Analyzing the Performance of Smart Industry 4.0 Applications on Cloud Computing Systems

Razin Farhan Hussain[1], **Alireza Pakravan**[2], **Mohsen Amini Salehi**[1]
[1]High Performance Cloud Computing (HPCC) Laboratory, University of Louisiana at Lafayette, LA, USA
[2]California State University, San Marcos, CA, USA
Email:[1]{razinfarhan.hussain1,amini}@louisiana.edu, [2]apakravan@csusm.edu

***Abstract*—Cloud-based Deep Neural Network (DNN) applications that make latency-sensitive inference are becoming an indispensable part of Industry 4.0. Due to the multi-tenancy and resource heterogeneity, both inherent to the cloud computing environments, the inference time of DNN-based applications are stochastic. Such stochasticity, if not captured, can potentially lead to low Quality of Service (QoS) or even a disaster in critical sectors, such as Oil and Gas industry. To make Industry 4.0 robust, solution architects and researchers need to understand the behavior of DNN-based applications and capture the stochasticity exists in their inference times. Accordingly, in this study, we provide a descriptive analysis of the inference time from two perspectives. First, we perform an application-centric analysis and statistically model the execution time of four categorically different DNN applications on both Amazon and Chameleon clouds. Second, we take a resource-centric approach and analyze a rate-based metric in form of Million Instruction Per Second (MIPS) for heterogeneous machines in the cloud. This non-parametric modeling, achieved via Jackknife and Bootstrap re-sampling methods, provides the confidence interval of MIPS for heterogeneous cloud machines. The findings of this research can be helpful for researchers and cloud solution architects to develop solutions that are robust against the stochastic nature of the inference time of DNN applications in the cloud and can offer a higher QoS to their users and avoid unintended outcomes.**

***Index Terms*—Deep Neural Network Applications, Industry 4.0, Cloud Platform, Heterogeneous Machines, Inference Time**

I. Introduction

Software solutions operating based on machine learning and, particularly, Deep Neural Network (DNN) models are becoming fundamental pillars of Industry 4.0 revolution [1]. In the industrial automation process, numerous smart sensors frequently produce and fed data to the DNN-based applications that can make smart latency-sensitive decisions to improve energy efficiency, production, and safety measures. Building robust Industry 4.0 solutions entail having an accurate estimation of the inference (execution) time of DNN-based applications hosted on the cloud or edge computing systems. Lack of such assessments often leads to missing the applications' latency constraints and lowers their quality of service (QoS) [2] or increase the incurred cost of cloud resources. In critical industrial sectors, such as oil and gas, the penalty of such inaccurate estimations can be disastrous and cause unintended consequences, such as an unsafe workplace, environmental footprints, energy wastage, and damaging devices [3], [4]. Accordingly, our goal in this study is to measure and model the stochasticity that exists in the execution time of industrial DNN-based applications that are commonly deployed in the cloud.

Our motivation in this study is the critical industry of Oil and Gas (O&G) that is aimed at becoming clean and ultimately unmanned, thereby safe, in Industry 4.0. O&G is one of the main environmental pollutants and even minor improvements in this industry can have major impacts in the global scale. In this context, there are several time-sensitive operations (*e.g.,* fire detection [5] and toxic gas monitoring [6]) that failing to timely process them can potentially lead to a disasters, scuh as oil spill, explosion, and even death. Understanding the uncertainties exist in execution time of different application types and properly modeling them is crucial in architecting software solutions that are robust against these uncertainties. Note that DNN-based applications encompass both the training and inference stages [7]. While the training stage is generally carried out offline, our focus in this study is on modeling the inference execution time that has to be accurately estimated for latency-sensitive and mission critical applications [8]. For instance, accurate estimation of inference time is instrumental in calculating the completion time of arriving tasks that can, in turn, help to make more precise resource allocation decisions [9].

Public or private Cloud datacenters, such as Amazon [10], are widely used as the back-end platform to execute DNN-based industrial applications [11]. The cloud providers often offer heterogeneous machine types, such as CPU-Optimized, Memory-Optimized, and GPU, that provide different execution time for various application types. For instance, a big data application type has its lowest execution time on the Memory-Optimized machine type whereas an image rendering application is best fitted to the GPU-based machine type. This form of heterogeneity is known as *inconsistent heterogeneity* [12], [13]. For each machine type, cloud providers offer a *consistent heterogeneity* in form of various virtual machine (VM) instance types with different number of allocated resources. For example, in Amazon cloud, for CPU-Optimized machine type, there is $c5d.18xlarge$ VM type with 36 number of cores that is faster than $c5.xlarge$ VM type with only 2 cores. As each application type can potentially have different inference time on distinct machine types, it is critical to consider the resource heterogeneity in estimating the execution time of different DNN-based applications.

For that purpose, in this study, we evaluate and analyze the execution time of DNN-based applications on heterogeneous cloud machine types. Our study encompasses both the application-centric perspective, by the way of modeling inference time, and the resource-centric perspective, by the way of measuring the Million Instruction Per Seconds (MIPS) metric. MIPS is considered as a rate-based metric that reflects the performance of cloud machine instance in terms of execution speed. As we consider latency-constrained applications, the underlying systems is considered as a dynamic (online) platform that processes each task upon arrival.

Prior studies on evaluating and modeling DNN-based applications [14] mostly focus on the core DNN model and ignore the end-to-end latency of the application that includes at least two other factors: (a) the latency of non-DNN parts of the application (*e.g.,* those for pre- and post-processing); and (b) the latency imposed due to uncertainties inherent to the cloud platform. Nonetheless, for critical industrial applications, such as those in O&G, a holistic analysis that considers the end-to-end latency of DNN-based applications is needed. The lack of such study hinders the path to develop a robust smart O&G solutions [15]. Accordingly, the main contributions of this work are as follows:

- Providing an application-centric analysis by developing a statistical model of the inference time of various DNN-based applications on heterogeneous cloud resources.
- Providing a resource-centric analysis of various DNN-based applications on heterogeneous cloud resources by developing a statistical model of MIPS, as a rate-based metric.
- Providing a publicly available[1] collection of pre-trained DNN-based industrial applications in addition to their training and testing datasets. Moreover, a trace of inference execution times of the considered applications on heterogeneous machines of two public cloud platforms (namely AWS and Chameleon Cloud [16]) is presented.

