U. Elordi, L. Unzueta, J. Goenetxea, S. Sanchez-Carballido, I. Arganda-Carreras and O. Otaegui, "Benchmarking Deep Neural Network Inference Performance on Serverless Environments With MLPerf," in IEEE Software, vol. 38, no. 1, pp. 81-87, Jan.-Feb. 2021, https://doi.org/10.1109/MS.2020.3030199.

Benchmarking DNN inference performance on serverless environments with MLPerf

U. Elordi
Vicomtech Foundation
Basque Research and Technology Alliance (BRTA)

L. Unzueta
Vicomtech Foundation
Basque Research and Technology Alliance (BRTA)

J. Goenetxea
Vicomtech Foundation
Basque Research and Technology Alliance (BRTA)

S. Sanchez-Carvallido
Vicomtech Foundation
Basque Research and Technology Alliance (BRTA)

I. Arganda-Carreras
University of the Basque Country (UPV/EHU)
Ikerbasque, Basque Foundation for Science
Donostia International Physics Center (DIPC)

O. Otaegui
Vicomtech Foundation
Basque Research and Technology Alliance (BRTA)
University of the Basque Country (UPV/EHU)

***Abstract*—Current computing requirements for high-scale inference of Deep Neural Networks (DNNs) demand distributed execution environments. The advantages of serverless functions in distributed computation offloading and automatic resource scalability make them a very suitable environment for such a task. However, finding the optimal workload at minimum resource usage requires extensive benchmarking analysis such as those of the MLPerf standard. However, due to its monolithic nature, the MLPerf execution model collides with serverless function platforms. To address this issue, we provide a novel decomposition methodology from the current MLPerf benchmark to the serverless function execution model. Moreover, we have tested our approach in Amazon Lambda to benchmark the processing capabilities of OpenCV and OpenVINO IE DNN inference engines using Caffe, Tensorflow, and OpenVINO models for Computer Vision tasks. Experimental results prove Amazon Lambda a suitable platform for DNN inference following our proposed architecture.**

■ **SERVERLESS COMPUTING** is an emerging cloud execution model whose resource scalability is dynamically managed on-demand by the cloud providers and billed only by usage time. A clear use case of this execution model is the serverless function paradigm. Under the scope of the Function

as a Service (FaaS), a serverless function architecture offloads the computation into functions, so developers can focus on the source code without worrying about the resource provisioning management.

In parallel, Machine Learning (ML) has strongly gained momentum in recent years, thanks to the emerging Deep Neural Networks (DNNs). DNN-based Computer Vision (CV) and Neural Language Processing (NLP) methods currently constitute the basis of the most advanced ML-based applications. The increasing popularity of such kind of applications have brought newer specialized ML hardware (e.g., TPUs, VPUs, referred as xPUs, in general) and software tools to optimize the inference of the computationally demanding DNNs.

However, the produced myriad combinations of ML hardware and software tools make the assessment of ML system performance challenging. Several attempts to solve this problem have been made both by academic and industrial organizations. In particular, the MLPerf Inference benchmarking suite has currently become the "de-facto" standard, driven by more than 30 organizations and more than 200 ML engineers and practitioners [1]. Its first call for submissions garnered more than 600 reproducible inference performance measurements in October 2019, and the obtained results were published in its webpage for comparison.

The current version of MLPerf Inference (version 0.5) was developed following a monolithic design (i.e., single execution unit), which collides with the nature of serverless function platforms. Fortunately, the MLPerf community is open to further revisions, as ML is still evolving, and newer needs arise. The MLPerf community is very active and their discussions are organized by topics and working groups. However, the fitting of serverless function platforms in MLPerf has not been discussed yet in that forum, so we aim at opening this possibility, explaining our insights and experiences which have resulted in the presented methodology, tested in Amazon Lambda, which is a popular serverless computing platform. Nevertheless, our approach is a FaaSified platform for benchmarking ML and could also be considered under the scope of alternative benchmarking suites.

