Distributed Context-Aware Resource Allocation for Dynamic Sensor Fusion in Edge Inference

Yashuo Wu
Dept. of EECS, University of California, Irvine
Irvine, USA
yashuow@uci.edu

Carla Fabiana Chiasserini
Politecnico di Torino
Torino, Italy
carla.chiasserini@polito.it

Marco Levorato
Dept. of ICS, University of California, Irvine
Irvine, USA
levorato@uci.edu

***Abstract*—The fusion of multi-modal information, such as images and LiDAR scans, is instrumental to maximize the performance of many computer vision tasks in next generation systems and applications. However, supporting fusion necessitates considerable effort, challenging the availability of computing and communication resources in edge systems. This work addresses this challenge by maximizing resource efficiency in systems where *mobile devices collect multi-modal sensor data* and use *dynamic multi-branched DNN models* to adapt inference to the operating context. To tune the overall system response to the *context* (e.g., weather conditions), we propose a dual-scale control approach: *centralized* orchestration of spectrum resources, and distributed *individual device-level* control of the execution path of the dynamic DNN fusion models. The control agents are driven by a novel context-aware decision-making method combined with game theory, named Context-Aware Network Slicing Auction (CANSA), which optimizes DNN inference performance, network slicing, and energy consumption. The decision-making performs such optimization by: (i) selecting data and features that best fit the current context; (ii) deciding on the appropriate DNN model complexity, including the use of multi-modal sensor fusion techniques for better data integration, and (iii) deploying these models on the most appropriate nodes (local nodes or edge servers). Results, obtained using real-world multi-modal data, show that CANSA surpasses conventional allocation methods by up to 52.3% in terms of inference task success rate.**

***Index Terms*—Edge computing, Semantic-driven system orchestration, Network Slicing**

I. Introduction

The evolution of edge computing has been a cornerstone of the rapidly expanding Internet of Things (IoT), enabling data processing closer to data sources and reducing latency in applications based on Deep Neural Networks (DNNs) [1]. Indeed, DNNs, which have become essential in a broad range of data analysis tasks, require a large amount of computing resources [2]. Edge computing addresses these challenges by positioning infrastructure-level computing resources at the network edge to take over the execution of DNNs.

Edge computing, however, suffers from a high degree of complexity, dynamicity, and heterogeneity, depending on characteristics of the data, tasks, load and requirements [3]. For example, when considering vehicular autonomy, the integration and fusion of information from multiple sensors (e.g., multi-camera, LiDAR, and radar) is crucial to enabling safe operations in a broad range of conditions [4]. However, transmitting all raw sensor data to the edge is often infeasible in real-time due to bandwidth constraints. Additionally, as applications offloading inference to edge servers become more common, the need for strategies to efficiently use spectrum and computing resources grows [5].

To address these issues, split computing has emerged as a promising mid-ground solution, where mobile devices and edge servers perform the inference collaboratively [6]. Intuitively, partial execution of a DNN performed by the mobile device means that the output tensor of a DNN section is transmitted rather than the initial data, providing compression opportunities, as well as an increased flexibility in the network load and computing load distribution. Furthermore, recent studies, such as [7], show that processing and transmitting data from all sensor data is not only not necessary in all the operating contexts, but also potentially detrimental. For instance, in some adverse weather conditions, additional modalities may degrade performance by introducing noise or undesirable redundancy. Both the structure of the model and the amount of data needed to complete analysis, thus, can be adapted in response to the operating context as well as the resources available to the system.

In this paper, we design a context-aware edge inference framework, which we refer to as Context-Aware Network Slicing Auction (CANSA). We take multi-branched DNNs for sensor fusion as a starting point, and build a network-level adaptation strategy that jointly optimizes *which* branches to activate, *where* to deploy them (at the mobile device, at the edge server, or collaboratively via split execution), *what* data to transmit, and *how* to allocate the limited communication resources to individual mobile devices. By integrating the following key components, our solution jointly addresses dynamic model inference, data selection, and game-theoretic radio resource blocks (RBs) allocation.

First, CANSA performs context-aware model and data configuration to intelligently determine the essential data to process and transmit, to improve overall efficiency while preserving task performance. *Second,* CANSA uses dynamic multi-branch architectures models, and context-aware model partitioning to adapt to varying channel conditions. In order to ensure efficient allocation strategies that optimally trade-off resource efficiency with DNN model complexity and, hence, quality of service (QoS), our framework is based on an auction-based game theoretical engine to optimize mobile-

side decisions. *Third,* based on individual devices' decisions, an edge orchestrator CANSA iteratively optimizes network slicing, allowing for the division of available resources to serve task streams with different requirements. The edge orchestrator performs constrained optimization to allocate resource blocks (RB) to reach equilibrium as bid for RBs under latency and accuracy constraints.

In summary, our key contributions are as follows:

- We propose CANSA, a context-aware resource framework that supports the distributed execution of multi-branched DNNs for sensor fusion under performance requirements.
- We develop a distributed game-theoretic optimization mechanism that dynamically balances resource efficiency and DNN deployment performance.
- We present a comprehensive decision making system architecture for sensor fusion and split inference, ensuring optimal context-aware utilization of resource blocks.
- Our results show that CANSA optimizes both network slicing and computing resource utilization, providing an average 52.3% gain compared to conventional resource allocation baselines.

II. Related Work

Our work is positioned within three main research domains.

Network Slicing and Resource Allocation. Network slicing facilitates the coexistence of services with diverse QoS needs and provides the flexibility and scalability required for effective network resource allocation [8]. The primary objective is to dynamically distribute network resources – such as bandwidth, computing power, and storage – across the *slices* to satisfy specific QoS requirements while optimizing overall performance. Existing work proposed reservation-based methods [9] that allocate resources to network slices based on explicit reservation request, as well as share-based approaches [10] that predefine shares associated with each slice. Machine learning techniques have also been used to enhance resource allocation by predicting traffic patterns, enabling proactive adjustments, improving decision-making processes, and dynamically optimizing resource allocation [11].

Dynamic (Multi-branch) DNNs and Context-Based Fusion. In computer vision and especially object detection, the utilization of multi-branch DNN models and context-based fusion has gathered significant attention. [12] proposed a gate mechanism to determine the optimal expert model for a given task. [13] leveraged an analogous structure between branches with similar tasks in object detection, resulting in faster training. HydraFusion [7] developed a two-stage training approach, initially pretraining branches and then training the gate to select branches for improved results. Context-based fusion methods, such as those applied in the case of autonomous vehicles [14], and context-aware CNNs [15], have also contributed to improving object detection performance. Fusion strategies with the use of multiple sensors consist of both early fusion, which involves feature extraction, and late fusion, which combines the results. However, the success of fusion strategies is not guaranteed. [16] explored branch selection in fusion and extended it to autonomous vehicles. Our work leverages the structure of dynamic multi-branched DNN models for sensor fusion in conjuntion with semantic network slicing to improve the overall efficiency of edge computing systems.