The rest of the paper is organized as follows: Section II discusses four different DNN applications and their underlying architectures. Section III states various cloud execution platforms with brief discussion. Section IV demonstrates the experimental setup to execute the DNN-based applications. The section V provides the application-centric analysis, whereas section VI presents the resource-centric analysis. Section VII presents related works. Finally, section VIII concludes the paper with discussion and future avenues for exploration.

II. DNN-Based Applications in O&G Industry 4.0

Table I summarizes different types of DNN-based applications used in the smart O&G industry. The table shows the abbreviated name for each application, its DNN (network) model, type of its input data, the scope of deployment in O&G Industry 4.0 [17], and the code base to build the model. All the applications, the input data, and analysis results are publicly available for reproducibility purposes[1]. In the rest of this section, we elaborate on the characteristics of each application type.

[1]https://github.com/hpcclab/Benchmarking-DNN-applications-industry4.0

Application Type	DNN Model	Input Type	Scope	Code Base
Fire Detection (Fire)	Customized Alexnet	Video Segment	Control & Monitoring	Tensorflow (tflearn)
Human Activity Recognition (HAR)	Customized Sequential Neural Network	Motion sensors	Workers Safety	keras
Oil Spill Detec. (Oil)	FCN-8	SAR Images	Disaster Management	keras
Acoustic Impedance Estimation (AIE)	Temporal Convolutional Network	Seismic Data	Seismic Exploration	PyTorch

TABLE I: DNN-based applications used in O&G Industry 4.0 along with their network model, input data type, usage scope, and code base.

A. Fire Detection (abbreviated as Fire)

Smart fire detection, a critical part of monitoring systems, aims at providing safety and robustness in Industry 4.0. We analyzed a fire detection application developed by Dunnings and Breckon [5] using convolutional neural network (CNN). It automatically detects fire regions (pixels) in the frames of a surveilled video in a real-time manner. Among other implementations, we deploy the FireNet model that accurately identifies and locate fire in each frame of a given video segment. FireNet is a lightweight variation of AlexNet model [18] with three convolutional layers of sizes 64, 128, and 256. In this model, each convolutional layer is augmented by a max-pooling layer and a local response normalization to achieve high frequency features with a large response from previous layer. To analyze the inference time of the fire detection system, we constructed a benchmarking dataset of 240 videos with different backgrounds. For fair and realistic analysis, the length of all videos is considered two seconds.

B. Human Activity Recognition (abbreviated as HAR)

Human Activity Recognition (HAR) systems are widely used in Industry 4.0 to ensure workers safety in hazardous zones. For this purpose, motion sensors, such as accelerometer and gyroscope, that are widely available on handheld PDA devices are utilized. The HAR system we use operates based on the sequential neural network model with four layers to identify the worker's activities (namely, walking, walking upstairs, walking downstairs, sitting). For analysis, we use a dataset of UCI machine learning repository, known as Human Activity Recognition Using Smartphones [19].

C. Oil Spill Detection (abbreviated as Oil)

Detecting the oil spill is of paramount importance to have a safe and clean O&G Industry 4.0. The accuracy of DNN-based oil spill detection systems has been promising [20]. We utilize a detection system that operates based on the FCN-8 model [21], which is depicted in Figure 1. As we can see, the model contains five Fully Convolutional Network (FCN) blocks and two up-sampling blocks that collectively perform semantic segmentation (*i.e.,* classifying every pixel) of an input image and output a labeled image. The FCN-8 model functions based on the satellite (a.k.a. SAR) [22] images. We configure the analysis to obtain the inference time of 110 SAR images collected by MKLab [20].

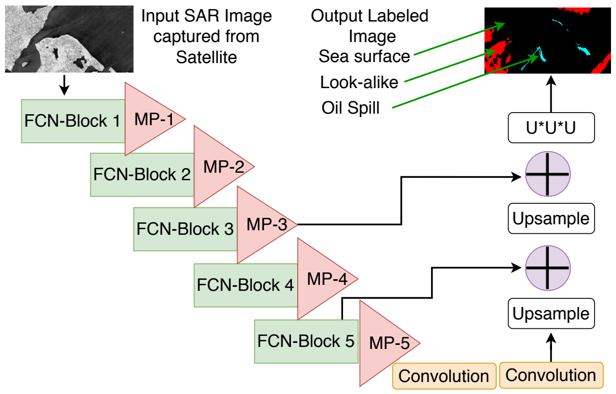

Fig. 1: The FCN-8 model is presented in block diagram that consist of 5 fully convolutional network blocks, and 2 up-sampling blocks. The model receives input as a SAR image and perform pixel-wise classification to output a labeled image.

D. Acoustic Impedance Estimation (abbreviated as AIE)

Estimating acoustic impedance (AI) from seismic data is an important step in O&G exploration. To estimate AI from seismic data, we utilize a solution functions based on the temporal convolutional network [23], shown in Figure 2. The solution outperforms others in terms of gradient vanishing and overfitting. Marmousi 2 dataset [24] is used to estimate AI.

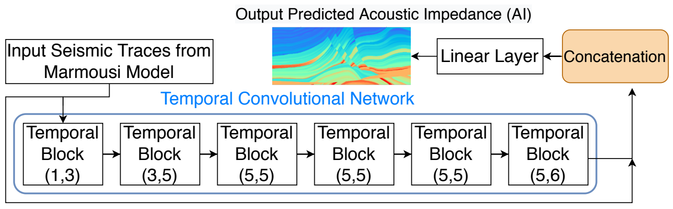

Fig. 2: Schematic view of Temporal Convolutional Network (TCN) model that consists of six temporal blocks, the input data, and the output in form of the predicted AI.