CHALLENGES OF BENCHMARKING DNN INFERENCE IN FAAS PLATFORMS

The FaaS platform workloading is managed by function instances. Serverless function instances are stateless (no dependency from function execution state), include ephemeral storage (the data is erased when the function instance finishes), and they are executed in isolated containers. Besides, each instance of the function contains an allocated amount of memory (function memory) and it is executed in CPU backend hardware [2].

Benchmarking DNN inference in FaaS platforms must cope with the following challenges:

- *How to deploy DNN models and their inference engine*: this includes choosing a suitable DNN inference engine and loading and processing the trained DNN models, considering the space and memory constraints of serverless platforms.
- *Cold starts:* additional latencies that occur when the serverless function is invoked for the first time.
- *Handling the performance results:* considering that the FaaS platforms are stateless with ephemeral storage, how do we manage the persistency of the measured results?

Zhang *et al.* [3] presented MArk, a general-purpose ML inference serving system for optimal DNN inference workloading, from GPUs to serverless runtime, followed by a predictive resource autoscaling algorithm. Romero *et al.* proposed the INFaaS [4] inference serving system, which automatically determines the model-variant, hardware, and scaling configuration, based on user-defined inference tasks and performance/accuracy requirements for queries. However, these systems do not tackle the goal of fair benchmarking across the high variety of hardware and software architectures for DNN inference, as MLPerf Inference does.

ENABLING SERVERLESS RUNTIME IN MLPERF

MLPerf Inference is designed to benchmark common ML-based CV tasks (image classification and object detection), and NLP tasks (translation). To do so, its architecture contains the following components:

- *System Under Test (SUT):* it runs the DNN inference and the performance measurements are sent back to the Load Generator (LoadGen).
- *LoadGen:* it feeds the SUT with the input data and calculates all performance measurements for benchmark calculation.
- *Data set:* to configure the input data to be ready for the benchmark.

MLPerf (version 0.5) considers the following scenarios, when feeding the SUT:

- *Single stream:* an inference query is sent and only upon completion, the next query is sent.
- *Multi stream:* LoadGen sends a set of inferences per query periodically (between 50 and 100 ms).
- *Server*: inputs arrive according to a Poisson distribution.
- *Offline*: the complete input data set is sent in a unique query.

Regarding the benchmarks, MLPerf Inference has two divisions for submitting results: closed and open. Strict rules govern the closed division, such as using specific DNN model implementations, to address the lack of a standard inference-benchmarking workflow. The open division, on the contrary, allows submitters to change the model and demonstrate different performance and quality targets.

These scenarios assume a monolithic design of the system's architecture. This means that the DNN inference is processed in the same execution unit (the targeted CPU, GPU or xPU), which is not feasible in a serverless system. Thus, we propose a new scenario, in which we execute a burst of several instances with no time interval between consecutive inference queries, and a FaaSified platform for benchmarking ML.

FaaSification is the process of transforming existing code into functions in conformance with the programming conventions expected by the target provider. According to [5], this process can be classified in three levels depending on the considered Atomic Unit (AU): shallow (AU: functions or methods), medium (AU: lines of code) and deep (AU: instructions).

Figure 1 depicts our FaaS benchmarking architecture and the life cycle of the benchmarking process, numbered from 1 to 11. The SUT is designed following a shallow FaaSification process, i.e., the AUs are functions and methods.

