Received January 15, 2021, accepted February 4, 2021, date of publication February 8, 2021, date of current version February 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3058021

*L*earning-*i*n-the-*Fo*g (*LiFo*): Deep Learning Meets Fog Computing for the Minimum-Energy Distributed Early-Exit of Inference in Delay-Critical IoT Realms

ENZO BACCARELLI, MICHELE SCARPINITI, (Senior Member, IEEE), ALIREZA MOMENZADEH, AND SIMA SARV AHRABI

Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome, 00184 Rome, Italy

Corresponding author: Michele Scarpiniti (michele.scarpiniti@uniroma1.it)

This work was supported in part by the Sapienza University of Rome Bando 2018 and 2019 through the Project SoFT: Fog of Social IoT, in part by the Sapienza University of Rome Bando Acquisizione di Medie e Grandi Attrezzature Scientifiche 2018 through the Project End-to-End Learning for 3-D Acoustic Scene Analysis (ELeSA), and in part by the Sapienza University of Rome Bando 2020 through the Project DeepFog–Optimized Distributed Implementation of Deep Learning Models Over Networked Multitier Fog Platforms for IoT Stream Applications.

ABSTRACT Fog Computing (FC) and Conditional Deep Neural Networks (CDDNs) with early exits are two emerging paradigms which, up to now, are evolving in a standing-alone fashion. However, their integration is expected to be valuable in IoT applications in which resource-poor devices must mine large volume of sensed data in real-time. Motivated by this consideration, this article focuses on the optimized design and performance validation of *L*earning-*i*n-the-*Fo*g (*LiFo*), a novel virtualized technological platform for the minimum-energy and delay-constrained execution of the inference-phase of CDDNs with early exits atop multi-tier networked computing infrastructures composed by multiple hierarchically-organized wireless Fog nodes. The main research contributions of this article are threefold, namely: (i) we design the main building blocks and supporting services of the *LiFo* architecture by explicitly accounting for the *multiple* constraints on the per-exit maximum inference delays of the supported CDNN; (ii) we develop an *adaptive* algorithm for the *minimum-energy distributed joint* allocation and reconfiguration of the available computing-plus-networking resources of the *LiFo* platform. Interestingly enough, the designed algorithm is capable to *self*-detect (typically, *unpredictable*) environmental changes and quickly *self*-react them by properly re-configuring the available computing and networking resources; and, (iii) we design the main building blocks and related virtualized functionalities of an *Information Centric*-based networking architecture, which enables the *LiFo* platform to perform the aggregation of spatially-distributed IoT sensed data. The energy-vs.-inference delay *LiFo* performance is numerically tested under a number of IoT scenarios and compared against the corresponding ones of some state-of-the-art benchmark solutions that do not rely on the Fog support.

INDEX TERMS Conditional deep neural networks, distributed multi-tier fog platforms, early exit of IoT inference, per-exit inference delays, virtualized networked computing architectures, adaptive resource allocation and reconfiguration.

I. MOTIVATIONS OF THE WORK AND REFERENCE FRAMEWORK

With the advent of the Internet of Things (IoT) era, it is expected that a huge number of (mainly wireless and spatially distributed) resource-limited devices will produce and/or consume large volumes of (possibly, heterogeneous and quickly changing) sensory data. Depending on the nature of the supported IoT application, the raw data generated by these devices may translate into big data streams which may require real-time mining. A white paper by Cisco [1] forecasts that more than 50 billion IoT devices will connect

The associate editor coordinating the review of this manuscript and approving it for publication was Xueqin Jiang.

to the Internet by 2021, so that about 850 Zettabytes of data per-year will be locally generated/consumed by these devices outside the cloud, with the global intra-data center traffic remaining limited up to 21 Zettabytes. This implies, in turn, that producers/consumers of big data will progressively move from large-scale centralized cloud-hosted data centers to a wide range of spatially distributed ''tiny'' edge devices [2]. However, it is expected that stand-alone cloud computing will become gradually unable to process these massive data, mainly because: (i) the IoT traffic generated in upstream towards cloud data centers will inflate the core networks; and, (ii) a spectrum of emerging IoT-supported applications, such as human activity recognition and autonomous car driving, will require to mine of the acquired data under strict real-time constraints, so to make infeasible the transportation of IoT data to remote cloud data centers [2].

Hence, in order to effectively exploit the IoT-driven Big Data Stream tsunami for discovering in real-time fresh information, the carried out analytic and supporting execution technological platforms should meet three main requirements [3]. First, the analytic should be *powerful enough* to effectively mine the heterogeneous and possibly noisy sensing data generated by resource-limited IoT devices. Second, it should be *fast enough*, in order to cope with the stream nature of the IoT data. Third, the analytic should be amenable to *distributed execution*, in order to be compliant with the spatially distributed nature of the IoT devices and, then, give rise to low processing/communication delays.

In principle, the first two requirements could be simultaneously met by employing the emerging paradigm of the so-called Conditional Deep Neural Networks (CDNNs) with early exits [4]–[6] (also referred to as BranchyNets [7], [8]), which provides an effective means to perform Deep Learning (DL) on structured/unstructured IoT data in real-time. A sketch of a CDNN with L layers and N_{EE} early exits is presented in Fig. 1a. Essentially, a CDNN with early-exits is obtained by augmenting the stack topology of a feedforward Baseline Deep Neural Network (B-DNN) with a number of side output branches (i.e., the early exits), which are connected to intermediate local classifiers. The introduction of side branch classifiers enables Early Exit of Inference (EEoI) to make easy-to-classify input data exit early via these branches, with high enough confidence (i.e., high enough reliability).

However, in order to actually implement spatial scaling and guarantee per-layer processing of the input data (see Fig. 1a), the technological platform supporting the execution of a CDNN with early exits cannot rely only on a standing-alone remote and centralized cloud data center. It should be composed, indeed, of the networked interconnection of a number of hierarchically-organized computing nodes, which are spatially scattered and operate nearby the IoT devices. This is the native layout of the emerging paradigm of Fog Computing (FC) [9]–[11]. As sketched in Fig. 1b, an FC technological platform is typically composed of a set of $(M-$

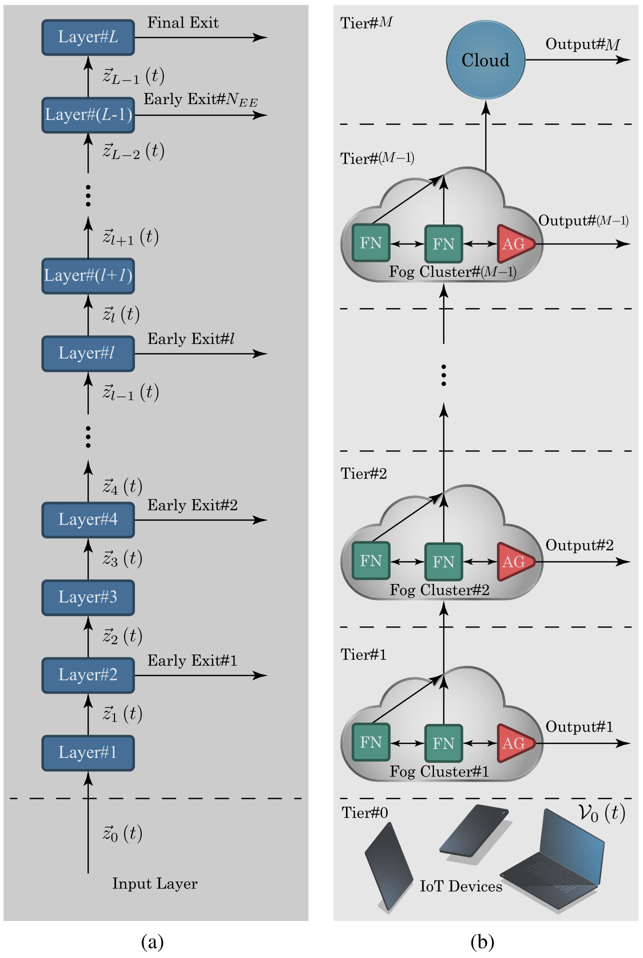


FIGURE 1. (a) Stack topology of a CDNN with L layers, N_{EE} early exits, and a final exit; (b) Hierarchical topology of a networked Fog computing platform with a bottom IoT tier, $(M-1)$ middle clusters of Fog nodes, and a final Cloud node. FN: = Fog Node; AG: = Aggregator.

1) clusters of medium-size virtualized data centers (i.e., Fog Nodes (FNs)), which are hierarchically-organized into tiers and exploit (typically wireless) transport links for enabling inter-tier communication, so to implement a communication path from the lowermost IoT realm at *tier* #0 to the uppermost Cloud Node (CN) at *tier* #M. At each intermediate *tier* #m, with $1 \leq m \leq M-1$, an intra-tier local network allows an Aggregator (AG) node to provide EEoI by suitably merging the outputs of the corresponding FNs into a local output (see the per-tier side branches in Fig. 1b). So doing, it is expected that only a (small) fraction of the volume $\mathcal{V}_0(t)$ of data generated by the IoT devices at time t needs to be transported up to the remote CN for analytics, while a (hopefully, large) part of $\mathcal{V}_0(t)$ early exits through the available intermediate per-tier local outputs.

IoT-CDNN-FC convergence: added value and main challenges — We anticipate that an overview of the related work of Section II points out that, at the present time, two main sets of challenges must be afforded, in order to fully exploit the IoT-CDNN-FC convergence.

A first set of challenges concerns the optimized design and training of the CDNN of Fig. 1a for IoT-oriented real-time applications and embraces:

1) the optimal setting of the number of local exits and their corresponding placement along the stack of layers of Fig. 1a;
2) the design of suitable criteria that allow an optimized delay-vs.-reliability tradeoff, so to balance the two contrasting requirements of generating *large* volumes of early exits with *high* confidence levels;
3) the design of suitable algorithms for the distributed training of the designed CDNN; and,
4) the optimized mapping of the layers of the designed CDNN of Fig. 1a onto the tiers of the underlying execution Fog platform of Fig. 1b.

A second set of challenges involves the optimized design, implementation and running of the distributed multi-tier networked FC technological platform of Fig. 1b and covers:

5) the support of the *virtualized execution* of data analytics, in order to allow the technological platform of Fig. 1b to run in parallel multiple IoT applications without interference;
6) the optimized *joint allocation* of the virtualized computing and networking resources to the FNs of Fig. 1b, so to minimize the resulting total consumed networking-plus-computing energy;
7) the design of suitable *adaptation* strategies that enable the Fog platform of Fig. 1b to track the (possibly unpredictable) time-variations of the IoT operating environment;
8) the design of effective re-configuration mechanisms that enable the Fog platform to be *resilient* against the failures induced by the inherently unreliable nature of the wireless IoT realm; and,
9) the simultaneous support of *multiple delay-constraints* on the inference flows generated by the local outputs of Fig. 1b, as dictated by the real-time nature of the supported IoT applications.

A. THE CONSIDERED VIRTUALIZED FOG EXECUTION PLATFORM

According to the aforementioned challenges, Fig. 2 provides a sketch of the ordered steps to be carry out for the design, training and execution of the CDNN with early exits of Fig. 1a over the distributed Fog technological platform of Fig. 1b.

The first four steps of Fig. 2 concern the training/setup of the considered CDNN with early exits. All these steps have been the focus of our previous contributions in [6], [12], which tackle with the first four challenges in items 1–4 of the previous list. The last two steps of Fig. 2 concern the inference phase of the CDNN's life-cycle and cover the second set of the aforementioned challenges which are listed in items 5–9. These last two steps and the associated (still open) five challenges are the *explicit* focus of this contribution.

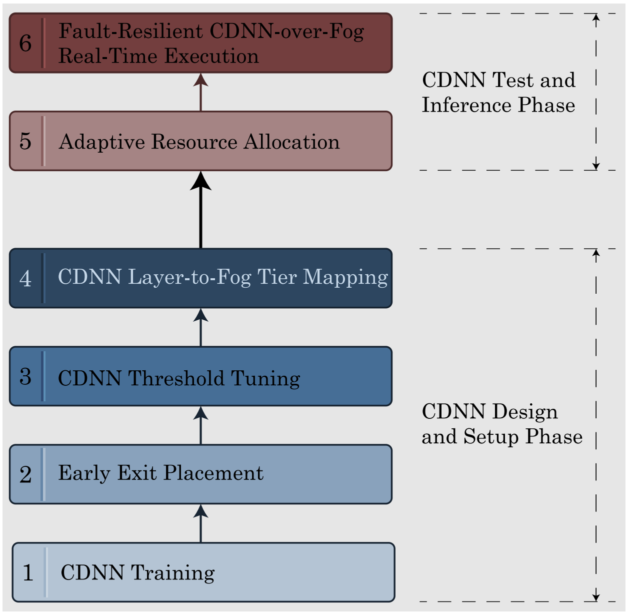

6 Fault-Resilient CDNN-over-Fog Real-Time Execution
5 Adaptive Resource Allocation
4 CDNN Layer-to-Fog Tier Mapping
3 CDNN Threshold Tuning
2 Early Exit Placement
1 CDNN Training
CDNN Test and Inference Phase
CDNN Design and Setup Phase

FIGURE 2. **A sketch of the ordered step sequence of the CDNN's life-cycle.**

According to this last consideration, Fig. 3 shows a sketch of the envisioned multi-tier networked *L*earning-*i*n-the-*Fo*g (*LiFo*) technological platform for the distributed execution of the inference phase of an (already designed, trained and mapped) CDNN with early exits.

Basically, the proposed *LiFo* platform is composed of the hierarchically organized cascade of three main segments, namely:

1) the *lowermost* segment (i.e., *tier* #0), where a set of spatially distributed resource-poor IoT sensors operate. Since current IoT devices (for example, smartphone, tablets and personal assistants, just to name a few) are natively equipped with built-in sensors, IoT devices at *tier* #0 are assumed to be co-located with Fog nodes at *tier* #1 (see the dark red triangles at the bottom of Fig. 3);
2) the *middle* segment, where a number of spatially distributed and networked Fog nodes operates. According to Fig. 3, this segment is hierarchically organized into $(M-1)$ stacked tiers numbered from *tier* #1 to *tier* #$(M-1)$. At *tier* #m, with $1 \leq m \leq M-1$, a cluster: $\{FN(j,m),\ 1 \leq j \leq m_m\}$, with $m_m \geq 1$, of Fog nodes operates in a Peer-to-Peer fashion by exchanging data with a local aggregator $AG(m)$ over an intra-tier Local Area Network (LAN). From time to time, the final *Detector* #m generates a local decision $\hat{c}_m$ (i.e., a class label) on the current input data generated by the sensors at *tier* #0 when the confidence level of the per-tier aggregated data $\overrightarrow{\overrightarrow{Y}}_m$ delivered by $AG(m)$ is high enough. In the opposite case (i.e., when the confidence level of $\overrightarrow{\overrightarrow{Y}}_m$ is estimated to not be high enough), the local output is the empty set: $\emptyset$, that is, no data is generated by the m-th local output;

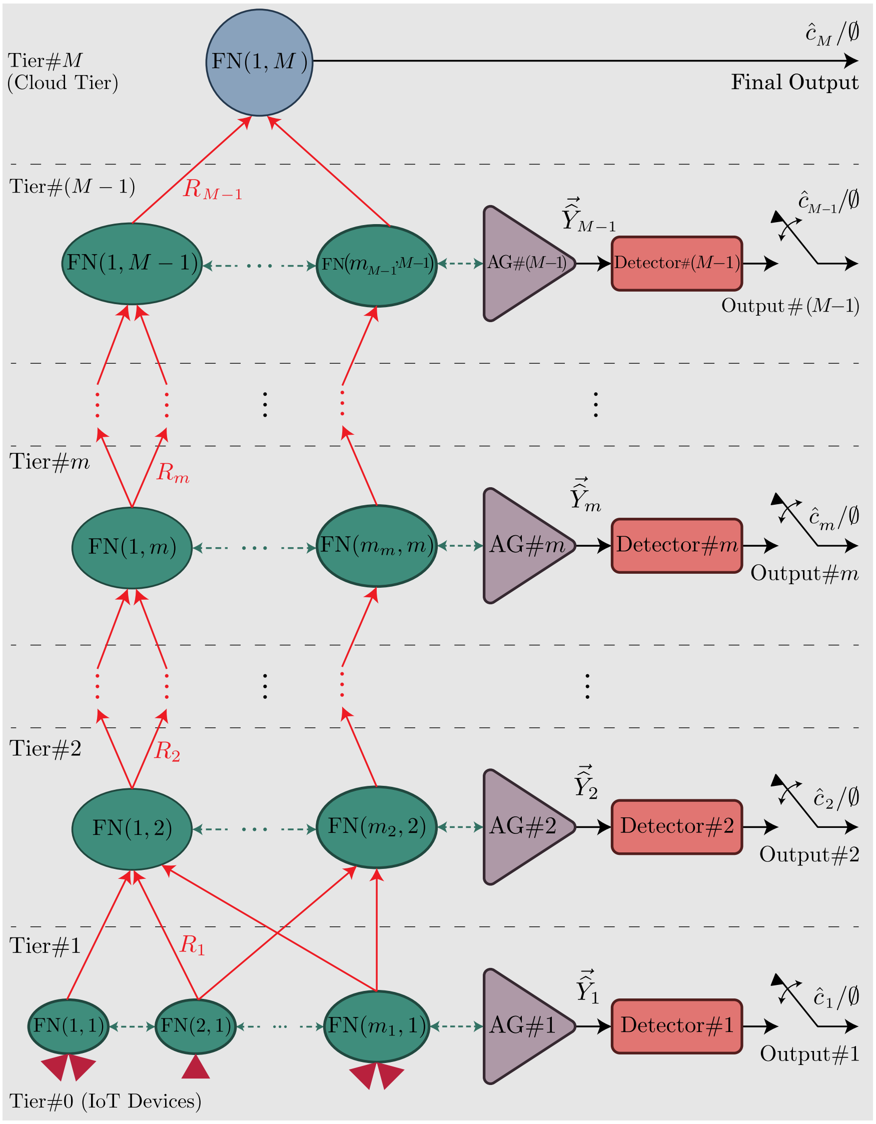


FIGURE 3. **A sketch of the proposed *LiFo* technological platform for the distributed execution of a CDNN with early exits. Continuous red (resp., dashed azure) arrows denote one-way (resp., two-way) TCP/IP (resp., UDP) inter-tier (resp., intra-tier) wireless transport flows. Red triangles denote IoT sensing devices placed at *tier* #0. FN: = Fog Node; AG: = Aggregator; $\hat{c}_m$: = local decision at *tier* #m.**

3) the *uppermost* segment (i.e., *tier* #M), where a single remote Cloud node (labeled as $FN(1, M)$ in Fig. 3) operates. Its task is to perform complex analytics on the most hard-to-classify input instances, so to provide a final decision $\hat{c}_M$ on the class label of the currently sensed input data, regardless of its actual confidence level.

From time to time, the sensors at the bottom of the *LiFo* technological platform of Fig. 3 sense the local surrounding environment and then pass the sensed data in upstream for its tier-by-tier hierarchical mining. At each intermediate *tier* #m, with $1 \leq m \leq (M-1)$, an (average) fraction ρ_m of the input data is passed to the next *tier* #$(m+1)$ for further processing, while the remaining fraction $(1-\rho_m)$ undergoes local exit, in order to produce the corresponding decision $\hat{c}_m$. Due to the real-time nature of the considered IoT application scenario, the inference-delay at which the decision $\hat{c}_m$ exits at each *tier* #m, $1 \leq m \leq M$, is assumed to be limited up to a per-tier upper bound $T_{EXIT}^{(m)}$ (s), $1 \leq m \leq M$, whose actual value is set on the basis of the Quality of Service (QoS) requirements of the supported application.

In order to implement the aforementioned hierarchically-organized data processing with intermediate local outputs, the Fog nodes of the *LiFo* technological platform in Fig. 3 must be equipped with both networking and computing capabilities. Specifically, the required inter-node message passing is implemented by a number of inter-tier and intra-tier transport connections (see the arrows of Fig. 3).

Up-link communication between Fog nodes falling in adjacent tiers is supported by a number of TCP/IP one-way reliable connections, which are sustained by wireless (possibly, single-hop and WiFi-based) communication channels (see the continuous red arrows in Fig. 3 between adjacent tiers). Since load balancing is assumed to be performed at the outputs of each tier, all the transport connections going from Fog nodes at *tier* #m to Fog nodes at *tier* #$(m+1)$ operate at a same bit-rate of R_m (bit/s).

Horizontal communication between the Fog nodes and the Aggregator falling into a same tier is assured by an intra-tier (wired or wireless) LAN which relies on (fast, i.e., connection-less) UDP/IP two-way transport connections (see the dashed green arrows in Fig. 3). Being of local-type and used only for (sporadic) aggregation operations, these intra-tier connections are assumed to operate at low bit-rates, so that the impact of their energy consumption is expected not to be so substantial.

B. FOG VIRTUALIZATION

Consider the computing capability needed by the Fog nodes for carrying out the real-time processing of the sensed input data, we note that, in principle, the technological platform of Fig. 3 may run in parallel *multiple* CDNNs which carry out different analytics on the same set of sensed data by resorting to the virtualization of the full spectrum of available physical resources [3].

Hence, according to these considerations, we assume that all Fog/Cloud nodes of Fig. 3 are equipped with software clones of the (possibly, multiple) run CDNNs. The number of clones simultaneously hosted by each FN equates to the number of (possibly, multiple) CDNNs which are running in parallel over the *LiFo* technological platform of Fig. 3. So doing, each clone is fully dedicated to the execution of a single associated CDNN, then it acts as a virtual "server" by providing resource augmentation to the tied "client" CDNN. For this purpose, each clone is executed by a container that is instantiated atop the hosting FN, so that it is capable of using (through resource multiplexing) a slice of the physical computing, storage, and networking resources of the hosting FN.

Fig. 4 details a logical view of the resulting virtualized container-based FN.

Specifically, according to Fig. 4, $FN(j, m)$, $1 \leq j \leq m_m$, $1 \leq m \leq M$, in Fig. 3 hosts a number $nc(j, m)$ of containers equal to the number of the executed CDNNs. All containers hosted by $FN(j, m)$ share: (i) a same Host Operating System (HOS); and, (ii) the pool of computing (i.e., CPU cycles), networking (i.e., bandwidth and I/O Network Interface Cards (NICs) and switches) and storage physical resources done available by the hosting FN. The task of the Container Engine of Fig. 4 is to allocate these physical resources to the requiring containers by performing dynamical multiplexing.

The execution of the tasks assigned to each clone is orchestrated and synchronized by the corresponding Clone Manager of Fig. 4. For this purpose, under the *LiFo* paradigm, the logical architecture envisioned for a clone is shown in Fig. 5.

Specifically, the j-th clone implemented at *tier* #m (in short, $clone(j, m)$, with $1 \leq j \leq m_m$, and $1 \leq m \leq M$) is equipped with: (i) Virtual Input/Output Ports (see the blocks labeled as *VIP* and *VOP* in Fig. 5); (ii) a virtual processor composed of the cascade of two virtual cores (see the blocks labeled as *VC* #1 and *VC* #2 in Fig. 5); and, (iii) a virtual input cache and two virtual output caches (see the blocks in Fig. 5 labeled as *VIC*, *VOC* #1 and *VOC* #2, respectively). The inter-tier and intra-tier flows received by the input port of the clone are temporarily queued at the input cache for attaining inter-flow synchronization, and then are processed by *VC* #1 and *VC* #2 at the processing speeds f_{jm} (bit/s) and $\tilde{f}_{jm}$ (bit/s) dictated by the Clone Manager of Fig. 4. The tasks of *VC* #1 and *VC* #2 are to perform the processing operations required for the implementation of the assigned CDNN layers and associated early exits, respectively (see Fig. 1a). After the buffering at the *VOC* #1 and *VOC* #2 output ports for synchronization purpose, the data flow generated by *VC* #1 (resp., *VC* #2) is forwarded to the FNs at the next *tier* #$(m+1)$ (resp., to the Aggregator at *tier* #m) through the (previously introduced) inter-tier (resp., intra-tier) transport connections of Fig. 3.

C. CONTRIBUTIONS OF THE PAPER

On the basis of the overview of the related work presented in Section II, and by referring to the inference phase of the CDNNs' life-cycle in Fig. 2, the main contributions of this article may be summarized as follows:

1) we define the main building blocks, design the virtualized functional architecture and define the supported service model of *LiFo*, i.e., the proposed virtualized technological platform for the real-time distributed execution of the inference phase of CDNNs with early exits under IoT realms. In particular, we propose an architecture for the implementation of the intra-tier local networks of Fig. 3, which is inspired by the emerging paradigm of the so-called Information Centric Networking (ICN) [13];
2) we analyze and model the computing and networking energy consumed by the virtualized Fog nodes of Fig. 3 by explicitly accounting for the real-time nature of the supported IoT-oriented CDNN applications;
3) we formally define the *LiFo* *O*ptimization *P*roblem (shortly, *LOP*) for the minimum-energy joint allocation of the networking $\{R_m,\ 1 \leq m \leq M\}$ (see Fig. 3) and computing $\{f_{jm}, \tilde{f}_{jm},\ 1 \leq m \leq m_m, 1 \leq m \leq M\}$ (see Fig. 5) virtualized resources over the nodes of the technological platform of Fig. 3 under the (aforementioned) delays $\{T_{EXIT}^{(m)},\ 1 \leq m \leq M\}$ on the per-tier inference decisions generated by the per-tier local outputs of the platform in Fig. 3. Furthermore, we also provide necessary and sufficient conditions for the convexity and feasibility of the considered *LOP*;
4) we develop an *adaptive* solving approach to the *LOP* that is based on primal-dual gradient-based iterations

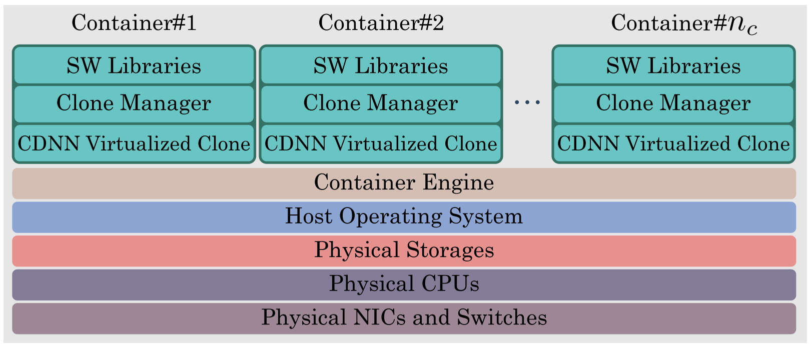


FIGURE 4. **Logical view of a virtualized container-based Fog node.**

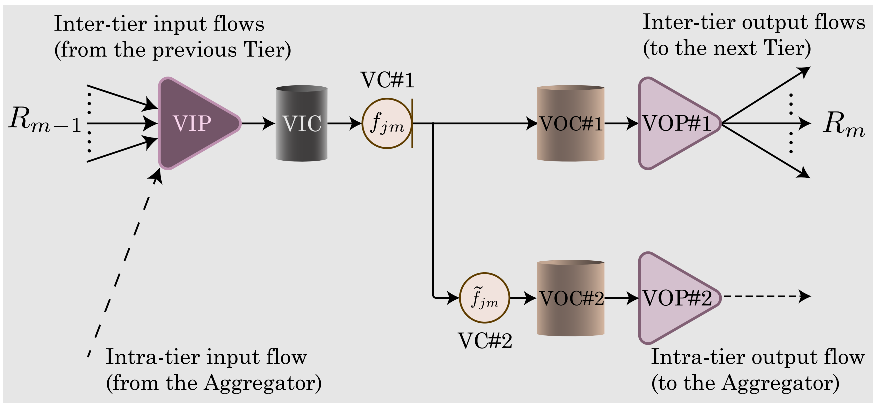

Inter-tier input flows (from the previous Tier)
Inter-tier output flows (to the next Tier)
R_{m-1}
VIP
VIC
VC#1
f_{jm}
VOC#1
VOP#1
R_m
$\tilde{f}_{jm}$
VOC#2
VOP#2
VC#2
Intra-tier input flow (from the Aggregator)
Intra-tier output flow (to the Aggregator)

FIGURE 5. **Logical view of the *j*-th *LiFo* clone implemented at *tier #m*. Continuous (resp., dashed) arrows denote inter-tier (resp., intra-tier) network flows. f_{jm} (resp., $\tilde{f}_{jm}$) denotes the processing speed of *VC* #1 (resp., *VC* #2). R_{m-1} (resp., R_m) is the bit rate of each input (resp., output) inter-tier data flow. VIP: = Virtual Input Port; VOP: = Virtual Output Port; VIC: = Virtual Input Cache; VOC: = Virtual Output Cache; VC: = Virtual Core.**

with time-varying step-size parameters. It allows the *LiFo* technological platform of Fig. 3 to *self*-detect and *self*-track the (typically, unpredictable) time-variations of the underlying operating environment, so to *self*-reconfigure the available computing-plus-networking resources. Interestingly enough, we develop a (seemingly new) design of the involved step-size parameters, which allows the *joint adaptation* of *both* the step-size values *and* their allowed value ranges on the basis of the current gaps between the actually experienced inference delays and their target values $\{T_{EXIT}^{(m)}, 1 \leq m \leq M\}$. In this way, we have numerically ascertained that the resulting reconfiguration delays suffered by the *LiFo* technological platform after (unpredicted and abrupt) failure events are up to 10 times less than the corresponding ones which are experienced when the values ranges of the step-sizes are held to be fixed;

5) we numerically test and compare the sensitivity of the steady-state energy performance of the proposed *LiFo* technological platform on a number of system parameters (i.e., target inference delays, volume of the sensed input data, actual topology of the Fog platform and volume of the actually generated early-exit inference) under some test CDNNs with early exits of practical interest in the IoT realm [12]. Interestingly enough, we anticipate that the carried out numerical performance comparisons support the conclusion that the (average) energy reduction provided by the proposed *LiFo* technological platform compared to traditional Cloud-based computer networks, is quite impressive and it may be larger than 50% under tree-shaped Fog topologies with depth (e.g., number of tiers) larger than 3–4.

D. ROADMAP OF THE PAPER

As reported by the roadmap of Fig. 6, the paper is organized into four main parts.

The first part embraces the Sections I and II on the reference framework and related work, respectively.