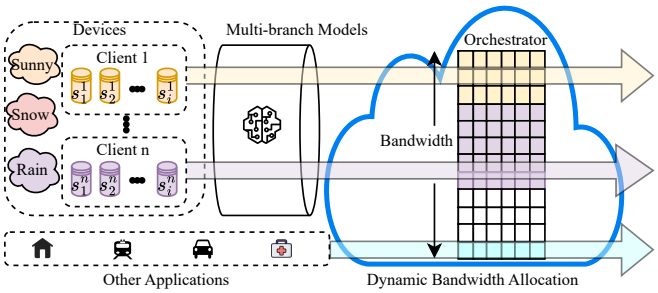

Figure 1. Overview of our scenario we consider: N clients deploy dynamic (multi-branch) DNN models with personalized requirements on inference performance and latency. The clients share the same bandwidth and rely on an edge orchestrator for radio resource allocation, as demands and context vary.

DNN Deployment. Efficient DNN deployment and resource allocation are crucial considerations, particularly when addressing scenarios with diverse and input data structures [17]. Achieving a balanced resource allocation is essential for the reliable operations of these applications, necessitating techniques such as model splitting and resource-aware machine learning. [18], [19].

In contrast with previous work on isolation sensor fusion, inference execution, and radio resource allocation, our system unifies these components into a context-aware, adaptive framework. It dynamically adjusts data selection, model complexity, and deployment strategies based on environmental context, varying channel conditions, and real-time availability of radio resource blocks (RBs).

III. System Model and Problem Formulation

We consider a dynamic system, as depicted in Fig. 1, comprising two main components: a group of clients $i \in \mathcal{N}$, (corresponding to a mobile device) that deploy a DNN-based application, and an orchestrating edge server. Specifically, we consider an extension of the dynamic DNN for object detection proposed in [7]. To execute inference tasks, each client selects a dynamic model configuration $m \in \mathcal{M}$, adapting to current system and network conditions. The underlying dynamic model architecture, illustrated in Fig. 2, indeed supports flexible deployment and consists of a shared, pretrained feature extractor with stems tailored for different sensor modalities, followed by multiple "analysis" branches that differ in complexity and process the output of either single-sensor or fused-sensor inputs. The architecture includes both early fusion — where intermediate features from multiple modalities are integrated before branching—and late fusion, which aggregates final outputs from multiple branches. The

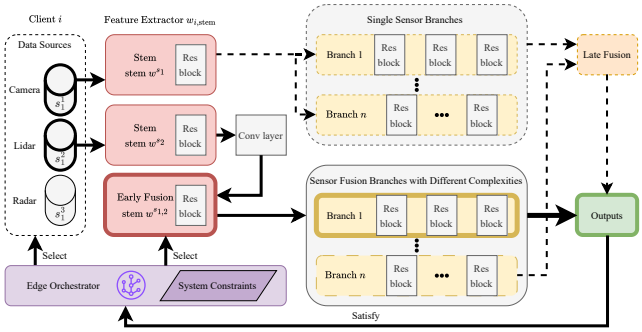

Figure 2. Overview of the dynamic model architecture across clients and an orchestrator running on the edge server. A shared feature extractor is followed by branches of varying complexity. Early fusion integrates intermediate features, while late fusion combines final outputs.

Table I
NOTATION

Symbol	Definition
$i \in \mathcal{N}$	Set of clients, each requiring a DNN model configuration
d_i	Distance between client i and base station
D_i	Total data size transmitted by client i
C_i	Channel capacity of client i
b_i	Bid submitted by client i
$s \in \mathcal{S}$	Sensor data sources
$m \in \mathcal{M}$	Set of DNN model configurations
R_i	Number of RBs allocated to client i
v_i	urgency factor of client i
g	Channel gain
$h \in \mathcal{H}$	Set of possible contexts
$A_{\min}(h)$	Minimum required accuracy under context h
$L_{\max}$	Maximum required latency
f	Model configuration selector
$x \in \mathcal{X}$	Set of deployment options
$U_i(m,x)$	Utility function of client i
A_i^m	Accuracy under model configuration m
$L_i^{m,x}(R_i)$	Latency of client i under m, x, and R_i
R_{rem}	No. of remaining RBs
α	Success rate

system dynamically allocates the available radio resources – expressed as resource blocks (RB), R_{total}, to address various client requirements and network conditions under different contexts, with $h{\in}\mathcal{H}$ denoting the possible contexts experienced by clients. In the following we describe in details the operations of the system. The notation is summarized in Table I.

A. Mobile Clients and Dynamic DNN Models

Each client $i \in \mathcal{N}$ is characterized by a context-dependent minimum accuracy requirement $A_{\min}(h)$, a maximum latency constraint $L_{\max}$, and an available channel bandwidth C_i. Clients compete for radio resources by requesting allocation requests to the edge orchestrator. Upon receiving the number of allocated RBs, R_i, client i adaptively configures its DNN inference model by selecting an appropriate model configuration $m{\in}\mathcal{M}$ based on the experienced context h, the accuracy and latency requirements, and the current channel conditions. Each client performs the following steps: *(i)* selecting data inputs from available sensors $s{\in}\mathcal{S}$, *(ii)* choosing an appropriate DNN model configuration $m{\in}\mathcal{M}$ (that is, the selection of the stems and branches), which implies different complexity and accuracy levels as well as inference latency and energy consumption, and *(iii)* choosing where to deploy the selected model configuration depending on RBs allocation determined by the edge orchestrator.

The possible deployment options, $x_i{\in}\mathcal{X}$, are as follows: (*1*) The entire model (stems and branches) is deployed on the mobile device; (*2*) The stems are deployed on the mobile device and the branches on the edge server; (*3*) the entire model is deployed at the edge server. We define the total latency $L_i^{m,x}(R_i)$ of client i, which includes communication and computing components and clearly is a function of the configuration and offloading strategy. For each client i, we then define a model configuration selector $f_i : \mathcal{H} \to \mathcal{M}$ that selects a model configuration $m{\in}\mathcal{M}$ based on the context h, such that the selected model meets the constraints: $A_i^m \geq A_{\min}(h)$ and latency $L_i^{m,x}(R_i) \leq L_{\max}$.

B. Edge Orchestrator Radio Resource Allocation

The edge orchestrator has total RB budget of R_{total} available and allocates R_i to the client i, which, as mentioned above, directly affects the client's latency. The objective of the edge orchestrator ensures that at least $\alpha_{\text{thr}}\%$ of the clients meet their latency and accuracy constraints, i.e.,

$$\frac{1}{|\mathcal{N}|}\sum_{i\in\mathcal{N}}\mathbb{I}\left(A_i^m \geq A_{\min}(h) \wedge L_i^{m,x}(R_i) \leq L_{\max}\right) \geq \alpha_{\text{thr}}, \quad (1)$$

where $\mathbb{I}(\cdot)$ is the indicator function.