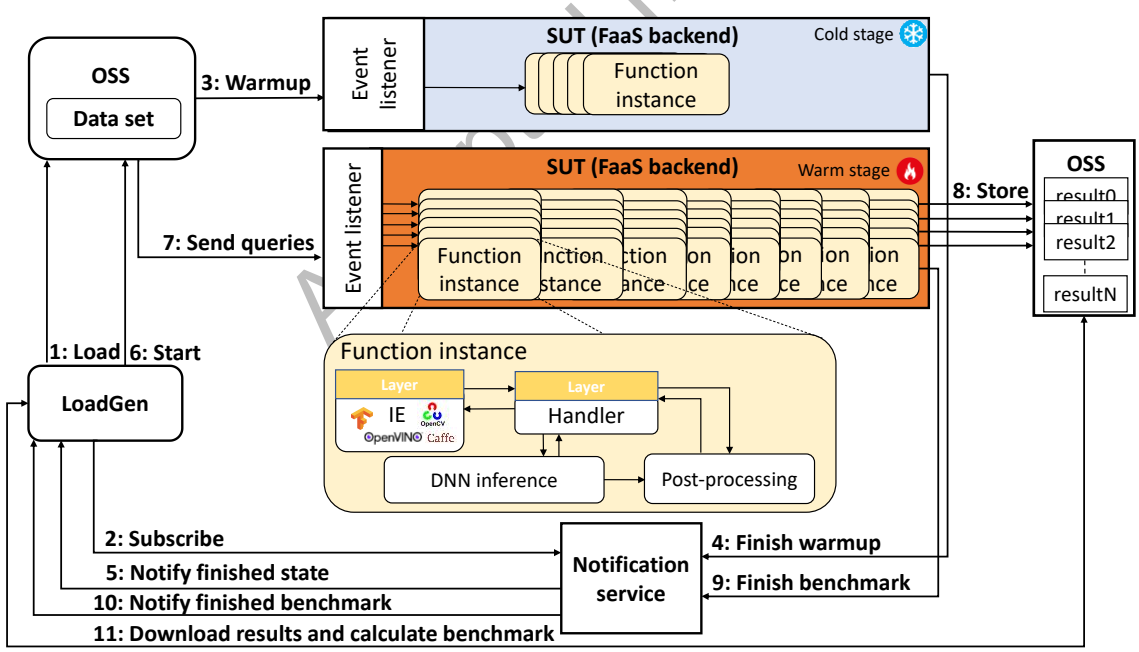


Figure 1. The proposed benchmarking FaaS architecture and life cycle.

More specifically, our SUT implementation is composed of two layers:

- *Inference Engine (IE) layer*: this layer encompasses the software tools to infer trained DNN models with ML task-related algorithms.
- *Handler layer:* this layer manages the DNN inference algorithm depending on the selected DNN framework and the post-processing operations to obtain the inference results.

For this FaaS architecture design we have relied on MLPerf Inference's components, but it could also be considered under the scope of alternative benchmarking suites, as any suite should contain components like SUT, LoadGen and data set.

As shown in Figure 1, firstly, the LoadGen configures the data set source from the Online Storage Service (OSS) (step 1). Also, the LoadGen is subscribed to a Notification Service (NS) to handle the benchmarking life cycle (step 2). Next, the warmup process executes few function instances to avoid cold start delays during the benchmarking process, and hence, changing the state of the FaaS container to warm stage (step 3). During the first function instance execution, trained DNN models and software tools are downloaded to the containers and then, these DNN resources are loaded to be available for the next warmed function instances. Finally, the LoadGen receives the warmup finish notification from the SUT (steps 4 and 5) and the system is ready to start benchmarking.

For each input element from the data set, the LoadGen uploads a .json file to the OSS (step 6). This file, that contains the paths to the input data set, is comprised of parameters such as database input data references, the result delivering output data, and benchmarking action commands. Next, following the event-driven design of the FaaS platforms [6][7], each uploading action triggers an event automatically creating a function instance (see event listener in Figure 1). At this point in the benchmarking process, the SUT invokes several function instances (one per query). So, each function performs the inference and post-processing tasks of the data coming from the OSS, and measures the following output values:

- The start and end timestamps.
- The processing time (the function's latency).
- The post-processing results of the function for accuracy evaluation.

To preserve the FaaS data persistency, each function instance saves the measured output values in a separate file in the OSS (step 8). The last function instance sends a finishing action message to the LoadGen (step 9-10).