The second part focuses on the (multi-facet) *LiFo* architectural aspects and covers: (i) Section III, where the CDNN architecture is reviewed; (ii) Section IV, where we develop the proposed virtualized architecture of the *LiFo* Fog nodes and carry out a related analysis of the entailed inference delays; and, (iii) Section V, where we introduce some models

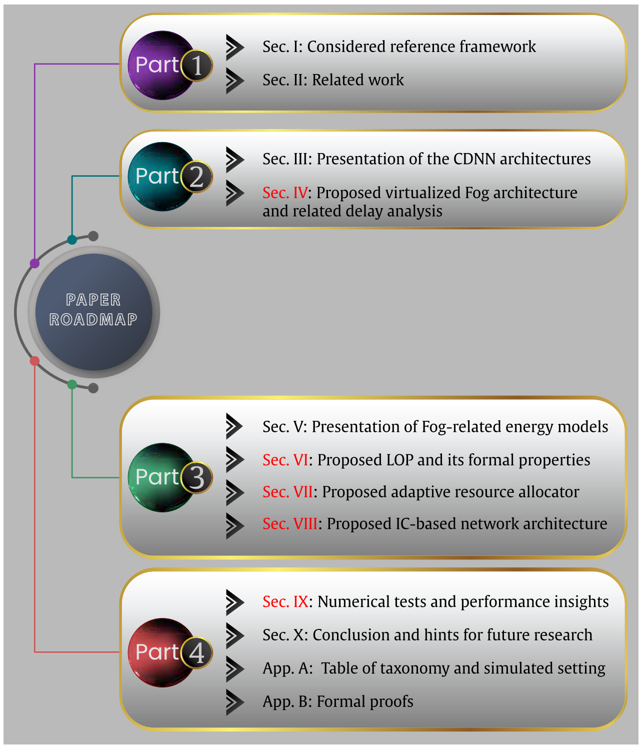

Part 1
Sec. I: Considered reference framework
Sec. II: Related work
Part 2
Sec. III: Presentation of the CDNN architectures
Sec. IV: Proposed virtualized Fog architecture and related delay analysis
PAPER ROADMAP
Part 3
Sec. V: Presentation of Fog-related energy models
Sec. VI: Proposed LOP and its formal properties
Sec. VII: Proposed adaptive resource allocator
Sec. VIII: Proposed IC-based network architecture
Part 4
Sec. IX: Numerical tests and performance insights
Sec. X: Conclusion and hints for future research
App. A: Table of taxonomy and simulated setting
App. B: Formal proofs

FIGURE 6. **The paper roadmap. The red-marked Sections are the ones where the major research contributions are provided. Sec.: = Section, LOP: = *LiFo* Optimization Problem; IC: = Information Centric; App.: = Appendix.**

for formally characterizing the computing and networking energy consumptions of the designed Fog-node architecture.

The third part of the paper is mainly devoted to: (i) the definition of the afforded *LOP* and the analysis of its structural properties (see Section VI); (ii) the development of the *LOP* solving approach and the presentation of the resulting adaptive *LiFo* resource allocator (see Section VII); and, (iii) the design of the ICN-inspired proposed architecture for the support of intra-tier local networking (see Section VIII).

Finally, the last part of the paper covers: (i) Section IX, where we present a spectrum of numerical tests which allow us to gain insight about the performance sensitivity of the proposed *LiFo* technological platform on the main parameters featuring the underlying Fog operating scenario; (ii) Section X, where we draw some conclusions and address some hints for future research; (iii) Appendix A, where we summarize the adopted paper's taxonomy, together with the simulated setting; and, (iv) Appendix B, where the formal proofs of some main formal results are provided.

Passing to consider the adopted notation, we note that $\vec{z}$ indicates an n-dimensional column vector whose i-th scalar component is $z(i)$, $|\mathcal{S}|$ is the cardinality of the set $\mathcal{S}$, $\mathcal{S}^n$ indicates the n-fold Cartesian product of the set $\mathcal{S}$ by itself, while $[A] \triangleq [a_{ij}]$ denotes a matrix, whose (i, j)-th element is a_{ij}. Furthermore, $\dim(\vec{z})$ is the size in bit of the binary vector $\vec{z}$, $\triangleq$ means equal by definition, $\delta(x)$ is the Kronecker's delta function (i.e., $\delta(0) = 1$, and $\delta(x) = 0$ for $x \neq 0$), and: $\emptyset$ denotes the empty set. Finally, $E(\cdot)$ is the expectation operator, while main assumptions are marked by bullets.

II. RELATED WORK AND PAPER POSITIONING

An overview of the recent literature on DL in the wireless realm points out that resource-augmentation of resource-limited IoT devices through edge computing may be provided according to four basic service modes, namely the *Offloading*, *Hierarchical*, *Peer-to-Peer* and *Hybrid* service modes [14]. Fig. 7 shows the (abstract) logical views of these service modes. According to this observation, in the following, we provide a review of the main literature on the convergence-integration of the Deep Learning and Fog Computing paradigms by using these service modes as roadmap.

A. OFFLOADING SERVICE MODE

Under this service mode, a resource-poor IoT device operating at *tier* #0 may decide to offload a (more or less large) number of its pending tasks to a nearby Fog node and/or a remote Cloud located at *tier* #1 by exploiting the Radio Access Network (RAN) as inter-tier communication medium [14]. Specifically, under the *Full Offloading* service mode, the IoT device offloads all its tasks or executes all them by itself, while the *Partial Offloading* service mode allows the IoT device to locally execute a part of its tasks (see the self-loop in Fig. 7a) and offloads the remaining ones.

The common focus of the *MCDNN* [15] and *DeepDecision* [16] frameworks is on the design and management of technological platforms for supporting the Full Offloading service mode. Specifically, the authors of [15] address the problem to execute DNNs on resource-poor IoT devices which are connected to the Cloud in an intermittent way. The proposed solution combines an optimization engine that generates a set of different versions of the DNN to be executed and a run-time scheduler which decides whether the (full) execution of each DNN version must happen at the device or at the Cloud. For this purpose, the decision criterion pursued by the scheduler is the maximization of the resulting DNN inference accuracy, while meeting the constraints on the device computing resources. *DeepDecision* in [16] generalizes the decision criterion pursued by *MCDNN* by including in the decision process also the effects of the local-vs.-remote inference latency and network conditions. Overall, like our paper, these contributions afford the optimized execution of a DNN over distributed computing platforms, but, unlike our work, the resulting *MCDNN* and *DeepDecision* frameworks: (i) do not consider the minimization of the energy consumption as key objective; and, (ii) do not consider the presence of early exits as a means for speeding-up the inference process.

The target of the contributions in [17], [18] and [19] is to enable partial offloading for the distributed execution of DNNs characterized by hierarchically-organized stack-shaped DAGs. For this purpose, *DeepWear* in [17] dynamically offloads DL tasks from a wearable device to its paired handheld device over short-range (typically, Bluetooth-based) connections. For this purpose, it relies on

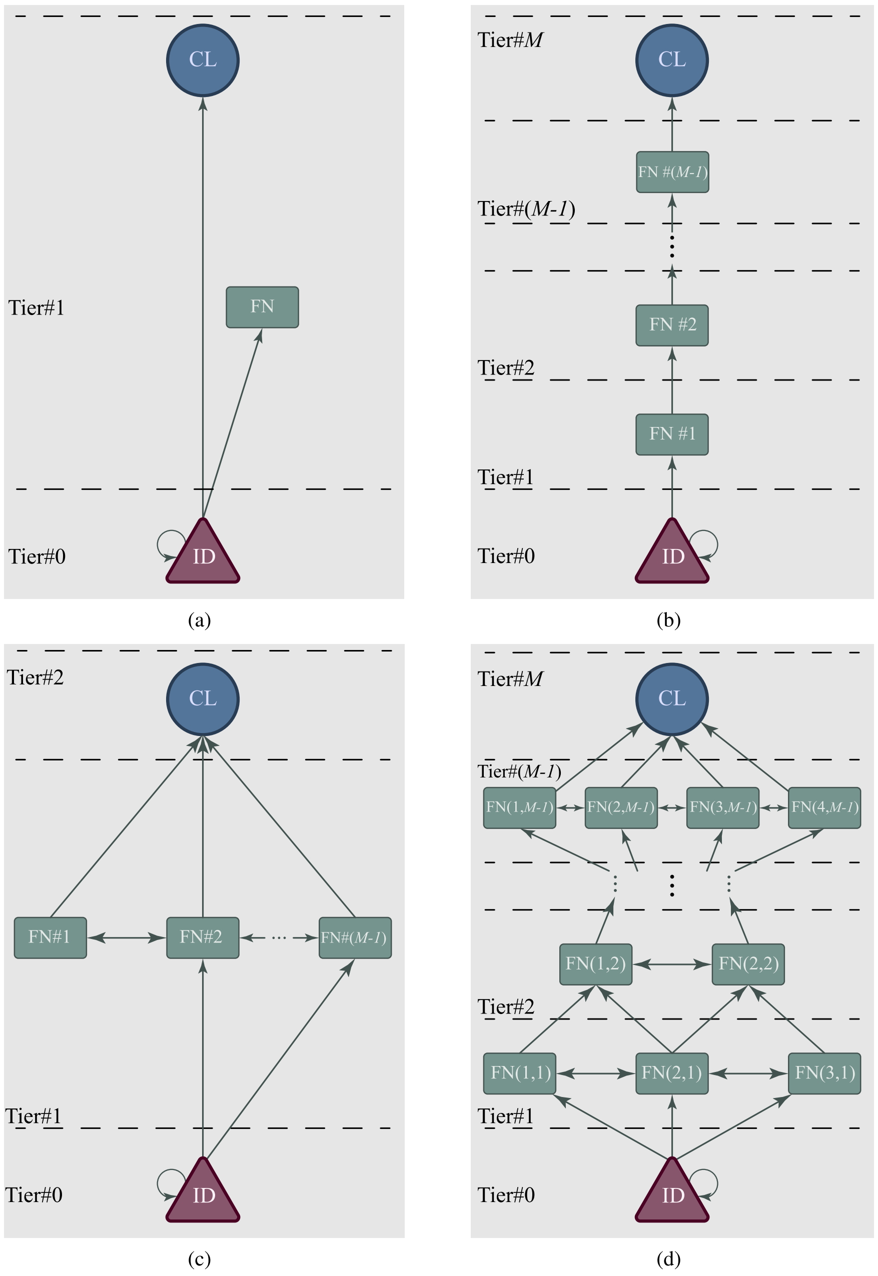


FIGURE 7. **Service modes in IoT-Fog-Cloud ecosystems for the support of DL wireless applications. (a) Offloading mode; (b) Hierarchical mode; (c) Pier-to-Pier mode; (d) Hybrid mode. The arrows indicate data paths. ID: = IoT Device; FN: = Fog Node; CL: = CLoud.**

the synergic exploitation of a number of strategies, like, for example, context-awareness, DNN model partitioning and pipelining. In order to reduce the start-up delay, *IONN* in [18] proposes a partitioning-based technique in which the device divides the DNN layers into a few subsets and offloads them one by one, so to allow the receiving edge server to immediately start up the DNN execution, while incrementally building the overall DNN model as each DNN partition arrives. In order to dynamically plan the best DNN partition and the offloading ordered sequence, *IONN* proposes a novel graph-based algorithm. *Edgent* in [19] considers the partial offloading of a CDNN with early-exits over a two-node execution platform composed of the device and a nearby edge data center. The target is the joint selection of the 2-way device-edge partition of the CDNN layers and the placement of their early-exits, in order to maximize the resulting inference accuracy under hard constraints on the allowed inference delay. Overall, like our proposal, all the contributions in [17], [18] and [19] consider the problem of the optimized execution of DNNs over distributed execution platforms under various system constraints, but, unlike our paper, they do not afford the problem of the energy-saving dynamic joint scaling of the available computing and networking resources.

Finally, the recent contribution in [20] proposes *DeSVig*, a decentralized swift vigilance framework for the detection of adversarial attacks in artificial intelligence-empowered industrial applications. Interestingly, by leveraging a two-tier decentralized architecture which exploits proximate virtualized Fog servers for resource augmentation, *DeSVig* allows the reliable detection of abnormal inputs with sub-millisecond latencies by exploiting the mining capability of suitably designed Generative Adversarial Networks (GANs). Hence, like our proposal, *DeSVig* exploits proximate Fog nodes for providing reliable resource-augmentation with low latencies. However, unlike our contribution, the authors of [20] do not address the topics of the adaptive resource allocation and resource re-configuration.

B. HIERARCHICAL SERVICE MODE

Relying on the resource augmentation provided by a limited number of edge nodes, it is expected that the previously described offloading strategies may be effective for the execution of computation-light shallow Machine Learning (ML) applications, while they may fail to adequately support more cumbersome DL-based algorithms [14]. Motivated by this consideration, the Hierarchical service mode of Fig. 7b leverages the hierarchical topology typically featuring the supported DNNs of Fig. 1a, in order to organize a set of FNs as a *stack* composed of an (in principle, arbitrary) number of interconnected tiers and then provide vertical-scaling capability to the resulting execution platform (see Fig. 7b).

This is, indeed, the reference service mode considered by the *Neurosurgeon* [21], *BranchyNet* [7] and *Cascading NN* [22] frameworks. Specifically, the contribution in [21] performs the profiling of all layers of a DNN both at the device and edge node, in order to generate suitable performance prediction models. Afterwards, based on these prediction models and the currently experienced operating conditions, *Neurosurgeon* evaluates each feasible layer partitioning in terms of inference latency and energy consumption, then partition the DNN accordingly, in order to execute it over a two-tier distributed computing platform composed of the mobile device and a remote cloud.

The main feature common to *BranchyNet* in [7] and *Cascade NN* in [22] is the presence of multiple early-exits in the topologies of the considered DNNs. Specifically, the authors of [7] consider the problem of the optimized placement of the early-exits over the stack of the layers of a baseline DNN. The pursued goal is the reduction of the communication and computing resource usages, while attaining a good tradeoff among the two contrasting requirements of reliable and fast EEoI. A similar goal is pursued by the authors of [22], who propose a new architecture (referred to as ''cascade network'') for the distributed execution of a DNN between a local device and the cloud. The cascade network begins to run on the device, and resorts to employ an edge data center, when the local part of the network does not provide reliable inference. For this purpose, the cascading network allows for an early-stopping mechanism during the inference phase of the network. The performance of the proposed cascading network is numerically evaluated in terms of the attained inference accuracy-vs.-inference delay tradeoff.

Overall, like our contribution, the works in [7], [21] and [22] explicitly consider early-exits as a means to attain good inference reliability-vs.-inference delay tradeoffs, while reducing the communication traffic in the distributed multi-tier execution of DNNs. However, unlike our proposal, these works consider static and failure free-operating scenarios and then do not afford the topic of the dynamic scaling of the available computing-plus-communication resources.

C. PEER-TO-PEER SERVICE MODE

The emerging spectrum of DL applications requires to gather data sensed by a number of spatially scattered (and possibly mobile) IoT devices for data fusion/mining [14]. This requires, in turn, that the underlying execution platform is equipped with spatial-scaling capability, as natively provided by the Peer-to-Peer service mode of Fig. 7c. Under this service mode, a number of spatially-distributed and horizontally interconnected FNs operate at *tier* #1 of the underlying execution platform in a peer-to-peer way, so to allow a (possibly mobile) IoT device at *tier* #0 to change, from time to time, its service point *without* service interruption. Sporadically, the FNs may require further resource augmentation to a remote Cloud node which operates at *tier* #2 of the resulting technological platform (see Fig. 7c).

This is, indeed, the service mode exploited by the *MoDNN* [23], *DeepThings* [24], *DeepCham* [25], *LAVEA* [26] and *ECO* [27] frameworks.

Specifically, user mobility is the focus of the *MoDNN* in [23], which proposes a distributed mobile computing system for the parallel execution of DNN applications. Main feature of *MoDNN* is its capability to partition already trained DNN models over several cooperative mobile peer devices. The involved peer devices run in parallel, so to speed up the DNN execution through resource pooling, while minimizing the resulting non-parallel data delivery delay.

DeepThings in [24] proposes a framework for the parallel execution of Convolutional Neural Networks (CNNs) on peer-to-peer distributed computing platform. The peculiar feature of the *DeepThings* approach is that it exploits the convolutional nature of the underlying DNN for performing a vertical (i.e., bottom-up) partition of the DNN layers into independent parallel tasks, i.e., the so-called Fused Tile Partitioning. Furthermore, it also develops a distributed work stealing scheduler, in order to dynamically enable workload balancing among the available set of co-working nodes at runtime.

Customizing edge infrastructure for supporting DL is the common feature of the contributions in [25], [26] and [27]. Specifically, the paper [25] focuses on the development on an edge-placed orchestration platform for coordinating the DNN training over a set of cooperative peer devices, while the authors of [26] design a number of distributed algorithms for optimizing the DNN task placement, so to enable inter-peer collaboration and minimize the resulting service response time. Finally, in order to enable the distributed execution of DNN models by spatially-scattered peer nodes, *ECO* in [27] develops a graph-based peer-to-peer overlay network in order to: (i) track pipelines and task dependencies; and, (ii) map them onto the available geographically distributed computing system.

Overall, like our paper, all the reviewed contributions aim at providing spatial-scaling capabilities to the underlying technological execution platform, so to support device mobility and distributed sensing. However, unlike our proposal, the reviewed papers: (i) do not tackle aspects related to the adaptation and re-configuration of the considered peer-to-peer execution platforms; and, (ii) do not address the topic of EEoI.

D. HYBRID SERVICE MODE

According to Fig. 7d, the goal of the (recently proposed) Hybrid service mode is to provide both vertical and horizontal scaling capabilities for the implemented execution platform through the hierarchical inter-connection of multiple tiers of clusters, with each cluster being composed of the peer-to-peer horizontal inter-connection of a number of cooperative FNs. If required, local outputs may be also provided by equipping the clusters with Aggregator nodes.

Being still an emerging paradigm, only a limited number of contributions explicitly rely on this service mode [8], [28] and [29]. Specifically, *D-DDN* in [8] develops and discusses the main implementation aspects of a computing architecture for the distributed execution of a CDNN with early-exits over spatially and vertically-organized hierarchies, composed of Cloud nodes, Fog nodes and IoT devices. Besides some architectural aspects, the focus of the paper is on the role played by the aggregator nodes in achieving distributed data fusion, as well as on the distributed training of the supported CDNN. The *DeepFog* contribution in [28] is more application-oriented. It presents a Fog-based deep learning model for the execution of DNNs *without* early-exits that aims at collecting data from users. The final goal is to predict the user wellness states by exploiting the mining capability of a supported DNN that can process heterogeneous and multi-dimensional data. For attaining this goal, three abnormalities in the user wellness (namely diabetes, hypertension attacks and stress type classification) are considered for the numerical DNN test. Finally, the authors of [29] propose and validate *EdgeLaaS*. It is an edge learning-as-a-service framework that relies on an ICN-inspired networking architecture and exploits the emerging Knowledge-Centric Connected Healthcare (KCCH) paradigm, in order to process patients' healthcare supervision data with the right knowledge of the right guardians (e.g., nurses and/or doctors). For attaining this goal, *EdgeLaaS* relies on a hybrid service mode in which: (i) different Reinforcement Learning-based processing models are hierarchically organized over different Fog tiers; and, (ii) all Fog nodes falling into a same tier cooperate to complete their shared learning tasks.

Overall, like our paper, the contributions in [8], [28], and [29] consider aspects related to the spatially distributed mining of DNNs, but, unlike our proposal, all these contributions do not consider adaptive resource allocation and do not formally analyze the performance effects of the enforced maximum delays on the per-exit inference time.

Table 1 provides a summarizing synoptic view of the aforementioned related work. Overall, on the basis of the carried out research overview, we may conclude that the peculiar features of the proposed *LiFo* technological platform are the following ones. Its ultimate goal is the *minimization* of the computing-plus-networking energy wasted by the *distributed* execution of a CDNN with early-exits over a Device-Fog-Cloud networked infrastructure composed of the *hierarchical* interconnection of a number of *spatially-distributed* Fog clusters placed at *multiple tiers*. In order to provide EEoI under *delay-constraints* on the maximum (i.e., worst-case) per-exit inference delays, a scheduler is designed that exploits the *virtualization* of the available computing-networking resources by *dynamically* performing resource allocation and platform reconfiguration. This is accomplished by accounting for: (i) the (possibly) time-varying conditions of the TCP/IP inter-tier wireless connections; (ii) the random fluctuations of the volume of the sensed data to be processed by the supported CDNN; and, (iii) the maximum bandwidth and computing resources actually available for the execution of the virtualized clones hosted by each Fog node.

III. DISTRIBUTED EXECUTION OF CDNNs WITH EARLY EXITS – FUNCTIONAL ARCHITECTURES

The main task of the IoT devices operating at *tier* #0 of Fig. 1b is to sense the environment at discrete instants and, then, generate a set of input data which is mined by running the CDNN of Fig. 1a over the Fog-Cloud computing platform of Fig. 1b. The goal is to deliver a (hopefully, reliable and fast) inference decision on the pattern actually present in the input data. For this purpose, in the *LiFo* framework, the time axis is partitioned into time-slots which are labeled by a discrete-time slot-index $t \geq 0$. In the case of periodic sensing, the slot duration is fixed at T_S (measured in second, i.e., (s)), so that the t-th slot spans the semi-open time-interval: $[tT_S, (t+1)T_S)$. In the more general case of event-driven sensing, the duration $\Delta_S(t)$ of the t-th slot depends on the slot-index t, so that the t-th slot covers the semi-open interval: $[\xi(t), \xi(t) + \Delta_S(t))$, where: $\xi(t) \triangleq \sum_{i=0}^{t-1} \Delta_S(i)$ is the starting time of the t-th slot. We anticipate that all the formal results presented in the sequel hold verbatim in both cases of periodic and event-driven sensing, provided that, in the second case, the minimum slot duration is lower bounded by T_S, that is, $\Delta_S(t) \geq T_S, \forall t$.

It is further assumed that the sensing action is performed by the IoT devices of Fig. 1b at the *beginning* of each slot t and it generates an n_0-dimensional (with $n_0 \geq 1$) *binary*-valued data vector $\vec{z}_0(t)$, which, by design, gathers the set of bits of the input feature to be mined by the CDNN of Fig. 1b at slot t. The corresponding size (measured in

TABLE 1. Comparative overview of main related work and *LiFo* positioning.

Service Mode	Reference	Main Focus/Target
Offloading Mode	*MCDNN* [15]	A full offloading decision is taken by optimizing the inference accuracy-vs.-battery capacity tradeoff.
	DeepDecision [16]	The criterion followed for planning the full offloading decision accounts for multiple factors, like inference accuracy, inference delay, residual battery capacity, network bandwidth, network fading level and network congestion level.
	DeepWear [17]	Design model partitioning, context-aware offloading and pipelining schemes for the best utilization of edge computing in wearable environments.
	IONN [18]	Perform dynamic partitioning and incremental offloading of the DNN layers, in order to minimize the startup execution delay.
	Edgent [19]	Develop a collaborative and on-demand co-inference framework for enabling EEoI by exploiting the device-edge synergy.
	DeSVig [20]	Design and test a Fog-supported decentralized swift vigilance framework for the GAN-enabled detection of cyber-attacks with sub-millisecond delays.
Hierarchical Mode	*Neurosurgeon* [21]	Generate profiling-based prediction models for a number of feasible partitions of the DNN layers and then select the one which jointly reduces both the inference delay and energy consumption.
	BrachyNet [7]	Consider the problem of the minimization of the communication and computing resource usage, while allowing low-delayed pattern classification through multiple early-exits.
	Cascade NN [22]	Propose and numerically test the accuracy-vs.-delay performance of various mechanisms for the dynamic stopping of the distributed device-edge execution of a baseline DNN.
Peer-to-Peer Mode	*MoDNN* [23]	Propose a local distributed mobile computing architecture to enable the parallel inference of DNNs on mobile platforms. Several partition schemes are provided for attaining inter-node workload balancing.
	DeepThings [24]	Propose a vertical Fused Tile Partitioning of the CNN layers for attaining task parallelism and design a novel workload scheduler to reduce the resulting inference time over execution platforms composed by peer nodes.
	DeepCham [25]	Orchestrate mobile devices for enabling the collaborative DNN training over peer-to-peer distributed parallel platforms.
	LAVEA [26]	Develop and test a spectrum of task-placement algorithms for enabling inter-peer collaboration, while minimizing the resulting service response delay.
	ECO [27]	Design an overlay network for modeling the inter-task dependencies and enable the distributed execution of DNNs by a set of cooperative peer-to-peer computing nodes.
Hybrid Mode	*D-DNN* [8]	Propose a distributed deep neural network architecture that is spatially distributed across computing hierarchies, which embrace Cloud nodes, Fog nodes and IoT devices. The related problem of the distributed training of the supported CDNN is also investigated.
	DeepFog [28]	Develop a Fog-based framework for the DL-aided mining of multidimensional data related to user wellness.
	EdgeLaaS [29]	Develop and test the performance of a Fog-supported framework for Knowledge-Centric Connected Healthcare (KCCH), in order to locally mine health supervision data through Reinforcement Learning-based engines.
	LiFo [Proposed]	Design, optimize and numerically test a hierarchically-organized spatially-distributed multi-tier IoT-Fog-Cloud virtualized platform for the execution of CDNNs with multiple early-exits. The main goal is to minimize the energy consumption by performing adaptive allocation of the virtualized resources under constraints on the allowed per-exit inference delays.

(bit)): $V_0(t) \triangleq \dim(\vec{z}_0(t))$ (bit) of the input feature vector $\vec{z}_0(t)$ is the workload to be processed by the execution platform of Fig. 1b during the t-th slot interval. Since the nature of the per-slot sensing actions performed by the IoT devices of Fig. 1b is inherently random and the inference process must be carried out in real-time, we formally assume that:

- the sequences $\{\vec{z}_0(t),\ t \geq 0\}$ and $\{V_0(t),\ t \geq 0\}$ of the feature vectors and workload at the input of the CDNN and Fog-Cloud computing platform of Fig. 1a and Fig. 1b are random, with *a priori* unknown steady-state probability distributions;
- each input feature vector $\vec{z}_0(t)$ is an instance drawn from an (*a priori* unknown) class c of a finite-size set: $\mathcal{C} \triangleq \{1, 2, \ldots, |\mathcal{C}|\}$ of allowed input classes (also referred to as input labels). The goal of the mining process performed by the CDNN of Fig. 1a is to deliver a (hopefully, reliable) estimate $\hat{c} \in \mathcal{C}$ of the actual class $c \in \mathcal{C}$ of the input feature vector $\vec{z}_0(t)$; and,

- the workload $V_0(t)$ (bit) arriving at the input of the platform of Fig. 1b at the beginning of slot t must be fully processed during the corresponding t-th slot interval, i.e., the t-th maximum inference time is upper-bounded by the t-th slot duration.

Hence, since the inference process is assumed to be carried out on a slot-by-slot basis, we refrain to explicitly indicate the slot-index t when not strictly needed and, then, we refer to $\vec{z}_0$ and V_0 as the per-slot input feature vector and input workload, respectively.

A. PER-LAYER FUNCTIONAL ARCHITECTURE

The task of this sub-section is to describe the transformation undergone by the information flow when it crosses a layer of the CDNN with early-exits of Fig. 1a, in order to formally define the per-layer fraction of the locally exited inference. For this purpose, Fig. 8a (resp., Fig. 8b) presents the basic architecture of the l-th hidden layer, with $1 \leq l \leq (L-1)$, (resp., the final L-th output layer) of the CDNN of Fig. 1a.

Since an in-depth analysis of the architectures of Fig. 8 is provided in [6] and [12], in the following we only summarize some basic functional aspects, which, in turn, are instrumental for the analysis of the volume of the locally delivered inference data. Therefore, regarding Fig. 8a, we briefly point out that [6], [12]:

1) $\vec{z}_{l-1}$ (resp., $\vec{z}_l$) is the binary-valued input (resp., output) vector at *layer #l*. $\vec{z}_{l-1}$ (resp., $\vec{z}_l$) is composed of $n_{l-1} \geq 1$ (resp., $n_l \geq 1$) scalar components, with each scalar component coded by a word of $n_{l-1}^{(b)} \geq 1$ (resp., $n_l^{(b)} \geq 1$) bits, so that we have: $\vec{z}_{l-1} \in \left(\{0,1\}^{n_{l-1}^{(b)}}\right)^{n_{l-1}}$ (resp., $\vec{z}_l \in \left(\{0,1\}^{n_l^{(b)}}\right)^{n_l}$), and also: $\dim(\vec{z}_{l-1}) = n_{l-1}^{(b)} n_{l-1}$ (bit) (resp., $\dim(\vec{z}_l) = n_l^{(b)} n_l$ (bit));
2) the l-th Digital-to-Analog Converter (DAC) $\mathcal{Q}_l^{-1}$ (resp., the l-th Analog-to-Digital Converter (ADC) $\mathcal{Q}_l$) transforms the n_{l-1}-dimensional binary input vector $\vec{z}_{l-1}$ (resp., the n_l-dimensional real-valued input vector $\vec{\tilde{z}}_l \in \mathbb{R}^{n_l}$) into the n_{l-1}-dimensional real-valued output vector $\vec{\tilde{z}}_{l-1} \in \mathbb{R}^{n_{l-1}}$ (resp., into the n_l-dimensional binary output vector $\vec{z}_l \in \left(\{0,1\}^{n_l^{(b)}}\right)^{n_l}$);
3) $[W_l]$ is the $(n_l \times n_{l-1})$ matrix of the real-valued weights of the l-th layer, while $\vec{b}_l$ is the (possibly, vanishing) n_l-dimensional real-valued bias column vector;
4) $\Phi_l(\cdot)$ is the l-th scalar-valued nonlinear activation function. It acts element-wise on the n_l-dimensional real-valued column vector $v_l \triangleq [W_l]\vec{\tilde{z}}_{l-1} + \vec{b}_l$, which is generated by the l-th fully connected layer of Fig. 8a;
5) $Pol_l(\cdot)$ is the (nonlinear and/or not invertible) pooling operator possibly present at *layer #l*. When it is present, this operator may also perform down sampling;
6) $[\widetilde{W}_l]$ is the $(|\mathcal{C}| \times n_l)$ real-valued weight matrix of the local classifier LC_l possibly present at the local output of the l-th layer, while $\vec{\tilde{b}}_l$ is the corresponding (possibly, vanishing) $|\mathcal{C}|$-dimensional real-valued bias column vector;
7) $\hat{c}_l \in \mathcal{C}$ is the scalar decision (i.e., the class label) delivered by the local classifier LC_l (possibly) present at the output of the l-th hidden layer. In principle, various criteria may be adopted for generating the local decision $\hat{c}_l$ [30]. To fix the ideas and without loss of generality, we assume that $\hat{c}_l$ is obtaining by applying the so-called *softmax* criterion as in [31]:

$$\hat{c}_l = \text{softmax}\left(\vec{\widehat{Y}}_l\right) \triangleq \arg\max_{1 \leq c \leq |\mathcal{C}|} \left\{\widehat{Y}_l(c)\right\}. \quad (1)$$

In Eq. (1), $\widehat{Y}_l(c)$ is the c-th scalar component of the $|\mathcal{C}|$-dimensional soft-decision vector $\vec{\widehat{Y}}_l$, and it is formally defined as [31]:

$$\widehat{Y}_l(c) \triangleq \frac{e^{\tilde{b}_l(c) + \widetilde{W}_l(c,.)\vec{\tilde{z}}_l}}{\sum_{k \in \mathcal{C}} e^{\tilde{b}_l(k) + \widetilde{W}_l(k,.)\vec{\tilde{z}}_l}}, \quad 1 \leq c \leq |\mathcal{C}|, \quad (2)$$

where $\widetilde{W}_l(c,.)$ indicates the c-th row of the matrix $[\widetilde{W}_l]$. The rationale behind the adoption of the *softmax* criterion is that, under some technical assumptions detailed in [31], $\widehat{Y}_l(c)$ in Eq. (2) provides a reliable estimate of the probability: $Prob(c \,|\, \vec{z}_{l-1})$ that the input feature $\vec{z}_0$ in Fig. 1a falls into the c-th decision class conditioned on the vector $\vec{z}_{l-1}$ present at the input of *layer #l*.