Given the above objective and system constraints, the problem that the orchestrator has to solve can be expressed as

$$\mathbb{P}_1 : \min \sum_{i\in\mathcal{N}} R_i, \quad (2)$$

s.t. (1) and

$$0 \leq R_i \leq R_{total}, \ \forall i \in \mathcal{N}. \quad (3)$$

C. Latency and Energy Estimation

Channel Model. The achievable data rate C_i for client i follows from 3GPP-approved Floating Intercept (FI) channel model [20]:

$$C_i = R_i \cdot \eta_i \cdot \frac{B_{\text{sys}}}{N_{\text{RB}}} \quad (4)$$

where $R_i{\in}\mathbb{Z}^+$ is the number of RBs allocated to client i by the edge orchestrator; $\eta_i{=}\log_2(1{+}\text{SINR}_i)$ represents spectral efficiency per RB; B_{sys} is the system bandwidth; $N_{\text{RB}}{=}\lfloor\frac{B_{\text{sys}}}{12\cdot\Delta f}\rfloor$ is total number of available RBs ($\Delta f{=}30$ kHz).

The signal-to-interference-plus-noise ratio (SINR) for client i is modeled as $\text{SINR}_i{=}\frac{P_t\cdot g\cdot\chi_i}{P_{\text{noise}}\cdot PL(d_i)}$, where $PL(d_i)$ is path loss with shadowing [20]; P_t is the transmit power; g is the antenna gain; P_{noise} is noise power spectral density; χ_i is small-scale fading coefficient. We consider path loss and shadowing where d_i is the distance between the client's device and the base station providing access to the edge server.

Let D_i denote the size of data to be transferred from the local device to the edge server. Let μ_i represent the total

number of Floating Point Operations Per Second (FLOPS) necessary to complete the task for client i. This total number is further divided into $\mu_{i,stems}$ and $\mu_{i,branches}$. We define o_i^{LC}, local device's computational capacity, and o_i^{EC}, edge server's computational capacity, both measured in FLOPS. Thetotal latency in the different computing modalities is:

Local Processing. The whole model (stemsa and branches) is executed by the client, and the total latency is equal to $L_i^{m,x}(R_i) = \frac{\mu_i}{o_i^{\text{LC}}}$.

Partial Offloading. The clients execute the initial DNN stages (stems) and offload subsequent stages (branches) to the edge server, the total latency includes the processing time on the device, the processing time on the edge server, and the data transmission time: $L_i^{m,x}(R_i) = \frac{\mu_{i,\text{stems}}}{o_i^{\text{LC}}} + \frac{\mu_{i,\text{branches}}}{o_i^{\text{EC}}} + \frac{D_i}{C_i R_i}$, where C_i is client i data rate.

Full Offloading. The whole data and execution is offloaded to the edge server. The total latency takes into account the time for data transfer and processing on the edge server: $L_i^{m,x}(R_i) = \frac{\mu_i}{o_i^{\text{EC}}} + \frac{D_i}{C_i R_i}$

Our focus on energy consumption primarily concerns the user-end devices, under the premise that the energy required for communication is significantly smaller than that used for execution the tasks. We draw on the energy consumption analysis in [21], which employs an evaluation of energy usage based on FLOPS in different GPUs. We can, then, express client i's energy consumption as $E_i^{m,x}{=}\tau_i{\cdot}\mu_i$, where τ_i is the consumed energy per FLOPS.

D. Client Utility Maximization

Each client i aims to maximize its utility:

$$U_i(m,x){=}w_1 \cdot \sigma_A(A_i^m) + w_2 \cdot \sigma_L(L_i^{m,x}(R_i)) - w_3 \cdot E_i^{m,x} - w_4 \cdot \mathbb{I}\left(A_i^m < A_{\min}(h) \vee L_i^{m,x}(R_i) > L_{\max}\right), \quad (5)$$

through joint optimization of the selected model configuration m and model deployment strategy x, under the system and applications constraints. In the above expression, $\sigma_A(A_i^m){=}\frac{1}{1+e^{-k_A(A_i^m - A_{\min}(h))}}$, $\sigma_L(L_i^{m,x}(R_i)) = \frac{1}{1+e^{-k_L(L_{\max} - L_i^{m,x}(R_i))}}$; w_1, w_2, w_3, w_4 are weights that balance the trade-off between benefit and cost. The indicator function $\mathbb{I}$ penalizes constraint violations. Thus, the problem that each client has to solve is as follows:

$$\mathbb{P}_2: \quad \max_{m,\,x} U_i(m,x) \quad (6)$$

IV. The CANSA Algorithm

Given the distributed nature of the optimization problem, an effective solution strategy must allow the edge orchestrator and the clients to collaboratively solve $\mathbb{P}_1$ and $\mathbb{P}_2$. Our solution framework enables efficient clients-server interaction through strategic bidding, adaptive model execution, and a stable resource allocation.

We first provide an overview of CANSA algorithm (Section 1), including client-side bidding mechanism, the server-side two-phase RB allocation, and the post-allocation adaptation strategy by clients to fine-tune their inference execution based on the received resources. Subsequently (Section IV-B), we provide a detailed Nash equilibrium analysis of CANSA.