Finally, the LoadGen downloads all files with the mentioned output values from OSS (step 11). This information is organized into different lists to calculate the inference latency, throughput, and accuracy results. These are calculated like this:

- *Latency:* instead of using average latency as the definitive metric, the 90th percentile of the latency list is calculated. We do this to reduce the impact of outliers.
- *Throughput:* the number of queries divided by the total time. This total time refers to the time difference between the maximum value of end timestamp and the minimum value of start timestamp.
- *Accuracy calculation:* the post-processing results are compared with respect to the ground truth data according to a measurement protocol, which depends on the ML task.

IMPLEMENTATION AND EVALUATION

We tested our approach in Amazon Lambda, with Amazon S3 to store the input and output data, and Amazon Simple Notification Service (SNS) as the NS. As mentioned above, we focused our implementation and tests on CV tasks to benchmark the processing capabilities of the DNN inference engines of OpenCV [8] and OpenVINO Inference Engine (IE) [9] using Caffe, Tensorflow, and OpenVINO Intermediate Representation (IR) models. In particular, we have taken the monolithic MLPerf algorithm class and its functions as AUs, and we manually deployed to a FaaS function, supported by Amazon Lambda layers. We make the source code available (https://github.com/Vicomtech/serverless-mlperf) to enable a follow-up discussion about the proposed design, implementation and experiments with the MLPerf and serverless computing communities.

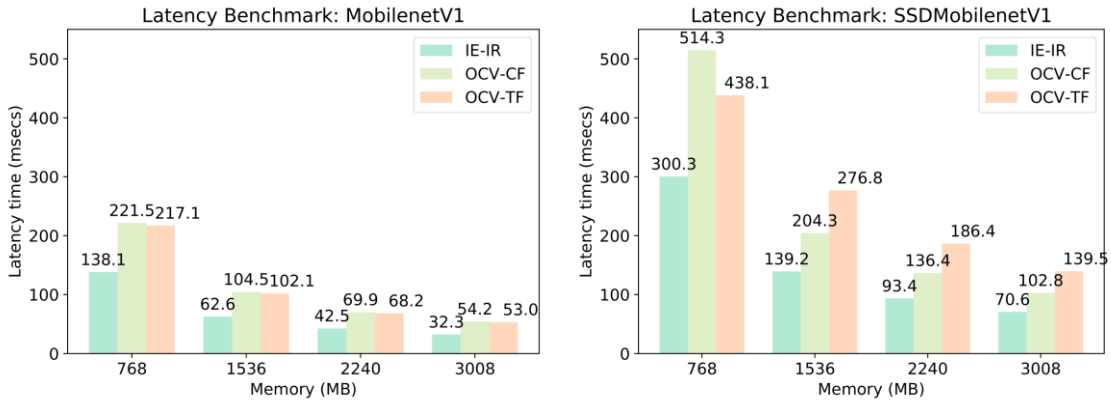


Figure 2. Inference latency results with OpenVINO IR (IR), Caffe (CF) and TensorFlow (TF) DNN models, and OpenVINO IE (IE) and OpenCV (OCV) as inference engines.

To evaluate the feasibility of our implementation, we benchmarked two DNN models of the MLPerf closed division with the following configuration:

- *Data set:* ImageNet [10] and COCO [11] (subset of randomized 10K images per data set).
- *Performance metrics:* latency and throughput.
- *DNN models:* we used *MobileNetV1* and *SSDMobileNetV1* for image classification and object detection respectively, with 32 Floating Point (FP) precision in Caffe, TensorFlow and OpenVINO IR formats.
- *FaaS memory configurations:* 768MB, 1536MB, 2240MB, 3008MB.

The baseline to compare these results with monolithic implementations would be the results published in MLPerf's webpage.

Figure 2 depicts the latency time for different function memory configurations, from 768MB to 3GB. We observed that increasing the allocated memory for each function instance the latency improves in both benchmarked models, especially in ranges between 768-1536MB. This performance improvement is between 2.12-2.20X times larger for MobileNetV1 and 1.58-2.55X for SSDMobileNetV1. This confirms the observations of Maissen *et al.* [2] about the latency reduction when the allocated memory is increased, and therefore, the CPU power increases linearly.