About the architecture in Fig. 8b of the final *layer #L*, let us note that its output:

$$\hat{c}_l = \text{softmax}(\vec{z}_L), \quad (3)$$

is still obtained by applying the previously defined *softmax* transformation to the vector: $\vec{z}_L = [W_L]\mathcal{Q}_L^{-1}(\vec{z}_{L-1}) + \vec{b}_L$, which is generated by the final output classifier of Fig. 8b.

Before proceeding, we explicitly point out that, although we have considered fully connected layers in the above description, all the framework developed in the following apply verbatim to the case in which convolutional layers are present. Intuitively, this a consequence of the formal fact that a convolutional layer may be described as a fully connected layer with circulant weight matrix $[W_l]$ (see [30] for additional details on this aspect).

The performed review of the per-layer functional architecture of a CDNN with early-exits is, indeed, instrumental for deriving three first formal results.

First, the input-output relationship implemented by the l-th hidden layer of a CDNN may be formally characterized by composing the ordered cascade of the chain of transformations shown in Fig. 8a, i.e.,

$$\vec{z}_l = \mathcal{Q}_l\left(\vec{\tilde{z}}\right) \equiv \mathcal{Q}_l\left(Pol_l\left(\Phi\left([W_l]\mathcal{Q}_l^{-1}(\vec{z}_{l-1}) + \vec{b}_l\right)\right)\right), \quad (4)$$

for $1 \leq l \leq L-1$.

Second, in order to formally characterize the volume of the inference data which exits at *layer #l*, let ρ_l, $1 \leq l \leq (L-1)$, be the steady-state fraction of the input vectors at *layer #l*

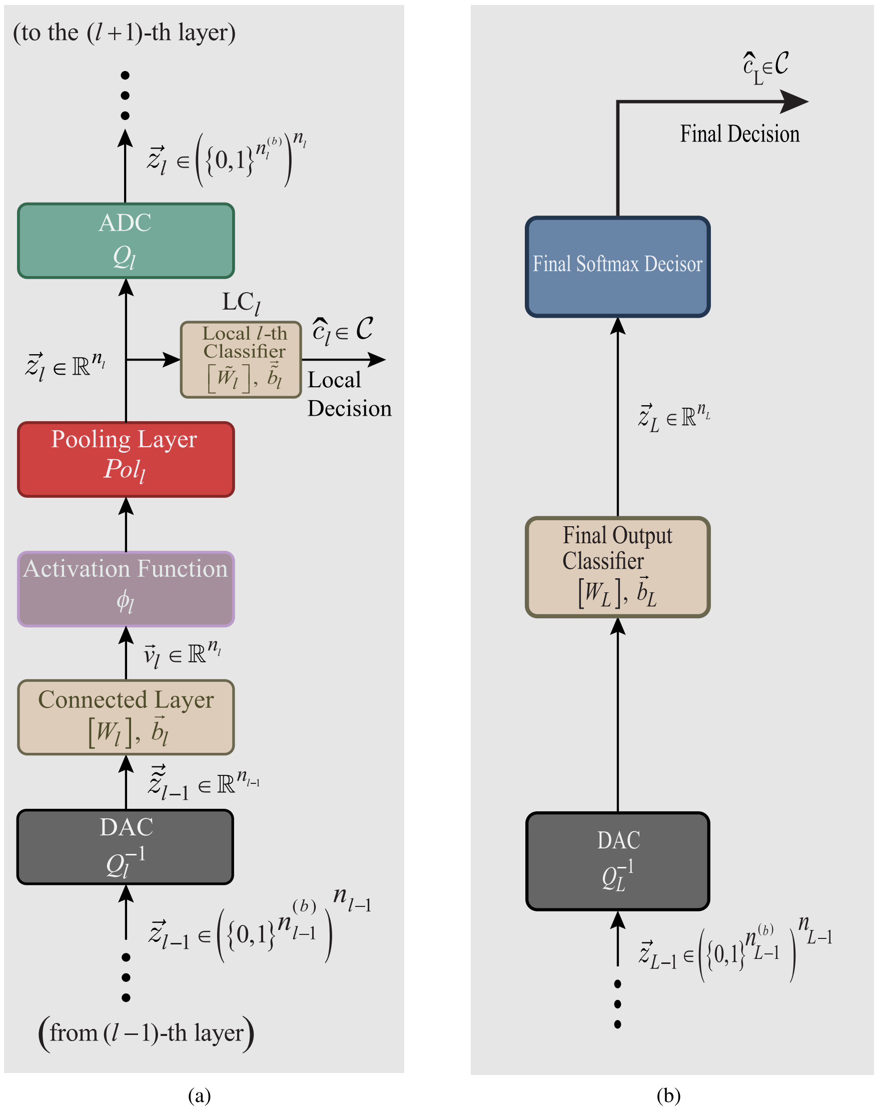

(to the $(l+1)$-th layer)
$\vec{z}_l \in \left(\{0,1\}^{n_l^{(b)}}\right)^{n_l}$
ADC
Q_l
LC$_l$
Local l-th Classifier
$[\tilde{W}_l]$, $\vec{\tilde{b}}_l$
$\hat{c}_l \in \mathcal{C}$
Local Decision
$\vec{z}_l \in \mathbb{R}^{n_l}$
Pooling Layer
Pol_l
Activation Function
ϕ_l
$\vec{v}_l \in \mathbb{R}^{n_l}$
Connected Layer
$[W_l]$, $\vec{b}_l$
$\vec{z}_{l-1} \in \mathbb{R}^{n_{l-1}}$
DAC
Q_l^{-1}
$\vec{z}_{l-1} \in \left(\{0,1\}^{n_{l-1}^{(b)}}\right)^{n_{l-1}}$
(from $(l-1)$-th layer)
(a)
$\hat{c}_L \in \mathcal{C}$
Final Decision
Final Softmax Decisor
$\vec{z}_L \in \mathbb{R}^{n_L}$
Final Output Classifier
$[W_L]$, $\vec{b}_L$
DAC
Q_L^{-1}
$\vec{z}_{L-1} \in \left(\{0,1\}^{n_{L-1}^{(b)}}\right)^{n_{L-1}}$
(b)

FIGURE 8. (a) Functional architecture of the l-th hidden layer of a CDNN with early exits, $1 \le l \le L-1$. (b) Functional architecture of the final L-th layer. LC_l: l-th Local Classifier.

which do not undergo early-exit, that is,

$$\rho_l \triangleq \lim_{t \to \infty} \frac{\mathcal{N}_l(t-1)}{t}, \quad 1 \le l \le L-1, \tag{5}$$

where $\mathcal{N}_l(t-1)$ is the number of the input vectors at *layer #l* which do *not* undergo early exit.

Hence, by using the definition in (5), the early exit-induced average compression factor: $cm_l \triangleq E\{\dim(\vec{z}_l)\}/E\{\dim(\vec{z}_{l-1})\}$ of the volume of data crossing the l-th hidden layer may be formally evaluated through the following formula:

$$\begin{aligned} cm_l &\triangleq \frac{E\{\dim(\vec{z}_l)\}}{E\{\dim(\vec{z}_{l-1})\}} = \frac{n_l n_l^{(b)} \rho_l}{n_{l-1} n_{l-1}^{(b)} \rho_{l-1}} \\ &\equiv \left(\frac{n_l n_l^{(b)}}{n_{l-1} n_{l-1}^{(b)}}\right) \times \left(\frac{\rho_l}{\rho_{l-1}}\right), \quad 1 \le l \le L-1, \end{aligned} \tag{6}$$

with $\rho_0 \triangleq 1$. The above formula jointly accounts for: (i) the dimensions n_l, n_{l-1} of the involved input-output vectors; (ii) the number of bits $n_l^{(b)}$, $n_{l-1}^{(b)}$ used for the binary coding of their scalar components (see Fig. 8); and, (iii) the fraction of the data which undergo early exit at *layer #l* (see the ratio ρ_l/ρ_{l-1}).

Third, by exploiting the defining relationship in (6), the per-slot average volume of data $\overline{\mathcal{V}}_l \triangleq E\{\dim(\vec{z}_l)\}$ (bit) generated in output by the l-th hidden layer and passed to the input of $(l+1)$-th layer admits the following closed-form expression:

$$\overline{\mathcal{V}}_l = cm_l \times E\{\dim(\vec{z}_{l-1})\} \equiv \left(\prod_{k=1}^{l} cm_k\right) \times \overline{\mathcal{V}}_0, \tag{7}$$

for $1 \le l \le L-1$, where $\overline{\mathcal{V}}_0 \triangleq E\{\mathcal{V}_0(t)\}$ (bit) is the average size of the workload generated by the IoT devices of Fig. 1b during a slot interval.

According to [12], all the ρ's coefficients in (5) and compression factors in (6) are profiled during the training phase of the considered CDNN and, then, in our framework, they play the role of known constants (see Fig. 2).

Remark 1 (On the Role Played by the DAC and ADC Blocks in the LiFo Framework):

Under the *LiFo* framework, there are (at least) three main reasons for explicitly considering the presence of the DAC and ADC (i.e., quantizer) blocks in the per-layer functional architecture of Fig. 8.

First, the modulation formats utilized by current Fog-based technological platforms for IoT applications are all of numerical type (see Ch. 4 of [3]), and this requires, in turn, the presence of the DAC and ADC blocks of Fig. 8.

Second, fixing the set $\left\{n_l^{(b)}\right\}$ of the per-layer per-sample numbers of used quantization bits is the key step for evaluating the resulting rates $\{R_m\}$ of the inter-tier connections of Fig. 3.

Third, a main new of the recent contribution in [32] is that it utilizes quantized weights, activation functions and inter-tier numerical flows during both the training and inference phases. In this case, all the analog multiply-and-add operations are replaced by binary XNOR operations. The numerical results presented in [32] support the conclusion that 1-bit quantized weights, 2-bit quantized activation functions and 6-bit quantized scalar gradients suffice to obtain classifying accuracy within $5.5\% - 6\%$ the corresponding ones of the analog counterparts. For attaining this performance, per-layer scalar (i.e., memory-less) logarithmic quantizers are used in [32]. In this regard, we anticipate that all results developed in this article apply verbatim to any considered quantization scheme, because they rely *only* on the spectrum $\left\{n_l^{(b)}\right\}$ of the adopted quantization bits.

B. ON THE ENERGY-EFFICIENT LAYER-TO-TIER MAPPING AND RESULTING INTER-TIER NETWORK TOPOLOGY

A factor impacting on the energy consumed by the execution of the inference phase of the CDNN of Fig. 1a on the distributed multi-tier networked computing platform of Fig. 1b is the adopted strategy for mapping the CDNN layers onto the available Fog/Cloud tiers. This problem has been afforded in deep in our recent contribution in [12]. Hence, in the remaining part of this section, we only recap some main results which will be employed in the sequel for the formal characterization of the energy consumed by the proposed *LiFo* technological platform of Fig. 3 during the inference phase.

Therefore, according to [12], let us consider a partition:

$$\mathcal{S}_m^* \subseteq \{1, 2, \ldots, L\}, \quad m = 1, \ldots, M, \tag{8}$$

of the set of the layers of the CDNN of Fig. 1a into M subsets, which is characterized by the following three defining properties:

- the first (resp., last) layer of the CDNN of Fig. 1a is mapped onto an element of the set $\mathcal{S}_1^*$ (resp., $\mathcal{S}_M^*$) of the considered partition;
- the $\left|\mathcal{S}_m^*\right|$ elements of $\mathcal{S}_m^*$ are the indexes of *consecutive* (that is, adjacent) CDNN layers;
- the cluster of computing nodes at *tier* #m, $1 \leq m \leq M$, exhibits sufficient computing power and communication bandwidth to host all the layers of the m-th partition set $\mathcal{S}_m^*$.

Therefore, *Proposition 8* of [12] formally proves that, at least in the case in which all the involved computing nodes exhibit the same power-consumption profile, such a kind of partition individuates a feasible Layer-to-Tier mapping which minimizes the average energy wasted by the multi-tier platform of Fig. 1b for the execution of the CDNN of Fig. 1a. Motivated by this formal result, we assume that the mapping of the layers of the CDNN of Fig. 1a onto the tiers of the execution platform of Fig. 1b has been already performed according to the just reviewed minimum-energy criterion (see Fig. 2).

In this regard, we also note that both the actual network interconnections from Fog nodes at *tier* #$(m-1)$ to the ones at *tier* #m, $2 \leq m \leq M$, and the resulting numbers of the per-node input/output ports are dictated by the nonzero elements of the weight matrix: $[W_l]$, with $l \equiv \sum_{i=1}^{m-1} \left|\mathcal{S}_i^*\right|$, which describes the last (i.e., highest order) layer of the supported CDNN of Fig. 1a, which is mapped by the performed partition in (8) onto the $(m-1)$-th tier of the execution platform in Fig. 1b. Hence, after implementing the Layer-to-Tier mapping according to the partition in (8), we may also assume that:

- the topologies of the inter-tier interconnections in Fig. 3 are defined, as well as the corresponding numbers of input and output ports equipping the Fog nodes.

Motivated by these considerations, we directly focus on the (still open) challenges of the design of the per-node virtualized architecture and per-tier adaptive allocation of the computing-plus-networking resources.

IV. THE LIFO VIRTUALIZED PLATFORM: NODE ARCHITECTURE, SERVICE MODEL AND EXECUTION TIMES

The goal of this section is threefold. First, we detail the virtualized architecture envisioned for a *LiFo* Fog node. Second, we describe the resulting *LiFo* protocol stack and define the roles of the various protocol layers. Third, we formally characterize the per-node and per-tier execution time and, then, we model their impact on the resulting early-exit inference times.

To begin with, Fig. 9 shows the internal functional architecture envisioned for the (j, m)-th Fog node (briefly, $FN(j, m)$) of Fig. 3, so to detail the corresponding logical view already sketched in Fig. 5.

Specifically, in Fig. 9 we have that:

1) the virtualized (j, m)-th *Main Processor* (in short, *MP*(j, m)) provides computing support for the execution of the (previously described) blocks labeled as *DAC*, *Connected Layer*, *Activation Function*, *Pooling Layer* and *ADC* in Fig. 8a. In principle, *MP*(j, m) in Fig. 9 is implemented by the first virtual core *VC* #1

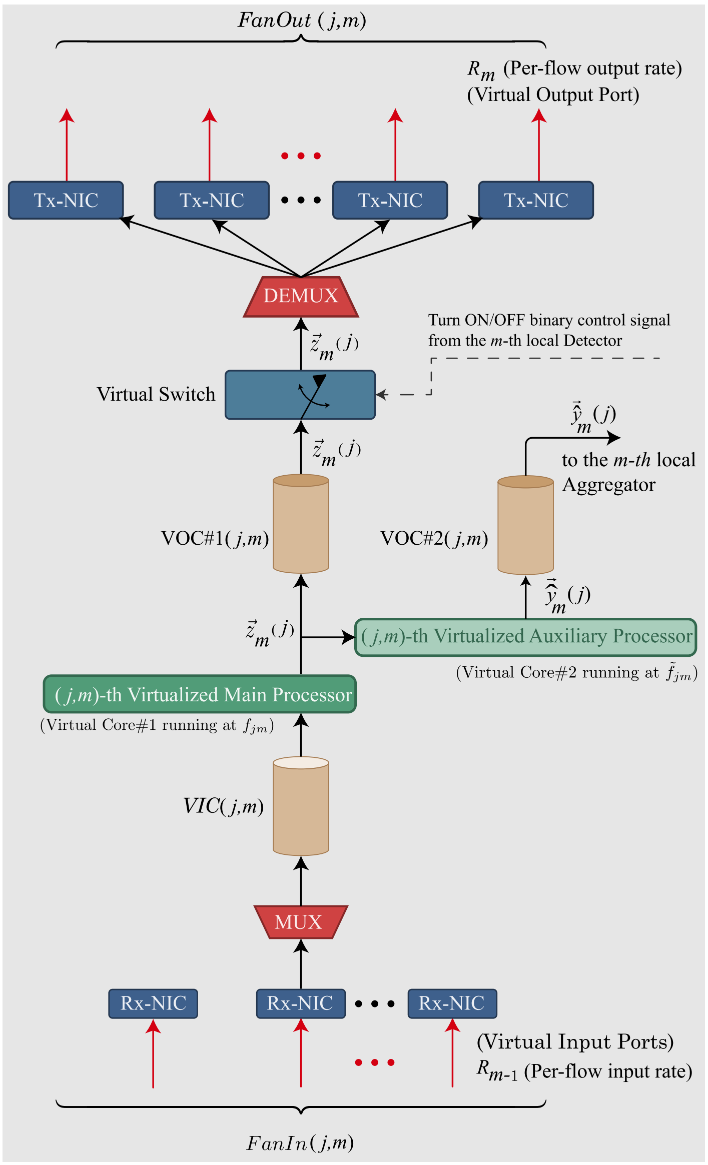

FanOut (j,m)
R_m (Per-flow output rate)
(Virtual Output Port)
Tx-NIC
Tx-NIC
Tx-NIC
Tx-NIC
DEMUX
Turn ON/OFF binary control signal from the m-th local Detector
$\vec{z}_m(j)$
Virtual Switch
$\vec{z}_m(j)$
$\vec{y}_m(j)$
to the m-th local Aggregator
VOC#1(j,m)
VOC#2(j,m)
$\vec{y}_m(j)$
$\vec{z}_m(j)$
(j,m)-th Virtualized Auxiliary Processor
(Virtual Core#2 running at $\tilde{f}_{jm}$)
(j,m)-th Virtualized Main Processor
(Virtual Core#1 running at f_{jm})
VIC(j,m)
MUX
Rx-NIC
Rx-NIC
Rx-NIC
(Virtual Input Ports)
R_{m-1} (Per-flow input rate)
FanIn(j,m)

FIGURE 9. **Functional architecture of the virtualized (j, m)-th Fog node under the *LiFo* paradigm. NIC: = Network Interface Card; VOC: = Virtual Output Cache; VIC: = Virtual Input Cache; MUX: = Multiplexer; DEMUX: = De-multiplexer.**

equipping the corresponding clone in Fig. 5, and its operating processing speed f_{jm} (bit/s) is orchestrated at runtime by the associated *Clone Manager* of Fig. 4. In the envisioned framework, the input data $\vec{z}_{m-1}(j)$ at the $MP(j, m)$ is a feature vector received from the Fog nodes working at the previous *tier* #$(m-1)$, while the corresponding output data $\vec{z}_m(j)$ is provided to the associated *Auxiliary Processor*, as well as forwarded to the Fog nodes operating at the next *tier* #$(m+1)$;

2) the virtualized (j, m)-th *Auxiliary Processor* (briefly, $AP(j, m)$) provides computing support for the execution of the block labeled as *Local Classifier* in Fig. 8a. It is implemented by the second virtual core *VC* #2 present in the clone of Fig. 5, and its operating processing speed $\tilde{f}_{jm}$ (bit/s) is still orchestrated at runtime by the *Clone Manager* of Fig. 4. The task of $AP(j, m)$ is to process the feature vector generated by the corresponding *Main Processor*, so to produce the output vector $\hat{y}_m(j)$ to be delivered to the corresponding m-th *Aggregator* of Fig. 3 for the (possible) generation of an early-exit;

3) $FN(j, m)$ is also equipped with a number $FanIn(j, m) \geq 1$ of virtualized input ports. Hence, the task of the *MUltipleXer* (*MUX*) at the bottom of Fig. 9 is to merge the corresponding information flows received by the Fog nodes operating at the previous *tier* #$(m-1)$, so to generate a (single) aggregate feature vector $\vec{z}_{m-1}(j)$. According to the architecture of the overall *LiFo* technological platform of Fig. 3, all input flows

at the bottom of Fig. 9 operate at a same bit-rate R_{m-1} (bit/s), which, in turn, is orchestrated at runtime by the *Clone Manager* of Fig. 4;

4) the main task of the *De-MUltipleXer* (*DEMUX*) at the top of Fig. 9 is to replicate the received feature vector $\vec{z}_m(j)$ over the $FanOut(j, m) \geq 1$ virtualized output ports equipping $FN(j, m)$. Each output flow is forwarded to the Fog nodes at the next *tier* #$(m + 1)$ at a bit-rate R_m (bit/s) which is still dictated by the *Clone Manager*;
5) the task of the *Virtual Switch* (*VS*) of Fig. 9 is to enable the transmission of the output feature $\vec{z}_m(j)$ to the Fog nodes at the next *tier* #$(m + 1)$ only when early-exit does not occur at *tier* #*m*. For this purpose, a binary turn ON/OFF control signal is generated by the *m*-th *Detector* of Fig. 3 on a per-slot basis. This signal triggers the *VS* of Fig. 9 to open (resp., to shut) when a local exit is delivered (resp., is not delivered) at *tier* #*m*;
6) the common task of the (j, m)-th *Virtual Input Cache* ($VIC(j, m)$), *Virtul Ouput Cache* #1 ($VOC\#1(j, m)$) and *Virtul Ouput Cache* #2 ($VOC\#2(j, m)$) is to allow the temporarily buffering of the received data, in order to enable: (i) inter-processor synchronization (i.e., $VIC(j, m)$); (ii) inter-tier synchronization (i.e., $VOC\#1(j, m)$); and, (iii) delayed retrieving of stored data by the *m*-th Aggregator of Fig. 3 (i.e., $VOC\#2(j, m)$). The storing/retrieving operations on these cache units take place under the supervision of the corresponding *Clone Manager* in Fig. 4.

The presented per-node architecture is the key to formally characterize the volumes of input data $\mathcal{V}_I(j, m) \triangleq \dim(\vec{z}_{m-1}(j))$ (bit) (resp., output data $\mathcal{V}_O(j, m) \triangleq \dim(\vec{z}_m(j))$ (bit)) received (resp., transmitted) by the input (resp., output) ports of $FN(j, m)$ in Fig. 9 over a slot interval. In this regard, we note that, by design, the *LiFo* paradigm requires that all Fog nodes at *tier* #*m* cooperate, in order to process the aggregate workload needed for the execution of the CDNN layers embraced by the *m*-th set $\mathcal{S}_m^*$ of the performed Layer-to-Tier partition of Eq. (8). As a consequence, after performing workload balancing over the available input/output ports of Fig. 9, the volume of data received by all input ports of $FN(j, m)$ over a slot interval can be analytically evaluated as:

$$\mathcal{V}_I(j, m) = \mathcal{V}_0 \times \left(\prod_{k=1}^{\left(\sum_{j=1}^{m-1} \left|\mathcal{S}_j^*\right|\right)} cm_k \right) \times \left(\frac{FanIn(j, m)}{\sum_{r=1}^{m_m} FanIn(r, m)} \right) \quad \text{(bit)}, \qquad (9)$$

for $1 \leq j \leq m_m$, $2 \leq m \leq M$, with:

$$\mathcal{V}_I(j, 1) = \mathcal{V}_0 \left(\frac{FanIn(j, 1)}{\sum_{r=1}^{m_m} FanIn(r, 1)} \right) \quad \text{(bit)}, \qquad (9.1)$$

while the following companion expressions hold for the corresponding per-slot volume of data generated by all the output ports of $FN(j, m)$:

$$\mathcal{V}_O(j, m) = \mathcal{V}_0 \times \left(\prod_{k=1}^{\left(\sum_{j=1}^{m-1} \left|\mathcal{S}_j^*\right|\right)} cm_k \right) \times \left(\frac{FanIn(j, m)}{\sum_{r=1}^{m_m} FanIn(r, m)} \right) \quad \text{(bit)}, \qquad (10)$$

for $1 \leq j \leq m_m$, $2 \leq m \leq M$, with:

$$\mathcal{V}_O(j, 1) = \mathcal{V}_0(j) \left(\prod_{k=1}^{\left(\sum_{j=1}^{m-1} |\mathcal{S}_1^*|\right)} cm_k \right) \quad \text{(bit)}, \qquad (10.1)$$

for $1 \leq j \leq m_1$. In the above expressions, we have that: (i) $\{cm_k\}$ are the per-layer compression factors in Eq. (6) of the supported CDNN; (ii) $\{S_j^*\}$ are the sets of the implemented Layer-to-Tier partition in Eq. (8); and, (iii) $\mathcal{V}_0(j)$ (bit) is the input traffic received by $FN(j, 1)$ at *tier* #1 of Fig. 3 from the attached IoT devices during the current slot. Since, by design, the per-slot aggregate workload generated by all IoT devices operating at *tier* #0 of Fig. 3 equates to $\mathcal{V}_0$ (see Fig. 1b), then, we must have: $\sum_{j=1}^{m_1} \mathcal{V}_0(j) = \mathcal{V}_0$.

Before proceeding, we point out that Eqs. (9) and (10) are derived by recursively applying the defining relationship in (6) from: *tier* #*m* down to *tier* #0, while simultaneously accounting for the following two architectural properties. First, the cluster of Fog nodes at *tier* #*k*, $1 \leq k \leq m$, provides the support for the execution of the CDNN layers embraced by the *k*-th set S_k^*, $1 \leq k \leq m$, of the implemented Layer-to-Tier mapping in Eq. (8) (see the second factors at the r.h.s.'s of Eqs. (9) and (10)). Second, due to the performed workload balancing, each input (resp., output) port of a Fog node at *tier* #*m* conveys the same per-slot volume of data (see the third factors at the r.h.s.'s of Eqs. (9) and (10)). As a consequence, the expressions in Eqs. (9) and (10) differ only in the upper limits of the therein present product forms. This is due to the fact that, by design, the performed Layer-to-Tier partition in Eq. (8) enforces the first $(m - 1)$ (resp., m) tiers of the *LiFo* execution platform of Fig. 3 to host the first $\left(\sum_{j=1}^{m-1} \left|\mathcal{S}_j^*\right|\right)$ (resp., the first $\left(\sum_{j=1}^{m} \left|\mathcal{S}_j^*\right|\right)$) layers of the supported CDNN.

A. LOCAL AGGREGATORS AND DETECTORS: FUNCTIONAL ARCHITECTURES AND ROLES

The envisioned functional architecture for the *m*-th local output of the *LiFo* platform of Fig. 3 is composed of the cascade of the blocks detailed in Fig. 10.