III. Cloud Computing Platforms for Industry 4.0

a) Amazon Cloud: AWS is a pioneer in the Cloud computing industry and provides more than 175 services, including Amazon EC2 [25], across a large set of distributed data centers. Amazon EC2 provides inconsistently heterogeneous machines (*e.g.,* CPU, GPU, and Inferentia) in form of various VM instance types (*e.g.,* general purpose, compute-optimized, and machine learning (ML)). Within each VM type, a range of VM configurations (*e.g.,* `large`, `xlarge`, `2xlarge`) are offered that reflect the consistent heterogeneity within that VM type. To realize the impact of machine heterogeneity on the inference time of various applications, we choose one representative VM type of each offered machine type. Table II represents the type of machines and their specification in terms of number of cores and memory. We note that all the machine types use SSD storage units. Although General Purpose machines are not considered suitable for latency-sensitive DNN-based applications, the reason we study them is their similarity to the specifications of machine types often used in the edge computing platforms. As such, considering these types of machines (and similarly $m1.small$ in the Chameleon cloud) makes the results of this study applicable to cases where edge computing is employed for latency-sensitive applications [26].

Machine Type	**VM Config.**	**vCPU**	**GPU**	**Mem. (GB)**
Mem. Optimized	`r5d.xlarge`	4	0	32
ML Optimized	`inf1.xlarge`	4	0	8
GPU	`g4dn.xlarge`	4	1	16
General Purpose	`m5ad.xlarge`	4	0	16
Comp. Optimized	`c5d.xlarge`	4	0	8

TABLE II: Heterogeneous machine types and VM configurations in Amazon EC2 that are considered for performance modeling of DNN-based applications. In this table, ML Optimized represents Inferentia machine type offered by AWS.

VM Config.	**vCPUS**	**Mem. (GB)**
`m1.xlarge`	8	16
`m1.large`	4	8
`m1.medium`	2	4
`m1.small`	1	2

TABLE III: Various VM flavors in Chameleon cloud are configured to represent a consistently heterogeneous system.

b) Chameleon Cloud: Chameleon cloud [16] is a large-scale public cloud maintained by National Science Foundation (NSF). Chameleon cloud supports VM-based heterogeneous computing. It offers the flexibility to manage the compute, memory, and storage capacity of the VM instances. In this study, we use the Chameleon cloud to configure a set of consistently heterogeneous machines. We configure four VM flavors, namely `m1.xlarge`, `m1.large`, `m1.medium`, and `m1.small`, as detailed in Table III. We note that VMs offered by Chameleon cloud uses HDD unit as storage.

IV. Environmental Setup for Performance Modeling

The focus of this study is on latency-sensitive DNN-based applications that are widely used in Industry 4.0. Accordingly, we consider a dynamic (online) system that is already loaded with pre-trained DNN-based applications, explained in the previous section, and executes arriving requests on the pertinent application. This means that we measure the hot start inference time [27] in the considered applications. The DNN-based applications mostly use TensorFlow, and the VMs both in AWS and Chameleon are configured to use the framework on top of Ubuntu 18.04.

Our initial evaluations in AWS (shown in Figure 3) demonstrate that, in different attempts, the inference execution time of an application (Oil Spill) on the same machine type can be highly stochastic. Similar stochasticity is found for chameleon cloud while we run the oil spill detection application 30 times within same VM instance. Hence to capture this randomness (aka consistent heterogeneity) that is caused by several factors, such as transient failures or multi-tenancy [28], [29], we base our analysis on 30 times execution of the same request within same VM.

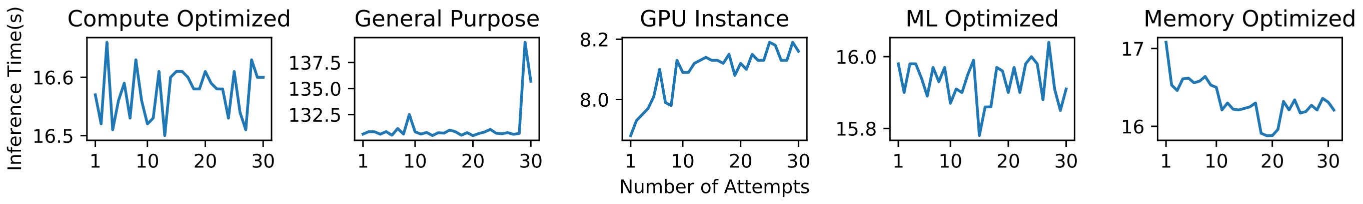

Fig. 3: The stochastic nature of inference execution time of oil spill application while running on heterogeneous VMs in the AWS. For every VM instance, the oil spill detection application is executed 30 times and those executions are plotted as number of attempts along x-axis. The y-axis represents the inference time for every attempts.

Execution Time Distribution with Shapiro-Wilk Test in AWS Cloud					
App. Type	**`Mem. Opt.`**	**`ML Opt.`**	**`GPU`**	**`Gen. Pur.`**	**`Compt. Opt.`**
Fire	Not Gaussian (P=2.73e^{-16})	Not Gaussian (P=5.42e^{-16})	Not Gaussian (P=6.59e^{-16})	Not Gaussian (P=2.06e^{-13})	Not Gaussian (P=3.9e^{-16})
HAR	Not Gaussian (P=7.12e^{-8})	Not Gaussian (P=1.04e^{-8})	Gaussian (P=0.19)	Not Gaussian (P=1.76e^{-8})	Not Gaussian (P=0.4.62e^{-5})
Oil	Not Gaussian (P=8e^{-4})	Not Gaussian (P=2.9e^{-16})	Not Gaussian (P=0.012)	Not Gaussian (P=1.27e^{-16})	Not Gaussian (P=5.86e^{-14})
AIE	Gaussian (P=0.46)	Gaussian (P=0.23)	Gaussian (P=0.08)	Not Gaussian (P=1.99e^{-10})	Gaussian (P=0.96)

TABLE IV: The execution time distributions of DNN-based applications in AWS clouds machines using Shapiro-Wilk test.

Execution Time Distribution with Shapiro-Wilk Test in Chemeleon				
App. Type	**`m1.xlarge`**	**`m1.large`**	**`m1.medium`**	**`m1.small`**
Fire	Not Gaussian (P=4.05e^{-5})	Not Gaussian (P=1.e^{-4})	Not Gaussian (P=7.58e^{-6})	Not Gaussian (P=1.32e^{-7})
HAR	Gaussian (P=0.74)	Not Gaussian (P=0.02)	Gaussian (P=0.18)	Gaussian (P=0.84)
Oil	Not Gaussian (P=0.01)	Not Gaussian (P=5.5e^{-7})	Not Gaussian (P=0.01)	N/A
AIE	Not Gaussian (P=2.77e^{-10})	Not Gaussian (P= 3.46e^{-6})	Not Gaussian (P= 1.4e^{-4})	Not Gaussian (P=2.46e^{-6})

TABLE V: The execution time distributions of DNN applications in Chameleon cloud using Shapiro-Wilk test.

