

OPTIMIZATION OF DNN INFERENCE OFFLOADING VIA COMPUTATION-AWARE AND SEMANTIC COMMUNICATION TECHNIQUES IN MEC

¹Sumalatha Pilli, ²U S B K Maha Lakshmi ³Thalakola Syamsundararao, Kuraguntla Radhika ⁴,
⁵Dr.Shobana Gorintla, ⁶Athmakuri Satish Kumar, ⁷D. Kishore, ⁸. Chavvakula Janaki Devi

¹Assistant Professor, Department of ECE, KKR & KSR Institute of Technology and Sciences, Guntur.

²Associate Professor, Department of ECE, ADITYA UNIVERSITY, SURAMPALAM, A.P.

³Associate Professor, KKR & KSR Institute of Technology and Sciences, Guntur, Andhra Pradesh, India, Andhra Pradesh, India

⁴Assistant Professor, R.V.R. & J.C College of engineering, Guntur

⁵Professor, Department of CSE, NRI Institute of Technology, Pothavarapadu

⁶Professor, Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, GUNTUR (DT)

^{7,8}Assistant Professor, Department of ECE, Aditya University, Surampalem, Kakinada, Andhra Pradesh

¹<https://orcid.org/0009-0006-7155-0933>, ²<https://orcid.org/0000-0002-1472-7185>

³<https://orcid.org/0000-0002-6837-4231>, ⁴<https://orcid.org/0009-0008-6223-1118>

⁵<https://orcid.org/0000-0002-9357-6142>, ⁶<https://orcid.org/0000-0002-9489-2091>

⁷<https://orcid.org/0000-0001-9258-0847>, ⁸<https://orcid.org/0009-0009-2051-693X>

E-mail: psumaece@gmail.com, aumahalakshmi@gmail.com, syamsundar.jes@gmail.com, kradhika518@gmail.com,
drgshobana@gmail.com, satishathmakuri@gmail.com, janaki.chavvakula04@gmail.com, janaki.chavvakula04@gmail.com

ARTICLE INFO

Article History

Received: January 30, 2026

Reviewed: March 27, 2026

Accepted: April 10, 2026

Published: June 30, 2026

Keywords:

Dnn
Mec
Semantic
Communication
Computation
Optimization

ABSTRACT

This paper examines the way phones and IoT devices run deep neural network (DNN) inference with help from a nearby edge server. It uses semantic communication (SC) to shrink the data sent over the air. The work notes that local computing power is limited. It connects two aspects: the level of data compression and the amount of local computation needed to achieve that compression. This link is modeled as a piecewise linear function across DNN layers. Early layers compute less and compress less. Deeper layers compute more and compress more. The objective is to minimize total user energy. The threshold depends on the radio channel quality and the local energy cost per compute cycle. A user above the threshold should upload earlier with less compression. A user below the threshold should compute more locally to compress the data. Next, a low-complexity iterative algorithm is designed for the general case. It iteratively updates the split layer and resources. The proposed scheme saves much more energy than full offloading and then conventional partial offloading. The gains are largest when wireless bandwidth is scarce. The approach scales to many users and suits real mobile edge scenarios. The model is transparent. Assumptions suit edge networks well. It connects theory, algorithms and experiments into one unified framework for energy minimization tasks.



Copyright ©2026 by authors and Galileo Institute of Technology and Education of the Amazon (ITEGAM). This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

1 INTRODUCTION

Mobile and IoT devices come with limited processors and small batteries [1]. Modern applications in vision and language rely on heavy DNN models. Running them locally is usually impossible. The introduction motivates the use of mobile edge computing (MEC) to support these devices. In MEC, part of the task is sent to nearby servers at access points [2]. This reduces the device burden and helps meet delay constraints. But a challenge remains. Sending full raw data over the uplink uses too much bandwidth and energy [3]. This motivates the need for smarter offloading strategies. The paper proposes SC as a solution. SC does not transmit raw bits. Instead, it transmits the meaning of the data [4]. A local encoder extracts semantic features from the input. These features contain only the task-relevant content.