In this regard, we note that, by design, the functional role of the *m*-th local *Aggregator* in Fig. 10 is to perform a suitable component-wise fusion of the set: $\left\{ \overrightarrow{y}_m(j),\ 1 \leq j \leq m_m \right\}$ of the $|\mathcal{C}|$-dimensional feature vectors generated by the (previously described) auxiliary processors of the Fog nodes at *tier* #*m* (see Fig. 9), in order to produce a $|\mathcal{C}|$-dimensional fused feature vector $\widehat{\overrightarrow{Y}}_m$ (see Fig. 10). According to [8],

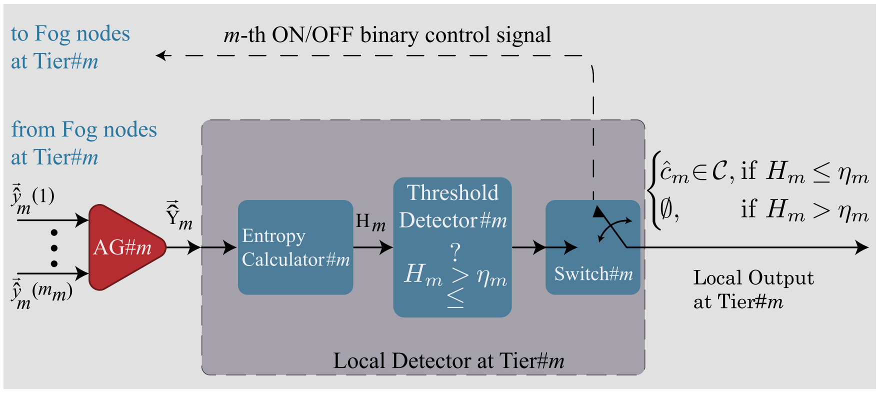

to Fog nodes at Tier#m
m-th ON/OFF binary control signal
from Fog nodes at Tier#m
$\vec{y}_m(1)$
$\vec{y}_m(m_m)$
AG#m
$\vec{Y}_m$
Entropy Calculator#m
H_m
Threshold Detector#m
$H_m \overset{?}{\gtrless} \eta_m$
Switch#m
$\hat{c}_m \in \mathcal{C}$, if $H_m \le \eta_m$
$\emptyset$, if $H_m > \eta_m$
Local Output at Tier#m
Local Detector at Tier#m

FIGURE 10. Planned functional architecture for the *m*-th local output of the proposed *LiFo* technological platform.

examples of aggregation rules are the *max* one, i.e.,

$$\widehat{Y}_m(k) \triangleq \max_{1\le j\le m_m} \{\widehat{y}_m(j,k)\}, \quad k = 1, \ldots, |\mathcal{C}|, \tag{11}$$

and the *average* one, i.e.,

$$\widehat{Y}_m(k) \triangleq \frac{1}{|\mathcal{C}|}\sum_{j=1}^{m_m} \widehat{y}_m(j,k), \quad k = 1, \ldots, |\mathcal{C}|. \tag{12}$$

Overall, the task of the *Aggregator* is twofold. First, per-tier aggregation enables cooperation between multiple spatially distributed Fog nodes which support the same set of CDNN layers, so to provide the key for attaining spatial scaling. Second, the *Aggregator* allows multiple local exits equipping adjacent layers of the supported CDNN to be fused into a single per-tier local output. As a consequence, without loss of generality, in the sequel, we assume that the number N_{EE} of the CDNN local exits in Fig. 1a coincides with the number $(M-1)$ of the local outputs of the supporting execution platform, i.e., we assume that:

$$N_{EE} \equiv M - 1. \tag{13}$$

For describing the architecture planned for the m-th local *Detector* (see the dashed box in Fig. 10), we observe that the reliability level of the inference data delivered by the m-th local output may be evaluated on the basis of a suitable statistic computed by the *Calculator* block of Fig. 10 by exploiting the corresponding aggregate vector $\overrightarrow{\widehat{Y}}_m$. Without loss of generality and according to [7], we may assume that this statistic is the normalized entropy:

$$H_m \triangleq -\frac{1}{|\mathcal{C}|}\sum_{k=1}^{|\mathcal{C}|} \widehat{Y}_m(k)\log_2 \widehat{Y}_m(k) \in [0,\ 1], \tag{14}$$

of the aggregate vector $\overrightarrow{\widehat{Y}}_m$, which the *Threshold Detector* of Fig. 10 compares, in turn, against to a suitably set entropy threshold $\eta_m \in (0, 1)$. Afterwards, on the basis of the performed comparison,

1) if $H_m \le \eta_m$, the m-th *Detector* in Fig. 10 assumes that the decision:
$$\widehat{c}_m \triangleq \arg\max_{1\le k\le|\mathcal{C}|} \left\{\widehat{Y}_m(k)\right\}, \tag{15}$$
is reliable enough. As a consequence, the switch in Fig. 10 is closed and the decision in (15) is locally delivered. At the same time, a feedback control signal is generated that forces all Fog nodes operating at *tier* #m to open own virtual switches in Fig. 9, so to stop the forwarding of the corresponding set: $\{\vec{z}_m(j),\ 1 \le j \le m_m\}$ of output feature vectors to the next *tier* #$(m+1)$; however,
2) if $H_m > \eta_m$, the m-th *Detector* considers the decision in (15) be unreliable. Then, the switch in Fig. 10 is open, so that no data is delivered by the m-th local output. Furthermore, the control signal generated by the m-th *Detector* forces all the virtual switches in Fig. 9 to shut, so to enable the forwarding of the feature vectors $\{\vec{z}_m(j),\ 1 \le j \le m_m\}$ to the next *tier* #$(m+1)$ for further processing.

Before proceeding, two explicative remarks are in order.

First, we observe that the tuning of the set: $\{\eta_m,\ 1 \le m \le (M-1)\}$ of the per-tier decision thresholds in Fig. 10 is performed during the setup phase of the supported CDNN (see Step 3 of Fig. 2). In this regard, we note that too low (resp., too high) threshold values increase (resp., decrease) the average reliability levels of the delivered local decisions in (15), but also increase (resp., decrease) the fraction of the input data that propagate up to the last *tier* #M of the platform of Fig. 3, so to annihilate (resp., emphasize) the benefits arising from the exploitation of early-exits. The challenging topic of the optimized tuning of these thresholds is afforded in depth in the contribution in [12], where an adaptive algorithm is proposed for balancing the two contrasting requirements of reliable local decisions and small per-layer compression factors (see Eq. (6)).

Second, since, by design, both the *Aggregator* and *Detector* in Fig. 10 operate *locally* at *tier* #m (see Fig. 3), in the sequel, we will assume that:

- the networking/computing resources needed for the support of the per-tier *Aggregators* and *Detectors* may be considered *negligible* with respect to the corresponding ones required for supporting the computing operations performed by the Fog nodes and the associated inter-tier communication flows.

B. THE ENVISIONED LIFO PROTOCOL STACK

In agreement with the per-node functional architecture of Fig. 9, we plan the protocol stack for the *LiFo* technological platform drawn at the left of Fig. 11. For comparison purposes, this figure also illustrates: (i) the general functional stack of an IoT ecosystem recently proposed by Cisco [3]; and, (ii) the protocol stack of a traditional networked platform for the support of remote Cloud applications in which all Fog nodes at the intermediate tiers: *tier* #1 – *tier* #$(M-1)$ of Fig. 1b are replaced by network switches/routers.

A comparative view of the stacks of Fig. 11 unveils the peculiar features retained by the proposed *LiFo* paradigm and leads, indeed, to four main remarks.

First, the three-layered architecture of the *LiFo* protocol stack of Fig. 11 reflects the corresponding partition of the *LiFo* technological platform of Fig. 3 into three hierarchically-organized main segments, namely (see Fig. 3): (i) the lower-most segment, at which the IoT devices operate; (ii) the middle segment, which embraces the clusters of Fog nodes; and, (iii) the upper-most segment, where the remote Cloud works.

Second, adaptive resource and reliability managements are cooperatively carried out by all layers of the *LiFo* protocol stack of Fig. 11. We anticipate that this feature is in agreement with the distributed and adaptive nature of the resource allocation strategy we develop in Sections VI and VII. This is, indeed, of *cross-tier* type and, then, it *jointly* involves the computing and networking resources of *all* tiers of the *LiFo* platform of Fig. 3.

Third, interestingly enough, besides *fully* embracing the Communication Layer of the corresponding IoT Functional stack, the intermediate Fog Layer of the *LiFo* protocol stack of Fig. 11 also *partially encloses* the corresponding Analytics sub-Layer. This is in agreement, indeed, with the per-node functional architecture of Fig. 9. In fact, this architecture allows each *LiFo* Fog node to *simultaneously* act as network switch (see the input/output ports of Fig. 9) and computing node (see the main and auxiliary processors of Fig. 9), so to perform packet forwarding, while enabling EEoI through the associated local outputs (see Fig. 10 and related text). This twofold role played by the *LiFo* Fog nodes is in *sharp* contrast with the corresponding role covered by the switches of a traditional networked platform for Cloud applications (see the protocol stack at the right side of Fig. 11), whose *only* task is that of data forwarding. In this regard, we anticipate that the numerical results of Section IX support the conclusion that the average energy consumption of the *LiFo* technological platform may be significantly below the corresponding ones of a traditional switch-based network supporting remote Cloud applications.

Finally, the spatially-scalable and virtualized nature of the proposed *LiFo* technological platform of Fig. 3 enables both: (i) *multi-tenant single-mined sensing applications*, in which the IoT devices operating at *tier* #0 of Fig. 3 are owned and managed by *different* spatially-scattered clients, but the resulting aggregate data is collectively mined through the execution of a *single* CDNN; and, (ii) *single-tenant multi-mined sensing applications*, where the containers in Fig. 4 equipping each Fog node run in parallel *multiple* different CDNNs that carry out *different* inference processes on the data set of a *single* client.

C. CHARACTERIZING THE LIFO INFERENCE TIMES

According to the architecture of Fig. 9, the following two tasks are executed by Fog node $FN(j,m)$ on a per-slot basis: (i) the Main Processor (resp., Auxiliary Processor) in Fig. 9 processes the input workload $\mathcal{V}_I(j,m)$ (bit) in Eqs. (9), (9.1) (resp., the output workload $\mathcal{V}_O(j,m)$ (bit) in Eqs. (10), (10.1) at a processing speed of f_{jm} (bit/s) (resp., $\tilde{f}_{jm}$ (bit/s)); and, (ii) the input (resp., output) ports in Fig. 9 receive (resp., transmit) in parallel the workload $\mathcal{V}_I(j,m)$ (resp., $\mathcal{V}_O(j,m)$) at a per-port rate of R_{m-1} (bit/s) (resp., R_m (bit/s)). However, by design, the transmission time from *tier* #$(m-1)$ must overlap with the receive time at *tier* #m, so that the resulting per-slot execution time $T_{EXE}(j,m)$ (measured in seconds) of $FN(j,m)$ is given by the following three contributions:

$$\begin{aligned} T_{EXE}(j,m) &\triangleq T_{MP}(j,m) + T_{AP}(j,m) + T_{NET}(j,m) \\ &\equiv \frac{\mathcal{V}_I(j,m)}{f_{jm}} + \frac{\mathcal{V}_O(j,m)}{\tilde{f}_{jm}} + \frac{\mathcal{V}_O(j,m)}{R_m}, \end{aligned} \tag{16}$$

for $1 \le j \le m_m$, $1 \le m \le (M-1)$, with the following auxiliary relationship:

$$T_{EXE}(1,M) \equiv T_{XE}^{(CLOUD)} = \frac{\mathcal{V}_I(1,M)}{f_{1M}} + \frac{\mathcal{V}_O(1,M)}{\tilde{f}_{1M}}, \tag{16.1}$$

accounting for the fact that the Cloud node operates at the uppermost tier.

In order to evaluate the per-tier execution time, we observe that the m-th aggregator $AG(m)$ in Fig. 10 may perform the fusion of the received information features: $\left\{\vec{y}_j(m),\ 1 \le j \le m_m\right\}$ only after that *all* the corresponding Fog nodes $\{FN(j,m),\ 1 \le j \le m_m\}$ finished their tasks. Hence, the resulting per-tier execution time $T_{EXE}(m)$ is dictated by the execution time of the slowest Fog node, and, then, it equates:

$$T_{EXE}(m) = \max_{1\le j\le m_m} \{T_{EXE}(j,m)\}, \quad 1 \le m \le M. \tag{17}$$

Finally, the hierarchical organization of the tiers of the *LiFo* technological platform of Fig. 3 guarantees that the aggregate execution time $T_{EXE}^{(1,m)}$ needed for generating a local exit at *tier* #m equates the summation of the corresponding per-tier

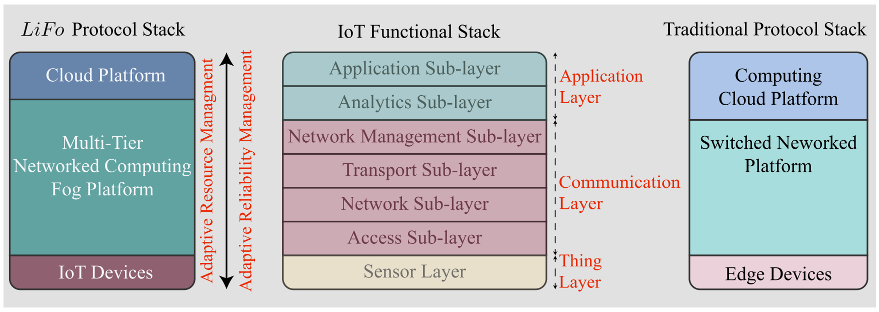


FIGURE 11. ***LiFo*-vs.-traditional protocol stacks: who does what under the *LiFo* paradigm.**

execution times in (17) up to the m-th one, i.e.,

$$T_{EXE}^{(1,m)} = \sum_{k=1}^{m} T_{EXE}(k) \equiv \sum_{k=1}^{m} \left(\max_{1 \le j \le m_k} \{T_{EXE}(j,k)\} \right), \quad (18)$$

for $1 \le m \le M$. This relationship provides the formal characterization of the inference time (i.e., delivering delay) at the m-th early exit of the *LiFo* platform in Fig. 3.

V. FEATURING THE LIFO ENERGY CONSUMPTION

The goal of this section is to formally model the power and energy consumption of the virtualized processors and input/output ports which equip each Fog node in Fig. 9. Since an in-depth presentation of general power and energy models for virtualized networked computing platforms has been carried out in the recent contribution in [33], in the sequel, we directly focus on the aspects of the models developed in Section V of [33] which are more specific to the *LiFo* framework considered here.

A. MODELING THE COMPUTING POWER AND ENERGY

In this sub-section, we focus on the formal characterization of the power and energy consumed for computing purpose by the main and auxiliary processors of Fig. 9.

1) MODELING THE COMPUTING POWER AND ENERGY OF THE VIRTUALIZED MAIN PROCESSOR

According to [33], the total power $\mathcal{P}_{MP}(j,m)$ (Watt) consumed by the main processor equipping the virtualized (j,m)-th Fog in Fig. 9 is the summation of the idle (i.e., static) power $\mathcal{P}_{MP}^{(IDLE)}(j,m)$ (Watt) and the dynamic power $\mathcal{P}_{MP}^{(DYN)}(j,m)$ (Watt).

Under the virtualized architecture featured by Fig. 9, the (j,m)-th idle power equates [33]:

$$\mathcal{P}_{MP}^{(IDLE)}(j,m) = 0.5 \times \frac{\mathcal{P}_{CPU}^{(IDLE)}(j,m)}{nc(j,m)}, \quad (19)$$

where: (i) $\mathcal{P}_{CPU}^{(IDLE)}(j,m)$ (Watt) is the total power consumed by the (j,m)-th Fog node for sustaining the hosted $nc(j,m) \ge 1$ containers in the idle state (see Fig. 4); and, (ii) the scaling factor: 0.5 accounts for the fact that the total idle power $\left(\mathcal{P}_{CPU}^{(IDLE)}(j,m)/nc(j,m)\right)$ consumed by the Fog node is evenly split over the main and auxiliary processors of Fig. 9.

Under the *LiFo* framework, the dynamic power $\mathcal{P}_{MP}^{(DYN)}(j,m)$ consumed by the main processor in Fig. 9 depends on three main factors, namely: (i) the processing frequency f_{jm} (bit/s) at which the main processor runs; (ii) the processing density ε_{jm} (measured in (CPU cycle/bit)) of the workload processed by the main processor; and, (iii) the number $|\mathcal{S}_m^*|$ of layers of the supported CDNN in Fig. 1a which are mapped onto the m-th tier of the *LiFo* platform of Fig. 1b (see Eq. (8) and related text). Specifically, according to [33], $\mathcal{P}_{MP}^{(DYN)}(j,m)$ may be modeled as follows:

$$\mathcal{P}_{MP}^{(DYN)}(j,m) = \mathcal{K}_{MP}^{(j,m)} \left(|\mathcal{S}_m^*| \, \varepsilon_{jm} f_{jm} \right)^{\gamma_{MP}^{(j,m)}}, \quad (20)$$

for $1 \le j \le m_m$, and $1 \le m \le M$. In the above expression, we have that [33], [34]:

- $\gamma_{MP}^{(j,m)}$ is a dimensionless positive exponent whose actual value depends on the power-profile of the underlying physical CPUs equipping the considered Fog node;
- according to [35], the processing density ε_{jm} ((CPU cycle)/bit) is defined as the average number of CPU cycles that are required for the processing of a *single* bit of the workload generated by a *single* layer of the supported CDNN of Fig. 1a. As a consequence, the actual value of ε_{jm} depends on the average number of neurons that compose a single layer of the considered CDNN, as well as the average number of add-and-multiply algebraic operations performed by each neuron (see [35] for numerical examples of processing densities of some DL-supported applications of practical interest);
- being $|\mathcal{S}_m^*|$ the number of layers of the supported CDNN to be executed by the m-th cluster of Fig. 1b, the presence of this factor in Eq. (20) accounts for the fact that the workload to be processed by the main processor in Fig. 9 scales up in a linear way for increasing values of $|\mathcal{S}_m^*|$; and,
- $\mathcal{K}_{MP}^{(j,m)}$ (measured in (Watt/(CPU cycle)/s)$^{\gamma_{MP}^{(j,m)}}$) is a scaling factor, whose actual value depends on the power profile of the virtual core which implements the main processor of Fig. 9 (see, for instance, [36] for some profiled examples).

Now, in order to characterize the resulting computing energy $\mathcal{E}_{MP}(j,m)$ (measured in (Joule)) consumed by the main processor of Fig. 9 over a slot time, we note that: (i) the main processor must remain turned ON for the time $T_{EXE}^{(1,M)}$ in Eq. (18) needed by the Cloud node at the top *tier #M* of Fig. 3 to deliver its final inference; and, (ii) according to Eq. (16), the time $T_{MP}(j,m)$ (s) needed by the main processor for processing the workload received by its input ports of Fig. 3 equates to:

$$T_{MP}(j,m) = \frac{\mathcal{V}_I(j,m)}{f_{jm}}. \tag{21}$$

Hence, on the basis of these considerations, the per-slot computing energy $\mathcal{E}_{MP}(j,m)$ consumed by the main processor of Fig. 3 may be formally characterized as follows:

$$\mathcal{E}_{MP}(j,m) = \underbrace{\mathcal{P}_{MP}^{(IDLE)}(j,m) \times T_{EXE}^{(1,M)}}_{Idle\ Energy} + \underbrace{\mathcal{P}_{MP}^{(DYN)}(j,m) \times T_{MP}(j,m)}_{Dynamic\ Energy}, \tag{22}$$

for $1 \leq m \leq m_m$, and $1 \leq m \leq M$.

2) MODELING THE COMPUTING POWER AND ENERGY OF THE VIRTUALIZED AUXILIARY PROCESSOR

The power and energy consumed by the auxiliary processor of Fig. 9 can be formally characterized by formulas analogous to those presented for the main processor. In this regard, we observe that:

1) being different the workloads to be processed (see Fig. 9), the virtual cores in Fig. 5 implementing the main and auxiliary processors can be *heterogeneous*, i.e., they may exhibit different dynamic-power profiles;
2) the processing speed $\tilde{f}_{jm}$ (bit/s) at which operates the auxiliary processor can be different from the corresponding one f_{jm} of the main processor; and,
3) being application-depending, the processing density β_{jm} (CPU cycle/bit) of the workload processed by the auxiliary processor can be different from the corresponding one ε_{jm} of the main processor. More importantly, β_{jm} does *not* longer scale proportionally to the number $\left|\mathcal{S}_m^*\right|$ of sustained CDNN layers, because, by design, the functional architecture of the supported m-th local branch in Fig. 10 does *not* depend on the number of CDNN layers which are mapped onto the m-th tier of the *LiFo* technological platform of Fig. 3.

Hence, on the basis of the above considerations, the idle power $\mathcal{P}_{AP}^{(IDLE)}(j,m)$ (Watt) consumed by the (j,m)-th auxiliary processor of Fig. 9 is similar to that presented in Eq. (19), that is:

$$\mathcal{P}_{AP}^{(IDLE)}(j,m) = 0.5 \times \frac{\mathcal{P}_{CPU}^{(IDLE)}(j,m)}{nc(j,m)}, \tag{23}$$

while, for the corresponding dynamic power $\mathcal{P}_{AP}^{(DYN)}(j,m)$ (Watt), the following expression holds (see Eq. (20)):

$$\mathcal{P}_{AP}^{(DYN)}(j,m) = \mathcal{K}_{AP}^{(j,m)} \left(\beta_{jm}\tilde{f}_{jm}\right)^{\gamma_{AP}^{(j,m)}}, \tag{24}$$

for $1 \leq m \leq m_m$, and $1 \leq m \leq M$. Therefore, the resulting per-slot energy consumption $\mathcal{E}_{AP}(j,m)$ (Joule) of the auxiliary processor of Fig. 9 is given by (see Eq. (22)):

$$\mathcal{E}_{AP}(j,m) = \underbrace{\mathcal{P}_{AP}^{(IDLE)}(j,m) \times T_{EXE}^{(1,M)}}_{Idle\ Energy} + \underbrace{\mathcal{P}_{AP}^{(DYN)}(j,m) \times T_{AP}(j,m)}_{Dynamic\ Energy}, \tag{25}$$

for $1 \leq m \leq m_m$, and $1 \leq m \leq M$, with (see (16)):

$$T_{AP}(j,m) = \frac{\mathcal{V}_O(j,m)}{\tilde{f}_{jm}}. \tag{26}$$

Remark – Fog devices-vs.-power models fitting. By referring to the virtualized architecture of a Fog node sketched in Fig. 4, we point out that the actual values of the CPU idle powers, scaling $\mathcal{K}$-factors and γ-power exponents present in the previously reported power models mainly depend on: (i) the virtualization strategy actually implemented by the Container Engine; and, (ii) the power profiles of the underlying Physical CPUs (such as, for example, Raspberry Pi, Arduino or Intel Galileo board, just to name a few). Detailed analysis of this topic are recently provided in the contributions [33], [37] and, then, they are not duplicated here. However, in order to give some insights on how the reported power/energy models reflect, indeed, the actual features of real-world virtualized Fog nodes, three main explicative remarks are in order.

First, the analysis reported, for example, in [33] supports the conclusion that the scaling $\mathcal{K}$-factors in Eqs. (20) and (24) depend on both: (i) the number of cores $n_{core} \geq 1$ equipping each virtual processor; and, (ii) the corresponding fraction $r_{core} \in [0,\ 1]$ of the dynamic power shared by all cores for common operations, as shown below:

$$\mathcal{K} = n_{core} \times (1 - r_{core}) \times \mathcal{C}_{CPU}. \tag{27}$$

In the above relationship, the positive constant $\mathcal{C}_{CPU}$ (measured in (Watt/(bit/s)$^{\gamma}$)) features the dynamic-power consumption of the physical CPU which is actually employed for implementing the Fog node.

Second, actual profiling of the power models of Eqs. (19), (20), (23) and (24) may be performed online by equipping each Clone Manager of Fig. 4 with the so-called *Joulemeter* tool in [38]. This is a software package tailored "ad hoc" on virtualized environments, which provides the same power and energy metering functionalities for virtual cores as there exist in hardware for physical cores [37]. As detailed in Section 5 of [38], *Joulemeter* uses container-observable hardware power states to track the per-virtual core energy usage on each hardware component. Interestingly enough, the experimental measurements reported in [38] lead to the general conclusion that the idle power wasted by a virtual core is proportional to the corresponding idle power consumed by the underlying hosting physical server, with the scaling factor being proportional to the number of virtual cores actually supported by the physical server. This is in agreement, indeed, with the relationships adopted in (19) and (23) for modeling the idle power consumed by a virtual clone.

Finally, we stress that, in our setting, all the parameters featuring the previously reported power and energy models play the role of known constants. In agreement with the numerical datasets reported, for example, in [33] and [37], typical values assumed by these power/energy-model parameters in Fog operating environments are reported in Table 9 of final Appendix A.

B. MODELING THE PER-FLOW INTER-TIER NETWORK ENERGIES

Before characterizing the energy consumption of the TCP/IP-based inter-tier transport connections of the *LiFo* platform of Fig. 3, two main remarks about the considered underlying network model are in order.

First, according to the functional protocol stack of Fig. 11, the bit rates: $\{R_m,\ 1 \le m \le (M-1)\}$ in Fig. 3 are to be understood as per-connection Transport layer throughputs. The reason is that these throughputs are, indeed, the actual rates at which the per-tier workloads generated by the supported CDNN of Fig. 1a are transported over the inter-tier networks of Fig. 3. However, both the resulting network power and energy scale up with the corresponding bit rates: $\left\{\widehat{R}_m,\ 1 \le m \le (M-1)\right\}$ measured at the Access sub-layer of the functional protocol stack of Fig. 11. In general, these two sets of bit rates scale proportionally according to [3]:

$$\widehat{R}_m = \psi_m R_m, \quad 1 \le m \le M-1. \tag{28}$$

In the above expression, ψ_m is a larger-than-the-unit dimension-less coefficient that accounts for [3]: (i) the protocol overhead (i.e., the sizes of the packet headers) of the actually implemented protocol stack; and, (ii) the average number of per-packet collisions experienced by the m-th inter-tier flows under the Multiple ACcess (MAC) protocol actually implemented at the Access sub-layer of Fig. 11.

Second, we assume that all the TCP connections going from *tier* #m to *tier* #$(m+1)$ share the same throughput R_m and Round-Trip-Time RTT_m (s). In principle, these assumptions reflect the consideration that all Fog nodes at *tier* #m support the *same* set $\mathcal{S}_m^*$ in (8) of CDNN layers, so that the corresponding traffic flows can be assumed to be load balanced. However, we point out that the generalization of the results presented in the sequel to the general case in which each flow generated by the output ports in Fig. 9 exhibits own throughput $R^k(j,m)$ and round-trip-time $RTT^k(j,m)$, $1 \le k \le$ *FanOut*(j,m), is quite direct and requires only a somewhat more cumbersome formal notation.

1) MODELING THE RECEIVE NETWORK ENERGIES

Let $\mathcal{P}_{NET}^{(IDLE-Rx)}(j,m)$ (Watt) be the network power consumed by each input port of the (j,m)-th Fog node in Fig. 9 in the idle state. According to [33], the corresponding per-receive port dynamic power $\mathcal{P}_{NET}^{(DYN-Rx)}(j,m)$ (Watt) can be modeled as:

$$\mathcal{P}_{NET}^{(DYN-Rx)}(j,m) = \Omega_{NET}^{(Rx)}(j,m) \times (\psi_{m-1}R_{m-1})^{\zeta^{(Rx)}(j,m)}. \tag{29}$$

In the above expression, we have that:

- $\zeta^{(Rx)}(j,m) \ge 2$ is a positive dimension-less exponent, whose actual value depends on the transmission technology actually implemented at the Access sub-layer of the functional stack of Fig. 11 [39], [40]; and,
- the positive coefficient $\Omega_{NET}^{(Rx)}(j,m)$ (measured in (Watt/((bit/s)$^{\zeta}\times$(s)$^{\eta}$)) accounts for the effect of the round-trip-time RTT_{m-1} of the received flows. Specifically, in the case of single-hop connections, it can be formally modeled as in the following [41], [42]:

$$\Omega_{NET}^{(Rx)}(j,m) = \frac{(RTT_{m-1})^{\eta}\ \chi(j,m)^{(Rx)}}{1+(l_{m-1})^{\alpha}}, \tag{30}$$

where: (i) $\eta \approx 0.6$ is a dimension-less positive exponent; (ii) l_{m-1} is the length (measured in meter (m)) of the wireless connections of Fig. 3 going from *tier* #$(m-1)$ to *tier* #m; (iii) $\alpha \ge 2$ is a fading-induced path-loss exponent; and, (iv) the positive coefficient $\chi(j,m)^{(Rx)}$ accounts for the power profile of the receive ports Fig. 9.

Therefore, after noting that the idle time is still given by $T_{EXE}^{(1,M)}$ (see Eq. (18) with $m = M$), while the per-port receive time $T_{NET}^{(Rx)}(j,m)$ of the (j,m)-th Fog node equates to (see Fig. 9):

$$T_{NET}^{(Rx)} = \frac{\mathcal{V}_I(j,m)}{R_{m-1} \times FanIn(j,m)}, \tag{31}$$

from the outset, it follows that the energy $\mathcal{E}_{NET}^{(Rx)}(j,m)$ (Joule) wasted by all receive ports of the (j,m)-th Fog node during a slot time is given by:

$$\begin{aligned}\mathcal{E}_{NET}^{(Rx)}(j,m) = {} & FanIn(j,m) \\ & \times \Bigg(\underbrace{\mathcal{P}_{NET}^{(IDLE-Rx)}(j,m) \times T_{EXE}^{(1,M)}}_{Idle\ Receive\ Energy} \\ & + \underbrace{\mathcal{P}_{NET}^{(DYN-Rx)}(j,m) \times T_{NET}^{(Rx)}}_{Dynamic\ Receive\ Energy}\Bigg).\end{aligned} \tag{32}$$

2) MODELING THE TRANSMIT NETWORK ENERGY

Companion expressions hold for the power and energy consumed by the output ports in Fig. 9. However, we note that wireless transmissions are more power expensive than wireless receptions, so that the power profiles of the output ports are typically different from the corresponding ones of the input ports. Therefore, on the basis of this consideration, the energy $\mathcal{E}_{NET}^{(Tx)}(j,m)$ (Joule) consumed by all output ports of the (j,m)-th Fog node of Fig. 9 during a slot interval may be evaluated as:

$$\mathcal{E}_{NET}^{(Tx)}(j,m) = FanOut(j,m)$$

$$\times\left(\underbrace{\mathcal{P}_{NET}^{(IDLE-Tx)}(j,m)\times T_{EXE}^{(1,M)}}_{Idle\ Transmit\ Energy} + \underbrace{\mathcal{P}_{NET}^{(DYN-Tx)}(j,m)\times T_{NET}^{(Tx)}}_{Dynamic\ Transmit\ Energy}\right). \quad (33)$$

Similarly to Eq. (29), the (j, m)-th dynamic transmit power $\mathcal{P}_{NET}^{(DYN-Tx)}(j, m)$ (Watt) present in Eq. (33) may be expressed as:

$$\mathcal{P}_{NET}^{(DYN-Tx)}(j,m) = \Omega_{NET}^{(Tx)}(j,m)\times(\psi_m R_m)^{\zeta^{(Tx)}(j,m)}, \quad (34)$$

while the following formula for the per-port transmit time:

$$T_{NET}^{(Tx)} = \frac{\mathcal{V}_O(j,m)}{R_m}, \quad (35)$$

accounts for the fact that, by design, the overall workload $\mathcal{V}_O(j, m)$ generated by the (j, m)-th Fog node is entirely replicated over each output port of Fig. 9.

Typical values of the various parameters featuring the networking power/energy models of Eqs. (29)–(34) in Fog operating environments are reported in Table 9 of final Appendix A.

