K-samples Anderson-Darling test for the inference task of seven different networks. For each box plot, the top and bottom sides represent the first and third quartiles (Q1 and Q3). The horizontal line represents the median (Q2). The bottom/upper fences limit the included values to 1.5*(Q3-Q1) distance from Q1/Q3 respectively. The outliers are omitted for legibility. The dashed horizontal line indicates the critical values for the 1% significance level.

Machine learning: PyTorch framework. We use PyTorch [2], version 1.6, as it is one of the most widely used deep learning frameworks. It allows developers and researchers to build deep learning networks, mainly using the Python programming language. The core functionality of PyTorch is implemented in C++ and exposed as Python C++ extensions to the users (i.e., LibTorch).

Machine learning: Pre-trained CNNs. We use Torchvision [15], version 0.8, to load the pretrained CNNs used in the experiments. Torchvision is a popular machine-vision package - compatible with and part of the PyTorch project - that encompasses popular data sets, models, and image transformations to process visual data. All classification CNNs included in Torchvision are trained and optimized on the 1000-class ImageNet dataset. Each vision network accepts inputs of a specific size (i.e., 4D Tensor). We generate a shape-compatible input tensor for each network (with randomly generated numbers), and use it during the inference task. We study ten different classification networks from Torchvision: AlexNet, GoogleNet, ResNet, MobilenetV2, MNASNet, SqueezeNet, ShuffleNetV2, DenseNet161, VGG16, and InceptionV3. We only include the results of six or seven networks for space reasons; however the results for the omitted networks are similar to the ones presented.

5 RESULTS AND ANALYSIS

For each CNN model, we collected one sample for the runtime, and one sample for power, at each configuration point, on each each board. This is about 100,000 samples and 8 million observations.

5.1 RQ1: Are the boards statistically different?

K-samples Anderson-Darling test. For each configuration and CNN model, we run the K-samples Anderson-Darling test (Figure 2). Each box plot in the figure represents the statistical summary (see legend) for runtime (or power) of multiple configuration points for a single CNN. The null hypothesis is that *all* boards are similar. To reject the null hypothesis, at the 1% and 5% significance levels, the critical values reported by the test have to be higher than ≈4.11 and 3.22 respectively. We find that, at the 1% significance level, the null

Table 1: Percentages of tests in which the p-values indicate that the null hypothesis can be rejected (i.e., samples do *not* belong to the same distribution) for two different significance levels (α).

	α	Alex Net	Google Net	Res Net	Mobile Netv2	Mnas Net	Squeeze Net	Shuffle Netv2	Mean
Runtime	1%	52.34	80.31	87.05	79.42	83.68	67.31	89.85	77.14
	5%	60.48	85.03	89.94	83.69	87.26	73.58	92.35	81.76
Power	1%	99.73	99.49	99.54	99.6	99.6	99.69	99.36	99.57
	5%	99.82	99.69	99.68	99.69	99.73	99.83	99.45	99.7

hypothesis can be rejected for 99.9% and 100% of the configurations sampled for the runtime and power consumption respectively.

Two-samples Kolmogorov-Smirnov test. The previous test indicates that, with high confidence, at least one board is different (for all configuration points). However, this could be simply because we have a single defective board, and does not necessarily imply the boards are all different from each other. To investigate this hypothesis, we turn to the Two-samples Kolmogorov-Smirnov test where we compare the boards pairwise. For each configuration and CNN, and for each board pair, we run the two-samples Kolmogorov-Smirnov test. Table 1 shows the percentages of tests where the *null hypothesis* (i.e., the samples belong to the *same* distribution) can be rejected. On average, across the seven CNNs, the results indicate that the null hypothesis can be rejected at the 1% significance level for ≈77% and ≈99% of the configurations sampled for runtime and power consumption respectively. This suggests that the performance and power consumption behavior are different across most boards pairs (recall that they are all sold as the same model).

5.2 RQ2: How large is the difference between the boards?

So far we have gathered statistical evidence that indicates that the boards are statistically different. We now aim to quantify this difference. Fig. 3 shows the distribution of the d_r values for the seven CNNs (see Section 3.2 for the definition of d_r). The figure highlights that there is significant spread in the d_r values we observe (particularly for power). These results imply that the difference in the distribution of the samples and its parameters (i.e., means, variances), obtained from two different boards, can vary significantly.