Algorithm 1 The CANSA Algorithm

Input:
  R_{total}: Total resource blocks
  γ: Bid cost coefficient (fixed)
  λ_0: Base penalty
  k_A, k_L: Sigmoid steepness parameters
  Context constraints: $\{A_{\min}(h), L_{\max}, \mathcal{M}\}$
Output:
  $\{R_i^*\}$: Final allocations
  $\{m_i^*\}$: Selected models
  α: System-wide success rate
Initialize:
  Priority queue $\mathcal{Q} \leftarrow \emptyset$
  $R_{\text{rem}} \leftarrow R_{\text{total}}$
for each client $i \in \mathcal{N}$ **do** ▷ Client.get_bid_commitment()
  Compute $R_{\min}^i$:
    $R_{\min}^i \leftarrow \min R$ s.t. $L_i^{m,x}(R_i) \le 0.9 L_{\max} \wedge A_i^m \ge A_{\min}(h)$
  Calculate urgency factor: $\upsilon_i \leftarrow \frac{R_{\text{total}} - R_{\min}^i}{R_{\text{total}}}$
  Compute initial bid: $b_i \leftarrow \frac{(R_{\min}^i)^2}{L_{\max}+\epsilon} \cdot \upsilon_i \cdot \frac{|\mathcal{N}|}{R_{\text{total}}}$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(R_{\min}^i, b_i, i)\}$
end for
Phase 1: Priority Allocation
  ▷ Edge Orchestrator.run_auction()
Sort $\mathcal{Q}$ by $(R_{\min}^i, -b_i)$ ▷ Smallest $R_{\min}$ first
for $(R_{\min}^i, b_i, i) \in \mathcal{Q}$ **do**
  if $R_{\text{rem}} \ge R_{\min}^i$ **then**
    $R_i^* \leftarrow R_{\min}^i$
    $R_{\text{rem}} \leftarrow R_{\text{rem}} - R_{\min}^i$
  end if
end for
Phase 2: Proportional Allocation
if $R_{\text{rem}} > 0$ **then**
  $B_{\text{total}} \leftarrow \sum b_i + \lambda_0$
  for each client $i \in \mathcal{N}$ **do**
    $\Delta R_i \leftarrow \min\left(\left\lfloor \frac{b_i}{B_{\text{total}}} R_{\text{rem}} \right\rfloor, R_{\text{rem}}\right)$
    $R_i^* \leftarrow R_i^* + \Delta R_i$
    $R_{\text{rem}} \leftarrow R_{\text{rem}} - \Delta R_i$
  end for
end if
Phase 3: Client Post-Allocation Adaptation
for each client $i \in \mathcal{N}$ **do**
  Select $m_i^* \leftarrow \arg\min_{m \in \mathcal{M}_i^{\text{feas}}(E_i^m)}$ ▷ Energy-optimal
  where $\mathcal{M}_i^{\text{feas}} = \{m | A_i^m \ge A_{\min}(h), L_i^{m,x}(R_i^*) \le L_{\max}\}$
  $u_i \leftarrow \sigma_A(A_i^{m^*}) + \sigma_L(L_i^{m^*,x^*}(R_i)) - \gamma b_i^2$
end for
Return $\{R_i^*\}, \{m_i^*, x_i^*\}, \alpha \leftarrow \frac{1}{|\mathcal{N}|} \sum \mathbb{I}_{\text{success}}^i$

A. Client-Server Interaction

1) Client-Side Bid Generation: Each client i computes the minimum required RBs, denoted with $R^i_{\min}$, by selecting feasible model configurations under latency and accuracy constraints. This calculation is based on the set of feasible DNN model configurations $\mathcal{M}_{\text{fea}}$ such that $R^i_{\min} \leftarrow \min R$ s.t. $L^{m,x}_i(R^i_{\min}) \leq 0.9 L_{\max} \wedge A^m_i \geq A_{\min}(h)$. The client then calculates an urgency factor $\upsilon_i \leftarrow \frac{R_{\text{total}} - R^i_{\min}}{R_{\text{total}}}$, reflecting how critical the resource need is, relative to the total capacity. For example, a client with poor channel conditions must bid higher to secure enough resources to meet its latency requirement. Using this urgency factor, the client generates a context-aware bid:

$$b_i \leftarrow \frac{(R^i_{\min})^2}{L_{\max} + \epsilon} \cdot \upsilon_i \cdot \frac{N}{R_{\text{total}}}. \tag{7}$$

Each client submits its bid b_i and resource estimate $R^i_{\min}$ to the edge orchestrator. The bidding mechanism incorporates a **truthful bidding incentive**: a quadratic penalty is applied in the client's utility to discourage overbidding or underbidding, promoting accurate reporting of resource needs.

2) Server-Side Two-Phase Allocation: Upon receiving all bids, the orchestrator allocates RBs in two phases:

Phase 1 - Priority Allocation: Clients are sorted by increasing R^i_{min} and, for ties, decreasing bid b_i. The orchestrator then sequentially allocates R^i_{min} to clients in this order until the remaining resource pool is insufficient. This process guarantees **Priority Preservation**: clients with the most urgent or minimal demands are satisfied first, which allows us to prove the following property.

Property 1. *Priority preservation ensures **Pareto-efficient satisfaction** where no reallocation can satisfy more clients without violating total resource constraints.*

Proof. Suppose, by contradiction, that there exists another allocation satisfying more clients than our priority allocation. Clients are allocated exactly their minimal feasible demand $R^i_{\min}$ in ascending order until exhaustion. Since each $R^i_{\min}$ is minimal and clients are sorted by increasing resource need, any reallocation that attempts to satisfy an additional client must either: reducing an already satisfied client below $R^i_{\min}$, violating feasibility, or allocating more total resources than available, violating the resource constraint. Thus, no alternative allocation can improve without violating feasibility or resource constraint, which proves the thesis. □

Phase 2 - Proportional Allocation: If any RBs remain, denoted by R_{rem}, the orchestrator distributes them proportionally to the submitted bids. I.e., defining $B_{\text{total}} = \sum b_i + \lambda_0$,

$$\Delta R_i(b_i, b_{-i}) \leftarrow \min\left(\left\lfloor \frac{b_i}{B_{\text{total}}} R_{\text{rem}} \right\rfloor, R_{\text{rem}}\right). \tag{8}$$

This proportional allocation ensures **Adaptive Residual Distribution**, allowing clients with higher urgency (hence bids) to receive additional resources, while maintaining fairness.

3) Client Post-Allocation Adaptation: Once the edge orchestrator allocates RBs R_i based on client bids, each client performs a post-allocation adaptation by selecting the most energy-efficient model that satisfies its constraints. Specifically, we enhance the expression of a client's utility as follows:

$$\begin{aligned}\widehat{U}_i(m, x, b_i, b_{-i}) = {} & w_1 \sigma_A(A^m_i) + w_2 \sigma_L(L^{m,x}_i(R_i(b_i, b_{-i}))) \\ & - w_3 \cdot E^{m,x}_i - \gamma b_i^2 \\ & - w_4 \mathbb{I}\left(A^m_i < A_{\min}(h) \vee L^{m,x}_i(R_i(b_i, b_{-i})) > L_{\max}\right),\end{aligned}$$

where b_{-i} denotes the bids of all players other than i and $R_i(b_i, b_{-i}) = R^i_{min} + \Delta R_i(b_i, b_{-i})$.

The system-level performance is then measured via the success rate $\alpha = \frac{1}{|N|} \sum_{i \in \mathcal{N}} \mathbb{I}^i_{\text{success}}$. This expression of the client's utility includes a quadratic bid penalty γb_i^2 to discourage both overbidding and underbidding. This encourages clients to truthfully report their resource needs, equipping design with *Truthful Bidding Incentives*.

