

Benchmarking Deep Neural Network Inference Performance on Serverless Environments With MLPerf

Unai Elordi, Luis Unzueta, Jon Goenetxea, and
Sergio Sanchez-Carvallido, Vicomtech Foundation,
Basque Research and Technology Alliance

Ignacio Arganda-Carreras, University of the Basque Country and
Basque Foundation for Science, Donostia International Physics Center

Oihana Otaegui, Vicomtech Foundation, Basque Research and Technology Alliance, University of the Basque Country

// We provide a novel decomposition methodology from the current MLPerf benchmark to the serverless function execution model. We have tested our approach in Amazon Lambda to benchmark the processing capabilities of OpenCV and OpenVINO inference engines. //



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 gained strong 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 has brought newer specialized ML hardware [for example, tensor processing units and vision processing units (referred to 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 by both 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 on its webpage for comparison.

The current version of MLPerf Inference (version 0.5) was developed following a monolithic design (that is, 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 the function execution state), include ephemeral storage (the data are 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 is executed in CPU back-end 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:* These are 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.⁴ proposed the INFaaS 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:* It is used 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 periodically sends a set of inferences per query (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 among 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 Spillner et al.,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 FaaS-ification process, that is, the AUs are functions and methods.

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

- *Interference engine 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 a data set. As shown in Figure 1, first, 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 lifecycle (step 2). Next, the warm-up process executes few function instances to avoid cold start delays during the benchmarking process and hence changing the state of the FaaS container to the warm-up 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 warm-up 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, which contains the paths to the input data set, comprises parameters, such as database input data references, the result-delivering output data, and the 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 postprocessing 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)