V. Application-Centric Analysis of Inference Time

A. Overview

In this part, we capture the inference time of the four DNN applications and try to identify their underlying statistical distributions using various statistical methods. Then, to describe the behavior of inference execution time using a single metric, we explore the central tendency of the distributions.

B. Statistical Distribution of Inference Execution Time

Among various statistical methods, the normality tests are widely employed to understand the distribution of the collected samples. Hence, we first use Shapiro-Wilk test [30] to verify the normality of the inference time distribution on each machine type. Next, we employ Kolmogorov-Smirnov test [31] to find the best fit distribution based on the sampled inference execution times.

1) Shapiro-Wilk Test to Verify Normality of the Sampled Data: The null hypothesis is that the inference execution times are normally distributed. To understand whether a random sample comes from a normal distribution, we perform the Shapiro-Wilk test. The result of this test is considered as W, whose low value (lower than w_α threshold) indicates that the sampled data are not normally distributed and vice versa. The value of W is used to perform the significant testing (*i.e.,* calculating P-value). The higher P-value, especially greater than a threshold value (typically 0.05), supports the null hypothesis that the sampled data are normally distributed.

The results of Shapiro-Wilk test on the collected inference times for AWS are presented in Table IV, where columns present the various machine types and rows define the application types. The table reflects that our initial assumption is not totally valid. The Shapiro-Wilk test result for the Chameleon cloud, depicted in Table V, shows that for only three of the total cases, the normality assumption holds. Considering the lack of normality in several cases, in the next section, we utilize Kolmogorov-Smirnov test to more granularly explore the best fitting distribution for the inference time of each application and also cross validate the prior results we obtained using another statistical method.

Execution Time Distribution with Kolmogorov-Smirnov Test in AWS Cloud					
App. Type	**`Mem. Opt.`**	**`ML Opt.`**	**`GPU`**	**`Gen. Pur.`**	**`Compt. Opt.`**
Fire	No Distr.	No Distr.	No Distr.	No Distr.	No Distr.
HAR	Student's t (P=0.08)	Student's t (P=0.77)	Student's t (P=0.99)	Student's t (P=0.57)	Student's t (P=0.95)
Oil	Student's t (P=0.44)	Student's t (P=0.96)	Student's t (P=0.5)	Student's t (P=0.20)	Exponential (P=0.21)
AIE	Normal (P=0.99)	Normal (P=0.54)	Normal (P=0.47)	Exponential (P=0.16)	Normal (P=0.99)

TABLE VI: Inference time distributions of DNN-based applications in AWS cloud machines using Kolmogorov-Smirnov test.

2) Kolmogorov-Smirnov Goodness of Fit Test: The null hypothesis for the Kolmogorov-Smirnov test is that the inference times of a certain application type on a given machine type follows a statistical distribution. The Kolmogorov-Smirnov Goodness of Fit test (a.k.a. *K-S test*) identifies whether a set of samples derived from a population fits to a specific distribution. Precisely, the test measures the largest vertical distance (called test statistic D) between a known hypothetical probability distribution and the distribution generated by the observed inference times (a.k.a. empirical distribution function (EDF)). Then, if D is greater than the critical value obtained from the K-S test P-Value table, then the null hypothesis is rejected.

The results of the K-S test on the observed inference times in AWS and Chameleon clouds are depicted in Table VI and VII, respectively. From Table VI, we find that, in AWS, majority of the entries either represent Normal distribution or Student's t-distribution that exposes similar properties. However, we observe that the inference time of Fire Detection application does not follow any particular distribution with an acceptable P-Value. We also observe that the inference times of both Oil Spill application on Compute Optimized machine and AIE application on General Purpose machine follow Exponential distribution. However, low P-Value in both of these cases indicate a weak acceptance of the null hypothesis.

Execution Time Distribution with Kolmogorov-Smirnov test in Chameleon				
App. Type	**`m1.xlarge`**	**`m1.large`**	**`m1.medium`**	**`m1.small`**
Fire	No Distr	No Distr	No Distr	Log-normal
HAR	Normal (P=0.98)	Student's t (P=0.88)	Normal (P=0.66)	Normal (P=0.96)
Oil	Log-normal (P=0.36)	Log-normal (P=0.99)	Log-normal (P=0.81)	N/A
AIE	Student's t (P= 0.47)	Student's t (P=0.12)	Student's t (P=0.25)	Student's t (P=0.83)

TABLE VII: Inference time distributions of DNN-based applications in Chameleon's machines using the K-S test.

On the contrary, Table VII reflects the dominance of Log-normal (*i.e.,* the logarithm of the random variable is normally distributed) and Student's t-distribution over other distributions in the Chameleon cloud. Analysing the execution traces shows us that the inference times in Chameleon are predominantly larger than the ones in AWS that causes right-skewed property, hence, the distribution tends to Log-normal. Consistent to AWS observations, we see that Fire Detection application, in most of the cases, does not follow any distribution. Our further analysis showed that the lack of distribution is because of variety (*e.g.,* frame rate and resolution) in the input videos. In fact, when we reduced the degree of freedom in the input videos limited them to those with the same configuration (frame-rate), we noticed the inference time follows a Log-normal distribution. The observation shows that the characteristics and variation of input data can be decisive on the statistical behavior of inference times (mentioned in highlighted insight). We note that Oil Spill application cannot be run on `m1.small` machine owing to its limited memory.

Mean and Standarad Deviation of Inference Execution Times (ms) in AWS					
App. Type	**`Mem. Opt.`**	**`ML Opt.`**	**`GPU`**	**`Gen. Pur.`**	**`Compt. Opt.`**
Fire	μ=1461.8 σ=457.3	μ=1281.7 σ=387.93	μ=1349.5 σ=418.9	μ =1534.8 σ=494.7	μ=1421.4 σ=441.8
HAR	μ=1.27 σ=0.082	μ=0.66 σ=0.006	μ=0.51 σ=0.006	μ =1.17 σ=0.042	μ=0.66 σ=0.003
Oil	μ=269.9 σ=1.01	μ=218.8 σ=0.66	μ=65.98 σ=0.47	μ=667.1 σ=2.26	μ=242.9 σ=0.68
AIE	μ=7.02 σ=0.02	μ=6.41 σ=0.03	μ=7.55 σ=0.04	μ=9.35 σ=0.06	μ=7.95 σ=0.02

TABLE VIII: The measurement of central tendency metric (μ), and data dispersion metric (σ) on the observed inference times in AWS.