Cohen provided a guideline for interpreting d values where ±0.25, ±0.5, and ±1.0 values represent small, medium, and large effect sizes, respectively. With this interpretation, on average across all CNNs, for 49% of board pairs and configurations we observe *large* differences for runtime (and 90% for power). Even with a more conservative threshold of ±2.0, *large* differences are observed, on average, about 27% for runtime (and 82% for power).

5.3 RQ3: Does the observed variability matter from an application's perspective?

One drawback of effect-size measures such as Cohen's d, and scaled robust d is that interpreting the obtained values is domain specific, and may appear to be ad-hoc. Thus, we instead examine whether there are large differences between boards from an *application-level* perspective. We offer two data-points. First, our experiment presented in Fig. 1, indicates the high impact of inter-node variability for runtime predictions (≈20% to ≈80% reduction in the RMSEs).

Second, Figures 4a (runtime) and 4b (power) show three frequency configurations (lowest, midst, and highest frequency). For each of configuration and CNN, we group the observations per board, and attempt to characterize how much the observations vary

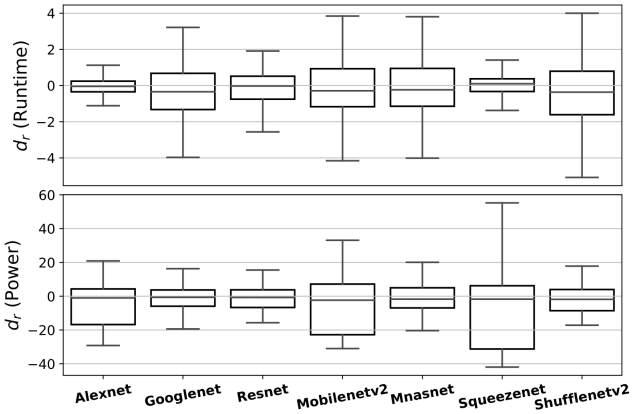


Figure 3: Pairwise d_r values distribution across six networks. The guideline for interpreting the effect size is: ±0.25, ±0.5, and ±1.0 representing small, medium, and high effect sizes respectively. The boxplots have the same meaning as in Fig. 2. The sample size is 10K for the runtime, and 6K for the power consumption.

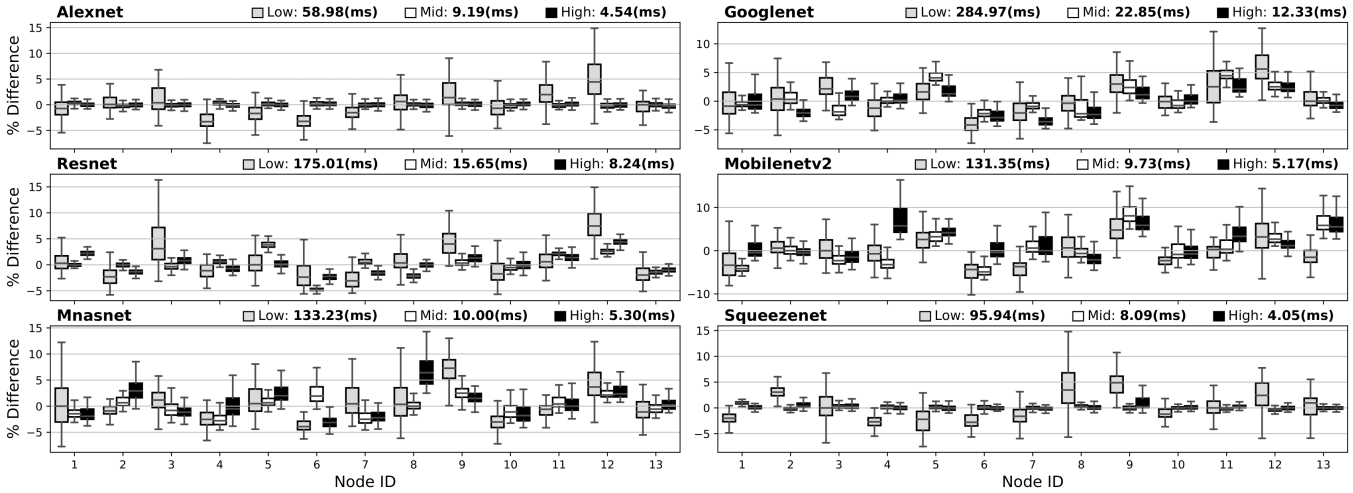
relative to the mean across all boards (the D_i box plots are constructed similar to those in Fig. 2). For the runtime, the D_i values fall mostly within the ±5% range of each other. However, in some cases (e.g., ResNet, MobileNetV2, ShuffleNetV2) D_i the range extends from -10% to 20% (for runtimes that are 10s to 100s of miliseconds). For the power consumption, most of the D_i values fall within the ±10% range, but in some cases (e.g., GoogleNet, MNASNet, ShuffleNetV2) variations extends from -30% to 20% range (for power drawn between a few watts to 10s of watts). Variability at this scale has been identified as an issue with a critical negative impact on the user experience of real-world mobile inference workloads [6].