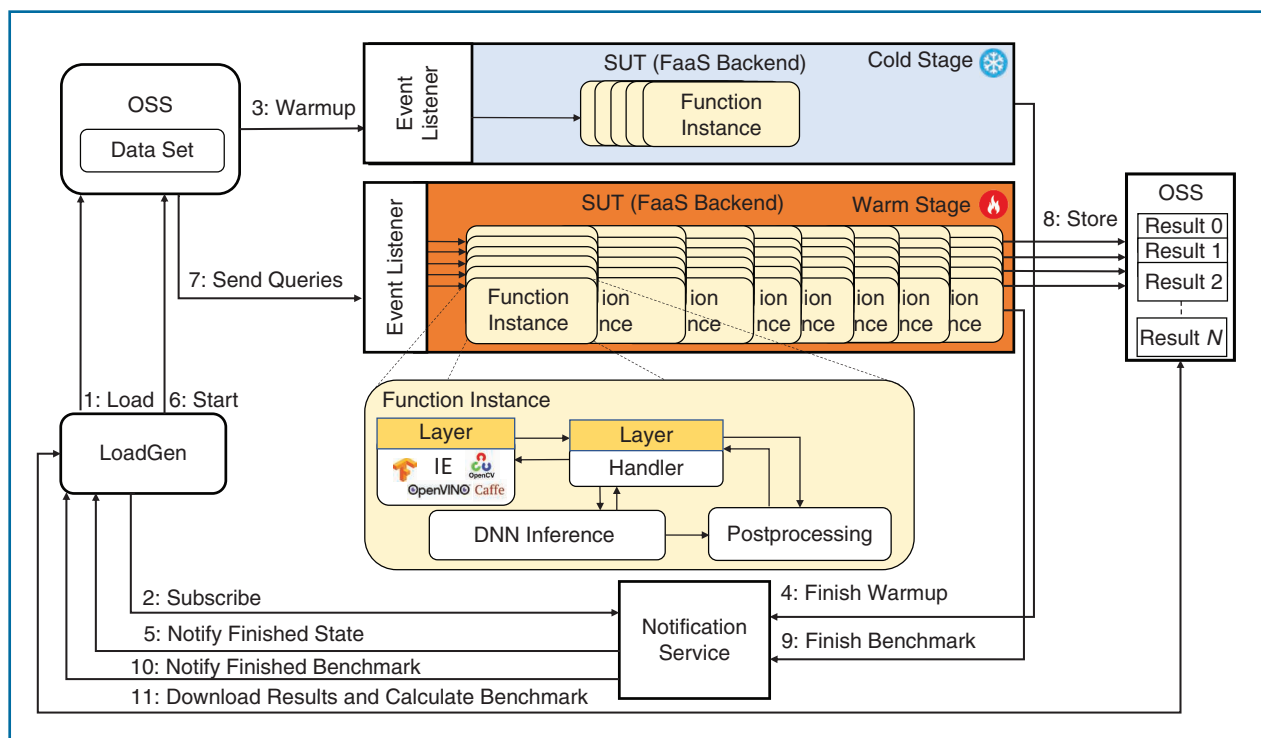


FIGURE 1. The proposed benchmarking FaaS architecture and lifecycle. IE: interference engine.

- the postprocessing 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 (steps 9 and 10). Finally, the LoadGen downloads all the files with the mentioned output values from the OSS (step 11). This information is organized into different lists to calculate the inference latency, throughput, and accuracy results. These are calculated like the following:

- **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:** This is the number of queries divided by the total time. This total time refers to the time difference between the maximum value of the end timestamp and the minimum value of the 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 as the NS. As mentioned previously, we focused our implementation and tests on CV tasks to benchmark the processing capabilities of the DNN inference engines of the OpenCV (OCV)⁸ and the OpenVINO IE,⁹ using Caffe (CF), Tensorflow (TF), 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 an FaaS function supported by Amazon Lambda layers. We make the source code available at (<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.

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

- **data set:** ImageNet¹⁰ and COCO¹¹ (subset of randomized 10K images per data set)
- **performance metrics:** latency and throughput
- **DNN models:** MobileNetV1 and SSDMobileNetV1 for image classification and object detection respectively, with 32 floating point precision in CF, TF, and OpenVINO IR formats
- **FaaS memory configurations:** 768, 1,536, 2,240, and 3,008 MB.

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

Figure 2 depicts the latency time for different function memory configurations, from 768 MB to 3 GB. We observed that while increasing the allocated memory for each function instance, the latency improves in both benchmarked models, especially in ranges between 768 and 1,536 MB. This performance improvement is between 2.12–2.20× times larger for

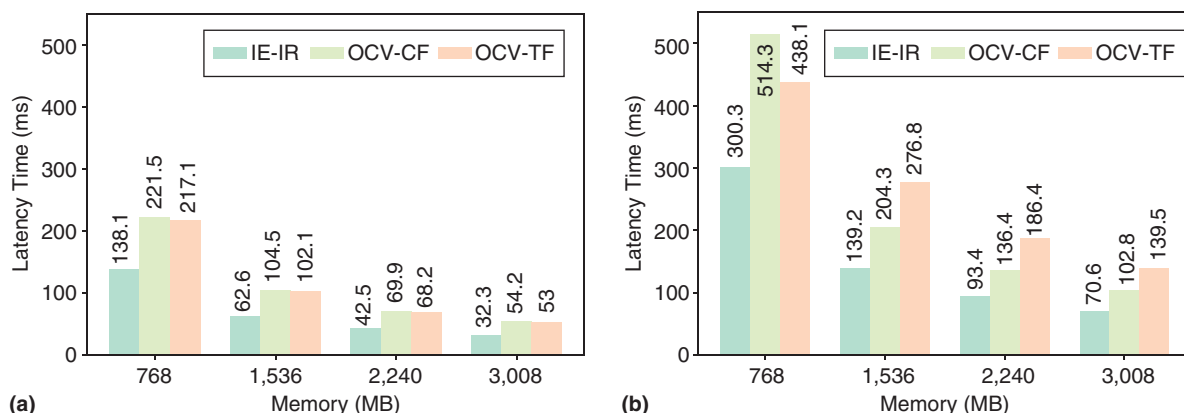


FIGURE 2. The inference latency results with OpenVINO IR (IR), CF, and TF DNN models and OpenVINO IE (IE) and OCV as inference engine. (a) Latency benchmark: MobilenetV1. (b) Latency benchmark: SSDMobilenetV1.

MobileNetV1 and $1.58\text{--}2.55\times$ 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 Amazon Lambda processors currently 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 CF and TF models is quite similar, in SSDMobileNet the reduction of the latency with the CF model is between 1.33 and $1.98\times$ compared to TF.