C. MODELING THE PER-SERVICE TOTAL ENERGY OF THE LIFO PLATFORM

The above analysis of the per-slot energy consumed by each Fog node leads to the conclusion that the resulting total energy $\mathcal{E}_{TOT}$ (Joule) consumed by the virtualized *LiFo* platform of Fig. 9 during a slot interval for the execution of the inference phase of the supported CDNN is the summation of a computing energy $\mathcal{E}_{COP}$ (Joule) and a network energy $\mathcal{E}_{NET}$ (Joule). Hence, it can be expressed as:

$$\mathcal{E}_{TOT} = \mathcal{E}_{COP} + \mathcal{E}_{NET}, \quad (36)$$

where, in turn, the computing and networking contributions are given by:

$$\mathcal{E}_{COP} = \sum_{m=1}^{M}\sum_{j=1}^{m_m}\left(\mathcal{E}_{MP}(j,m)+\mathcal{E}_{AP}(j,m)\right), \quad (37)$$

and:

$$\mathcal{E}_{NET} = \underbrace{\sum_{j=1}^{m_1}\mathcal{E}_{NET}^{(Tx)}(j,1)}_{Tx\ energy\ at\ tier\ \#1} + \underbrace{\sum_{m=2}^{M-1}\sum_{j=1}^{m_m}\left(\mathcal{E}_{NET}^{(Rx)}(j,m)+\mathcal{E}_{NET}^{(Tx)}(j,m)\right)}_{Tx-plus-Rx\ energy\ at\ the\ middle\ tiers} + \underbrace{\mathcal{E}_{NET}^{(Rx)}(1,M)}_{Rx\ energy\ at\ tier\ \#M}. \quad (38)$$

About the practical meaning of $\mathcal{E}_{TOT}$ in (36), we remark that this energy is evaluated on a per-slot basis, and accounts for the total energy consumed for the execution of the inference phase of a *single* CDNN. Since the virtualized nature of the proposed *LiFo* technological platform allow the isolated and *parallel* executions of multiple CDNNs through the exploitation of the $nc \geq 1$ containers equipping each Fog node (see Fig. 4), we point out that, in principle, the full energy consumed by the *LiFo* platform of Fig. 3 in practical operative scenarios can be up to nc times larger than the one modeled by Eq. (36).

Overall, the final Table 9 reports the adopted taxonomy, as well as the typical values of the parameters involved by the computing/networking power/energy models introduced in this Section V.

VI. THE LIFO OPTIMIZATION PROBLEM (LOP) AND ITS STRUCTURAL PROPERTIES

Let $Q \triangleq \sum_{m=1}^{M} m_m$ be the number of Fog-plus-Cloud computing nodes composing the *LiFo* technological platform of Fig. 3 and let:

$$\begin{aligned}\overrightarrow{F} &\triangleq [f_{11}, f_{21}, \ldots, f_{M1}]^T \in \mathbb{R}_+^Q,\\ \overrightarrow{\tilde{F}} &\triangleq \left[\tilde{f}_{11}, \tilde{f}_{21}, \ldots, \tilde{f}_{M1}\right]^T \in \mathbb{R}_+^Q,\\ \overrightarrow{R} &\triangleq [R_1, R_2, \ldots, R_M]^T \in \mathbb{R}_+^{M-1},\end{aligned} \quad (39)$$

be the (column) vectors of the (previously introduced) processing frequencies of the main and auxiliary processors and the (column) vector of the inter-tier network throughputs (see Fig. 9). Furthermore, let:

$$\overrightarrow{X} \triangleq \left[\overrightarrow{F}^T, \overrightarrow{\tilde{F}}^T, \overrightarrow{R}^T\right]^T \in \mathbb{R}_+^{2Q+(M-1)}, \quad (40)$$

be the resulting compound resource (column) vector.

By design, the *LiFo* technological platform of Fig. 3 aims at supporting DL-based applications in which: (i) the most relevant *QoS* metrics are the experienced per-exit inference delays in Eq. (18); (ii) the total energy consumption in Eq. (36) must be driven to the minimum; (iii) the per-clone processing frequencies embraced by the vectors $\overrightarrow{F}$ and $\overrightarrow{\tilde{F}}$ in (39) are upper limited, in order to allow each physical Fog node in Fig. 3 to maximize the number of containers which may run in parallel (see Fig. 4); and, (iv) the inter-tier throughputs gathered by the vector $\overrightarrow{R}$ in Eq. (39) must be also upper bounded, in order to allow the interference-free concurrent implementation of multiple transport connections over a same physical bandwidth [39].

According to these considerations, we formulate the *LiFo Optimization Problem* (*LOP*) for the constrained allocation of the available computing-plus-networking resources to the computing nodes of the platform in Fig. 3 as follows:

$$\min_{\overrightarrow{X}} \mathcal{E}_{TOT}, \quad (41a)$$

$$\text{s.t.: } 0 \leq T_{EXE}^{(1,m)} \leq T_{EXIT}^{(m)}, \quad (41b)$$

$$0 \leq f_{jm} \leq f_{im}^{(MAX)}, \quad (41c)$$

$$0 \leq \tilde{f}_{jm} \leq \tilde{f}_{im}^{(MAX)}, \quad (41d)$$

$$0 \leq R_m \leq R_m^{(MAX)}, \tag{41e}$$

for $m = 1, \ldots, M$ and $j = 1, \ldots, m_m$, where the elements of the sets: $\left\{T_{EXIT}^{(m)}\right\}$, $\left\{f_{jm}^{(MAX)}\right\}$, $\left\{\tilde{f}_{jm}^{(MAX)}\right\}$ and $\left\{R_m^{(MAX)}\right\}$ play the role of (positive) assigned constants. Specifically, in the presented *LOP* formulation, we have that:

- the objective function $\mathcal{E}_{TOT}$ in (41a) to be minimized is formally given by Eq. (36);
- the M constraints in (41b) guarantee that each per-exit inference time $T_{EXE}^{(1,m)}$ in Eq. (18) is limited up to a corresponding *QoS*-dictated maximum value $T_{EXIT}^{(m)}$ (s);
- the box constraints in Eq. (41c) (resp., in Eqs. (41d) and (41e)) limit each main processing frequency f_{jm} (resp., auxiliary processing frequency $\tilde{f}_{jm}$ and inter-flow throughput R_m) up to a corresponding maximum value $f_{jm}^{(MAX)}$ (bit/s) (resp., $\tilde{f}_{jm}^{(MAX)}$ (bit/s) and $R_m^{(MAX)}$ (bit/s).

From a formal point of view, the *LOP* in (41) is an optimization problem which embraces: $2Q + (M - 1)$ nonnegative *continuous-valued* variables and: $2(M + Q) - 1$ constraints. The constraints in Eqs. (41b) – (41e) are of box-type, while the M constraints in Eq. (41b) on the allowed per-exit inference times are *nonlinear* constraints involving all elements of the optimization vector $\overrightarrow{X}$ in (40).

Hence, regarding the convex/nonconvex nature of the *LOP*, the following result is proved in the final Appendix B.

Proposition 1 (On the convexity of the LOP): Assume that the values of all the γ's and ζ's exponents present in the power models previously presented in Eqs. (20), (21), (29) and (34) are not less than 2. Hence, the resulting *LOP* in Eq. (41) is a *convex* optimization problem. ■

To examine the feasibility of the considered *LOP*, let:

$$\begin{aligned}
\overrightarrow{F}^{(MAX)} &\triangleq \left[f_{11}^{(MAX)}, \ldots, f_{M1}^{(MAX)}\right]^T \in \mathbb{R}_+^Q, \\
\overrightarrow{\tilde{F}}^{(MAX)} &\triangleq \left[\tilde{f}_{11}^{(MAX)}, \ldots, \tilde{f}_{M1}^{(MAX)}\right]^T \in \mathbb{R}_+^Q, \\
\overrightarrow{R}^{(MAX)} &\triangleq \left[R_1^{(MAX)}, \ldots, R_M^{(MAX)}\right]^T \in \mathbb{R}_+^{M-1},
\end{aligned} \tag{42}$$

be the (column) vectors of the maximum allowable computing-networking resources (see the box constraints in Eqs. (41c) – (41e)), and, then, let :

$$\overrightarrow{X}^{(MAX)} \triangleq \left[\overrightarrow{F}^{(MAX)^T}, \overrightarrow{\tilde{F}}^{(MAX)^T}, \overrightarrow{R}^{(MAX)^T}\right]^T, \tag{43}$$

be the corresponding compound maximal (column) vector. Furthermore, let:

$$T_{EXE}^{(1,m)}\left(\overrightarrow{X}^{(MAX)}\right), \quad 1 \leq m \leq M, \tag{44}$$

be the value assumed by the m-th inference time in Eq. (18) when all the resources are clipped at their own maximal values. Hence, the following result holds.

Proposition 2 (On the feasibility of the LOP): Let the *LOP* be convex. Then, it is *feasible* if and only if the following inequalities are simultaneously met:

$$T_{EXE}^{(1,m)}\left(\overrightarrow{X}^{(MAX)}\right) \leq T_{EXIT}^{(m)}, \quad 1 \leq m \leq M. \tag{45}$$

■

Proof: In order to prove the sufficiency of the conditions in (45), we simply note that: $\overrightarrow{X} \equiv \overrightarrow{X}^{(MAX)}$ is a feasible solution of the *LOP* in (41), because, by design, it meets all the box constraints in (41c) – (41e) on the allowed maximal resources with the equality.

The necessary part can be proved by contradiction. For this purpose, let us assume that at least one of the condition in (45) fails, i.e., let us suppose that the $\tilde{m}$-th inequality in (45) falls short. Hence, in order to reduce the corresponding value $T_{EXE}^{(1,\tilde{m})}\left(\overrightarrow{X}^{(MAX)}\right)$ of the $\tilde{m}$-th inference time, we are forced to *increase at least one* scalar component of $\overrightarrow{X}^{(MAX)}$, because *Lemma 1.b* in the Appendix B proves that the function $T_{EXE}^{(1,\tilde{m})}\left(\overrightarrow{X}\right)$ is not increasing with respect to each scalar component of the resource vector $\overrightarrow{X}$ in (40). However, by doing so, we arrive to an *infeasible* vector $\overrightarrow{X}$, because all the components of $\overrightarrow{X}^{(MAX)}$ are already set, by definition, to their own allowed maximum allowable values. This proves, in turn, that the conditions in (45) are also necessary for the *LOP* feasibility. □

To formally characterize the *LOP* solution, let us assume that the *LOP* in (41) is convex and feasible, and, then, let:

$$\overrightarrow{X}^* \triangleq \left[\overrightarrow{F}^{*^T}, \overrightarrow{\tilde{F}}^{*^T}, \overrightarrow{R}^{*^T}\right]^T, \tag{46}$$

indicate its (possible, not unique) vector solution. Hence, the general results provided, for example, in Theorems 4.3.7 and 4.3.8 of [43] guarantee that the Karush-Kuhn-Tucker (KKT) conditions are necessary and sufficient for the evaluation of the *LOP* solution in (46) when the *LOP* also meets the Slater's qualification condition. In this regard, the following formal result can be proved.

Proposition 3 (Slater's qualification for the LOP): Let us assume that the *LOP* is convex and, in addition, all the feasibility conditions in (45) are met with the *strict* inequalities. Then, the *LOP* in (41) satisfies the Slater's qualification condition. ■

Proof: Since the *LOP* is assumed to be convex, all the delay constraints in (41b) are, by definition, convex. Therefore, as detailed, for example in [43], the Slater's qualification condition requires that there exists at least one resource allocation vector $\overrightarrow{X}$ in (40) which is feasible and meets all the M constraints in (41b) with the strict inequalities. However, if the feasibility conditions previously presented in (45) are met with the strict inequalities, then, the vector $\overrightarrow{X}^{(MAX)}$ in (43) of the maximal allowed resources is, by definition, feasible, and, in addition, it satisfies all the convex constraints in Eq. (41b) with the strict inequalities. This guarantees, in turn, that the Slater's qualification condition holds. □

Overall, according to the presented results and in order to formally characterize the *LOP* vector solution in (46), we will assume that the considered *LOP* in (41) is convex, feasible and meets the Slater's qualification condition.

VII. THE LOP SOLVING APPROACH

The previously described operative setting allows us to resort to a suitable application of the so-called *Primal-Dual Solving Approach* (*PDSA*) [43], in order to evaluate the solution $\overrightarrow{X}^*$ in (46) of the considered *LOP*. Towards this end, we note that, since the *LOP* is a convex optimization problem by *Proposition 1*, the box constraints in Eqs. (41c) – (41e) may be retained as implicit ones and, then, managed by performing clipping operations [43]. Hence, the Lagrangian function $\mathcal{L}\left(\overrightarrow{X}, \overrightarrow{\lambda}\right)$ (measured in (Joule)) associated to the *LOP* in (41) is defined as follows:

$$\mathcal{L}\left(\overrightarrow{X}, \overrightarrow{\lambda}\right) \triangleq \mathcal{E}_{TOT}\left(\overrightarrow{X}\right) + \sum_{m=1}^{M} \lambda_m \left(\frac{T_{EXE}^{(1,m)}\left(\overrightarrow{X}\right)}{T_{EXIT}^{(m)}} - 1 \right), \tag{47}$$

where λ_m (Joule) is the (scalar nonnegative) Lagrange multiplier associated to the m-th delay-constraint in Eq. (41b), and:

$$\overrightarrow{\lambda} \triangleq [\lambda_1, \lambda_2, \ldots, \lambda_M]^T \in \mathbb{R}_+^M, \tag{48}$$

is the corresponding M-dimensional multipliers' vector. Therefore, from a formal point of view, the constrained *max–min* optimization problem to be solved for the evaluation of the *LOP* solution $\overrightarrow{X}^*$ in (46) and the corresponding optimum multiplier vector $\overrightarrow{\lambda}^*$ is the following one [43]:

$$\max_{\overrightarrow{\lambda} \geq \overrightarrow{0}} \left\{ \min_{\overrightarrow{0} \leq \overrightarrow{X} \leq \overrightarrow{X}^{(MAX)}} \left\{ \mathcal{L}\left(\overrightarrow{X}, \overrightarrow{\lambda}\right) \right\} \right\}, \tag{49}$$

where $\overrightarrow{X}^{(MAX)}$ is the vector in (43) of the maximal allowable resources. Now, let: $\overrightarrow{\nabla}\mathcal{L}\left(\overrightarrow{X}, \overrightarrow{\lambda}\right)$ be the $(2(Q+M)-1)$-dimensional gradient vector of the Lagrangian function in (47) done with respect to both the $(2Q+M-1)$ scalar components of $\overrightarrow{X}$ in (40) (i.e., the vector of the so-called *primal* variables of the *LOP*; see [43]) and the M scalar components of $\overrightarrow{\lambda}$ in (48) (i.e., the vector of the so-called *dual* variables of the *LOP*; see [43]). Hence, *Theorem 6.2.6* of [43] guarantees that the solution of the *max–min* optimization problem in (49) (that is, the so-called saddle-point of the Lagrangian function in (47)) can be computed by performing the orthogonal projection over the Cartesian-product set: $\left[\overrightarrow{0}, \overrightarrow{X}^{(MAX)}\right] \times \mathbb{R}_+^M$ of the compound vector solution $\left[\overrightarrow{X}^T, \overrightarrow{\lambda}^T\right]^T$ of the following $(2(Q+M)-1)$-dimensional algebraic equation system:

$$\overrightarrow{\nabla}\mathcal{L}\left(\overrightarrow{X}, \overrightarrow{\lambda}\right) = \overrightarrow{0}. \tag{50}$$

A. THE SOLVING APPROACH

Two remarks about the solution:

$$\left\{\overrightarrow{X}^*, \overrightarrow{\lambda}^*\right\}, \tag{51}$$

of the equation system in (50) are in order. First, since both the objective function $\mathcal{E}_{TOT}$ in (41a) and the inference times $\left\{T_{EXE}^{(1,m)}\right\}$ in (41b) are nonlinear functions of the resource allocation vector $\overrightarrow{X}$ in (40) (see Eqs. (18) and (36) and the related texts), the resulting system in (50) is also nonlinear in $\overrightarrow{X}$, so that its solution *resists* closed-form evaluation. Second, even if it would possible to solve in closed-form the system in (50) under some specific cases, the obtained closed-form solution should be *re-evaluated from scratch* when, due to the mobility of the IoT devices in Fig. 1b and/or fading effects, the operating conditions[1] of the *LiFo* platform of Fig. 3 undergo abrupt (and, typically, unpredictable) time variations.

In order to tackle with these challenges, we develop a suitable set of projected gradient-based primal-dual iterations whose step-sizes are *adaptively-scaled* on the basis of suitably designed *time-varying clipping thresholds*.

Towards this end, we begin to remark that, as detailed, for example, in [43], the primal-dual algorithm is an iterative procedure that updates step-by-step both the involved primal: $\overrightarrow{X}$, and dual: $\overrightarrow{\lambda}$ vector variables, in order to throttle the corresponding Lagrangian function in (47) towards its saddle-point, i.e., the solution of *max–min* optimization problem in (49). Hence, according to [43], after introducing the dummy position:

$$[z]_a^b \triangleq \max\{a;\ \min\{z;\ b\}\}, \tag{52}$$

the $(k+1)$-th updating of the i-th scalar component x_i, for $i = 1, \ldots, (2Q+M-1)$, of the compound resource vector $\overrightarrow{X}$ in (40) reads as in:

$$x_i(k+1) = \left[x_i(k) - \alpha_i(k) \frac{\partial \mathcal{L}\left(\overrightarrow{X}(k), \overrightarrow{\lambda}(k)\right)}{\partial x_i} \right]_0^{x_i^{(MAX)}}, \tag{53}$$

for $k \geq 0$ and $i = 1, \ldots, (2Q+M-1)$, while the $(k+1)$-th updating of the m-th scalar component λ_m, $m = 1, \ldots, M$, of the multiplier vector $\overrightarrow{\lambda}$ is given by:

$$\lambda_m(k+1) = \left[\lambda_m(k) - \xi_m(k) \frac{\partial \mathcal{L}\left(\overrightarrow{X}(k), \overrightarrow{\lambda}(k)\right)}{\partial \lambda_m} \right]_0^{\infty}, \tag{54}$$

for $k \geq 0$ and $m = 1, \ldots, M$. In the above iterations, we have that:

1) $k \geq 0$ is an integer-valued iteration index;
2) $\partial\mathcal{L}\left(\overrightarrow{X}(k), \overrightarrow{\lambda}(k)\right)/\partial x_i$ (resp., $\partial\mathcal{L}\left(\overrightarrow{X}(k), \overrightarrow{\lambda}(k)\right)/\partial \lambda_m$) is the partial derivative of the Lagrangian function in (47) with respect to the i-th scalar component x_i of the resource vector $\overrightarrow{X}$ (resp., the m-th scalar component of the Lagrange multiplier vector $\overrightarrow{\lambda}$) at the k-th iteration;
3) $x_i^{(MAX)}$ is the i-th scalar component of the maximal resource vector $\overrightarrow{X}^{(MAX)}$ in (43); and,

[1] That is, the components of the maximal resource vector $\overrightarrow{X}^{(MAX)}$ in (43), and/or the volume $\mathcal{V}_0$ of the data generated by the IoT devices over a slot interval.

4) $\{\alpha_i(k),\ i = 1, \ldots, (2Q + M - 1)\}$ and $\{\xi_m(k),\ m = 1, \ldots, M\}$ are nonnegative k-varying step-size sequences.

B. ACHIEVING FAST ADAPTATION AND FAILURE RESILIENCE: THE PROPOSED OPTIMIZED STEP-SIZE DESIGN

A still open challenge is how the time-varying step-size sequences in Eqs. (53) and (54) should be actually designed, in order to allow the resulting *LiFo* technological platform of Fig. 3 to: (i) quickly adapt to abrupt changes of the wireless (possibly, mobile) IoT operating scenario; and/or, (ii) fast react to (typically, unpredictable) failure events. For this purpose, we *further optimize* the approach presented in [44] by introducing a "clipped" version with *adaptive per-tier clipping thresholds* of the step-size design presented in [44].

In this regard, we observe that *Theorem 3.3* of [44] proves that it suffices to update each step-size sequence present in Eqs. (53) and (54) proportionally to the *current* squared value of the *corresponding* primal-dual variable, in order to guarantee the *asymptotic* convergence to the global optimum of the full set of primal-dual iterations. However, too large (resp., too low) instantaneous values of the step-sizes may stretch the resulting convergence times by introducing unstable transient oscillatory phenomena (resp., by reducing the capability of the system to react to environmental changes).

Hence, motivated by these considerations, we planned to implement the *adaptively clipped* relationships for updating the step-sizes in (53) and (54), given by Eqs. (55) and (56) as shown at the bottom of next page, where m in (55) is the index of the tier in Fig. 3 at which the i-th (scalar) resource x_i is allocated.

1) THE PROPOSED OPTIMIZED DESIGN OF THE STEP-SIZE CLIPPING THRESHOLDS

The goal of the (positive-valued) clipping threshold $a_{MAX}^{(m)}(k)$ in (55) and (56) is to avoid that the corresponding step-sizes assume too small or too large instantaneous values, in order to attain both quick reactions to environmental changes and small oscillations in the steady-state. However, *unlike* state-of-the-art contributions (see, for example, [45] and the references therein), the *peculiar* feature of the clipping threshold $a_{MAX}^{(m)}(k)$ in (55) and (56) is that it is *no longer* fixed as a static hyper-parameter, but it *may vary* over *both* the time k and the spatial m dimensions of the underlying *LiFo* technological platform of Fig. 3. For this purpose, we *propose* the following relationship for the *adaptive* updating of $a_{MAX}^{(m)}(k)$ over the space-time indexes:

$$a_{MAX}^{(m)}(k) = A_0\left(1 + \sigma\left|\frac{T_{EXE}^{(1,m)}(k)}{T_{EXIT}^{(m)}} - 1\right|\right), \tag{57}$$

for $k \geq 0$ and $m = 1, \ldots, M$.

In the above relationship, we have that: (i) k (resp., m) is the iteration (resp., tier) index; (ii) $T_{EXE}^{(1,m)}(k)$ is the actual value assumed by the m-th inference time in (18) at the k-th iteration, while $T_{EXIT}^{(m)}$ is the corresponding tolerated maximum value in (41b); and, (iii) A_0 and σ are positive dimension-less hyper-parameters.

The rationales leading to the proposed relationship in (57) rely on the following four formal considerations.

First, values of the ratio $T_{EXE}^{(1,m)}(k)/T_{EXIT}^{(m)}$ larger than the unit would violate the *LOP* constraint in (41b) and, then, are infeasible. However, since *Lemma 1.b* in Appendix B proves that $T_{EXE}^{(1,m)}$ is a *not increasing* function with respect to each scalar component of the resource vector $\overrightarrow{X}$ in (40), values of the ratio $T_{EXE}^{(1,m)}(k)/T_{EXIT}^{(m)}$ too below the unit would cause useless resource wasting, and, then, they should be avoided. As a consequence, the optimal target value of the ratio $T_{EXE}^{(1,m)}(k)/T_{EXIT}^{(m)}$ is the unit one, and this motivates the presence of the absolute term $|\cdot|$ in (57).

Second, since larger (resp., smaller) values of the absolute term $|\cdot|$ in (57) increase (resp., decrease) the current value of $a_{MAX}^{(m)}(k)$, they dynamically enlarge (resp., narrow) the ranges of the values allowed to the corresponding step-sizes in (55) and (56). So doing, the capability of the associated iterations in (53) and (54) to explore new solutions falling into a neighborhood of the current one $\overrightarrow{X}(k)$, $\overrightarrow{\lambda}(k)$ is increased (resp., decreased).

Third, the (positive) hyper-parameter $A_0 > 0$ in (57) lowers bound, by design, the allowable range of values $a_{MAX}^{(m)}(k)$. Therefore, its role is to forbid vanishing values of $a_{MAX}^{(m)}(k)$, which, in turn, would give rise to vanishing step-sizes in (55) and (56) and, then, would annihilate the adaptive capability of the iterations in (53) and (54) to *self*-react to environmental changes.

Forth, the main role of the nonnegative sensitivity hyper-parameter $\sigma \geq 0$ in (57) is to amplify the effect on $a_{MAX}^{(m)}(k)$ induced by nonzero values of the absolute gap: $\left|\left(T_{EXE}^{(1,m)}(k)/T_{EXIT}^{(m)}\right) - 1\right|$, so to magnify the dynamic of the allowable ranges of the step-sizes values in (55) and (56). In this regard, we explicitly note that the limit case of vanishing σ is the state-of-the-art case in which the iterations in (53) and (54) work with their maximal step-size values *statically fixed* at A_0.

Finally, regarding the actual effectiveness of the proposed dynamic relationship in (57) for the updating of the per-tier maximal step-sizes values, we anticipate that the numerical results presented in Section IX support the conclusion that its utilization allows to (*at least*) *halve* the convergence time of the set of primal-dual iterations in (53) and (54) with respect to the corresponding static cases in which all the step-size thresholds in (57) are taken fixed at A_0.

C. IMPLEMENTATION COMPLEXITY OF THE LOP ITERATIONS

The pseudo-code of Algorithm 1 presents the ordered list of relationships that are needed for the software implementation of the set of *LOP* iterations in (53) and (54).

After noting that I_{MAX} in Algorithm 1 is the number of performed primal-dual iterations, an examination of the presented pseudo-code leads to two main conclusions.

Algorithm 1 Computing the LOP Resource Allocation Vector and Related Energy Consumptions

Input: Hyper-parameters A_0, σ and number I_{MAX} of iterations to be carried out.
Output: Resource allocation vector $\overrightarrow{X}$ and corresponding total, computing and network energies $\mathcal{E}_{TOT}$, $\mathcal{E}_{COP}$ and $\mathcal{E}_{NET}$.
▷ **Begin LOP function**

```
1:  Initialize: $\overrightarrow{X} = \overrightarrow{0}$ and $\mathcal{E}_{TOT} = \mathcal{E}_{COP} = \mathcal{E}_{NET} = 0$;
2:  if the set of LOP feasibility conditions in (45) is met then
3:      for $k = 0 : (I_{MAX} - 1)$ do
4:          Update the set of Lagrangian derivatives in (53) and (54);
5:          Update the inference times in (18);
6:          Update the step-size thresholds in (57);
7:          Update the step-sizes in (55) and (56);
8:          Update the set of primal-dual iterations in (53) and (54);
9:          Update the current values of the energies $\mathcal{E}_{TOT}$, $\mathcal{E}_{COP}$ and $\mathcal{E}_{NET}$ in (36), (37) and (38);
10:     end for
11: end if
12: return the final resource allocation vector $\overrightarrow{X}$ and the related energies $\mathcal{E}_{TOT}$, $\mathcal{E}_{COP}$ and $\mathcal{E}_{NET}$.   ▷ End LOP function
```

First, the computational complexity, needed for the implementation of the *LOP* iterations in (53) and (54), scales up in a *linear way* with I_{MAX} as:

$$\mathcal{O}(I_{MAX} \times (2 \times (Q + M) - 1)). \tag{58}$$

Second, this complexity is *fully independent* of the (possibly large) number L of the layers of the supported CDNN of Fig. 1a. Furthermore, it scales in a *linear way* with respect to the summation of the number $Q \equiv \sum_{m=1}^{M} m_m$ of the computing nodes and the number M of the tiers of the implemented *LiFo* platform of Fig. 3.

Overall, since the actual values of both Q and M are typically limited due to the monetary cost of the underlying IoT technological platform [3] and the iterations in (53) and (54) are to be carried out *only* in response to (substantial) changes of the surrounding operating conditions, we expect that, in practice, the computation complexity required for the implementation of proposed Algorithm 1 remains, indeed, limited.

VIII. IMPLEMENTATION ASPECTS OF THE PROPOSED LIFO TECHNOLOGICAL PLATFORM

The goal of this section is to details a number of implementation aspects of the proposed *LiFo* technological platform of Fig. 3 which mainly concern: (i) the centralized-vs.-distributed implementation of the primal-dual iterations in (53) and (54); (ii) the online profiling of the power and energy models of Section V of a *LiFo* virtualized Fog node; and, (iii) the design of the main building blocks and functionalities of the network architecture for the support of intra-tier message passing and information retrieving.

A. *IMPLEMENTING THE* LOP *ITERATIONS*

Regarding the implementation of the iterations in (53) and (54), the following three main aspects should be investigated, namely: (i) who runs the iterations; (ii) when starting the iterations; and, (iii) when stopping the iterations.

1) WHO RUNS THE ITERATIONS

In principle, the primal-dual iterations in (53) and (54) could be implemented in a centralized or distributed way.

Under the centralized implementation, the Cloud node at the top of the hierarchy of Fig. 3 runs the *full* set of iterations and, then, after their convergence, broadcasts down to the underlying Fog nodes the vector $\overrightarrow{X}^*$ in (46) of the optimized resource allocation.

In the distributed case, we have that: (i) the (j, m)-th Fog node of Fig. 3 runs only two primal iterations, i.e., the ones needed for updating the frequencies $f_{jm}(k)$ and $\tilde{f}_{jm}(k)$ of own main and auxiliary virtual processors of Fig. 9; (ii) the Cloud node of Fig. 3 runs the $(2M - 1)$ primal and dual iterations for updating the sets of the inter-tier throughputs $\{R_m(k),\ m = 1, \ldots, (M-1)\}$ and Lagrange multipliers $\{\lambda_m(k),\ m = 1, \ldots, M\}$, whose values are, indeed, shared by all Fog nodes of Fig. 3; and, (iii) the Cloud node also updates *all* the Lagrangian derivatives needed for computing the iterations in (53) and (54), whose computations require

$$\alpha_i(k) = \max\left\{a_{MAX}^{(m)}(k);\ \min\left\{a_{MAX}^{(m)}(k) \times \left(x_i^{(MAX)}\right)^2;\ (x_i(k))^2\right\}\right\}, \tag{55}$$

$$\xi_m(k) = \max\left\{a_{MAX}^{(m)}(k);\ \min\left\{a_{MAX}^{(m)}(k) \times \max_i\left\{\left(x_i^{(MAX)}\right)^2\right\};\ (\lambda_m(k))^2\right\}\right\}. \tag{56}$$

the *joint* utilization of *all* components of the resource vector $\overrightarrow{X}$ and Lagrangian multipliers' vector $\overrightarrow{\lambda}$.

Overall, the centralized (resp., distributed) solution does not require (resp., demand for) synchronized inter-Fog signaling, but focuses (resp., scatters) the overall computational effort of Eq. (58) on a single computing node (resp., over multiple cooperating computing nodes).

2) WHEN RUNNING THE ITERATIONS

The iterations in (53) and (54) must be run *from scratch* at the *beginning* of the inference phase, i.e., when the *LiFo* platform of Fig. 3 is bootstrapped for the first time. After their convergence, they *no longer* run, provided that *no changes* occur in the operating conditions of the underlying *LiFo* platform. However, if some environmental changes happens at slot t (i.e., the volume $\mathcal{V}_0(t)$ of the data generated by the IoT devices at the beginning of slot t is very different from the one experienced in the previous slot $(t-1)$ or a failure event occurred at the current slot t), the iterations in (53) and (54) must be run once a time by starting from the *most* recently computed resource allocation vector $\overrightarrow{X}^*(t-1)$. This means that, under the envisioned *LiFo* paradigm: (i) the execution of the iterations for resource allocation is inherently *event-driven*; and, (ii) the proposed dynamic design in (57) of the ranges of the allowable step-size values enables the *Lifo* platform to *both self*-detect environmental changes and *self*-react to them.