Then each client selects the model configuration m^*_i from its feasible set such that:

$$\begin{aligned} m^*_i &= \arg\max_{m \in \mathcal{M}_{\text{fea}}} \widehat{U}_i(m, x, b_i, b_{-i}), \\ \mathcal{M}_{\text{fea}} &= \{m \in \mathcal{M} \mid A^m_i \geq A_{\min}(h),\ L^{m,x}_i(R_i(b_i, b_{-i})) \leq L_{\max}\} \end{aligned}$$

where $\mathcal{M}^i_{\text{fea}}$ is the set of models feasible under the accuracy and latency constraints.

If no feasible configuration exists for the allocated resources, the clients return these unused blocks to the edge orchestrator:

$$R^{\text{final}}_i = \begin{cases} R_i, & \text{if } m^*_i \in \mathcal{M}^i_{\text{fea}}, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

The unclaimed RBs are then unfrozen by the orchestrator and made available to other network slices, thus improving the overall resource utilization.

4) Analysis of CANSA's Properties: The CANSA algorithm enforces rational bidding behavior and stable allocation by design. As highlighted above, its key incentive-aligned components are: (*i*) truthful bidding incentives, (*ii*) priority preservation, and (*iii*) adaptive residual distribution. These components induce a strategic structure where clients' best responses are monotonic, leading to a unique Nash equilibrium, as formally analyzed in the following.

B. Nash Equilibrium Analysis

We first establish the existence of a Nash Equilibrium (NE) and then prove its uniqueness under monotonic best responses. Finally, we evaluate the efficiency of the CANSA discrete allocation mechanism.

1) Existence of a NE via Discontinuous Game Theory: We model the auction as a strategic game $\mathcal{G} = < \mathcal{N}, \mathcal{B}, \widehat{U}_i >$, where $\mathcal{N}$ is the set of players (clients), $\mathcal{B}$ is the set of possible moves (i.e., the collected bids from players defined in (7)), and $\widehat{U}_i(m, x, b_i, b_{-i})$ is the payoff (client's utility) function of player i in (5). For simplicity, in the following we drop from the notation of the clients' bids the dependency on m and x and write $\widehat{U}_i(b_i, b_{-i})$.

A NE occurs when a bid profile $(b_1^*, b_2^*, \ldots, b_N^*)$ for each client $i \in \mathcal{N}$ is such that:

$$\widehat{U}_i(b_i^*, b_{-i}^*) \geq \widehat{U}_i(b_i, b_{-i}^*), \quad \forall b_i \in \mathcal{B}_i\,. \tag{10}$$

Theorem IV.1 (Existence of Nash Equilibrium)**.** *At least one pure-strategy NE exists if:*

1) *Utility functions $\widehat{U}_i(b_i, b_{-i})$ are upper semi-continuous,*
2) *Strategy spaces are finite-dimensional compact sets,*
3) *Utility functions satisfy diagonal transfer quasi-concavity.*

Proof. **Upper semi-continuity**: For any bid sequence $b_i^{(k)} \to b_i$

$$\limsup_{k\to\infty} \widehat{U}_i(b_i^{(k)}, b_{-i}) \leq \widehat{U}_i(b_i, b_{-i}) \tag{11}$$

holds due to the floor function's right-continuity.

Compactness: Strategy spaces $\mathcal{M}$, $\mathcal{X}$, and $\mathcal{R}$ are finite. $R_{min}^i \leq R_{total}$ and $b_i \leq \frac{(R_{total}^2)}{L_{max}}$. Strategy $\mathcal{B}$ is a finite set of bids, hence compact.

Diagonal Transfer Quasi-Concavity: For any bid profiles b, b' if $\widehat{U}_i(b_i', b_{-i}) > \widehat{U}_i(b_i, b_{-i})$, then $\exists \alpha \in (0,1)$ such that:

$$\widehat{U}_i(\alpha b_i' + (1-\alpha) b_i, b_{-i}) \geq \min(\widehat{U}_i(b_i', b_{-i}), \widehat{U}_i(b_i, b_{-i}))$$

Satisfied through the proportional allocation mechanism in Phase 2, guaranteeing quasi-concavity. By Reny's existence theorem [22], a NE exists. □

2) Uniqueness Under Monotonic Best Responses: We now prove the uniqueness of Nash equilibrium by showing that clients' best responses are strictly monotonic.

Theorem IV.2 (Uniqueness of Nash Equilibrium)**.** *If the best-response function $b_i^*(\sum_{j\neq i} b_j)$ of each client $i \in N$ is strictly decreasing, i.e., $\frac{\partial b_i^*(\sum_{j\neq i} b_j)}{\partial \sum_{j\neq i} b_j} < 0$, then the NE is unique.*

Proof. Define the best-response mapping $F(b)$, where each element is $F_i(b_{-i}) = b_i^*(\sum_{j\neq i} b_j)$. By assumption, F is strictly monotonic decreasing. Consider two bid profiles b, b', if $\sum_{j\neq i} b_j > \sum_{j\neq i} b_j'$, then $b_i^*(\sum_{j\neq i} b_j) < b_i^*(\sum_{j\neq i} b_j')$. Thus, F is a contraction mapping on the finite discrete bid space.

By the Banach fixed-point theorem, this contraction guarantees a unique fixed point. Therefore, the NE is unique under the given monotonicity condition. □

The client's utility function $\widehat{U}_i(b_i, b_{-i})$ is differentiable with respect to its bid b_i, yielding the first-order condition for optimality:

$$\frac{\partial \widehat{U}_i(b_i, b_{-i})}{\partial b_i} = \frac{\partial(\sigma_A + \sigma_L)}{\partial R_i(b_i, b_{-i})} \cdot \frac{\partial R_i(b_i, b_{-i})}{\partial b_i} - 2\gamma b_i = 0. \tag{12}$$

This equilibrium condition ensures that the best-response function $b_i^*\left(\sum_{j\neq i} b_j\right)$ is strictly decreasing in the aggregate bids $\sum_{j\neq i} b_j$. As other clients bid higher, the residual resources allocated to client i shrink, reducing $\frac{\partial R_i(b_i, b_{-i})}{\partial b_i}$. The marginal benefit of client i declines of bidding higher and therefore induces a lower optimal bid b_i^*. It follows that our first-order condition yields the monotonicity condition required for uniqueness.

3) Efficiency Analysis with Discrete Allocation: Although bids are continuous, final resource allocations R_i are quantized into discrete RBs. We evaluate the efficiency of this discrete allocation mechanism via the Price of Anarchy (PoA), which is bounded by:

$$\text{PoA} \leq 1 + \frac{N \cdot \Delta R}{R_{total}}, \quad \text{where} \quad \Delta R < 1, \tag{13}$$

with ΔR representing the maximum quantization error per client i, bounded by: $\Delta R \leq \frac{b_i}{\sum_j b_j + \lambda} < 1$. The bounded PoA demonstrates that quantization introduces only minimal inefficiencies, as residual allocation and proportional distribution strategies explicitly constrain ΔR. Thus, the efficiency loss remains minimal in the case of discrete actions.