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 Amazon Web Services cloud provider scheduling resources.

The increasing need to deploy ML tasks at a 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 768 and 1,536 MB.

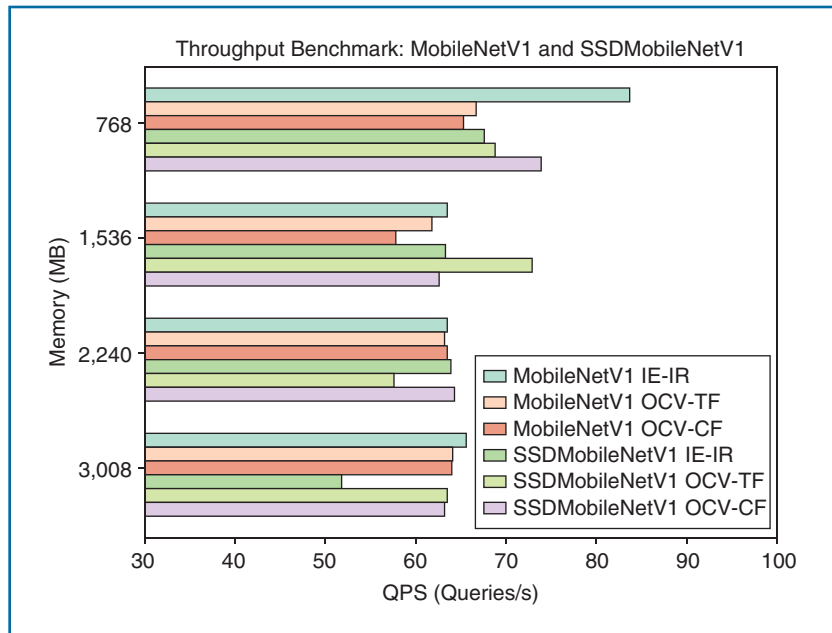



FIGURE 3. The inference throughput results with OpenVINO IR (IR), CF, and TF DNN models and OpenVINO IE (IE) and OCV as interference engine.

Also, the OpenVINO IE and the 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 queries/s (QPS) values make Amazon Lambda a suitable platform for DNN inference.

The design space is still very large—including different serverless environments, benchmarks (in addition to MLPerf), and hardware targets (CPUs, GPUs, xPUs, and so on)—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. 

Acknowledgment

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,” GitHub, San Francisco. Accessed: Oct. 26, 2020. [Online]. Available: <https://github.com/mlperf/inference>
2. P. Maissen, P. Felber, P. Kropf, and V. Schiavoni, “FaaSdom: A benchmark suite for serverless computing,” in *Proc. 14th ACM Int. Conf. Distributed Event-Based Syst. (DEBS '20)*, 2020, pp. 73–84. doi: 10.1145/3401025.3401738.
3. C. Zhang, M. Yu, and W. Wang, “MArk: Exploiting cloud services for cost-effective, SLO-aware



UNAI ELORDI is a researcher at the Vicomtech Foundation, Basque Research and Technology Alliance, Donostia-San Sebastian, Spain. His research interests include the optimization of deep neural networks in edge to cloud environments. Elordi received his master's degree in computational engineering and intelligent systems from the Basque Country University, Donostia-San Sebastian, Spain. Further information about him can be found at <https://www.vicomtech.org/en/vicomtech/team/390>. Contact him at uelordi@vicomtech.org.



SERGIO SANCHEZ-CARVALLIDO is a researcher at the Vicomtech Foundation, Basque Research and Technology Alliance, Donostia-San Sebastian, Spain. His research interests include machine learning and the deployment of architectural design of cloud services. Sanchez-Carvallido received his Ph.D in material science and engineering from Carlos III University of Madrid. Further information about him can be found at <https://www.vicomtech.org/es/vicomtech/equipo/890>. Contact him at ssanchez@vicomtech.org.



LUIS UNZUETA is a senior researcher at the Vicomtech Foundation, Basque Research and Technology Alliance, Donostia-San Sebastian, Spain. His research interests include video-surveillance systems and human-computer interaction. Unzueta received his Ph.D. in mechanical engineering from Tecnun, University of Navarra. Further information about him can be found at <https://www.vicomtech.org/en/vicomtech/team/274>. Contact him at luunzueta@vicomtech.org.