Insights: The key insights of the analysis we conducted on identifying the distribution of inference time are as follows:

- Shapiro-Wilk test for AWS and Chameleon rejects the null hypothesis that the inference times of DNN-based applications follow the Normal distribution.
- The K-S test reflects the dominance of Student's t-distribution of inference time in both AWS (Table VI), and Chameleon (Table VII).
- Various configurations of input data is decisive on the statistical distribution of the inference time.

C. Analysis of Central Tendency and Dispersion Measures

Leveraging the statistical distributions of inference times, in this part, we explore their central tendency metric that summarizes the behavior of collected observations in a single value. In addition, to analyze the statistical dispersion of the observations, we measure the standard deviation of the inference times. In particular, we estimate the arithmetic mean and standard deviation of the inference times. The central tendency metric of inference times for AWS and Chameleon clouds are shown in Tables VIII and IX, respectively. The **key insights** are as follows:

- Machine Learning Optimized and GPU instances often outperform other AWS machine types.
- In both clouds, the inference times of Fire and Oil experience a higher standard deviation in compare with other applications. The high uncertainty is attributed to the characteristics of their input data; Oil Spill input images suffer from class imbalance [20], whereas, Fire input videos are highly variant.
- In Chameleon VMs with a consistent heterogeneity, DNN applications with dense network models (*e.g.,* Oil and Fire) can take advantage of powerful machines (*e.g.,* `m1.xlarge`) to significantly reduce their inference times.
- Overall, AWS offers a lower inference time than Chameleon. The reason is utilizing SSD units in AWS rather than HDD in Chameleon. In addition, we noticed that Chameleon experiences more transient failures that can slow down the applications.

Mean and Std. of Inference Execution Times (ms) in Chameleon				
App. Type	**`m1.xlarge`**	**`m1.large`**	**`m1.medium`**	**`m1.small`**
Fire	μ=2155.20 σ=725.48	μ=2213.30 σ=731.50	μ=2330.80 σ=742.20	μ=3184.80 σ=1033.30
HAR	μ=22.14 σ=0.76	μ=47.69 σ=1.26	μ=49.24 σ=0.57	μ=52.69 σ=0.78
Oil	μ=147.16 σ=5.23	μ=222.22 σ=2.89	μ=412.78 σ=4.99	N/A
AIE	μ=6.23 σ=0.25	μ=6.23 σ=0.15	μ=6.40 σ=0.13	μ=7.72 σ=0.24

TABLE IX: Central tendency metric (μ), and data dispersion metric (σ) of the inference times in the Chameleon cloud.

The MIPS for DNN Applications in AWS Cloud					
App. Type	**`Mem. Opt.`**	**`ML Opt.`**	**`GPU`**	**`Gen. Pur.`**	**`Compt. Opt.`**
Fire	1938.63	2196.35	2092.72	1862.04	1989.56
HAR	838640.65	1595874.34	2040057.33	891754.48	1581709.12
Oil	164.54	168.58	331.98	20.46	162.01
AIE	145.58	180.28	150.25	131.25	160.32

TABLE X: MIPS values of heterogeneous machines in AWS for each DNN-based application.

VI. Resource-Centric Analysis of Inference Time

In performance analysis of computing systems, a rate-based metric [32] is defined as the normalization of number of computer instructions executed to a standard time unit. MIPS is a popular rate-based metric that allows comparison of computing speed across two or more computing systems.

The MIPS for DNN Applications in Chameleon				
App. Types	`m1.xlarge`	`m1.large`	`m1.medium`	`m1.small`
Fire	1327.81	1282.33	1249.63	871.36
HAR	91.78	102.51	124.76	136.62
Oil	18267.35	11233.41	6243.94	N/A
AIE	246366.52	249551.29	236300.93	201807.49

TABLE XI: MIPS vales for heterogeneous machines on Chameleon cloud for each DNN-based application.

Given that computing systems (*e.g.,* AWS ML Optimized and GPU) increasingly use instruction-level facilities for ML applications, our objective in this part is to analyze the performance of different machine types in processing DNN-based applications. The results of this analysis can be of particular interest to researchers and cloud solution architects whose endeavor is to develop tailored resource allocation solutions for Industry 4.0 use cases. As for rate-based metrics we do not assume any distribution [33], we conduct a non-parametric approach. In addition to MIPS, we provide the range of MIPS in form of *Confidence Intervals* (CI) for each case.

Let application i with n_i instructions have t_{im} inference time on machine m. Then, MIPS of machine m to execute the application is defined as $MIPS_{mi} = n_i/(t_{im} \times 10^6)$. Hence, before calculating MIPS for any machine, we need to estimate the number of instructions (n) of each DNN-based application. For that purpose, we execute each task t_i on a machine whose MIPS is known and estimated n_i. Then, for each machine m, we measure t_{im} and subsequently calculate $MIPS_{mi}$. Tables X and XI show the MIPS values for AWS and Chameleon, respectively.

To measure the confidence intervals (CI) of MIPS for each application type in each machine type, we use the non-parametric statistical methods [33] that perform prediction based on the sample data without making any assumption about their underlying distributions. As we deal with a rate-based metric, we use harmonic mean that offers a precise analysis for this type of metric rather than the arithmetic mean. We utilize Jackknife [33] re-sampling method and validate it using Bootstrap [33], which is another well-known re-sampling method. Both of these methods employ harmonic mean to measure the confidence intervals of MIPS.