V. Evaluation Setting

We simulate a system with 60 clients each assigned a random distance in Poisson distribution from the serving base station. These distances reflects a range of channel conditions from poor to excellent closed to real-world settings. Random contexts for each client were generated with uniform latency $L_{\max}$ and accuracy $A_{\min}(h)$ constraints. Clients submit bids through our utility-aware bidding mechanism (see Alg. 1) where bids reflect their ability to meet constraints given their channel quality and context.

Datasets and Dynamic Model Deployment. We use the RADIATE dataset [23], containing data gathered using cameras, radar, and LiDAR sensors to train and evaluate the multi-branch fusion model. This dataset represents a diverse array of weather conditions (i.e., sunny, night, rain, fog, and snow), and various driving scenarios (i.e., parked, urban, motorway, and suburban environments). These conditions present challenges in object detection tasks for autonomous driving. The dataset features annotations for objects detected in radar images, categorizing them into eight classes: cars, vans, trucks, buses, motorbikes, bicycles, pedestrians, and groups of pedestrians. These annotations were then mapped onto data from the camera and LiDAR sensors correspondingly.

We select ResNet architectures with varying depths – ResNet-18, ResNet-50, and ResNet-101 – as backbone models for object detection tasks [24]. The stem refers to the first residual block of each ResNet. The branches correspond to the remaining residual stages after the stem. ResNet-18 consists of four residual stages with a total of 18 layers, while ResNet-50 and ResNet-101 use deeper bottleneck architectures of 50 and 101 layers, respectively. The object detection framework is implemented using Detectron2 [25]. To assess the performance of our object detection models, we employ mean Average Precision (mAP) as performance metric and considered different threshold values.

Channel Model. We incorporate the key channel characteristics from [20] where the Floating Intercept (FI) path loss model is characterized by signal attenuation in a large-scale wireless network. Noise power spectral density P_{noise} = -174 dBm/Hz. Floating Intercept α_{d_0} = 73 dB with the distance range of [1, 100] m. The path loss exponent n = 1.9 and

shadow fading standard deviation χ = 5 dB. This setting is directly derived from [20] and aligned with measured propagation characteristics in open-area deployments at 3.5 GHz for 5G new radio.

VI. Results

We first illustrate the need for optimization by analyzing the trade-offs of dynamic model execution, context-aware inference adaptation, and channel-aware resource allocation under realistic variability across clients.

Next, we evaluate our proposed algorithm CANSA. We benchmark CANSA against three conventional resource allocation strategies: *(i)* Average Allocation, where RBs are evenly divided among all clients without considering channel variations; *(ii)* Proportional Channel-Based Allocation, where RBs are distributed proportionally to clients' instantaneous channel quality; *(iii)* Equal Data Rate Fairness Allocation, which aims to ensure all clients achieve similar data rates. Performance comparisons are made in terms of success rate, resource scalability, distribution of success rates across clients, and fairness quantified by Jain's index.

A. Preliminary Study: Motivation for Context-Aware Adaptation

Adaptive Execution Trade-offs. Fig. 3 illustrates the total inference latency for different sensor modalities and model complexities under three deployment strategies: edge only, split, and full local deployment. The latency includes both stem model inference, data transfer, and branch model inference. Each modality shows a trade-off where increasing model complexities improve accuracy but also lead to a huge rise in latency. Indeed, although fusion models can offer high accuracy, they result in significant communication and computation overheads if they are not adaptively executed. Fig. 4 shows while local execution yields constant latency regardless of the allowed data rate: edge and split strategies are highly sensitive to transmission bandwidth due to their dependence on data transfer time.

This result confirms the observations in [7], which noted that additional sensor modalities do not always lead to proportional accuracy gains, while significantly increasing execution latency. This motivates our design of an adaptive execution framework that dynamically selects the complexity of the model and the deployment strategy based on both the network condition and the input modality. The results in Fig. 3 and 4 emphasize the inefficiency of static inference approaches and motivate the need for joint optimization of model selection and execution strategy to accommodate dynamic network conditions.

Importance of Context-Aware Inference. The informativeness of different sensors varies by context. For instance, in sunny environment, LiDAR may not provide additional value, whereas in snow or fog, vision only models may fail.

Fig. 5 illustrates the number of RBs required to meet context-aware scaled accuracy thresholds (top 10%, 30%, and 50%) across six different environmental contexts, assuming full data offloading to the edge server. The bars reflect the minimum RBs needed to support model configurations that satisfy the required proportion of the achievable accuracy in each context. If multiple configurations satisfy the accuracy constraint, we calculate the smallest RBs needed for the configuration deployed on edge orchestrator assuming each RB can achieve 1 Mbps. Results highlight how sensing conditions and informativeness of modalities (e.g., vision degradation in fog or snow) impact the system's ability to meet performance requirements. Notably, challenging contexts such as fog and snow demand significantly more RBs to satisfy even modest accuracy thresholds, indicating the need for adaptive execution and context-aware inference.

B. Success Rate Comparisons

Channel-Aware Allocation Mechanism Comparison. We compare the performance of the proposed bidding-based allocation approach against our three benchmarks: Average, Proportional, and Fairness, as introduced before. We evaluate our bidding mechanism across clients with different channel conditions. Specifically, we generate clients' locations according to Poisson distribution with parameter $\lambda_{\text{Poisson}}=10^{-8}$. Also, we set the same distance clipping range [1, 100] m.

The results in Fig. 6 reveal that the bidding-based mechanism CANSA outperforms other strategies across all constraint levels, particularly under tight latency budgets. The left panel in Fig. 6 compares the considered policies in terms of success rate α—defined as the percentage of clients satisfying both accuracy and latency constraints. As the latency constraint increases, success rates α generally improve. However, occasional non-monotonicity is observed in other strategies due to resource contention and model selection behavior under relaxed constraints. Compared to the Average, Proportional, and Fairness allocation methods, our approach achieved success rate improvements of up to 52.3% under Poisson clients' distance distribution. These findings demonstrate the robustness of the proposed auction-based allocation, which dynamically adapts to context, channel quality, and constraint tightness.

Resource Allocation Under Varying RBs Budget. The right panel in Fig. 6 focuses on the CANSA solution, illustrating how the successful rate changes as the RB budget varies. As expected, results show that as $L_{\max}$ increases, the success rate improves across all RB budgets. At tight latency constraints, the system is resource-constrained, and additional RBs directly improve client success by enabling faster data transmission. However, as $L_{\max}$ grows, the system becomes latency-limited rather than resource-limited: even clients with modest RB allocations can satisfy their performance constraints without requiring large additional resources.