3) ADAPTIVE STOPPING OF THE ITERATIONS

In practical application scenarios, the convergence time of the iterations in (53) and (54) is not known in advance, and, which is the most, it could change from time to time, in response to the nature of the fluctuations/failures actually experienced by the environmental operating conditions. Hence, it would be worth designing a stopping mechanism which *adaptively* (i.e., autonomously) stops the execution of the iterations in (53) and (54) when they are approaching the steady-state. For this purpose, we designed the following rule for the adaptive setting of the stopping iteration index k_{STOP}:

$$k_{STOP}: \quad 0.99 \leq \max_{1\leq m\leq M}\left\{\frac{T_{EXE}^{(1,m)}(k_{STOP})}{T_{EXIT}^{(m)}}\right\} < 1. \tag{59}$$

The above stopping rule guarantees that, at the stopping instant k_{STOP}, the following two properties are retained: (i) all the constraints in (41b) on the inference times are met (see the maximum operator and the *strict* unit-valued upper bound in (59)); and, (ii) no wasting of computing/networking resources occurs at the stopping instant (see the *nearly-unit* lower bound in (59)). We anticiapte that all the numerical results of Section IX have been obtained under the stopping rule in (59).

B. IMPLEMENTING THROUGHPUT ADAPTATION

In order to consider the actual implementation of mechanisms for the adaptive scaling of the rates: $\{R_m, \ m=1,\ldots,M\}$ supported by the inter-tier transport connections of Fig. 3, we note that these mechanisms may be implemented through flow control [46]. In this regard, we remark that, by design, the inter-tier *LiFo* transport connections of Fig. 3 are *TCP* supported and, then, flow control is to be considered already implemented at the Transport sub-layer of the protocol stack equipping each *LiFo* Fog node (see Fig. 11). Hence, according to this observation, in the *LiFo* setting, the throughput R_m of the m-th *TCP* flow of Fig. 3 equates, by design, to:

$$R_m = \frac{wd_m}{RTT_m}, \tag{60}$$

where [46]: (i) wd_m (measured in (bit)) is the current size of the congestion window of each *TCP* connection operating at *tier* #m; and, (ii) RTT_m (measured in second) is the corresponding per-connection Round Trip Time. Therefore, since RTT_m can be considered constant under steady-state operating conditions [46], throughput adaptation can be directly implemented by dynamically scaling up/down the corresponding *TCP* window size wd_m in (60).

C. IMPLEMENTING THE INTRA-TIER LOCAL NETWORKS – AN INFORMATION CENTRIC APPROACH

We propose an *UDP*-supported and *Information Centric Network* (*ICN*) architecture for sustaining inter-Fog and Fog-Aggregator communication over each cluster of the *LiFo* technological platform of Fig. 3. The reason for this architectural proposal is threefold.

First, *ICN* is emerging as an effective paradigm to cope with the scarcity of address-space which typically affects standard *IP*-based networks in IoT realms [13]. The main feature doing interesting the utilization of *ICN* in IoT operating scenarios is that this network paradigm allows to enlarge the address space of the state-of-the-art Internet protocol by adopting a name-based content addressing. Doing so, *ICN* enables both name-driven routing and name-based content caching at the intermediate network nodes [13].

Second, the Aggregator equipping each cluster of the *LiFo* platform of Fig. 3 can carry out the fusion of the data locally generated by the corresponding Fog nodes by efficiently exploiting the aforementioned properties natively retained by the *ICN* paradigm. We note that, for this purpose, the Aggregator can directly exploit the publish-subscribe functionalities supported by the *ICN* suite, in order to issue request/response commands for the selective and name-based retrieving of data temporarily stored by the virtual caches of the Fog nodes (see Fig. 9).

Third, the choice of the (unreliable but connection-less) *UDP* as intra-tier transport protocol arises from the considerations that: (i) since intra-tier communication involves proximate nodes scattered over a same spatial cluster (see Fig. 3), Transport-layer reliability is expected not to be a major issue; and, (ii) the connection-less nature of *UDP* avoids the per-connection signaling overhead associated to the initial *SYN/SYN–ACK/ACK* three-way handshake and the final *FIN/ACK* two-way termination required by *TCP*-based network architectures. Doing so, the resulting intra-tier data delivery delays are reduced and, then, fast data fusion by the Aggregator is enabled.

In the following, we sketch the main building blocks and the associated data-retrieving mechanisms envisioned for the proposed *ICN* architecture.

1) MAIN BUILDING BLOCKS AND PROTOCOL MESSAGES

Fig. 12 shows the main blocks which should equip each *LiFo* Fog node of Fig. 3 for the support of the proposed intra-tier *ICN* architecture.

Specifically, the *ICN* setting envisioned for the *LiFo* paradigm relies on the following main data messages and signaling mechanisms:

1) each data chunk stored by the virtual caches in Fig. 9 of a *LiFo* Fog node is assumed to be identified by an unique, persistent and location-independent name, which is used by the corresponding Aggregator for search and retrieval purposes. Inter-Fog and Fog-Aggregator intra-tier communication uses two signaling messages, that is: (a) the *Interest* message, which is used to request a specific data addressed by its name; and, (b) the *Data* message, which carries out the data required by the corresponding Interest message. Since both such types of message are assumed to be conveyed by connection-less *UDP*-based pipes, unsatisfied Interest messages can be simply re-sent at any time by the requiring node, *without* the need to resume any previous data connection; and,
2) each Data message is, by design, *self-consistent*, so that it can be cached by any intermediate Fog node of Fig. 3, in order to serve future requests *without* involving the original source node. Doing so, both the intra-tier network traffics and the resulting retrieval delays are expected to be reduced with respect to the case of a host-centric *IP*-based network architecture. Furthermore, an *AgeTime* may be also included by the original source in each generated data message, so to allow the requiring node to evaluate the actual freshness of the received data.

According to Fig. 12, in the *LiFo* setting, each Fog node is assumed to be also equipped with the following building blocks, in order to enable the name-based data retrieving process by the corresponding Aggregator:

1) the (already introduced) *VIC* and *VOC* of Fig. 9, to temporarily cache input/output Data messages;
2) a *Pending Interest Table* (*PIT*), to temporarily buffer incoming Interest messages which are not still consumed by the corresponding Data messages;
3) a *Forwarding Information Data Base* (*FIDB*), to forward the received Interest messages to the corresponding data source nodes;
4) a *Routing Information Base* (*RIB*), to record already acquired route paths; and,
5) a *Strategy Rule Table* (*SRT*), to gather specific routing strategies chosen by the system administrator for tackling with specific operating conditions and/or Interest messages.

2) DESIGNED INTRA-TIER DATA RETRIEVING PROCESS

The processing chain followed by an Interest message is illustrated in Fig. 12a. Specifically, when a Fog node receives in upstream an Interest message by the Aggregator from an input port, it begins to search in its *VIC* for a name matching. If a name matching is found, then, the corresponding stored Data message is sent back to the Aggregator immediately. Otherwise, the Fog node passes to look into its *PIT*: if a name matching is found, it means that the current request has been already sent in upstream, and the Fog node is, indeed, waiting for the result. As a consequence, the Fog node only proceeds to update the already present *PIT* entry with the number of the input port from which the Interest message has been received. Otherwise, the Fog node passes to look into its *FIDB* of Fig. 12a. If a matching is found, the Fog node generates a new *PIT* entry and broadcasts the Interest message over all its output ports for further forwarding. Otherwise, the Interest message is dropped and a negative acknowledgment is returned back to the requesting Aggregator.

A retrieved Data message follows the reverse path, in order to reach the Aggregator. Specifically, according to Fig. 12b, at the reception of a Data message, the Fog node forwards the message to the ports from which the corresponding Interest message was been previously received. At the same time, the Fog node stores the forwarded data into its *VOC* of Fig. 12b and, then, deletes the associated *PIT* entry. Received Data messages that do not match any current *PIT* entry are directly dropped and not forwarded at all.

Overall, from the outset, it follows that the main expected benefit arising from the adoption of the described *UDP*-supported *ICN* architecture for intra-tier communication may be twofold. First, the planned architecture allows the Aggregator to perform the *on-demand* and *selective* retrieving of the data cached by the corresponding Fog nodes, *without suffering* from the delays induced by the dial-up/tear-down phases of the utilized transport pipes. Second, it is expected that the in-network caching mechanism natively supported by the planned *ICN* architecture *decreases* both the intra-tier network traffic and the resulting retrieving delays.

IX. NUMERICAL TESTS AND PERFORMANCE COMPARISONS

The goal of this section is to numerically evaluate and compare the adaptive capability, fault resilience and energy-vs.-inference delays performance of the proposed Algorithm 1 for the optimal allocation of the computing-plus-networking resources of the designed *LiFo* technological platform of Fig. 3. In order to account for the multiple aspects of the carried out tests, we organized this section according to the following roadmap. After specifying in Section IX-A the simulated settings and the considered benchmark solutions,

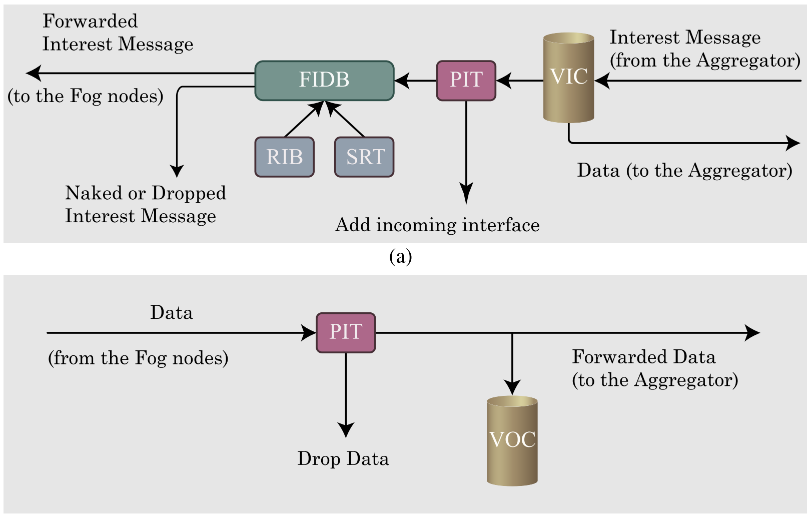

Forwarded Interest Message
(to the Fog nodes)
FIDB
PIT
VIC
Interest Message (from the Aggregator)
RIB
SRT
Data (to the Aggregator)
Naked or Dropped Interest Message
Add incoming interface
(a)
Data
(from the Fog nodes)
PIT
Forwarded Data (to the Aggregator)
VOC
Drop Data
(b)

FIGURE 12. **Planned forwarding mechanism and main supporting functional blocks for *ICN*-based intra-tier communication under the *LiFo* paradigm. (a) Upstream Aggregator-to-Fog data forwarding; (b) Downstream Fog-to-Aggregator data forwarding. PIT: = Pending Interest Table; VOC: = Virtual Output Cache; VIC: = Virtual Input Cache; FIDB: = Forwarding Interest Data Base; RIB: = Routing Information Base; SRT: = Strategy Rule Table.**

in Sections IX-B and IX-C we check the adaptive and resilient capabilities of the proposed Algorithm 1 to cope with (abrupt) time-variations of the operating IoT realm and (unpredicted) fault phenomena, respectively. Section IX-D numerically checks and compares against the considered benchmark solutions the performance sensitivity of Algorithm 1 of a number of parameters featuring the operating conditions of the underlying *LiFo* technological platform, like, the per-slot input workload, the per-exit inference delays and the per-layer compression factors of the supported CDNN with early exits. Afterwards, the goal of Section IX-E is to give insights about the effects on the energy-vs.-inference delay performance of Algorithm 1 induced by topology of the underlying *LiFo* technological platform of Fig. 3. Finally, Section IX-F numerically compares the energy performance of the proposed Fog-based *Lifo* solution to the corresponding ones of a traditional switch-based network for Cloud applications in which all the intermediate *LiFo* Fog nodes are replaced by switch nodes that perform only workload forwarding but not workload processing.

A. SIMULATED FRAMEWORK AND BENCHMARKS

The performed simulations have been carried out by exploiting the recently developed *DeepFogSim* toolbox [47] running over a hardware execution platform equipped with: (i) 40 GB of DDR 4 RAM; (ii) a 512 GB SSD with a 2 TB HHD; (iii) an Intel 12-core i9-7900X processor; and, (iv) a GPU ZOTAC GetForce GTX 1070. The adopted hardware platform is equipped with a software execution environment which relies on the release R2018a of MATLAB.

1) FEATURING THE SIMULATED CDNNs WITH EARLY EXITS

A joint examination of the: (i) expressions of $\mathcal{V}_I(j,m)$ and $\mathcal{V}_O(j,m)$ in Eqs. (9) and (10) for the volumes of the input and output workloads processed by the (j,m)-th Fog node; and, (ii) the related model in Eq. (20) for the dynamic power profile $\mathcal{P}_{MP}^{(DYN)}(j,m)$ of the (j,m)-th main processor, points out that the data about the CDNN with early exits, which are needed for the implementation of the proposed Algorithm 1 reduces to the set of the per-layer compression factors in Eq. (6) and the set of the Layer-to-Tier mappings featured by the actually performed optimal partition in Eq. (8). Hence, after selecting the CDNN with early exits to be run atop the *LiFo* technological platform of Fig. 3, it must be considered *known* both the corresponding $(L-1)$-dimensional (positive real-valued) vector:

$$\overrightarrow{cm} \triangleq [cm_1, cm_2, \ldots, cm_{L-1}]^T \in \mathbb{R}_+^{L-1}, \tag{61}$$

of the per-layer compression factors in (6), as well as the M-dimensional (positive integer-valued) vector:

$$\overrightarrow{L2T} \triangleq \left[\left|\mathcal{S}_1^*\right|, \left|\mathcal{S}_2^*\right|, \ldots, \left|\mathcal{S}_M^*\right|\right]^T \in \mathbb{N}_+^M, \tag{62}$$

of the numbers in (8) of the layers of the considered CDNN, which are mapped onto each tier of the underlying *LiFo* execution platform. As detailed in [12], we note that, in practice, the actual values of these vectors depend on a number of design factors, like, for example, the topology of the considered CDNN, the number and placement of the corresponding early exits, the setting of the decision thresholds $\{\eta_m\}$ in Fig. 10, the sets of examples used to train and validate the CDNN, just to name a few. An in-depth analysis of the overall topic of the optimized design of CDNNs with early exits is presented in [12], together with numerical examples of the vectors in (61) and (62) for some CDNNs of practical interest (see, in particular, Tables 4, 5, and 6 of [12] and the related texts). According to these mentioned results, we assume the following *default* numerical settings for the compression and mapping vectors in (61) and (62) [12]:

$$\overrightarrow{cm}_0 = [0.045,\ 0.1,\ 0.3,\ 0.5,\ 0.71,\ 0.7,\ 0.7,\ 0.9,\ 0.9]^T\ , \tag{63}$$

and,

$$\overrightarrow{L2T}_0 = [2,\ 3,\ 4]^T\ . \tag{64}$$

They refer to a suitably optimized version of the standard *AlexNet* [48] with 3 early exits, trained over the *SVHN* dataset[2] and, then, running over a 3-tier *LiFo* platform (see [12] for more details).

Finally, the default setting of the M-dimensional (with $M = 3$, our setting) vector $\vec{T}_0^{(EXIT)}$ of the per-tier maximum allowed inference times in Eq. (41b) is:

$$\vec{T}_0^{(EXIT)} = [0.30,\ 0.35,\ 0.40]^T \quad \text{(s)}, \tag{65}$$

with the corresponding default setting of the (minimum) slot-duration T_S equal to 1.0 (s).

2) CONSIDERED BENCHMARK SOLUTIONS

An examination of the related work of Section II points out that the overall topic of the optimized execution of CDNNs with early exits over multi-tier Fog platform under constraints on the per-exit inference delays is quite new, so that no specific algorithms for the related problem of the adaptive resource allocation appear to be still available in the open literature. Hence, on the basis of this consideration, the performance of the *LiFo* technological platform of Fig. 3 under the proposed resource allocation Algorithm 1 is compared with the corresponding ones of two benchmarks solutions, namely:

1) *Static Maximal Solution* (*SMS*): under the *SMS*, all available computing-plus-networking resources are held *fixed* at their corresponding maximal allowable values, so that, by definition, the resource allocation vector $\vec{X}^{(SMS)}$ returned by SMS coincides with the maximal one $\vec{X}^{(MAX)}$ in (43). In the following, the energy consumed by the *SMS* will be labeled with the upper-script $(\bullet)^{(SMS)}$;
2) *Only Switched Solution* (*OSS*): under the *OSS*, all Fog nodes at the intermediate *tier* #1 – *tier* #$(M-1)$ of Fig. 3 are replaced by conventional network switches, which perform data forwarding but do *not* execute any processing on the forwarded data. Hence, under the *OSS* setting, *only* the remote Cloud, at the top of Fig. 3, performs data processing and, then, may deliver inference data. In the following, the energy consumed by the *OSS* and the returned resource allocation vector will be labeled with the upper-script $(\bullet)^{(OSS)}$.

[2]Available at: `http://ufldl.stanford.edu//housenumbers`

3) CONSIDERED FOG TOPOLOGIES

According to the hierarchically organized structure of the *LiFo* platform of Fig. 3, we plan to test the corresponding performance by considering two broad families of *tree*-shaped Fog topologies.

The first family is composed of maximally-connected binary-tree topologies of depth M. By definition, a maximally-connected tree of depth M: $TR^{(MXC)}(2, M)$ is a directed binary tree which retains the following topological properties: (i) it is composed of $M \geq 2$ tiers, which are progressively numbered from the bottom to the top; (ii) each not-leaf node has 2 children and a single node (i.e., the root node) is present at the upper-most *tier* #M; and, (iv) each node at *tier* #m is connected to all nodes at *tier* #$(m+1)$, i.e., the tree is maximally connected. As an example, Fig. 13a illustrates the topology of $TR^{(MXC)}(2, 3)$. We anticipate that this family of maximally-connected tree-shaped topologies will be employed for testing the performance sensitivity of the *LiFo* paradigm on the depth M of the implemented technological platform of Fig. 3.

The second family of test topologies is composed of balanced r-ary trees of depth M. By definition, a balanced r-ary tree of depth M: $TR^{(BAL)}(r, M)$ is a directed tree featured by the following properties: (i) it is composed of $M \geq 2$ tiers, which are progressively numbered from the bottom to the top; (ii) each not-leaf node has $r \geq 2$ children and a single node (i.e., the root node) is present at the upper-most *tier* #M; (iii) each node at *tier* #m is connected to a single node at *tier* #$(m+1)$; and, (iv) all nodes at *tier* #m, with $2 \leq m \leq M$, share a same number of input edges, i.e., the topology is balanced. For illustrative purposes, Fig. 13b shows the topology of $TR^{(BAL)}(4, 3)$. We anticipate that this family of balanced tree-shaped topologies will be employed for testing the performance sensitivity of the *LiFo* paradigm on the *Communication-to-Computing Ratio* (*CCR*)[3] of the implemented *LiFo* platform.

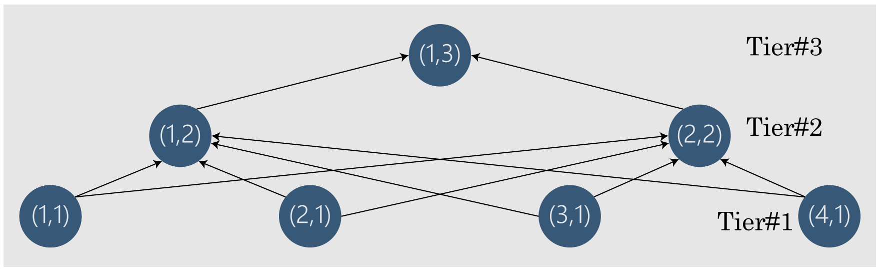
Finally, we note that, by default, the size of the input workload to be processed by the *LiFo* platform of Fig. 3 is set to $\mathcal{V}_0 = 6$ (Mb), and it is assumed to be even split over the Fog nodes at *tier* #1 of the implemented *LiFo* topology.

The last column of Table 9 in Appendix A recaps the default settings of the main parameters used in the carried out simulations.

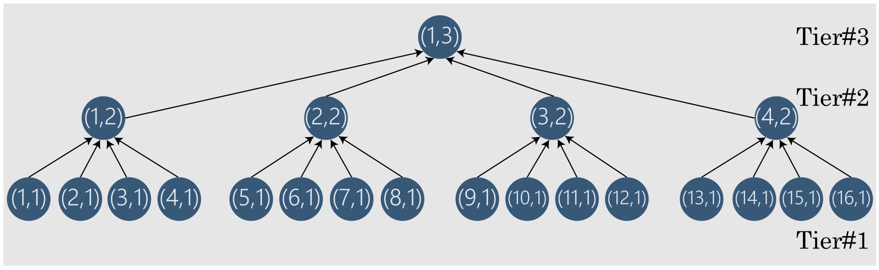
B. TESTING THE LIFO ADAPTIVE CAPABILITY

The twofold goal of this section is to numerically check the sensitivity of the tracking capability of the proposed *LiFo* resource-allocation iterations in (53) and (54) on both the hyper-parameters σ and A_0 in Eq. (57), as well as to evaluate their corresponding convergence time (in multiple of the iteration index k).

[3]In our setting, this is defined as the ratio of the number of the edges (*#edges*) of the considered tree to the corresponding number of nodes (*#nodes*), that is $CCR \triangleq \#edges/\#nodes$.



(a)



(b)

FIGURE 13. (a) Topology of $TR^{(MXC)}(2,3)$; (b) Topology of $TR^{(BAL)}(4,3)$.

Towards this end, we have simulated an event-driven dynamic operating scenario in which the size $\mathcal{V}_0$ of the input workload to be processed by the *LiFo* platform of Fig. 3 undergoes unpredicted up/down changes at the iteration indexes $k = 1$, 70, 140, 210 and 280, according to the following piece-wise constant pattern:

$$\begin{aligned}\mathcal{V}_0(k=1) &= 6\ (\text{Mb}) \rightarrow \mathcal{V}_0(k=70) = 9\ (\text{Mb})\\ &\rightarrow \mathcal{V}_0(k=140) = 6\ (\text{Mb}) \rightarrow\\ &\rightarrow \mathcal{V}_0(k=210) = 4.5\ (\text{Mb}) \rightarrow\\ &\rightarrow \mathcal{V}_0(k=280) = 6\ (\text{Mb}). \qquad (66)\end{aligned}$$

After each change, Algorithm 1 reacts in an *autonomous* way by *re-starting* the execution of the iterations in (53) and (54), so to properly *re-compute* the components of the resource vector $\overrightarrow{X}$ in (40).

The obtained dynamic behaviors of the total, computing and networking energies $\mathcal{E}_{TOT}^{(LiFo)}$, $\mathcal{E}_{COP}^{(LiFo)}$ and $\mathcal{E}_{NET}^{(LiFo)}$ returned by Algorithm 1 under the Fog topology $TR^{(MXC)}(2,3)$ of Fig. 13a are shown in Figs. 14 and 15. In this regard, we note that the plots of Fig. 14 (resp., Fig. 15) aim to check the performance sensitivity of Algorithm 1 on the hyper-parameter σ (resp., A_0) in (57), and then, they are evaluated at fixed $A_0 = 1.3 \times 10^{-5}$ (resp., at fixed $\sigma = 3.5$) and for four values of σ, namely $\sigma = 0$, 3.0, 3.5 and 3.8 (resp., for three values of A_0, namely $A_0 = 2.3 \times 10^{-5}$, 1.3×10^{-5} and 8.5×10^{-6}).

An examination of the plots of Figs. 14 and 15 leads to three main insights about the performance impact of the hyper-parameters A_0 and σ of Eq. (57). First, in Fig. 14, values of σ positive and around 3.5 allow Algorithm 1 to convergence within about 30 – 40 iterations after each environmental change, so to nearly *halve* the corresponding convergence time of the state-of-the-art approaches, which do not perform any adaptive tuning of the parameters $a_{MAX}^{(m)}$'s in Eq. (57) (see the plots of Fig. 14 at vanishing σ). Second, the convergence time exhibited by the plots of Fig. 14 for values of σ ranging over the simulated set: $\{3.0,\ 3.5,\ 3.8\}$ are nearly equal. Third, a comparative examination of the plots of Fig. 15 shows that, after setting $\sigma = 3.5$, the residual performance sensitivity of Algorithm 1 on the lower threshold A_0 in (57) is *not* very high, so that values of A_0 ranging over the interval: 8.0×10^{-6} – 2.5×10^{-5} are to be considered *almost equivalent*.

Overall, two main conclusions arise from the carried out comparative analysis. First, the performance sensitivity of the resource allocation Algorithm 1 on the σ hyper-parameter in (57) is *significantly larger* than the corresponding one on the lower threshold A_0 and vanishing values of σ *strongly* penalize, indeed, the tracking capability of Algorithm 1. Second, at least in the carried out tests, values of the sensitivity hyper-parameter σ (resp., A_0) in (57) *strictly* positive and around 3.5 (resp., around 10^{-5}) appear to guarantee the best tradeoff among the two contrasting requirements of fast response to changes of the operating conditions and stable behavior in the steady-state.

C. TESTING THE LIFO FAULT-TOLERANCE CAPABILITY

The twofold goal of this section is to: (i) check the reconfiguration capability of the *LiFo* resource-allocation Algorithm 1 in the presence of unpredicted failure events of the underlying technological platform of Fig. 3; and, (ii) evaluate the effects of non-vanishing values of the sensitivity hyper-parameter σ in (57) on the corresponding recovery time.

For this purpose, we have simulated a failure-affected scenario in which nodes: $FN(1,2)$ and $FN(2,2)$ at *tier* #2 of the topology of Fig. 13a sequentially fail and then resume at the

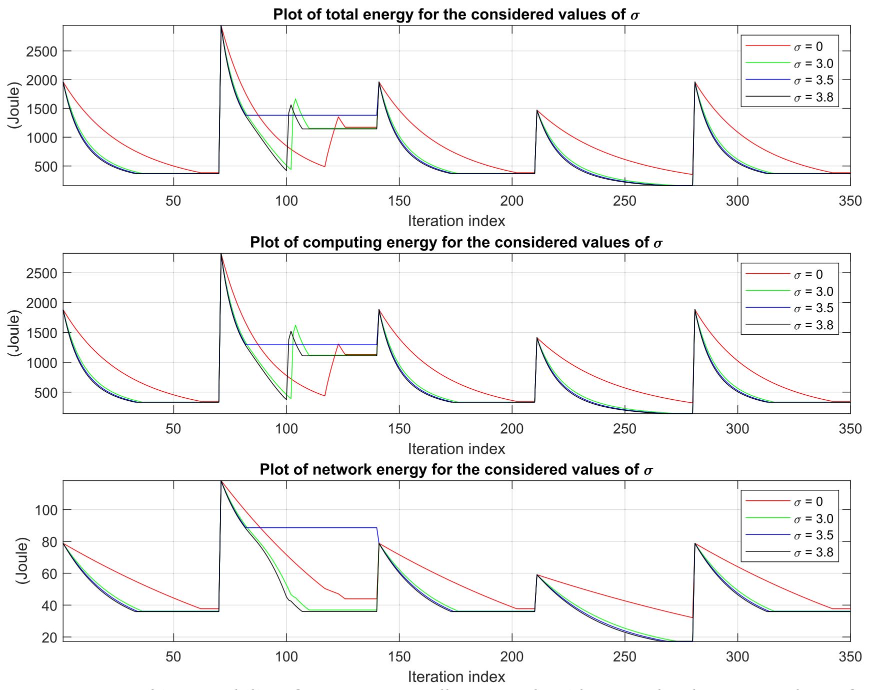


FIGURE 14. Tracking capability of *LiFo* resource-allocation Algorithm 1 under the Fog topology of Fig. 13a at fixed $A_0 = 1.3 \times 10^{-5}$ and for $\sigma = 0, 3.0, 3.5$ and 3.8. The simulated setting is the one of Section IX-B. (Top) Dynamic behaviors of $\mathcal{E}_{TOT}^{(LiFo)}$; (middle) Dynamic behaviors of $\mathcal{E}_{COP}^{(LiFo)}$; and, (bottom) Dynamic behaviors of the $\mathcal{E}_{NET}^{(LiFo)}$.