CI of MIPS using Jackknife Method in AWS cloud					
App. Type	`Mem. Opt.`	`ML Opt.`	`GPU`	`Gen. Pur.`	`Compt. Opt.`
Fire	[1549.42, 1975.65]	[1770.81, 2243.04]	[1671.78, 2131.66]	[1465.31, 1889.77]	[1594.78, 2028.36]
HAR	[812040.26, 856355.96]	[1592214.75, 1599426.64]	[2033084.47, 2046727.57]	[880417.69, 901345.49]	[1580275.10, 1585598.85]
Oil	[163.55, 165.47]	[168.36, 168.81]	[330.68, 333.22]	[20.35, 20.57]	[161.86, 162.17]
AIE	[139.02, 141.04]	[155.56, 156.01]	[141.57, 142.03]	[118.06, 119.82]	[148.35, 149.00]

TABLE XII: The confidence intervals of MIPS values for DNN-based applications in AWS machines, resulted from Jackknife re-sampling method.

1) Estimating Confidence Interval using Jackknife Method: Let p be the number of observed inference times. The Jackknife method calculates the harmonic mean in p iterations, each time by eliminating one sample. That is, each time it creates a new sample (re-sample) with size $p-1$. Let x_j be the jth observed inference time. Then, the harmonic mean of re-sample i is called the pseudo-harmonic value (denoted as y_i) and is calculated based on Equation 1.

CI of MIPS using Jackknife Method in Chameleon Cloud				
App. Type	`m1.xlarge`	`m1.large`	`m1.medium`	`m1.small`
Fire	[1032.11, 1341.75]	[1010.62, 1303.02]	[964.76, 1259.68]	[670.82, 872.85]
HAR	[88.27, 94.20]	[99.84, 104.49]	[122.33, 126.67]	[135.13, 137.92]
Oil	[18083.59, 18628.64]	[11159.71, 11662.41]	[6139.59, 6262.15]	N/A
AIE	[237710.12, 252686.82]	[247166.73, 251673.68]	[168804.58, 268273.11]	[199676.71, 203681.17]

TABLE XIII: Confidence intervals of MIPS values for different DNN-based applications in Chameleon machines, resulted from Jackknife re-sampling method.

$$y_i = \frac{p-1}{\sum_{j=1, j\neq i}^{p} \frac{1}{x_j}} \tag{1}$$

Next, the arithmetic mean (denoted $\bar{y}$) of the p pseudo-harmonic values is computed, and is used to estimate the standard deviation. Finally, the t-distribution table is used to calculate the CI boundaries with a 95% confidence level. The result of the Jackknife method for AWS machines is shown in Table XII that conforms with the MIPS calculation in Table X. Similarly, the results of analysis for Chameleon cloud using Jackknife method, shown in Table XIII, validate the prior MIPS calculations in Table XI. However, in the next part, we cross-validate these results using Bootstrap method.

2) Estimating Confidence Interval using Bootstrap Method: Bootstrap repeatedly performs random sampling with a replacement technique [33] on the observed inference times. The random sampling refers to the selection of a sample with the chance of non-zero probability and the number (represented as k) of re-sample data depends on the user's consideration. After re-sampling, the harmonic means of k number of samples are calculated and sorted in ascending order to estimate the confidence intervals. Finally, for a specific confidence level, the $(\alpha/2 \times k)$th and $((1-\alpha/2) \times k)$th values are selected from the sorted samples as the lower and upper bounds of the CI. We set the k value to 100 and α to 0.05 for 95% confidence level.

For both AWS and Chameleon, the results of CI analysis using the Bootstrap method are similar to, thus validate, the ranges estimated by the Jackknife method. Therefore, due to the shortage of space, we do not report the table of MIPS values for the Bootstrap method. However, we note that the CI ranges provided by the Bootstrap method are shorter (*i.e.,* have less uncertainty), regardless of the application type and the cloud platform. The reason for the shorter range is that Bootstrap performs re-sampling with a user-defined number of samples that can be larger than the original sample size.

To perform a cross-platform analysis of the MIPS values, in Figure 4, we compare the range of MIPS values for AWS `Compute Optimized` against `m1.large` that is a compatible machine type in Chameleon (see Tables II and III). The horizontal axis of this figure shows different application types

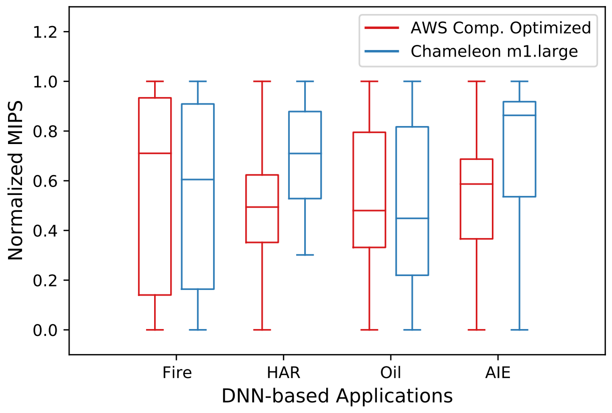

Fig. 4: Comparative analysis of the MIPS values of AWS and Chameleon machines for various DNN-based applications. For the sake of presentation, the MIPS values are normalized between [0,1].

and the vertical axis shows the MIPS values, normalized based on MinMax Scaling in the range of [0,1], for the sake of better presentation. Due to high variation in the input videos, we observe a broad CI range for Fire detection across both cloud platforms. However, for HAR, Oil Spill, and AIE applications, we observe that the first and third quartiles of the CI range in Chameleon (whose machines are prone to more transient failures [34]) is larger than those in AWS. This wide range indicates that, apart from variations in the input data, the reliability of underlying resources is also decisive on the stochasticity of the inference times.

VII. Related Work

The advent of Industry 4.0 brought revolution in the O&G industry, and Oil and Gas 4.0 [1] era arises. To deploy and improve the existing industrial DNN-based solutions for Oil and Gas 4.0, performance modeling of DNN-based applications is of great importance for cloud solution architects and researchers. Although, it is hard to find in-depth performance study of real-world DNN-based applications in cloud platforms, especially in the context of the O&G industry considering the inference time of DNN-based applications. As such, we explore research areas related to benchmarking of DNN-based applications in various contexts (*i.e.,* artificial intelligence, video streaming, image recognition) utilizing local machines, virtual machines, or even mobile devices. However, these works focus mainly on training times, whereas our work focuses on the inference time.