Moreover, OpenVINO IR models achieve the best performance results. This is because the OpenVINO IE DNN inference engine operations are optimized for Intel parallelization and vectorization instructions such as AVX, SSE2 or SSE4, and currently Amazon Lambda processors rely on Intel hardware [2].

As expected, since SSDMobileNetV1 has more parameters and layers than MobileNetV1, its latency is higher. While in MobileNetV1 the performance of Caffe and TF models is quite similar, in SSDMobileNet the reduction of the latency with the Caffe model is between 1.33 and 1.98X compared to TF.

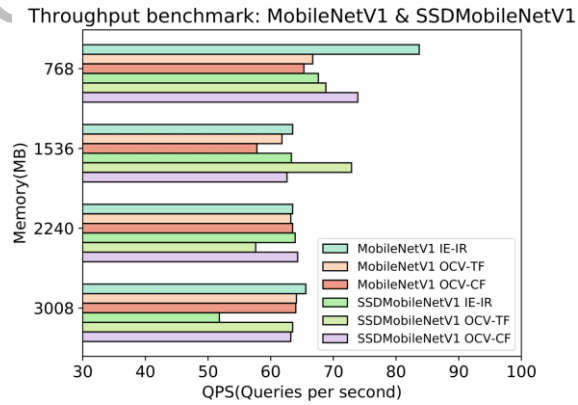


Figure 3. Inference throughput results with OpenVINO IR (IR), Caffe (CF) and TensorFlow (TF) DNN models, and OpenVINO IE (IE) and OpenCV (OCV) as inference engines.

Nevertheless, the inference throughput values calculated in **Figure 3** reveal that the inference latency time does not have any influence in the

throughput values. We believe the variations in inference throughput depend on the AWS cloud provider scheduling resources.

CONCLUSION

The increasing need to deploy ML tasks at high scale demands optimal execution models such as serverless functions. However, finding an efficient DNN inference workload using minimum resources requires an important benchmarking analysis. Throughout our benchmarking evaluation of DNN inference efficiency, we have observed that the amount of the allocated memory for each function instance plays an important role in inference latency time reduction, especially when the configured memory is between 768MB and 1536MB.

Also, the OpenVINO IE DNN inference engine and OpenVINO IR model optimizations contribute to reduce the inference latency in Amazon Lambda hardware. However, these latency results do not influence the inference throughput results. We hypothesize that this occurs due to the cloud provider scheduling capabilities. However, we believe that 51-83 QPS values make Amazon Lambda a suitable platform for DNN inference.

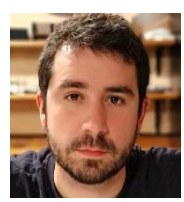
The design space is still very large –different serverless environments, different benchmarks (in addition to MLPerf), different hardware targets (CPUs, GPUs, xPUs, etc) – and requires further investigation. Therefore, we expect to expand these benchmarking evaluations to the most popular serverless function platforms. We will also explore how to benchmark more complex ML systems that consider a computing continuum formed by mobile, edge, and cloud resources [12], relying on standards such as MLPerf.

ACKNOWLEDGEMENT

This work has been partially supported by the program ELKARTEK 2019 of the Basque Government under project AUTOLIB.