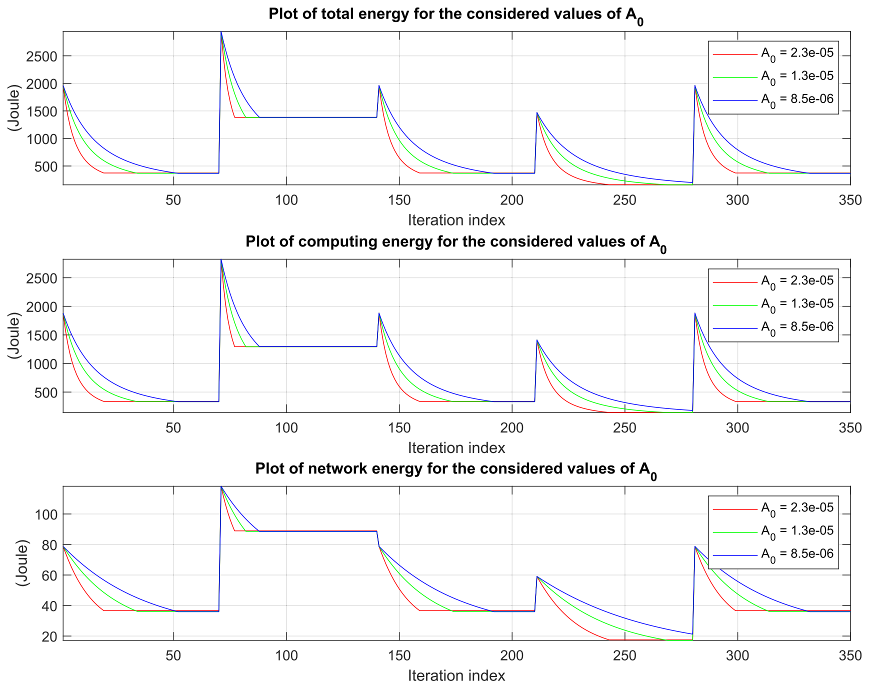


FIGURE 15. Tracking capability of *LiFo* resource-allocation Algorithm 1 under the Fog topology of Fig. 13a at fixed $\sigma = 3.5$ and for $A_0 = 2.3 \times 10^{-5}$, 1.3×10^{-5} and 8.5×10^{-6}. The simulated setting is the one of Section IX-B. (Top) Dynamic behaviors of $\mathcal{E}_{TOT}^{(LiFo)}$; (middle) Dynamic behaviors of $\mathcal{E}_{COP}^{(LiFo)}$; and, (bottom) Dynamic behaviors of the $\mathcal{E}_{NET}^{(LiFo)}$.

iteration indexes $k = 1, 80, 160, 240$ and 320 according to the following pattern: (i) at $k = 1$, all nodes and links of Fig. 13a are ON; (ii) at $k = 80$, the $(1, 2)$-th Fog node of Fig. 13a fails, i.e., both the operating frequencies of its main and auxiliary processors in Fig. 9 vanish; (iii) at $k = 160$, the $(1, 2)$-th Fog node resumes its normal operating conditions; (iv) at $k = 240$, the main and auxiliary processors of the (2, 2)-th Fog node in Fig. 13 fail; and, finally, (v) at $k = 320$, the (2, 2)-th Fog node turns to be operative. After each change in the operating conditions, Algorithm 1 *self*-reacts by *re-computing* the components of the resource vector $\overrightarrow{X}$ in (40), in order to suitably reconfigure the underlying *LiFo* technological platform of Fig. 3, so to attempt to still meet the constraints in (41b) on the inference times.

Fig. 16 illustrates the obtained dynamic behaviors of the total and networking energies $\mathcal{E}_{TOT}^{(LiFo)}$ and $\mathcal{E}_{NET}^{(LiFo)}$, as well as the Lagrange multipliers $\lambda_1^{(LiFo)}$ and $\lambda_3^{(LiFo)}$ associated to the first and last constraints on the inference times in (41b) for the two cases of $\sigma = 3.5$ (i.e., the proposed solution; red curves) and $\sigma = 0$ (i.e., the state-of-the-art solution; blue curves). In all simulated cases, the lower threshold A_0 in (57) is held fixed at 1.0×10^{-5}.

A comparative examination of these plots leads to two main conclusions about the impact on non-vanishing values of σ in (57) on the recovery time of the underlying simulated technological platform. First, an examination of Figs. 16a and 16b shows that, after each failure-resume event, the proposed solution in Eq. (57) at $\sigma = 3.5$ allows the convergences of the (red-marked) energy curves within 30–40 iterations, while 80 iterations are *not* sufficient for the convergence of the corresponding (blue-marked) plots obtained at vanishing σ (see the behaviors of the blue traces of Figs. 16a and 16b at the iteration indexes 80, 240 and 400). Second, this conclusion is supported by the corresponding behaviors of the Lagrange multipliers of Figs. 16c and 16d. In fact, a comparative examination of the traces presented in Figs. 16c and 16d confirms that the multipliers' behaviors at $\sigma = 3.5$ are more responsive to environmental changes than the corresponding ones at vanishing σ, thus giving further support about the actual effectiveness of the proposed dynamic relationship in Eq. (57).

Overall, the main lesson stemming from the above performance analysis is that the proposed adaptive tuning of the parameters $a_{MAX}^{(m)}$'s in Eq. (57) significantly improves, indeed, the resilient capability of the *LiFo* technological platform against failure events.

D. LIFO PERFORMANCE SENSITIVITY TO THE COMPUTING-NETWORKING PARAMETERS

The goal of this section is twofold. First, we check the sensitivity of the energy performance of the *LiFo* Algorithm 1 on: (i) the size $\mathcal{V}_0$ in (10) of the per-slot workload to be processed; (ii) the setting of the vector $\overrightarrow{T}^{(EXIT)} \triangleq \left[T_{EXIT}^{(1)}, \ldots, T_{EXIT}^{(M)}\right]^T$ in (41b) of the maximum allowable inference times; and, (iii) the setting of the vector $\overrightarrow{cm} \triangleq [cm_1, \ldots, cm_{L-1}]^T$ in (6) of the per-layer compression factors. Second, we compare the obtained *LiFo* energy performance to the corresponding ones of the (previously defined) *SMS* benchmark, in order to evaluate the actual energy saving arising from the adaptive mechanisms implemented by proposed Algorithm 1. All the results presented in Section IX-D refer to the test Fog topology $TR^{(MXC)}(2, 3)$ drawn in Fig. 13a.

1) PERFORMANCE SENSITIVITY TO THE PER-SLOT INPUT WORKLOAD

Table 2 provides the numerically evaluated steady-state energy performance of the proposed *LiFo* Algorithm 1 and the *SMS* benchmark under the default settings of Eqs. (63), (64) and (65).

A column-wise comparative examination of Table 2 unveils three main performance trends.

First, as it could be expected, all the obtained energy values increase for increasing values of the size $\mathcal{V}_0$ of the workload to be processed during each slot time.

Second, the percent values reported in the 6th column of Table 2 point out that the fraction $\left(\mathcal{E}_{NET}^{(LiFo)}/\mathcal{E}_{TOT}^{(LiFo)}\right)$ of the total *LiFo* energy $\mathcal{E}_{TOT}^{(LiFo)}$ consumed by the inter-tier networking infrastructure of Fig. 3 is *no negligible* at low/medium workload sizes, and it is higher than 10% for values of $\mathcal{V}_0$ up to 6 (Mb). This supports the basic design choice of the proposed Algorithm 1 to perform the adaptive tuning of the inter-tier transport rates, in order to *dynamically optimize* the usage of the available networking resources.

Third, the percent values reported in the last column of Table 2 show that the per-slot total energy $\mathcal{E}_{TOT}^{(LiFo)}$ consumed by the *LiFo* platform under the (adaptive) Algorithm 1 remains a fraction of about 69% of the corresponding energy $\mathcal{E}_{TOT}^{(SMS)}$ wasted by the (not adaptive) benchmark *SMS*, even at workload sizes exceeding 10 (Mb). This means, in turn, that the *adaptive joint* optimization of the networking-plus-computing resources pursued by proposed Algorithm 1 translates into energy savings *over* 40%, *even* under operating conditions featured by high input workload and stringent requirements on the allowed inference times (see Eq. (65)).

2) PERFORMANCE SENSITIVITY TO THE PER-TIER DELAY CONSTRAINTS

The goal of Table 3 is to allow a comparative analysis of the sensitivity of the energy performance of proposed *LiFo* Algorithm 1 and the benchmark *SMS* under more or less stringent constraints on the allowed per-tier inference delays. For this purpose, the scalar components of the vector $\overrightarrow{T}_0^{(EXIT)}$ in (65) of the default values of the maximum allowable inference time are scaled up/down as shown in the first column of Table 3. The presented numerical results refer to the default settings of Eqs. (64) and (65) at $\mathcal{V}_0 = 6$ (Mb).

An examination of Table 3 leads to four main conclusions about the impact of the actual setting of the allowed inference-time vector $\overrightarrow{T}^{(EXIT)}$ on the energy performance of the simulated resource-allocation algorithms.

First, since, by design, *SMS* fixes at their maxima all the available resources regardless of the actually enforced constraints, it is expected that the corresponding consumed energy $\mathcal{E}_{TOT}^{(SMS)}$ does not depend, indeed, on the required maximum inference time. This expectation is validated by the

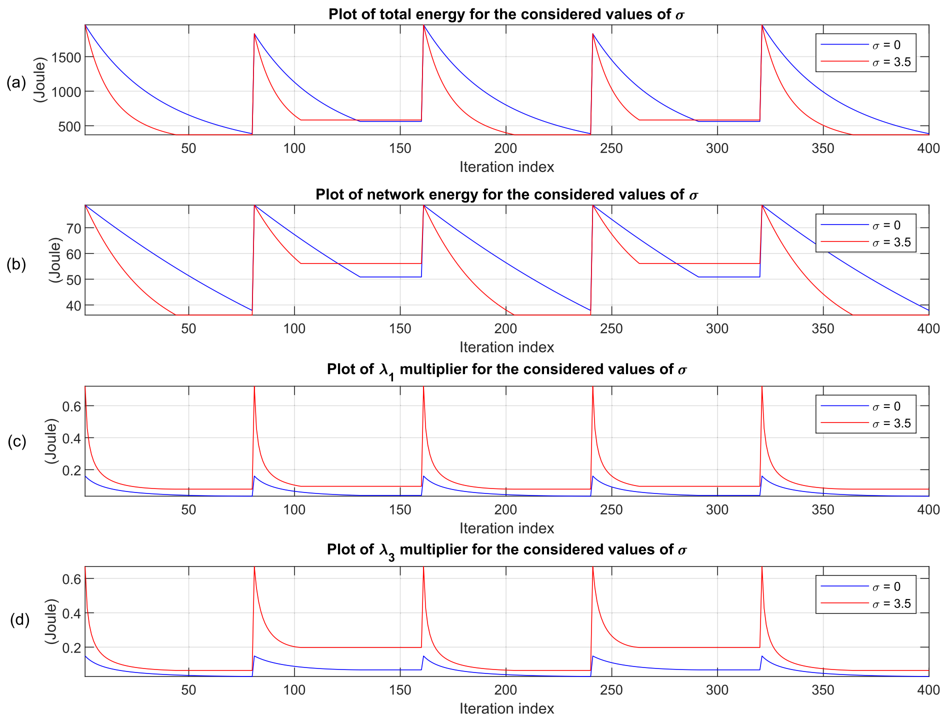


FIGURE 16. Reconfiguration capability of *LiFo* Algorithm 1 after unpredicted failure events under the Fog topology of Fig. 13a at $A_0 = 1.0 \times 10^{-5}$. Red (resp., blue) plots refer to the case of $\sigma = 3.5$ (resp., $\sigma = 0$). The simulated setting is the one of Section IX-C. (a) Dynamic behaviors of $\mathcal{E}_{TOT}^{(LiFo)}$; (b) Dynamic behaviors of $\mathcal{E}_{NET}^{(LiFo)}$; (c) Dynamic behaviors of the $\lambda_1^{(LiFo)}$ multiplier; and, (d) Dynamic behaviors of the $\lambda_3^{(LiFo)}$ multiplier.

TABLE 2. Energy sensitivity on the size of the per-slot input workload under the simulated setting of Section IX-D.

$\mathcal{V}_0$ (Mb)	$\mathcal{E}_{TOT}^{(SMS)}$ (J)	$\mathcal{E}_{TOT}^{(LiFo)}$ (J)	$\mathcal{E}_{COP}^{(LiFo)}$ (J)	$\mathcal{E}_{NET}^{(LiFo)}$ (J)	$\frac{\mathcal{E}_{NET}^{(LiFo)}}{\mathcal{E}_{TOT}^{(LiFo)}}$ (%)	$\frac{\mathcal{E}_{TOT}^{(LiFo)}}{\mathcal{E}_{TOT}^{(SMS)}}$ (%)
3.0	980.3	56.2	48.3	7.6	14 %	5.7 %
4.5	1468	161.7	142.2	19.5	12 %	11 %
6.0	1954	369.8	333.4	36.4	9.7 %	18.9 %
7.5	2445	744.7	687	57.7	7.7 %	30.5 %
9.0	2933	1363	1281	82.5	6 %	46.5 %
10.5	3421	2366	2255	111.3	4.8 %	69.1 %

entries in the 2nd column of Table 3, which share, indeed, a same value. However, since the proposed *LiFo* Algorithm 1 adapts, by design, the returned resource allocation vector in (46) to the constraints in (41b) on the inference times, all the corresponding *LiFo* energies in the 3rd, 4th and 5th columns of Table 3 scale down for increasing values of the maximum allowable inference time.

Second, the entries of the 6th column of Table 3 confirm that the fractions of the total *LiFo* energy $\mathcal{E}_{TOT}^{(LiFo)}$ needed for the support of the inter-tier network flows of Fig. 3 are *no negligible* when the imposed constraints on the inference time are not too stringent, i.e., under operating conditions which do not stress too much the corresponding computing resources of the simulated *LiFo* platform.

Third, the above conclusion is further corroborated by the entries of the last column of Table 3. They show that the adaptive mechanisms equipping the proposed Algorithm 1 allow the *LiFo* total energy $\mathcal{E}_{TOT}^{(LiFo)}$ to remain less than 61% than the corresponding energy $\mathcal{E}_{TOT}^{(SMS)}$ of the benchmark *SMS*, even when the imposed maximum inference time is reduced by 40% below the default setting of Eq. (65) (see the 6th row of Table 3).

Finally, the numerical results shown in the last row of Table 3 confirm the expectation that, when the constraints on

TABLE 3. Energy sensitivity on the vector $\vec{T}^{(EXIT)}$ (s) of the maximum allowed inference times under the simulated setting of Section IX-D.

$\vec{T}^{(EXIT)}$ (s)	$\mathcal{E}_{TOT}^{(SMS)}$ (J)	$\mathcal{E}_{TOT}^{(LiFo)}$ (J)	$\mathcal{E}_{COP}^{(LiFo)}$ (J)	$\mathcal{E}_{NET}^{(LiFo)}$ (J)	$\frac{\mathcal{E}_{NET}^{(LiFo)}}{\mathcal{E}_{TOT}^{(LiFo)}}$ (%)	$\frac{\mathcal{E}_{TOT}^{(LiFo)}}{\mathcal{E}_{TOT}^{(SMS)}}$ (%)
$2.5 \times \vec{T}_0^{(EXIT)}$	1954	78.8	70.3	8.5	12.1 %	4.0 %
$1.2 \times \vec{T}_0^{(EXIT)}$	1954	261.1	232.1	29	11.1 %	13.4 %
$1.0 \times \vec{T}_0^{(EXIT)}$	1954	369.8	333.4	36.4	9.7 %	18.9 %
$0.8 \times \vec{T}_0^{(EXIT)}$	1954	594.9	548.3	46.6	7.8 %	30.4 %
$0.6 \times \vec{T}_0^{(EXIT)}$	1954	1188.6	1126.7	61.9	5.2 %	60.8 %
$0.5 \times \vec{T}_0^{(EXIT)}$	1954	1911.7	1838.9	72.9	3.8 %	97.8 %

the allowed inference time become so stringent to push the operating point of the *LiFo* platform towards the boundary of the underlying feasibility region of *Proposition 2*, then the resulting *LiFo* energy consumptions approach the corresponding ones of the benchmark *SMS*, so to annihilate the resulting energy savings (see the last entry of Table 3).

3) PERFORMANCE SENSITIVITY TO THE PER-LAYER COMPRESSION FACTORS

Table 4 gives the energy consumption of proposed *LiFo* Algorithm 1 and the benchmark *SMS* when the scalar components of the default per-layer compression vector $\vec{cm}_0$ in (63) are scaled up/down by factors ranging from 2.0 to 0.1 (see the first column of Table 4). All the results presented in Table 4 refer to the default settings of Eqs. (64) and (65) at $\mathcal{V}_0 = 6$ (Mb).

A comparative examination of the columns of Table 4 allows us to arrive at three main conclusions about the impact of the early exits of the supported CDNN of Fig. 1a on the energy performance of the corresponding Fog execution platform of Fig. 1b.

First, the relationship in Eq. (9) points out that, at fixed size $\mathcal{V}_0$ of the input workload, the corresponding size: $\mathcal{V}_I(j,m)$, $1 \leq j \leq m_m$, $2 \leq m \leq M$, of the workload to be processed by the (j,m)-th Fog node *decreases* for *decreasing values* of the scalar components of the corresponding compression vector $\vec{cm}$. This is the reason why *all* the energies presented in Table 4 scales down passing from: $\vec{cm} = 2.0 \times \vec{cm}_0$ to: $\vec{cm} = 0.1 \times \vec{cm}_0$. This is also true for the energy $\mathcal{E}_{TOT}^{(SMS)}$ consumed by the (not adaptive) *SMS*, because also the execution time in (21) (and then the corresponding dynamic energy in (22)) scales down for decreasing values of workload to be processed by Fog nodes.

Second, an examination of the 6th column of Table 4 unveils that the behavior of the *LiFo* networking-to-computing energy ratio $\left(\mathcal{E}_{NET}^{(LiFo)}/\mathcal{E}_{TOT}^{(LiFo)}\right)$ is no longer monotonic for decreasing values of the components of the compression vector $\vec{cm}$, but it is, indeed, $\cap$-shaped. This means that the ratio $\left(\mathcal{E}_{NET}^{(LiFo)}/\mathcal{E}_{TOT}^{(LiFo)}\right)$ assumes larger values around the default setting of $\vec{cm}$ in (63), while it tends to decrease both for larger and smaller settings. A comparison of the 3rd and 4th columns of Table 4 supports the conclusion that this no monotonic behavior is induced by the corresponding behavior of the *LiFo* computing energy $\mathcal{E}_{COP}^{(LiFo)}$, which tends to dominate the total *LiFo* energy $\mathcal{E}_{TOT}^{(LiFo)}$ in both limit cases of high and low settings of the compression vector $\vec{cm}$.

Third, the last column of Table 4 points out that *LiFo* Algorithm 1 allows to attain *noticeable* energy savings with respect to the benchmark *SMS*, which range from: 89.4% at $\vec{cm} = 0.1 \times \vec{cm}_0$ (i.e., small setting of the compression vector $\vec{cm}$) to: 32.9% at $\vec{cm} = 2.0 \times \vec{cm}_0$ (i.e., large setting of the compression vector $\vec{cm}$). This leads, in turn, to the conclusion that proposed *LiFo* Algorithm 1 is capable of effectively capitalizing the reduction in the workload to be processed by the Fog platform of Fig. 1b which is induced by the early-exit-of-inference performed by the supported CDNN of Fig. 1a.

E. LIFO PERFORMANCE SENSITIVITY ON THE TOPOLOGY OF THE FOG EXECUTION PLATFORM

The goal of this section is to numerically evaluate and assess the sensitivity of the energy performance of the *LiFo* Algorithm 1 on the structural properties of the topology of the implemented Fog platform of Fig. 1b. For this purpose, the maximally connected and balanced tree-shaped topologies presented in the first column of Table 5 have been considered.

The 2nd, 3rd and 4th columns of Table 5 report the main formal properties (i.e., number of nodes, number of edges and resulting *CCR*) of the considered topologies, while the last two columns specify the Layer-to-Tier mapping vector and maximum inference-time vector under which each topology is simulated.

In this regard, three remarks are in order. First, the Layer-to-Tier mapping vector used for the simulation of each topology has been optimized by applying the minimum-energy mapping criterion developed on [12]. Second, since the considered topologies of Table 5 are composed of a variable number M of tiers ranging from $M = 3$ to $M = 5$, we cannot

TABLE 4. Energy sensitivity on the compression vector $\overrightarrow{cm}$ under the simulated setting of Section IX-D.

$\overrightarrow{cm}$	$\mathcal{E}_{TOT}^{(SMS)}$ (J)	$\mathcal{E}_{TOT}^{(LiFo)}$ (J)	$\mathcal{E}_{COP}^{(LiFo)}$ (J)	$\mathcal{E}_{NET}^{(LiFo)}$ (J)	$\frac{\mathcal{E}_{NET}^{(LiFo)}}{\mathcal{E}_{TOT}^{(LiFo)}}$ (%)	$\frac{\mathcal{E}_{TOT}^{(LiFo)}}{\mathcal{E}_{TOT}^{(SMS)}}$ (%)
$2.0 \times \overrightarrow{cm}_0$	7000	4698	4599	98.4	2.1 %	67.1 %
$1.5 \times \overrightarrow{cm}_0$	3680	948	898.5	49.5	5.2 %	25.8 %
$1.0 \times \overrightarrow{cm}_0$	1954	369.8	333.4	36.4	9.7 %	18.9 %
$0.5 \times \overrightarrow{cm}_0$	1647	251.6	237.7	13.9	5.5 %	15.3 %
$0.25 \times \overrightarrow{cm}_0$	1546	171.6	167.5	4.1	2.4 %	11.1 %
$0.1 \times \overrightarrow{cm}_0$	1537	163.3	160.1	3.6	2.2 %	10.6 %

TABLE 5. Main topologic properties of the test topologies of Section IX-E and corresponding simulated Layer-to-Tier vector $\overrightarrow{L2T}$ and maximum inference-time vector $\overrightarrow{T}^{(EXIT)}$.

Fog Topology	Number of nodes	Number of edges	CCR	Simulated $\overrightarrow{L2T}$	Simulated $\overrightarrow{T}^{(EXIT)}$ (s)
$TR^{(MXC)}(2,3)$	7	10	1.43	[2, 3, 4]	[1, 1, 0.4]
$TR^{(MXC)}(2,4)$	15	42	2.80	[3, 2, 2, 2]	[1, 1, 1, 0.4]
$TR^{(MXC)}(2,5)$	31	170	5.48	[4, 2, 1, 1, 1]	[1, 1, 1, 1, 0.4]
$TR^{(BAL)}(2,5)$	31	30	0.97	[2, 1, 2, 2, 2]	[1, 1, 1, 1, 0.4]
$TR^{(BAL)}(3,4)$	40	29	0.97	[2, 2, 2, 3]	[1, 1, 1, 0.4]
$TR^{(BAL)}(4,3)$	21	20	0.95	[3, 3, 3]	[1, 1, 0.4]

utilize a single setting for the vector $\overrightarrow{T}^{(EXIT)}$ of the maximum allowable inference times. Hence, in order to carry out fair energy comparisons, we pursued the unifying criterion to set the maximum inference times of all intermediate tiers of the simulated topologies to the default slot-duration $T_S = 1$ (s), while fixing at the default value of 0.4 (s) the inference time of the corresponding upper-most M-th tier (see the settings of the $\overrightarrow{T}^{(EXIT)}$) vectors in the last column of Table 5). Third, all the numerical results of this section refer to the default setting in Eq. (63) at $\mathcal{V}_0 = 6$ (Mb).

1) ENERGY PERFORMANCE SENSITIVITY ON THE DEPTH OF THE FOG TOPOLOGY

In order to check the sensitivity of the energy performance of the *LiFo* Algorithm 1 on the depth M (i.e., number of tiers) of the topology used for the implementation of the Fog platform of Fig. 1b, we have considered the first three topologies presented in Table 5. They refer to maximally-connected binary trees of depth M ranging from 3 to 5. The corresponding numerically evaluated energy consumptions are given in Table 6.

A column-wise comparative examination of the entries of Table 6 leads to three main remarks.

First, all the energy values presented in the 2nd, 3rd, 4th and 5th columns of in Table 6 scale down for increasing values of the depth M of the considered binary-tree topologies. This is due to the fact that, under the simulated topologies, both the numbers of computing nodes and transport links increase for increasing values of M (see the 2nd and 3rd columns of Table 5). This allows a *finer* distribution of the input workload over the computing nodes and network links of the underlying execution platform, which, due to the convexity of the adopted networking and computing energy models (see *Proposition 1*), leads, in turn, to a reduction of all involved energies.

Second, the percent values in the 6th column of Table 6 point out that the fraction $\left(\mathcal{E}_{NET}^{(LiFo)}/\mathcal{E}_{TOT}^{(LiFo)}\right)$ of the total *LiFo* energy $\mathcal{E}_{TOT}^{(LiFo)}$ consumed by the inter-tier networking infrastructure of Fig. 3 increases from 9.3% to 16.8% by passing from the 3-tier topology $TR^{(MXC)}(2,3)$ to the 5-tier topology $TR^{(MXC)}(2,5)$. This is due to the formal fact that the CCR of $TR^{(MXC)}(2,5)$ is about 3.8 times larger than the corresponding one of $TR^{(MXC)}(2,3)$, so that the $TR^{(MXC)}(2,5)$ topology is more communication expensive than the $TR^{(MXC)}(2,3)$ one.

Third, the entries in the last column of Table 6 show that the energy savings offered by the (adaptive) *LiFo* Algorithm 1

TABLE 6. Energy sensitivity on the depth of the considered binary-tree Fog topologies under the simulated setting of Section IX-E.

Fog Topology	$\mathcal{E}_{TOT}^{(SMS)}$ (J)	$\mathcal{E}_{TOT}^{(LiFo)}$ (J)	$\mathcal{E}_{COP}^{(LiFo)}$ (J)	$\mathcal{E}_{NET}^{(LiFo)}$ (J)	$\frac{\mathcal{E}_{NET}^{(LiFo)}}{\mathcal{E}_{TOT}^{(LiFo)}}$ (%)	$\frac{\mathcal{E}_{TOT}^{(LiFo)}}{\mathcal{E}_{TOT}^{(SMS)}}$ (%)
$TR^{(MXC)}(2,3)$	1954	239.1	216.8	22.3	9.3 %	12.2 %
$TR^{(MXC)}(2,4)$	1140	144.3	124.9	19.4	13.4 %	12.6 %
$TR^{(MXC)}(2,5)$	279.5	38.7	32.2	6.5	16.8 %	13.8 %

TABLE 7. Energy performances under 2-ary (i.e., tall), 3-ary (i.e., normal) and 4-ary (i.e., fat) tree-shaped Fog topologies and the simulated setting of Section IX-E.

Fog Topology	$\mathcal{E}_{TOT}^{(SMS)}$ (J)	$\mathcal{E}_{TOT}^{(LiFo)}$ (J)	$\mathcal{E}_{COP}^{(LiFo)}$ (J)	$\mathcal{E}_{NET}^{(LiFo)}$ (J)	$\frac{\mathcal{E}_{NET}^{(LiFo)}}{\mathcal{E}_{TOT}^{(LiFo)}}$ (%)	$\frac{\mathcal{E}_{COP}^{(LiFo)}}{\mathcal{E}_{TOT}^{(LiFo)}}$ (%)	$\frac{\mathcal{E}_{TOT}^{(LiFo)}}{\mathcal{E}_{TOT}^{(SMS)}}$ (%)
$TR^{(BAL)}(2,5)$	133.5	43.8	26.3	17.9	40.9 %	59.1 %	32.8 %
$TR^{(BAL)}(3,4)$	72.1	13.4	10.6	2.8	20.9 %	79.1 %	18.6 %
$TR^{(BAL)}(4,3)$	422.3	38.7	35.2	3.9	10.1 %	89.9 %	9.2 %

TABLE 8. *OSS-vs.-LiFo* energy performance comparisons under binary Fog topologies of depth 3, 4 and 5, and the simulated setting of Section IX-F.

Fog Topology	$\mathcal{E}_{TOT}^{(OSS)}$ (J)	$\mathcal{E}_{NET}^{(OSS)}$ (J)	$\frac{\mathcal{E}_{NET}^{(OSS)}}{\mathcal{E}_{TOT}^{(OSS)}}$ (%)	$\mathcal{E}_{TOT}^{(LiFo)}$ (J)	$\frac{\mathcal{E}_{TOT}^{(LiFo)}}{\mathcal{E}_{TOT}^{(OSS)}}$ (%)
$TR^{(MXC)}(2,3)$	270	108	40.0 %	239.1	88.5 %
$TR^{(MXC)}(2,4)$	329.8	167.8	50.8 %	144.3	43.7 %
$TR^{(MXC)}(2,5)$	347.2	185.2	53.3 %	38.7	11.1 %

over the (not adaptive) *SMS* are quite substantial and, under the simulated topologies, range over the interval: 86.2% – 87.8%.

2) COMPUTING-VS.-NETWORKING ENERGY CONSUMPTIONS

The goal of the set of tests presented in this sub-section is to numerically check and assess the *LiFo* energy performance under a number of (r, M)-tree topologies which exhibit *different* width r and/or depth M, but share a (nearly) *same* value of the corresponding *CCR* values. This is motivated by the following two considerations.

First, the main disadvantage of deeper tree-shaped topologies is that they exhibit longer leaf-to-root paths. Hence, under any fixed upper bound on the overall inference time, it is expected that deeper tree-shaped topologies would require higher per-connection inter-tier transmit rates, that, in turn, would *increase* the dynamic parts of the per-connection *network energy* in Eqs. (32) and (33). Hence, deeper tree topologies are expected to consume *more* network energy.

Second, the main advantage of wider tree-shaped topologies is that they exhibit bigger numbers of tiers. Hence, under a fixed number of layers of the CDNN to be supported, wider tree-shaped topologies allow finer (i.e., more equalized) Layer-to-Tier mappings, that, in turn, would reduce the average per-node processing workload and then would *decrease* the dynamic parts of the per-node *computing energy* in Eqs. (22) and (25). Hence, wider tree topologies are expected to consume *less* computing energy.

In order to numerically support these expectations and then acquire insight into the resulting computing-vs.-networking energy tradeoff, we have considered the last three topologies of Table 5. They refer to balanced r-ary tree topologies with $r = 2, 3, 4$, and depth $M = 5, 4, 3$, respectively. In this regard, we observe that the *CCR* values of all the considered balanced tree topologies are nearly the same. However, the nodes of the binary (resp., 4-ary) $TR^{(BAL)}(2, 5)$ (resp., $TR^{(BAL)}(4, 3)$) tree topology are mainly placed along the vertical (resp., horizontal) dimension, so that $TR^{(BAL)}(2, 5)$ (resp., $TR^{(BAL)}(4, 3)$) is an example of "tall" (resp., "fat") tree topology. An intermediate "normal" shape is retained by the 3-ary tree topology $TR^{(BAL)}(3, 4)$ of Table 5.

Table 7 shows the obtained energy performance under the default setting in Eq. (63) of the per-layer compression vector at $\mathcal{V}_0 = 6$ (Mb).

A comparative examination of the columns of Table 7 unveils the multi-facet nature of the computing-vs.-networking energy tradeoff and leads, indeed, to three main conclusions.

First, according to the previously reported expectations, the *LiFo* networking-to-total energy ratio $\left(\mathcal{E}_{NET}^{(LiFo)}/\mathcal{E}_{TOT}^{(LiFo)}\right)$ scales down by passing from the "tall" $TR^{(BAL)}(2, 5)$ topology to the "fat" $TR^{(BAL)}(4, 3)$ one, while an opposite trend is exhibited by the corresponding computing-to-total energy ratio $\left(\mathcal{E}_{COP}^{(LiFo)}/\mathcal{E}_{TOT}^{(LiFo)}\right)$ (compare the 6th and 7th columns of Table 7).