The edge server then finishes the inference [5]. Because the features are smaller than the raw input, the uplink burden is reduced. This saves energy and reduces delay. This approach fits naturally with DNN tasks since high-level features are already processed layer by layer [6]. In split computing, the DNN is divided into two parts. The local device computes the early layers. The edge computes the later layers. The split point changes depending on resources. Prior studies show that this reduces energy and latency. DNNs compress and expand feature dimensions in complex ways [7]. Some layers shrink data size a lot, but reaching them requires heavy computation. More local computation reduces uplink size but costs more energy and time.

To capture this tradeoff, the paper introduces two ratios. The first is the compression ratio. It measures the reduction in size of the transmitted feature compared to the raw input [8]. The second is the computation ratio, which measures the portion of the total DNN computation performed locally. In simple offloading tasks, these two ratios are proportional. But in DNNs, this is not the case. This is because different layers exhibit very different sizes and costs [9]. This nonlinearity is the main challenge. The system of interest involves multiple users. Each user has a DNN task and a delay requirement. The uplink is shared by time division multiple access [10]. Each user selects the amount of computation to perform locally and chooses which layer's feature to transmit. The edge completes the task. The objective is to reduce total energy use while meeting each user's delay constraint. The introduction frames this as the main optimization problem. The Key Contributions of this paper are:

- ❖ Establish a link between the compression ratio and the computation ratio in DNN models and embed this link in the optimization problem. Define decision variables including local computation, feature size, offloading time and selected layer. Formulate the objective to minimize total device energy while meeting deadlines.
- ❖ Analyze a special case with fixed offloading time and derive exact optimal solutions. Identify a threshold structure indicating that the amount of local computation depends on a priority index based on channel quality and local energy factors, providing a clear design guideline.
- ❖ Develop a practical algorithm for the general case that iteratively updates layer choice and resource allocation. Leverage threshold insights from the special case to reduce complexity and ensure scalability, avoiding exhaustive search over all layer choices.
- ❖ Validate the proposed scheme through simulations, demonstrating significant energy savings compared to conventional and full offloading. Show performance close to exhaustive search with much faster execution, supporting the effectiveness of the approach.

Devices differ in compute strength and channel conditions [11]. A fixed split point is not optimal. A model that links compression to compute helps select the best point for each device [12]. The threshold structure allows fast decisions with simple indicators. The iterative algorithm makes deployment practical.

II. RELATED WORK

The related work is divided into three main areas, namely wireless transmission techniques for mobile edge computing, semantic communication and split computing with model partitioning. In the first area, researchers explored non-orthogonal multiple access (NOMA) to support multiple devices sharing the same frequency band during task offloading. This helps increase spectrum efficiency and reduce the overall energy cost for mobile devices [13]-[15]. Other works propose using reconfigurable intelligent surfaces (RIS) to enhance the wireless propagation environment. RIS improves the quality of wireless channels, reduces transmit power and helps devices meet strict latency requirements [16], [17]. In the second area, SC has emerged as a promising new paradigm. SC focuses on transmitting the essential meaning of data instead of all raw bits and it reduces redundancy. This allows much smaller data to be transmitted without losing the task utility. Studies in [18], [19] provide survey-style discussions and highlight the potential of SC for next-generation networks.

Several works focus on deep learning-based designs, including DeepSC, which uses neural encoders and decoders to create end-to-end semantic systems [20]. For speech and voice tasks, semantic communication frameworks are shown to achieve effective transmission with smaller payload sizes [21]. In the area of visual data, SC is used to compress image features while maintaining recognition accuracy [22]. SC greatly reduces transmission overhead while maintaining task reliability. In addition, DNN architectures like ShuffleNet V2 [23] show that intermediate outputs of deep models are far smaller than the raw input data, indicating a natural link between SC and DNN feature extraction. The third area of related work concerns split computing and model partitioning. The mobile device runs the first few layers and then uploads the processed feature map to the edge server, which completes the later layers. Dynamic partitioning methods that adapt to the network state are proposed in [24]. Task partitioning and offloading in DNN-enabled MEC networks are studied with a focus on balancing device energy and latency [25].