The two presented data points suggest that the observed inter-node variability matters at the application-level, and this has key implications for the (runtime/power draw) model we plan to develop (as mentioned in the introduction): *(i)* the model must be exposed to this intrinsic variability during training to reduce prediction errors (i.e., samples must be collected across multiple boards not just from a single one), and *(ii)* the model's performance will be bounded by this intrinsic variability regardless of how finely tuned it is, or how many samples are used for training (i.e., the observed range of this intrinsic variability is the floor for the model's error).

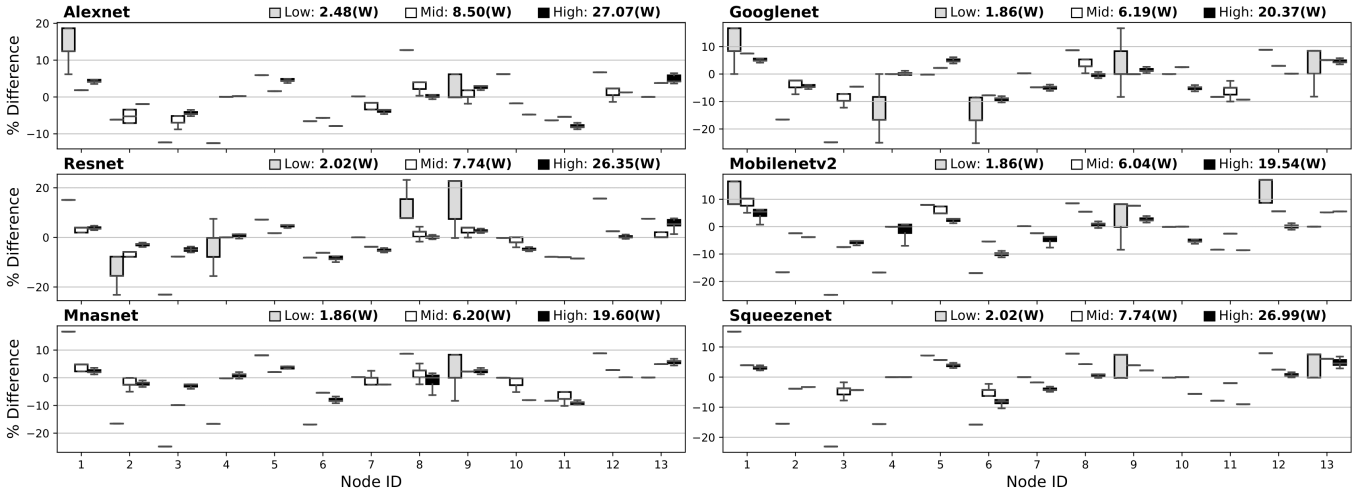
6 SUMMARY AND FUTURE WORK

Summary. We find that there is an inherent variability in both the performance and power consumption between different hardware (AGX) boards of the same model. The experimental data suggests that the differences are significant enough to lead to application-level impact, e.g., a performance or power prediction model trained on an individual board will have wide variation in the quality of its predictions when deployed over nominally similar boards. This leads to an important implication on our long-term objective of developing a runtime/power model for inference tasks: *we need to collect samples from different boards to train the runtime and power prediction models so that they are exposed to this variability.*

Future Work. Our results open up a number of intriguing questions, which we plan to explore in more detail in the near future: *(i)* is there a correlation between some parts of the configuration



(a) D_i **values for the runtime.**



(b) D_i **values for the power consumption.**

Figure 4: D_i **values for runtime and power consumption for six deep learning networks. The x-axis shows the node ID (AGX board identifier), and the y-axis indicate the** D_i **values. The three box plots per node represent the lowest, midst, and highest frequency configurations (colored gray, white, and black respectively). The legend on top of each plot shows the absolute value (ms) for the mean of means for each configuration.**

space and higher/lower inter-node variability?, *(ii)* is the change in variability smooth or abrupt across the configuration space?, and *(iii)* is variability also an issue for the other available specialized processing units (e.g., vision and deep learning accelerators)?