As an execution platform for video transcoding operation, heterogeneous cloud virtual machines are analyzed for performance evaluation of different video content by Li *et al.* in [35]. In this work, the authors elaborately discuss the correlation of video content type with the transcoding operation's execution time on heterogeneous VMs. In a similar context, Ghatrehsamani *et al.* in [36] performed a detailed study of performance overhead of various cloud execution platforms utilizing four categories of real-world applications. One of the interesting findings of this work is that the containers are not always suitable computing platforms in the cloud. The above mentioned works mainly focus on virtual computing platforms regardless of application domain whereas we explore specific DNN-based applications inference execution pattern.

To perform the layer-wise behavior of various DNN models in heterogeneous deployment platforms, Xia *et al.* in [14] proposed a DNN tuning framework. This work provides a detailed discussion of the performance and energy consumption of DNN models (*i.e.,* CNN, LSTM, MLP) concerning different deployment strategies (*i.e.,* cloud, mobile device, mobile-cloud hybrid). Similarly, Turner *et al.* in [37] proposed various mainstream DNN compression methods with the evaluation of the model accuracy, training time, and memory on two types of execution hardware (*i.e.,* CPU, GPU). In [38], Bianco *et al.* performed an extensive system-level analysis of a wide range of DNN models on two different computing architectures, namely NVIDIA Titan X Pascal and NVIDIA Jetson TX1. From an intuitively different perspective, the authors of [39] represent a detailed analysis of DNN-based application execution in optimized hardware and their overhead considering various prior and post operations during implementation on a special-purpose accelerator. This work proposes to utilize a specialized edge data center designed for DNN-based application that overcomes the data processing overhead. On the contrary, our work focuses on inference time behavior of DNN-based applications on heterogeneous cloud machine types, especially in the smart O&G industry.

VIII. Summary and Discussion

Accurately estimating the inference time of latency-sensitive DNN-based applications plays a critical role in robustness and safety of Industry 4.0. Such accurate estimations enable cloud providers and solution architects to devise resource allocation and load balancing solutions that are robust against uncertainty exists in the execution time of DNN-based applications. In this work, we provide application- and resource-centric analyses on the uncertainty exists in the inference times of several DNN-based applications deployed on heterogeneous machines of two cloud platforms, namely AWS and Chameleon. In the first part, we utilized the Shapiro-Wilk test to verify if the assumption of Normal distribution for the inference time holds. We observed that the inference times often do not follow a Normal distribution. Therefore, in the second part, we broaden our distribution testing investigation and utilized the Kolmogorov-Smirnov test to verify the underlying distributions in each case. The analysis showed that inference times across the two cloud platforms often follow Student's t-distribution. However, in several cases in Chameleon cloud we observed the Log-normal distribution that we attribute it to the uncertain performance of VMs in this platform. Next, to conduct a resource-centric analysis, we modeled MIPS (as a rate-based performance metric) of the heterogeneous machines for each application type. In the analysis, we took a non-parametric approach, which is suitable for rate-based metrics, and utilized the Jackknife and Bootstrap re-sampling methods with harmonic mean to determine the range of confidence intervals of the MIPS values in each case. The calculated MIPS values and their CI ranges reflect the behavior of different DNN-based applications under various machine types and cloud platforms. A comparative analysis of the CI ranges across AWS and Chameleon cloud demonstrate that the uncertainty in the inference time is because of variations in the input data and unreliability of the underlying platforms. In the future, we plan to incorporate the findings of this research to devise accurate resource allocation methods in IoT and edge computing systems. In addition, we plan to develop a predictive analysis to determine the execution of each inference task upon arrival.

Acknowledgments

The research was supported by Chameleon cloud and Amazon Cloud (AWS) research credit.

References

[1] H. Lu, L. Guo, M. Azimi, and K. Huang, “Oil and gas 4.0 era: A systematic review and outlook,” *Journal of Computers in Industry*, vol. 111, pp. 68–90, 2019.

[2] X. Li, M. A. Salehi, M. Bayoumi, N.-F. Tzeng, and R. Buyya, “Cost-efficient and robust on-demand video transcoding using heterogeneous cloud services,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 29, no. 3, pp. 556–571, 2017.

[3] R. Hussain, M. Amini, A. Kovalenko, Y. Feng, and O. Semiari, “Federated edge computing for disaster management in remote smart oil fields,” in *21st International Conference on High Performance Computing and Communications (HPCC)*, 2019.

[4] I. Dincer and C. Acar, “A review on clean energy solutions for better sustainability,” *International Journal of Energy Research*, vol. 39, no. 5, pp. 585–606, 2015.

[5] A. J. Dunnings and T. P. Breckon, “Experimentally defined convolutional neural network architecture variants for non-temporal real-time fire detection,” in *Proceedings of 25th IEEE International Conference on Image Processing (ICIP)*, pp. 1558–1562, IEEE, 2018.

[6] F. Aliyu and T. Sheltami, “Development of an energy-harvesting toxic and combustible gas sensor for oil and gas industries,” *Journal of Sensors and Actuators B: Chemical*, vol. 231, pp. 265–275, 2016.

[7] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, “Jointdnn: an efficient training and inference engine for intelligent mobile cloud computing services,” *Journal of IEEE Trans. on Mobile Computing*, 2019.

[8] Q. Zhang, M. Zhang, M. Wang, W. Sui, C. Meng, J. Yang, W. Kong, X. Cui, and W. Lin, “Efficient deep learning inference based on model compression,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 1695–1702, 2018.

[9] Z. Han, H. Tan, X.-Y. Li, S. H.-C. Jiang, Y. Li, and F. C. Lau, “Ondisc: Online latency-sensitive job dispatching and scheduling in heterogeneous edge-clouds,” *IEEE/ACM Trans. on Networking*, vol. 27, no. 6, pp. 2472–2485, 2019.

[10] I. Bermudez, S. Traverso, M. Mellia, and M. Munafo, “Exploring the cloud from passive measurements: The amazon aws case,” in *Proceedings of IEEE INFOCOM*, pp. 230–234, 2013.

[11] A. Luckow, M. Cook, N. Ashcraft, E. Weill, E. Djerekarov, and B. Vorster, “Deep learning in the automotive industry: Applications and tools,” in *Proceedings of International Conference on Big Data (Big Data)*, pp. 3759–3768, 2016.