IGNACIO ARGANDA-CARRERAS is an Ikerbasque research fellow at the University of the Basque Country, Donostia-San Sebastian, Spain; Basque Foundation for Science, Bilbao, Spain, and Donostia International Physics Center, Donostia-San Sebastian, Spain. His research interests include computer vision and bioimage analysis. Arganda-Carreras received his Ph.D. in computer science and electrical engineering from the Universidad Autonoma de Madrid. Further information about him can be found at <https://www.ikerbasque.net/en/ignacio-arganda-carreras>. Contact him at ignacio.arganda@ehu.eus.



JON GOENETXEA is a researcher at the Vicomtech Foundation, Basque Research and Technology Alliance, Donostia-San Sebastian, Spain. His research interests include computer vision and computer graphics. Goenetxea received his master's degree in software engineering from the Basque Country University, Donostia-San Sebastian, Spain. Further information about him can be found at <https://www.vicomtech.org/es/vicomtech/equipo/391>. Contact him at jgoenetxea@vicomtech.org.



OIHANA OTAEGUI is in charge of the ITS and Engineering Department at the Vicomtech Foundation, Basque Research and Technology Alliance, Donostia-San Sebastian, Spain, and external professor at the University of Basque Country. Her research interests include complex digital signal processing, computer vision, and machine learning techniques. Otaegui received her Ph.D. in electronic engineering from Tecnun, University of Navarra, Spain. Further information about her can be found at <https://www.vicomtech.org/es/vicomtech/equipo/202>. Contact her at ootaegui@vicomtech.org.

- machine learning inference serving,” in *Proc. USENIX Ann. Technical Conf. (USENIX ATC)*, 2019, pp. 1049–1062.
4. F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, “INFaaS: A model-less inference serving system,” 2019, arXiv:1905.13348.
 5. J. Spillner, C. Mateos, and D. A. Monge, “FaaSter, better, cheaper: The prospect of serverless scientific computing and HPC,” in *Proc. Commun. Comput. Inf. Sci. (CARLA 2017)*, 2018, vol. 796, pp. 154–168.
 6. I. Baldini et al., “Serverless computing: Current trends and open problems,” in *Research Advances in Cloud Computing*, S. Chaudhary, G. Somani, and R. Buyya, Eds. Berlin: Springer-Verlag, 2017, pp. 1–20.
 7. G. McGrath and P. Brenner, “Serverless computing: Design, implementation, and performance,” in *Proc. IEEE Int. Conf. Distributed Comput. Syst. Workshops (ICDCSW)*, 2017, pp. 405–410. doi: 10.1109/ICDCSW.2017.36.
 8. OpenCV. [Online]. Accessed: Oct. 26, 2020. Available: <https://opencv.org/>
 9. Intel, “OpenVINO Toolkit—Deep Learning Deployment Toolkit repository,” GitHub, San Francisco. Accessed: Oct. 26, 2020. [Online]. Available: <https://github.com/openvinotoolkit/openvino>
 10. J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.
 11. T.-Y. Lin et al., “Microsoft COCO: Common objects in context,” in *Proc. Eur. Conf. Comput. Vision (ECCV)*, 2014, pp. 740–755. doi: 10.1007/978-3-319-10602-1_48.
 12. L. Baresi, D. F. Mendonça, M. Garriga, S. Guinea, and G. Quattrocchi, “A unified model for the mobile-edge-cloud continuum,” *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 1–21, 2019. doi: 10.1145/3226644.

IEEE Computer Society Has You Covered!

WORLD-CLASS CONFERENCES — Over 215 globally recognized conferences.

DIGITAL LIBRARY — Over 800k articles covering world-class peer-reviewed content.

CALLS FOR PAPERS — Write and present your ground-breaking accomplishments.

EDUCATION — Strengthen your resume with the IEEE Computer Society Course Catalog.

ADVANCE YOUR CAREER — Search new positions in the IEEE Computer Society Jobs Board.

NETWORK — Make connections in local Region, Section, and Chapter activities.

Explore all of the member benefits
at www.computer.org today!