■ REFERENCES

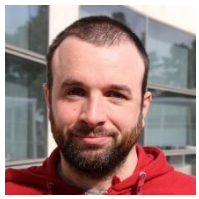
1. MLPerf Inference benchmark v0.5. [Online]. Available: https://github.com/mlperf/inference (URL)
2. P. Maissen, P. Felber, P. Kropf, and V. Schiavoni, “FaaSdom: A benchmark suite for serverless computing,” *Proceedings of the 14th ACM International Conference on Distributed and Event-Based Systems (DEBS '20),* pp. 73-84, 2020. (conference proceedings)
3. C. Zhang, M. Yu, and W. Wang, “MArk: Exploiting cloud services for cost-effective, SLO-aware machine learning inference serving,” *Proc. USENIX Ann. Technical Conf. (USENIX ATC)*, 2019. (conference proceedings)
4. F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, “INFaaS: A model-less inference serving system,” *arXiv preprint arXiv:1905.13348*, 2019. (PrePrint)
5. J. Spillner, C. Mateos, and D. A. Monge, “FaaSter, better, cheaper: The prospect of serverless scientific computing and HPC,” *Proc. CARLA 2017*, *Communications in Computer and Information Science*, vol 796., pp. 154–168, 2018. (conference proceedings)
6. I. Baldini, P. Castro, K. Chang, et al., “Serverless computing: Current trends and open problems,” *Research Advances in Cloud Computing*, pp. 1–20, 2017. (journal)
7. G. McGrath, and P. Brenner, “Serverless computing: Design, implementation, and performance,” *Proc. IEEE Int. Conf. Distributed Computing Systems Workshops (ICDCSW)*, pp. 405–410, 2017. (conference proceedings)
8. Intel, OpenCV [Online]. Available: https://opencv.org/ (URL)
9. Intel, OpenVINO Toolkit - Deep Learning Deployment Toolkit repository. [Online]. Available: https://github.com/openvinotoolkit/openvino (URL)
10. J. Deng, W. Dong, R. Socher, et al., "ImageNet: A large-scale hierarchical image database," *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pp. 248-255, 2009. (conference proceedings)
11. T.-Y. Lin, M. Maire, S. Belongie, et al., "Microsoft COCO: Common objects in context," *Proc. European Conf. Computer Vision (ECCV)*, pp. 740-755, 2014. (conference proceedings)
12. L. Baresi, D. F. Mendonça, M. Garriga, S. Guinea, and G. Quattrocchi, “A unified model for the mobile-edge-cloud continuum,” *ACM Transactions on Internet Technology*, vol. 19, no. 2, pp. 1–21, 2019. (journal)



Unai Elordi is a researcher at Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastian, Spain. His research interests include the optimization of deep neural networks in edge to cloud environments. Unai received a master’s degree in computational engineering and intelligent systems from the Basque Country University (EHU/UPV), Donostia-San Sebastian, Spain. Contact him at uelordi@vicomtech.org.

Luis Unzueta is a senior researcher at Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastian, Spain. His research interests include video-surveillance systems and

human-computer interaction. Luis received a Ph.D. in mechanical engineering from Tecnun, University of Navarra, Donostia-San Sebastian, Spain. Contact him at lunzueta@vicomtech.org.



Jon Goenetxea is a researcher at Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastian, Spain. His research interests include CV and computer graphics. Jon received a master's degree in software engineering from the Basque Country University (EHU/UPV), Donostia-San Sebastian, Spain. Contact him at jgoenetxea@vicomtech.org.

Sergio Sanchez-Carvallido is a researcher at Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastian, Spain. His research interests include machine learning and the deployment of architectural design of cloud services. Sergio received a Ph.D in material science and engineering from Carlos III University of Madrid. Contact him at ssanchez@vicomtech.org.

Ignacio Arganda-Carreras is an Ikerbasque Research Fellow at the University of the Basque Country (UPV/EHU), Donostia-San Sebastian, Spain; Ikerbasque, Basque Foundation for Science, Bilbao, Spain, and Donostia International Physics Center (DIPC), Donostia-San Sebastian, Spain. His research interests include computer vision and bioimage analysis. Ignacio received a Ph.D. in computer science and electrical engineering from the Universidad Autonoma de Madrid, Madrid, Spain. Contact him at ignacio.arganda@ehu.eus.

Oihana Otaegui is in charge of the ITS and Engineering department of Vicomtech Foundation, Basque Research and Technology Alliance (BRTA), Donostia-San Sebastian, Spain, and external professor at the University of Basque Country (UPV/EHU). Her research interests include complex digital signal processing, computer vision and machine learning techniques. Oihana received her Ph.D. in electronic engineering from Tecnun, University of Navarra, Spain. Contact her at ootaegui@vicomtech.org.