Second, all the energies reported in the 2nd, 3rd, 4th and 5th columns of Table 7 exhibit a no-monotonic $\cup$-shaped behaviors when we move from the "tall" $TR^{(BAL)}(2, 5)$ topology to the "fat" $TR^{(BAL)}(4, 3)$ one. This supports the conclusion that the *best tradeoff* among the computing-vs.-networking energy consumption is, indeed, attained under the "normal" $TR^{(BAL)}(3, 4)$ topology.

Finally, the decreasing trend of the numerical results reported in the last column of Table 7 points out that the energy saving attained by the (adaptive) *LiFo* Algorithm 1 over the (not adaptive) benchmark *SMS* (substantially) increases by passing from "tall" topologies to "fat" ones.

F. HOW MUCH COMMUNICATING-WHILE-COMPUTING IS GOOD? LIFO-VS.-OSS PERFORMANCE COMPARISONS

A key feature of the proposed *LiFo* technological platform is that the all Fog nodes at the intermediate *tier* $\#m$, $1 \leq m \leq (M-1)$, are equipped with *both* computing and networking capabilities (see Fig. 9). As a consequence, they may: (i) process the received workload; (ii) early exit part of it; and, (iii) forward the remaining part toward higher-tier Fog nodes. This enables early-exit-of-inference, which, *reduces*, in turn, the size of the input workload to be processed by the full M-tier execution stack of Fig. 1b. However, since nothing is for free, the computing energy in Eqs. (22) and (25) required by the main and auxiliary processors equipping each Fog node of Fig. 9 add to the corresponding receive and transmit network energy in Eqs. (32) and (33).

Therefore, the following crucial question naturally arises:

- how much communicating-while-computing is really energy saving?

To address this question, we have implemented the (previously introduced) benchmark *OSS*. It aims at emulating a traditional multi-tier switched network infrastructure for the support of remote Cloud applications, in which all the Fog nodes at the intermediate tiers *tier* $\#m$, $1 \leq m \leq (M-1)$, of the execution platform of Fig. 1b are replaced by network switches that perform only workload forwarding. Hence, under the benchmark *OSS*, we have, by design, that: (i) *only* the remote Cloud at the upper-most *tier* $\#M$ is equipped with computing capability; (ii) intermediate early exits are *no longer* present; and, then, (iii) *all* the input workload flows over the *full* stack of Fig. 1b from *tier* $\#1$ to *tier* $\#M$.

In order to carry out fair performance comparisons, the benchmark *OSS* is simulated under the following setting: (i) the computing capacity of the *OSS* Cloud node equates to the aggregate computing capacities of all *LiFo* Fog-plus-Cloud nodes; (ii) the aggregate networking capacity of the *OSS* equates to the corresponding ones of the competing *LiFo* platform; and, (iii) the maximum inference time at the *OSS* Cloud node equates to the corresponding value 0.4 (s) allowed the corresponding *LiFo* Cloud node. Finally, we point out that, in order to carry out fair performance comparisons, the proposed Algorithm 1 has been *still* applied for attaining the *optimized* adaptive tuning of *all* computing-plus-networking resources available under the benchmark *OSS*.

Table 8 shows the numerically evaluated *LiFo*-vs.-*OSS* energy performance under the maximally-connected binary-tree topologies of Table 5. The presented results refer to the default setting in (63) of the per-layer compression vector at $\mathcal{V}_0 = 6$ (Mb).

An examination of Table 8 allows us to unveil three main trends regarding the *OSS*-vs.-*LiFo* energy performance.

First, the entries of the 2nd column of Table 8 point out that the networking-plus-computing total energy $\mathcal{E}_{TOT}^{(OSS)}$ wasted by the benchmark *OSS* increases by increasing the depth of the simulated Fog topologies, that is, by passing from $TR^{(MXC)}(2, 3)$ to $TR^{(MXC)}(2, 5)$. We have numerically ascertained that this trend is due to the increment of the corresponding number of switches of the simulated *OSS* networks, which, in turn, induces companion increments in the corresponding *OSS* network energy (see the 3rd and 4th columns of Table 8).

Second, an *opposite* trend is exhibited, indeed, by the total energy $\mathcal{E}_{TOT}^{(LiFo)}$ wasted by the Fog-based *LiFo* platform, which *decreases* passing from $TR^{(MXC)}(2, 3)$ to $TR^{(MXC)}(2, 5)$. This is due to the fact that, since Fog nodes are equipped with computing capabilities under the *LiFo* paradigm, the number of computing nodes increments from 7 to 31 by passing from $TR^{(MXC)}(2, 3)$ to $TR^{(MXC)}(2, 5)$. This increment allows a *finer* per-tier distribution of the overall workload over the available computing nodes, which, in turn, leads to a *reduction* of the resulting total energy $\mathcal{E}_{TOT}^{(LiFo)}$.

Finally, the percent values of the ratio $\left(\mathcal{E}_{TOT}^{(LiFo)}/\mathcal{E}_{TOT}^{(OSS)}\right)$ given in the last column of Table 8 point out that the energy savings offered by the Fog-based *LiFo* paradigm over the traditional Switch-based *OSS* one are, indeed, *noticeable*, and may reach 88%–89% under deep tree topologies (see the last entry of 6th column in Table 8).

Overall, the final lesson arising from the carried out analysis is that communicating-while-computing is, indeed, a (very) good strategy for attaining energy savings.

X. CONCLUSION AND HINTS FOR FUTURE RESEARCH

The incoming convergence of the Deep Learning and Fog Computing paradigms is enabling the real-time and energy-efficient inference of aggregate volumes of data generated by spatially-scattered and resource-limited IoT devices. Motivated by this consideration, in this article, we focused on the optimized design and validation of *LiFo*, a virtualized Fog-based multi-tier technological platform for the energy-efficient and delay-constrained execution of the inference phase of a novel family of Deep Neural Networks, i.e., the so-called Conditional Deep Neural Networks with early exits. Specifically, after designing the main building blocks and associated virtualized functionalities of the *LiFo* paradigm, we develop a framework for the joint allocation and

TABLE 9. List of the main parameters, their meaning/role, measuring units and simulated default values.

Parameter	Meaning/Role	Measuring Units	Default Settings
L	Number of the CDNN layers	Dimensionless	$L = 9$
M	Number of tiers of the Fog topology	Dimensionless	Topology depending (see Table 5)
N_{EE}	Number of the CDNN early exits	Dimensionless	$M - 1$
$m = 1, \dots, M$	Tier index	Dimensionless	Topology depending (see Table 5)
$m_m, m = 1, \dots, M$	Number of Fog nodes at *tier* $\#m$	Dimensionless	Topology depending (see Table 5)
$j = 1, \dots, m_m$	Fog index at *tier* $\#m$	Dimensionless	Topology depending (see Table 5)
$FN(j,m)$	(j,m)-th Fog node	Dimensionless	Topology depending (see Table 5)
f_{jm} (resp., $\tilde{f}_{jm}$)	Processing frequency of the main (resp., auxiliary) processor at $FN(j,m)$	bit/s	Optimization variable
R_m	Per-connection transport rate at *tier* $\#m$	bit/s	Optimization variable
$f_{jm}^{(MAX)}$ (resp., $\tilde{f}_{jm}^{(MAX)}$)	Maximum processing frequency of the main (resp., auxiliary) processor at $FN(j,m)$	bit/s	9×10^6 (resp., 8×10^6)
$R_m^{(MAX)}$	Maximum per-connection transport rate at *tier* $\#m$	bit/s	9×10^6
$\overrightarrow{cm}$	Vector of the per-layer compression factors	Dimensionless	See Eq. (63)
$\vec{T}^{(EXIT)}$	Vector of the per-tier maximum inference time	s	Topology depending (see Table 5)
$\overrightarrow{L2T}$	Vector of the Layer-to-Tier mappings	Dimensionless	Topology depending (see Table 5)
$\mathcal{V}_0$	Per-slot input workload	bit	6×10^6
T_S	Time-slot duration	s	1.0
$\mathcal{E}_{MP}(j,m)$ (resp., $\mathcal{E}_{AP}(j,m)$)	Energy of the main (resp., auxiliary) processor at $FN(j,m)$	Joule	To be optimized
$\mathcal{E}_{NET}^{(Tx)}(j,m)$ (resp., $\mathcal{E}_{NET}^{(Rx)}(j,m)$)	Transmit (resp., receive) network Energy at $FN(j,m)$	Joule	To be optimized
$T_{EXE}^{(1,m)}$	Inference time (i.e., inference delay) at *tier* $\#m$	s	To be optimized
$\mathcal{P}_{MP}^{(IDLE)}(j,m)$ (resp., $\mathcal{P}_{AP}^{(IDLE)}(j,m)$)	Idle power of the main (resp., auxiliary) processor at $FN(j,m)$	Watt	10^{-7} (resp., 10^{-7})
$\gamma_{MP}^{(j,m)}$ (resp., $\gamma_{AP}^{(j,m)}$)	Exponent of the dynamic power consumption of the main (resp., auxiliary) processor at $FN(j,m)$	Dimensionless	3.2 (resp., 3.2)
$\mathcal{K}_{MP}^{(j,m)}$ (resp., $\mathcal{K}_{AP}^{(j,m)}$)	Scaling factor profiling the dynamic power consumption of the main (resp., auxiliary) processor at $FN(j,m)$	$\frac{\text{Watt}}{(\text{CPU cycles / s})^{\gamma}}$	5×10^{-36} (resp., 5×10^{-37})
ε_{jm} (resp., β_{jm})	Processing density of the main (resp., auxiliary) processor at $FN(j,m)$	CPU cycles/bit	2.5×10^3 (resp., 1.5×10^3)
ψ_m	Per-connection transport-to- physical scaling factor at *tier* $\#m$	Dimensionless	1.11
$\zeta^{(Tx)}(j,m)$ (resp., $\zeta^{(Rx)}(j,m)$)	Exponent of the per-connection transmit (resp., receive) dynamic power consumption at $FN(j,m)$	Dimensionless	2.4 (resp., 2.2)
$\Omega_{NET}^{(Tx)}(j,m)$ (resp., $\Omega_{NET}^{(Rx)}(j,m)$)	Scaling factor of the per-connection transmit (resp., receive) dynamic network power at $FN(j,m)$	$\frac{\text{Watt}}{(\text{bit/s})^{\zeta}}$	10^{-13} (resp., 10^{-14})
RTT_m	Per-connection average Round-Trip-Time at *tier* $\#m$	s	10^{-3}
η	Exponent of the per connection RTTs	Dimensionless	0.6
$FanIn(j,m)$ (resp., $FanOut(j,m)$)	Number of input (resp., output) ports at $FN(j,m)$	Dimensionless	Topology depending (see Table 5)
$\mathcal{P}_{Net}^{(IDLE-Tx)}(j,m)$ (resp., $\mathcal{P}_{NET}^{(IDLE-Rx)}(j,m)$)	Idle power consumed by each output (resp., input) port at $FN(j,m)$	Watt	0.5×10^{-8} (resp., 0.5×10^{-8})

TABLE 9. List of the main parameters, their meaning/role, measuring units and simulated default values.

$\mathcal{E}_{TOT}$ (resp., $\mathcal{E}_{COP}, \mathcal{E}_{NET}$)	Total (resp., computing, networking) energy consumed by the implemented Fog platform	Joule	To be optimized
λ_m	Lagrange multiplier of the m-th constraint on the allowed inference time	Joule	To be optimized
A_0	Clipping factor of the step-sizes of the *LOP* iterations	Dimensionless	1.0×10^{-5}
σ	Sensitivity factor of the step-sizes of the *LOP* iterations	Dimensionless	3.5
I_{MAX}	Maximum number of the performed *LOP* iterations	Dimensionless	70

re-configuration of the available computing-plus-networking resources. The final goal is to allow the resulting *LiFo* technological platform to quickly *self*-detect (typically, unpredictable) environmental changes and promptly *self*-react them. The sensitivity of the energy-vs.-inference delay performance of the designed *LiFo* platform on a number of system parameters is numerically checked and compared with the corresponding ones of some state-of-the-art benchmark solutions under various IoT-oriented operative scenarios.

Being the era of the Deep Learning-Fog Computing convergence just incoming, we think that the results of this article are only the tip of the iceberg, and, then, they could be amenable of further developing along (at least) four main research directions.

First, *Federated Learning* (*FL*) is emerging as an interesting paradigm for training complex Deep Neural Networks on the basis of heterogeneous data-sets generated on-line by spatially-distributed IoT devices [49]. Therefore, how exploit the spatial-scaling and data-aggregation capabilities natively supported by the proposed *LiFo* paradigm for performing the *real-time FL* of CDNNs with early exits in an *asynchronous* and *energy-efficient* way may be a first research line of potential interest.

Second, in the stack topology in Fig. 1a, the here considered CDNNs with early exits could be augmented by inter-layer feedback connections, so to give arise to recurrent-type CDNNs for the effective mining of time-correlated data sequences [6]. How generalize the proposed *LiFo* technological platform of Fig. 3 for the support of recurrent CDNNs with early exits may be a research topic of practical interest.

A third possible research line stems from the consideration that the envisioned 6G communication paradigm would adopt massive numbers of terminal antennas which would operate in the (up to date nearly unexplored) Terahertz band [50]. Hence, generalizing the networking energy models of Section V for accounting for the effects of spatial coding and multiplexing [51] operating over Terahertz communication channels may be a third valuable research line.

Forth, the ultimate goal of the emerging paradigm of the so-called Social IoT (*SIoT*) is to make scalable (very) large IoT networks through the self-establishment and self-management of suitable inter-thing social relationships [52]. How effectively exploiting the developed *LiFo* technological platform for performing in real-time the mining operations needed to discover the inter-thing social networks featuring the *SIoT* paradigm may be a further research topic of potential interest.

Overall, the final message of this article is: the *Learning-in-the-Fog* era is knocking at the door. Please, come in!

APPENDIX A
TAXONOMY AND DEFAULT SIMULATED SETTING

Table 9 details the main symbols used in the paper, their meaning/role, measuring units and simulated default values. These last are quite typical of the here considered Fog-based settings [33], [37].

APPENDIX B
PROOF OF THE *LOP* CONVEXITY

The proof of *Proposition 1* of Section VII uses some auxiliary formal results, which are of own interest and are presented by the following *Lemmas 1* and *2*.

Lemma 1 (*On the Convexity and Monotonic Behavior of the Per-Exit Inference Times*):

a) Each per-node execution time $T_{EXE}(j, m)$, $1 \leq j \leq m_m$, $1 \leq m \leq M$, in (16), (16.1) is a jointly strictly convex and decreasing function in the involved optimization variables $f_{jm}, \tilde{f}_{jm}$ and R_m;
b) each inference time function $T_{EXE}^{(1,m)}$, $1 \leq m \leq M$, in (18) is jointly convex and not increasing in the scalar components of the compound resource vector $\overrightarrow{X}$ in (40).

Proof:

a) Since the function $f(y) \triangleq 1/y$, $y \geq 0$, is strictly convex and decreasing in y, each per-node execution time in (16) and (16.1) retains, by design, the same properties, because it is a linear superposition with nonnegative coefficients of strictly convex and decreasing component functions.
b) The *max* function is convexity preserving and not decreasing with respect to its arguments [43]. Hence, being the summation of multiple *max* functions, each inference time in Eq. (18) is jointly convex in $\overrightarrow{X}$ and does not decrease for increasing values of each scalar component of $\overrightarrow{X}$. However, $T_{EXE}^{(1,m)}$ in (18) is *not* jointly *strictly* convex and/or strictly decreasing with respect to the scalar components of $\overrightarrow{X}$, because $T_{EXE}^{(1,m)}$ does not

depend, indeed, on the subset of optimization variables: $\left\{f_{jk}, \tilde{f}_{jk}, R_k\right\}$ for $k \geq m+1$.

$\square$

Lemma 2 (*On the Convexity of the Computing and Network Energies*): Let us assume that all the γ's and ζ's exponents present in the power models of Eqs. (20), (21), (28) and (33) are equal or larger than 2. Hence, all the main-auxiliary computing and receive-transmit network energies in Eqs. (22), (25), (32) and (33) are jointly convex in the scalar components of the compound resource vector $\overrightarrow{X}$ in (40).

Proof: The proof directly follows from the following three formal properties:

1) *Lemma 1.b* guarantees that the Cloud inference time $T_{EXE}^{(1,M)}$ is jointly convex in the components of $\overrightarrow{X}$ (see Eq. (18) with $m = M$);
2) the power-function $g(y) \triangleq y^{a-1}$, $y \geq 0$, is convex in y for each exponent $a \geq 2$; and,
3) any linear combination with nonnegative coefficients of jointly convex functions is still jointly convex [43].

$\square$

Afterwards, the proof of Proposition 1 directly follows from the following two observations.

First, each nonlinear constraint in (41b) is jointly convex in the involved optimization variables by *Lemma 1.b*. Second, under the assumption of Proposition 1 on the values of the exponents γ's and ζ's present in the power models of Section V, the resulting objective function $\mathcal{E}_{TOT}$ in (41a) is jointly convex in the scalar components of the vector variable $\overrightarrow{X}$ in (40). This is due to the fact that *Lemma 2* guarantees that $\mathcal{E}_{TOT}$ is the summation of jointly convex energy functions (see the defining relationships in Eqs. (36), (37) and (38)).

The proof of Proposition 1 is now completed.

REFERENCES

[1] *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*. Accessed: Nov. 10, 2020. [Online]. Available: https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf

[2] *Cisco Global Cloud Index: Forecast and Methodology*. Accessed: Nov. 10, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html

[3] D. Hanes, G. Salgueiro, P. Grossetete, R. Barton, and J. Henry, *IoT Fundamentals-Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Indianapolis, IN, USA: Cisco Press, 2017.

[4] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Dresden, Germany, Mar. 2016, pp. 475–480.

[5] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup, "Conditional computation in neural networks for faster models," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Juan, Puerto Rico, May 2016, pp. 1–4.

[6] S. Scardapane, M. Scarpiniti, E. Baccarelli, and A. Uncini, "Why should we add early exits to neural networks?" *Cognit. Comput.*, vol. 12, no. 5, pp. 954–966, Sep. 2020.

[7] S. Teerapittayanon, B. McDanel, and H. T. Kung, "BranchyNet: Fast inference via early exiting from deep neural networks," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*. Cancun, Mexico: IEEE, Dec. 2016, pp. 2464–2469.

[8] S. Teerapittayanon, B. McDanel, and H. T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*. Atlanta, GA, USA: IEEE, Jun. 2017, pp. 328–339.

[9] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of Everything*. Singapore: Springer, Oct. 2018, pp. 103–130.

[10] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, "Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study," *IEEE Access*, vol. 5, pp. 9882–9910, 2017.

[11] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Archit.*, vol. 98, pp. 289–330, Sep. 2019.

[12] E. Baccarelli, S. Scardapane, M. Scarpiniti, A. Momenzadeh, and A. Uncini, "Optimized training and scalable implementation of conditional deep neural networks with early exits for fog-supported IoT applications," *Inf. Sci.*, vol. 521, pp. 107–143, Jun. 2020.

[13] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Recent advances in information-centric networking-based Internet of Things (ICN-IoT)," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2128–2158, Apr. 2019.

[14] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2224–2287, 3rd Quart., 2019.

[15] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy, "MCDNN: An approximation-based execution framework for deep stream processing under resource constraints," in *Proc. 14th Annu. Int. Conf. Mobile Syst., Appl., Services*, Jun. 2016, pp. 123–136.

[16] X. Ran, H. Chen, X. Zhu, Z. Liu, and J. Chen, "DeepDecision: A mobile deep learning framework for edge video analytics," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 1421–1429.

[17] M. Xu, F. Qian, M. Zhu, F. Huang, S. Pushp, and X. Liu, "DeepWear: Adaptive local offloading for on-wearable deep learning," *IEEE Trans. Mobile Comput.*, vol. 19, no. 2, pp. 314–330, Feb. 2020.

[18] H.-J. Jeong, H.-J. Lee, C. H. Shin, and S.-M. Moon, "IONN: Incremental offloading of neural network computations from mobile devices to edge servers," in *Proc. ACM Symp. Cloud Comput.*, Oct. 2018, pp. 401–411.

[19] E. Li, Z. Zhou, and X. Chen, "Edge intelligence: On-demand deep learning model co-inference with device-edge synergy," in *Proc. Workshop Mobile Edge Commun.*, Aug. 2018, pp. 31–36.

[20] G. Li, K. Ota, M. Dong, J. Wu, and J. Li, "DeSVig: Decentralized swift vigilance against adversarial attacks in industrial artificial intelligence systems," *IEEE Trans. Ind. Informat.*, vol. 16, no. 5, pp. 3267–3277, May 2020.

[21] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proc. 22nd Int. Conf. Architectural Support Program. Lang. Operating Syst. (ASPLOS)*, 2017, pp. 615–629.

[22] S. Leroux, S. Bohez, E. De Coninck, T. Verbelen, B. Vankeirsbilck, P. Simoens, and B. Dhoedt, "The cascading neural network: Building the Internet of smart things," *Knowl. Inf. Syst.*, vol. 52, no. 3, pp. 791–814, Sep. 2017.

[23] J. Mao, X. Chen, K. W. Nixon, C. Krieger, and Y. Chen, "MoDNN: Local distributed mobile computing system for deep neural network," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 1400–1405.

[24] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "DeepThings: Distributed adaptive deep learning inference on resource-constrained IoT edge clusters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2348–2359, Nov. 2018.

[25] D. Li, T. Salonidis, N. V. Desai, and M. C. Chuah, "DeepCham: Collaborative edge-mediated adaptive deep learning for mobile object recognition," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Washington, DC, USA, Oct. 2016, pp. 64–76.

[26] S. Yi, Z. Hao, Q. Zhang, Q. Zhang, W. Shi, and Q. Li, "LAVEA: Latency-aware video analytics on edge computing platform," in *Proc. 2nd ACM/IEEE Symp. Edge Comput.*, Oct. 2017, pp. 1–13.

[27] N. Talagala, S. Sundararaman, V. Sridhar, D. Arteaga, Q. Luo, S. Subramanian, S. Ghanta, L. Khermosh, and D. Roselli, "ECO: Harmonizing edge and cloud with ML/DL orchestration," in *Proc. USENIX Workshop Hot Topics Edge Comput. (HotEdge)*, 2018, pp. 1–7.

[28] R. Priyadarshini, R. Barik, and H. Dubey, "DeepFog: Fog computing-based deep neural architecture for prediction of stress types, diabetes and hypertension attacks," *Computation*, vol. 6, no. 4, p. 62, Dec. 2018.

[29] G. Li, G. Xu, A. K. Sangaiah, J. Wu, and J. Li, "EdgeLaaS: Edge learning as a service for knowledge-centric connected healthcare," *IEEE Netw.*, vol. 33, no. 6, pp. 37–43, Nov. 2019.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[31] J. Cid-Sueiro, J. I. Arribas, S. Urban-Munoz, and A. R. Figueiras-Vidal, "Cost functions to estimate a posteriori probabilities in multiclass problems," *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 645–656, May 1999.

[32] I. Hubara, M. Courbariaux, and D. Soudry, "Quantized neural networks: Training neural networks with low precision weights and activations," *J. Mach. Learn. Res.*, vol. 18, pp. 1–30, Jan. 2018.

[33] E. Baccarelli, M. Scarpiniti, and A. Momenzadeh, "EcoMobiFog–design and dynamic optimization of a 5G mobile-fog-cloud multi-tier ecosystem for the real-time distributed execution of stream applications," *IEEE Access*, vol. 7, pp. 55565–55608, 2019.

[34] E. Baccarelli, P. G. Vinueza Naranjo, M. Shojafar, and M. Scarpiniti, "Q*: Energy and delay-efficient dynamic queue management in TCP/IP virtualized data centers," *Comput. Commun.*, vol. 102, pp. 89–106, Apr. 2017.

[35] J. Kwak, Y. Kim, J. Lee, and S. Chong, "DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, pp. 2510–2523, Dec. 2015.

[36] R. Basmadjian and H. de Meer, "Evaluating and modeling power consumption of multi-core processors," in *Proc. 3rd Int. Conf. Future Energy Syst. Where Energy, Comput. Commun. Meet e-Energy*. New York, NY, USA: IEEE, May 2012, pp. 1–10.

[37] W. Li, I. Santos, F. C. Delicato, P. F. Pires, L. Pirmez, W. Wei, H. Song, A. Zomaya, and S. Khan, "System modelling and performance evaluation of a three-tier cloud of things," *Future Gener. Comput. Syst.*, vol. 70, pp. 104–125, May 2017.

[38] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, "Virtual machine power metering and provisioning," in *Proc. 1st ACM Symp. Cloud Comput. (SoCC)*, 2010, pp. 39–50.

[39] E. Baccarelli, M. Biagi, R. Bruno, M. Conti, and E. Gregori, "Broadband wireless access networks: A roadmap on emerging trends and standards," in *Broadband Services: Business Models and Technologies for Community Networks*. Hoboken, NJ, USA: Wiley, Oct. 2005, ch. 14, pp. 215–240.

[40] E. Baccarelli and M. Biagi, "Power-allocation policy and optimized design of multiple-antenna systems with imperfect channel estimation," *IEEE Trans. Veh. Technol.*, vol. 53, no. 1, pp. 136–145, Jan. 2004.

[41] E. Baccarelli, M. Biagi, C. Pelizzoni, and N. Cordeschi, "Optimized power allocation for multiantenna systems impaired by multiple access interference and imperfect channel estimation," *IEEE Trans. Veh. Technol.*, vol. 56, no. 5, pp. 3089–3105, Sep. 2007.

[42] E. Baccarelli, R. Cusani, and S. Galli, "A novel adaptive receiver with enhanced channel tracking capability for TDMA-based mobile radio communications," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 9, pp. 1630–1639, Dec. 1998.

[43] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 3rd ed. Hoboken, NJ, USA: Wiley, 2017.

[44] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Trans. Netw.*, vol. 24, no. 1, pp. 596–609, Feb. 2016.

[45] A. H. Sayed and X. Zhao, "Asynchronous adaptive networks," in *Cooperative and Graph Signal Processing*. Amsterdam, The Netherlands: Elsevier, 2018, ch. 1, pp. 3–68.

[46] E. Baccarelli, M. Scarpiniti, and A. Momenzadeh, "Fog-supported delay-constrained energy-saving live migration of VMs over MultiPath TCP/IP 5G connections," *IEEE Access*, vol. 6, pp. 42327–42354, 2018.

[47] M. Scarpiniti, E. Baccarelli, A. Momenzadeh, and S. S. Ahrabi, "DeepFogSim: A toolbox for execution and performance evaluation of the inference phase of conditional deep neural networks with early exits atop distributed Fog platforms," *Appl. Sci.*, vol. 11, no. 1, pp. 1–42, Jan. 2021, Paper 377.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Lake Tahoe, NV, USA, Dec. 2012, pp. 1097–1105.

[49] L. U. Khan, W. Saad, Z. Han, E. Hossain, and C. S. Hong, "Federated learning for Internet of Things: Recent advances, taxonomy, and open challenges," 2020, *arXiv:2009.13012*. [Online]. Available: https://arxiv.org/abs/2009.13012

[50] M. Giordani, M. Polese, M. Mezzavilla, S. Rangan, and M. Zorzi, "Towards 6G networks: Use cases and technologies," *IEEE Commun. Mag.*, vol. 58, no. 3, pp. 55–61, Mar. 2020.

[51] E. Baccarelli and M. Biagi, "Optimized power allocation and signal shaping for interference-limited multi-antenna 'ad hoc' networks," in *Personal Wireless Communications* (Lecture Notes in Computer Science), vol. 2775, M. Conti, S. Giordano, E. Gregori, and S. Olariu, Eds. Berlin, Germany: Springer, 2003, pp. 138–152.

[52] E. Baccarelli, M. Scarpiniti, P. G. V. Naranjo, and L. Vaca-Cardenas, "Fog of social IoT: When the fog becomes social," *IEEE Netw.*, vol. 32, no. 4, pp. 68–80, Jul. 2018.

ENZO BACCARELLI received the Laurea degree in electronic engineering and the Ph.D. degree in communication theory and systems from the Sapienza University of Rome and the Post-Doctorate degree in information theory and applications from the INFOCOM Department, Sapienza University of Rome, in 1995. He is currently a Full Professor in information and communication engineering with the DIET Department, Sapienza University of Rome. He has coauthored "Fog of Everything: Energy-Efficient Networked Computing Architectures, Research Challenges, and a Case Study," which was one of the most cited IEEE Access articles published in 2017. His current research interests include data networks, distributed computing, networked data centers, and fog computing. He was the National Coordinator of several MIUR projects. He has been listed in the 2020 World's Top 2% Scientists ranked by the Stanford University. From 2005 to 2010, he has served as an Associate Editor for IEEE Communications Letters.

MICHELE SCARPINITI (Senior Member, IEEE) received the Laurea degree (Hons.) in electrical engineering from the Sapienza University of Rome, Italy, in 2005, and the Ph.D. degree in information and communication engineering in 2009. He is currently an Associate Professor of circuit theory and multimedia signal processing with the Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome. He is a member of the Intelligent Signal Processing and MultiMedia (ISPAMM) Laboratory, an interdisciplinary research group working at the DIET Department aiming at the design and development of innovative methodologies for multimedia processing. He has coauthored "Fog of Everything: Energy-Efficient Networked Computing Architectures, Research Challenges, and a Case Study," which was one of the most cited IEEE Access articles published in 2017. His current research interests include nonlinear adaptive filters, audio processing, blind signal processing, and neural networks for signal processing. He is also a member of the Audio Engineering Society (AES) and the Società Italiana Reti Neuroniche (SIREN).

ALIREZA MOMENZADEH received the Bachelor of Civil Engineering degree from Estahban University, Iran, and the master's degree in structural engineering from the University Technology of Malaysia, in 2015. He is currently a Researcher with the DIET Department, Sapienza University of Rome. His current research interests include nonlinear optimization through hybrid methods, fog computing architectures, and related sensor/actuator-based control applications.

SIMA SARV AHRABI received the Ph.D. degree in mathematical models for engineering, electromagnetism, and nanoscience from the Sapienza University of Rome, in 2018. She is currently a Researcher with the Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome. Her current research interests include mathematical optimization, deep learning, and fog computing architectures.