Success Rate Distribution. The results in Fig. 7 reveal that the bidding-based allocation, CANSA, consistently outperforms all baselines, particularly in adverse conditions such as motorway, fog, and night. In high-visibility scenarios like sunny or rain, even simpler strategies (e.g., average or proportional allocation) achieve relatively competitive performance. However, under challenging conditions such as snow, only

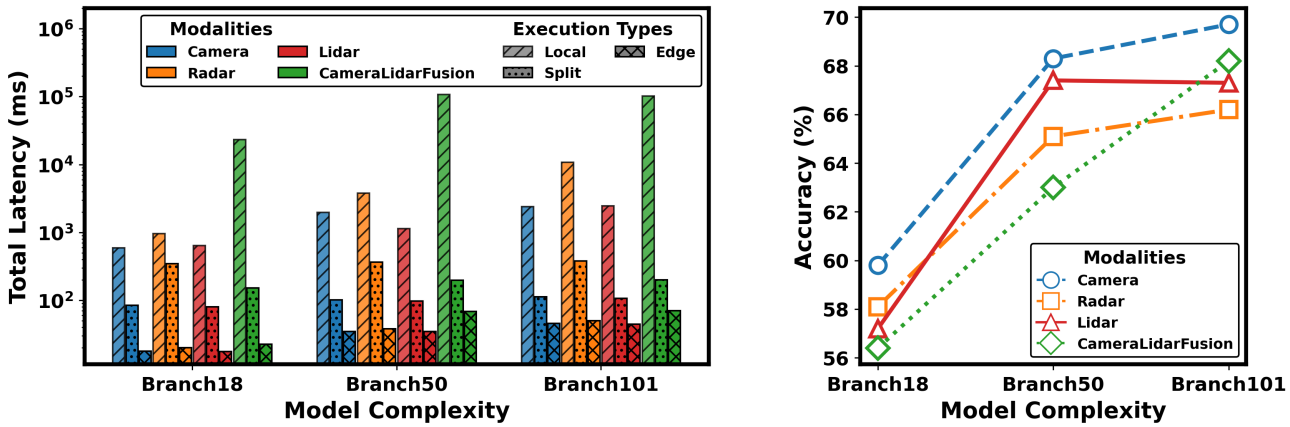

Figure 3. Left: comparison of total latency and model accuracy across four sensing modalities—Camera (blue), Radar (orange), LiDAR (red), Camera-LiDAR Fusion (green)—under three deployment options: Local (solid bars), Split (hatched bars), and Edge (transparent bars). Right: accuracy achieved by these four modalities in sunny context. The x-axis denotes model complexity (Branch18, Branch50, Branch101) in both panels.

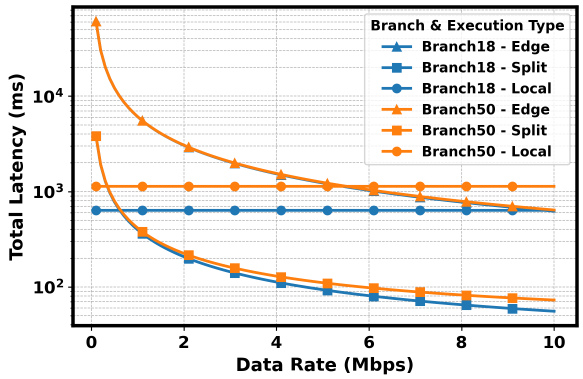

Figure 4. Total inference latency vs. data rate for LiDAR modality under Branch18 and Branch50, comparing local (circle), split (square), and edge (triangle) execution strategies.

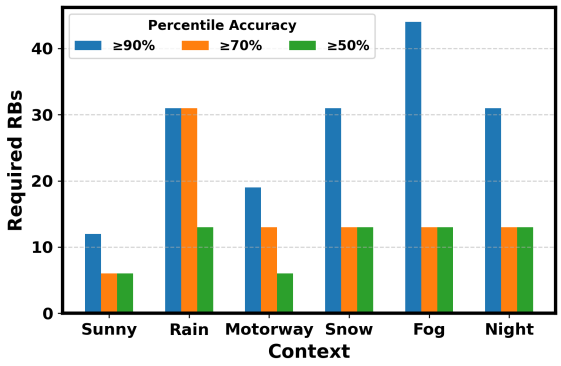

Figure 5. RBs needed to meet top 10%, 30%, and 50% of context-specific accuracy under full data offloading, across six environmental contexts.

CANSA maintains non-zero success, highlighting its ability to prioritize critical users when resources are scarce. *These findings demonstrate that context-unaware strategies, while acceptable in simple environments, often fail under complicated environment. In contrast, our proposed CANSA leverages context and urgency to achieve higher overall utility and fairness.*

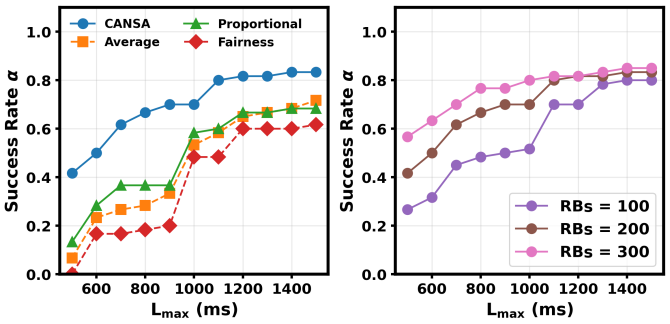

Figure 6. Left: Success rate α under four allocation strategies—CANSA, Average, Proportional, and Fairness—with Poisson-distributed client distances. Right: Resource block usage of the bidding strategy under varying total RB budgets.

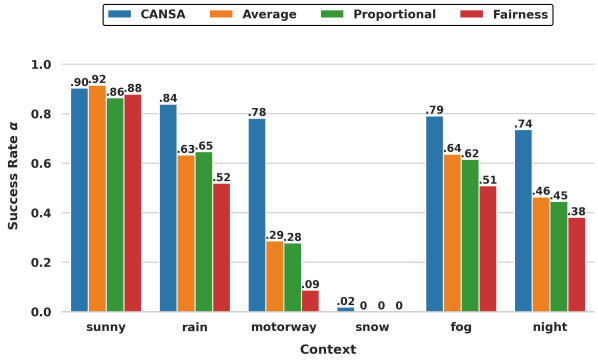

Figure 7. Success rate α across different environmental contexts.