[12] A. Mokhtari, C. Denninnart, and M. A. Salehi, “Autonomous task dropping mechanism to achieve robustness in heterogeneous computing systems,” *Proceedings of the 29th Heterogeneity in Computing Workshop (HCW 2019)*, May 2020.

[13] J. Gentry, C. Denninnart, and M. Amini Salehi, “Robust dynamic resource allocation via probabilistic task pruning in heterogeneous computing systems,” in *Proceedings of the 33rd IEEE International Parallel & Distributed Processing Symposium*, IPDPS ’19, May 2019.

[14] C. Xia, J. Zhao, H. Cui, X. Feng, and J. Xue, “Dnntune: Automatic benchmarking dnn models for mobile-cloud computing,” *ACM Trans. on Architecture and Code Optimization (TACO)*, vol. 16, no. 4, 2019.

[15] R. F. Hussain, M. A. Salehi, A. Kovalenko, S. Salehi, and O. Semiari, “Robust resource allocation using edge computing for smart oil fields,” in *Proceedings of the 24th International Conference on Parallel and Distributed Processing Techniques & Applications*, 2018.

[16] K. Keahey, P. Riteau, D. Stanzione, T. Cockerill, J. Mambretti, P. Rad, and P. Ruth, “Chameleon: a scalable production testbed for computer science research,” in *Contemporary High Performance Computing: From Petascale toward Exascale*, vol. 3, ch. 5, pp. 123–148, May 2019.

[17] T. Nguyen, R. G. Gosine, and P. Warrian, “A systematic review of big data analytics for oil and gas industry 4.0,” *IEEE Access*, vol. 8, pp. 61183–61201, 2020.

[18] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. Van Esesn, A. A. S. Awwal, and V. K. Asari, “The history began from alexnet: A comprehensive survey on deep learning approaches,” *arXiv preprint arXiv:1803.01164*, 2018.

[19] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “A public domain dataset for human activity recognition using smartphones.,” in *Esann*, 2013.

[20] M. Krestenitis, G. Orfanidis, K. Ioannidis, K. Avgerinakis, S. Vrochidis, and I. Kompatsiaris, “Oil spill identification from satellite images using deep neural networks,” *Journal of Remote Sensing*, vol. 11, no. 15, p. 1762, 2019.

[21] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[22] Z. Huang, C. O. Dumitru, Z. Pan, B. Lei, and M. Datcu, “Classification of large-scale high-resolution sar images with deep transfer learning,” *Journal of Geoscience and Remote Sensing Letters*, 2020.

[23] A. Mustafa, M. Alfarraj, and G. AlRegib, “Estimation of acoustic impedance from seismic data using temporal convolutional network,” *arXiv preprint arXiv:1906.02684*, 2019.

[24] R. Versteeg, “The Marmousi experience; velocity model determination on a synthetic complex data set,” *Journal of the Leading Edge*, vol. 13, pp. 927–936, 09 1994.

[25] J. Varia, S. Mathew, *et al.*, “Overview of amazon web services,” *Amazon Web Services*, pp. 1–22, 2014.

[26] V. Veillon, C. Denninnart, and M. A. Salehi, “F-FDN: Federation of Fog Computing Systems for Low Latency Video Streaming,” in *Proceedings of the 3rd IEEE International Conference on Fog and Edge Computing*, pp. 1–9, 2019.

[27] S. S. Ogden and T. Guo, “Characterizing the deep neural networks inference performance of mobile applications,” *arXiv preprint arXiv:1909.04783*, 2019.

[28] H. Moradi, W. Wang, and D. Zhu, “Adaptive performance modeling and prediction of applications in multi-tenant clouds,” in *21st International Conference on High Performance Computing and Communications(HPCC)*, pp. 638–645, 2019.

[29] X. Li, M. A. Salehi, Y. Joshi, M. K. Darwich, B. Landreneau, and M. Bayoumi, “Performance analysis and modeling of video transcoding using heterogeneous cloud services,” *IEEE Trans. on Parallel and Distributed Systems*, vol. 30, no. 4, pp. 910–922, 2019.

[30] Z. Hanusz, J. Tarasinska, and W. Zielinski, “Shapiro-wilk test with known mean,” *REVSTAT-Statistical Journal*, vol. 14, no. 1, 2016.

[31] I. M. Chakravarty, J. Roy, and R. G. Laha, “Handbook of methods of applied statistics,” 1967.

[32] D. J. Lilja, *Measuring computer performance: a practitioner’s guide*. 2005.

[33] S. Patil and D. J. Lilja, “Using resampling techniques to compute confidence intervals for the harmonic mean of rate-based performance metrics,” *IEEE Computer Architecture Letters*, vol. 9, no. 1, 2010.

[34] B. Charyyev, A. Alhussen, H. Sapkota, E. Pouyoul, M. H. Gunes, and E. Arslan, “Towards securing data transfers against silent data corruption.,” in *CCGRID*, pp. 262–271, 2019.

[35] X. Li, M. A. Salehi, Y. Joshi, M. K. Darwich, B. Landreneau, and M. Bayoumi, “Performance analysis and modeling of video transcoding using heterogeneous cloud services,” *Journal of Trans. on Parallel and Distributed Systems*, vol. 30, no. 4, pp. 910–922, 2018.

[36] D. Ghatrehsamani, C. Denninnart, J. Bacik, and M. Amini Salehi, “The art of cpu-pinning: Evaluating and improving the performance of virtualization and containerization platforms,” in *Proceedings of the 49th International Conference on Parallel Processing*, Aug 2020.

[37] J. Turner, J. Cano, V. Radu, E. J. Crowley, M. O’Boyle, and A. Storkey, “Characterising across-stack optimisations for deep convolutional neural networks,” in *Proceedings of IEEE International Symposium on Workload Characterization (IISWC)*, pp. 101–110, 2018.

[38] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, “Benchmark analysis of representative deep neural network architectures,” *IEEE Access*, vol. 6, pp. 64270–64277, 2018.

[39] D. Richins, D. Doshi, M. Blackmore, A. T. Nair, N. Pathapati, A. Patel, B. Daguman, D. Dobrijalowski, R. Illikkal, K. Long, *et al.*, “Ai tax: The hidden cost of ai data center applications,” *arXiv preprint arXiv:2007.10571*, 2020.