7 ACKNOWLEDGMENTS

This project was sponsored in part by the Natural Sciences and Engineering Research Council of Canada (NSERC), and a gift from Huawei's Compilers and Programming Languages Lab in Canada. We also thank the reviewers of EdgeSys'21 for their comments.

REFERENCES

[1] J. Cohen. 2013. *Statistical power analysis for the behavioral sciences.* A. Press.
[2] A. Paszke et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. arXiv:1912.01703 [cs.LG]
[3] B. Gaudette et al. 2016. Improving smartphone user experience by balancing performance and energy with probabilistic QoS guarantee. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 52–63.
[4] B. Gaudette et al. 2019. Optimizing User Satisfaction of Mobile Workloads Subject to Various Sources of Uncertainties. *IEEE Transactions on Mobile Computing* 18, 12 (2019), 2941–2953. https://doi.org/10.1109/TMC.2018.2883619
[5] B. Kocoloski et al. 2018. Varbench: an Experimental Framework to Measure and Characterize Performance Variability. In *Proceedings of the 47th International Conference on Parallel Processing*. 1–10.
[6] C Wu et al. 2019. Machine learning at facebook: Understanding inference at the edge. In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 331–344.
[7] F. Scholz et al. 1987. K-Sample Anderson–Darling Tests. *J. Amer. Statist. Assoc.* 82, 399 (1987), 918–924. https://doi.org/10.1080/01621459.1987.10478517
[8] H. Weisbach et al. 2018. Hardware Performance Variation: A Comparative Study Using Lightweight Kernels. In *High Performance Computing*. Springer International Publishing, Cham, 246–265.
[9] J. Algina et al. 2005. An alternative to Cohen's standardized mean difference effect size: a robust parameter and confidence interval in the two independent groups case. *Psychological methods* 10, 3 (2005), 317.
[10] J. McCullough et al. 2011. Evaluating the effectiveness of model-based power characterization. In *USENIX Annual Technical Conf*, Vol. 20.
[11] L. Wanner et al. 2013. Hardware Variability-Aware Duty Cycling for Embedded Sensors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21, 6 (2013), 1000–1012. https://doi.org/10.1109/TVLSI.2012.2203325

[12] M. G. Sarwar Murshed et al. 2020. Machine Learning at the Network Edge: A Survey. arXiv:1908.00080 [cs.LG]

[13] N. Razali et al. 2011. Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics* 2, 1 (2011), 21–33.

[14] P. Gupta et al. 2012. Underdesigned and opportunistic computing in presence of hardware variability. *IEEE Transactions on Computer-Aided Design of integrated circuits and systems* 32, 1 (2012), 8–23.

[15] S. Marcel et al. 2010. Torchvision the Machine-Vision Package of Torch. In *Proceedings of the 18th ACM International Conference on Multimedia* (Firenze, Italy) *(MM '10)*. Association for Computing Machinery, New York, NY, USA, 1485–1488. https://doi.org/10.1145/1873951.1874254

[16] J. Hodges. 1958. The significance probability of the Smirnov two-sample test. *Arkiv för Matematik* 3, 5 (1958), 469–486.

[17] J. Li. 2016. Effect size measures in a two-independent-samples case with non-normal and nonhomogeneous data. *Behavior research methods* 48, 4 (2016), 1560–1574.

[18] NVIDIA 2020. *Jetson AGX Xavier Series: Thermal Design Guide*. NVIDIA. Retrieved February, 2021 from https://tinyurl.com/r7zeehya

[19] Texas Instruments 2016. *INA3221 Power Monitors*. Texas Instruments. Retrieved January, 2021 from https://www.ti.com/product/INA3221

[20] The PyTorch Team. 2021. *CUDA Timing Events*. NVIDIA. Retrieved January, 2021 from https://pytorch.org/docs/stable/cuda.html#torch.cuda.Event