C. Fairness Analysis

Jain's Fairness Index. Fig. 8 depicts the Jain's Fairness Index, a measure of equity in resource utilization, across different latency constraints $L_{\max}$. The fairness metric here is computed based on the actual number of RBs used by each client after the post-allocation adaptation step rather than the initially assigned RBs. This captures the realistic scenario where clients may only use a portion of their allocated RBs due to infeasibility and optimization of their local execution. The average allocation method exhibits the highest fairness

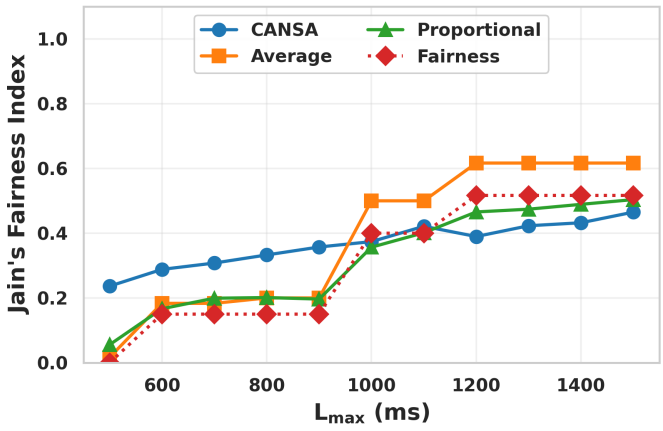

Figure 8. Jain's Fairness Index of actual resource usage after post-allocation adaptation under four allocation strategies as a function of latency constraint $L_{\max}$. Results reflect the fairness of RBs effectively used, not assigned.

in higher $L_{\max}$, as increased tolerance enables more clients to make efficient use of their assignments. In contrast, the bidding mechanism is more effective in maximizing overall success rate but achieves lower fairness in higher $L_{\max}$. This is because the bidding mechanism prioritizes resource efficiency and contextual urgency. These results thus underscore the inherent trade-off between overall system performance and equitable resource use.

VII. Conclusions

We considered a mobile-edge network where multi-modal data collected at mobile devices can be processed with the help of an edge orchestrator, by jointly leveraging dynamic DNN models and split computing. To account for the requirements of the different DNN-based applications inference tasks and the scarcity of computational and network resources, we designed an optimization framework that combines centralized orchestration with mobile devices' personalized game-theoretical strategies. Our method optimizes communication network resources and energy usage, while ensuring low-latency and accurate inference along with real-time response to changes in the operational context. Results, obtained using real-world, multi-modal data, demonstrated that our solution achieves substantial performance improvements, delivering up to 52.3% higher success rates compared to average, proportional and fairness-based methods.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," 2022.

[3] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE Trans. on Neural Networks and Learning Systems*, 2021.

[4] U. M. Gidado, H. Chiroma, N. Aljojo, S. Abubakar, S. I. Popoola, and M. A. Al-Garadi, "A survey on deep learning for steering angle prediction in autonomous vehicles," *IEEE Access*, vol. 8, pp. 163 797–163 817, Aug. 2020.

[5] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark analysis of representative deep neural network architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018.

[6] Y. Matsubara, M. Levorato, and S. Banerjee, "Split computing for complex object detectors," *arXiv:1905.09712*, 2019.

[7] A. V. Malawade, T. Mortlock, and M. A. A. Faruque, "Hydrafusion: Context-aware selective sensor fusion for robust and efficient autonomous vehicle perception," 2022.

[8] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network slicing to enable scalability and flexibility in 5g mobile networks," *IEEE Communications Mag.*, vol. 55, no. 5, pp. 72–79, 2017.

[9] A. Banchs, G. de Veciana, V. Sciancalepore, and X. Costa-Perez, "Resource allocation for network slicing in mobile networks," *IEEE Access*, vol. 8, pp. 214 696–214 706, 2020.

[10] P. Caballero, A. Banchs, G. de Veciana, X. Costa-Pérez, and A. Azcorra, "Network slicing for guaranteed rate services: Admission control and resource allocation games," *IEEE Trans. on Wireless Communications*, vol. 17, no. 10, pp. 6419–6432, 2018.

[11] J. S. Assine, J. C. S. Santos Filho, E. Valle, and M. Levorato, "Slimmable encoders for flexible split dnns in bandwidth and resource constrained iot systems," in *2023 IEEE 24th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. IEEE, Jun. 2023, p. 1–9. [Online]. Available: http://dx.doi.org/10.1109/WoWMoM57956.2023.00014

[12] R. Aljundi, P. Chakravarty, and T. Tuytelaars, "Expert gate: Lifelong learning with a network of experts," 2017.

[13] Y. Li, Y. Chen, N. Wang, and Z. Zhang, "Scale-aware trident networks for object detection," 2019.

[14] L. Snidaro, J. Garcia, and J. Llinas, "Context-based information fusion: A survey and discussion," *Information Fusion*, vol. 25, pp. 16–31, 2015.

[15] Y. Gong, Z. Xiao, X. Tan, H. Sui, C. Xu, H. Duan, and D. Li, "Context-aware convolutional neural network for object detection in vhr remote sensing imagery," *IEEE Trans. on Geoscience and Remote Sensing*, vol. 58, no. 1, pp. 34–44, 2020.

[16] C. Chen, S. Rosa, Y. Miao, C. X. Lu, W. Wu, A. Markham, and N. Trigoni, "Selective sensor fusion for neural visual-inertial odometry," in *CVPR*, 2019.

[17] C. Singhal, Y. Wu, F. Malandrino, M. Levorato, and C. F. Chiasserini, "Resource-aware deployment of dynamic dnns over multi-tiered interconnected systems," in *IEEE INFOCOM*, 2024, pp. 1621–1630.

[18] S. L. Contreras and M. Levorato, "Context-aware heterogeneous task scheduling for multi-layered systems," in *2023 IEEE 24th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2023, pp. 195–204.

[19] C. Singhal, Y. Wu, F. Malandrino, S. L. de Guevara Contreras, M. Levorato, and C. F. Chiasserini, "Resource-efficient sensor fusion via system-wide dynamic gated neural networks," 2024. [Online]. Available: https://arxiv.org/abs/2410.16723

[20] E. I. Adegoke, E. Kampert, and M. D. Higgins, "Channel modeling and over-the-air signal quality at 3.5 ghz for 5g new radio," *IEEE Access*, vol. 9, pp. 11 183–11 193, 2021.

[21] R. Desislavov, F. Martínez-Plumed, and J. Hernández-Orallo, "Trends in ai inference energy consumption: Beyond the performance-vs-parameter laws of deep learning," *Sustainable Computing: Informatics and Systems*, vol. 38, p. 100857, Apr. 2023. [Online]. Available: http://dx.doi.org/10.1016/j.suscom.2023.100857

[22] P. J. Reny, "On the existence of pure and mixed strategy nash equilibria in discontinuous games," *Econometrica*, vol. 67, no. 5, pp. 1029–1056, 1999. [Online]. Available: http://www.jstor.org/stable/2999512

[23] M. Sheeny, E. D. Pellegrin, S. Mukherjee, A. Ahrabian, S. Wang, and A. Wallace, "Radiate: A radar dataset for automotive perception in bad weather," 2021.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[25] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.