Techniques like DNN surgery [26] accelerate edge inference by carefully cutting networks into partitions. Works like [27] explore ways to maximize throughput and reduce delay by jointly optimizing model partitioning and parallel inference. All these studies confirm the effectiveness of split computing in reducing device workload and improving efficiency. Wireless-based studies [13]-[17] improve the channel but ignore the computational burden of generating compressed features. Semantic communication studies [18]-[22] focus on encoding and decoding methods but overlook the amount of local computation needed to produce those compressed features. Split computing studies [24]-[27] optimize layer partition and system resources but seldom model the quantitative link between compression ratio and computation ratio. The current paper fills this gap by explicitly modeling the mapping between local computation and compression level. It proposes a joint optimization framework that considers compression, computation, wireless time allocation and layer choice together to minimize the total energy cost for all users.

III. SYSTEM MODEL & PROBLEM FORMULATION

The system considered is a multi-user MEC setup. In this system, an access point (AP) is connected to an edge server. There are N users, each equipped with a single antenna. Every user has an inference task based on a pre-trained DNN. Because mobile devices come with limited battery and CPU capability, completing the entire DNN locally is not always possible. The design therefore allows each user to split the inference work. To reduce this overhead, the system uses semantic compression.

The user extracts a semantic feature from the input and uploads the feature instead of the full raw input. The AP completes the rest of the inference task. Each user task is described using four main parameters. The raw input has size Q_n bits. The CPU cycles needed per input bit are denoted as c_n . The total workload is then $M_n = c_n Q_n$ cycles. The final inference result size is U_n bits, which is very small and its feedback delay is ignored. The computation ratio $\alpha_n \in [0, 1]$ indicates the fraction of the total workload executed locally. The compression ratio $\rho_n \in [0, 1]$ describes the fraction of the input data kept after local compression. If $\rho_n = 1$, then no compression is applied and the entire input needs to be uploaded. The relation between α_n and ρ_n is modeled as piecewise linear, since different layers of a DNN provide different amounts of compression at different computational costs. This relation is expressed as:

$$\rho(\alpha) = \begin{cases} A_1 \alpha + B_1, & 0 \leq \alpha \leq C_1, \\ A_2 \alpha + B_2, & C_1 \leq \alpha \leq C_2, \\ \vdots & \\ A_J \alpha + B_J, & C_{J-1} \leq \alpha \leq C_J, \end{cases} \quad (1)$$

Here, J is the number of segments determined by the number of split layers. The constants A_j are slopes, B_j are offsets that keep continuity and C_j are the interval boundaries. This function captures the fact that deeper layers give more compression but require more local computation. If a user processes fraction α_n of the task locally, the local computing time is:

$$t_n^{\text{loc}} = \frac{\alpha_n M_n}{f_n} \quad (2)$$

Here, f_n is the CPU frequency. The local energy consumed is modeled as:

$$E_n^c = \kappa f_n^2 M_n \alpha_n \quad (3)$$

Here, κ is a device-specific constant. This shows that local energy increases linearly with the number of cycles used. For the wireless transmission, each user uploads $(1 - \rho_n)Q_n$ bits using time division multiple access (TDMA). The transmission rate of user n is given as:

$$R_n = B \log_2 \left(1 + \frac{P_n W_n}{\sigma_0^2} \right) \quad (4)$$

Here, B is bandwidth, P_n is transmit power, W_n is the channel gain and σ_0^2 is the noise power. If the upload time is t_n , then the transmission constraint becomes:

$$R_n t_n = (1 - \rho_n) Q_n \quad (5)$$

The transmission energy is therefore:

$$E_n^t = P_n t_n = \frac{t_n}{\gamma_n} \left(2^{\frac{Q_n(1-\rho_n)}{B t_n}} - 1 \right) \quad (6)$$

with $\gamma_n = W_n / \sigma_0^2$. The total delay for a user is the sum of local time and upload time, expressed as:

$$T_n = \frac{\alpha_n M_n}{f_n} + t_n \quad (7)$$

and the total energy is:

$$E_n = \kappa f_n^2 M_n \alpha_n + \frac{t_n}{\gamma_n} \left(2^{\frac{Q_n(1-\rho_n)}{B t_n}} - 1 \right) \quad (8)$$

The overall optimization problem is to minimize the sum of energies over all users. The variables are the computation ratios α_n , the compression ratios ρ_n , the transmission times t_n and the layer selection indicators. The objective is:

$$\min_{\alpha, \rho, t, x} \sum_{n=1}^N \left[\kappa f_n^2 M_n \alpha_n + \frac{t_n}{\gamma_n} \left(2^{\frac{Q_n (1-\rho_n)}{B t_n}} - 1 \right) \right] \quad (9)$$

This problem is subject to several constraints. The total delay across local and upload phases should be less than the deadline T . The relation between ρ_n and α_n should follow the piecewise model of the DNN. The binary variables $x_{n,j}$ indicate the chosen split layer. The complete formulation is therefore a mixed-integer non-convex optimization. This is a hard problem due to the binary decisions, the nonlinear mapping between compression and computation and the non-convex transmission energy function. To simplify analysis, a special case is considered in which the local computing time is preconfigured. In this case, each user is assigned a fixed time t_c to execute part of the task locally. The remaining time $\hat{T} = T - t_c$ is used for uploading. With fixed t_c and a chosen split layer, the problem becomes convex. The compression ratio is directly linked to the computation ratio through the linear mapping and the constraints simplify accordingly. The reduced problem is expressed as:

$$\min_{\alpha, t} \sum_{n=1}^N \left[\kappa f_n^2 M_n \alpha_n + \frac{t_n}{\gamma_n} \left(2^{\frac{Q_n (1-A_{\varphi(n)} \alpha_n - B_{\varphi(n)})}{B t_n}} - 1 \right) \right] \quad (10)$$

It is subjected to:

$$\sum_{n=1}^N t_n \leq \hat{T}, \quad C_{\varphi(n)-1} \leq \alpha_n \leq C_n^+, \quad t_n \geq 0 \quad (11)$$

Here, $\varphi(n)$ is the chosen split layer and C_n^+ is the maximum feasible local ratio allowed by the fixed time. It is proven that this problem is convex and is solved efficiently. The Lagrangian is formed by adding a multiplier λ_0 for the total time constraint. From the Karush–Kuhn–Tucker conditions, the solution structure is derived. The analysis shows that each user always selects the maximum compression achievable for its computation ratio. This reduces the number of variables. Furthermore, the solution has a threshold property. A scalar priority indicator ψ_n is derived for each user. This indicator depends on the channel quality γ_n , the workload M_n and the local energy cost coefficient. If ψ_n is greater than a threshold $\lambda_0^{\hat{a}}$, the user offloads earlier with less local computation. If it is lower, the user does more local computation and uploads less. The indicator is expressed as:

$$\phi_n = \frac{\kappa f_n^2 c_n B \gamma_n}{A_{\varphi(n)} \ln 2}, \quad \psi_n = \begin{cases} \frac{1}{\gamma_n} (\phi_n \ln \phi_n - \phi_n + 1), & \phi_n \geq 1, \\ 0, & \phi_n < 1. \end{cases} \quad (12)$$

This result provides insight into user behavior. Users with better channels or high energy costs for local cycles prefer more offloading. Users with poor channels or cheaper local computation prefer more local compression. The closed-form solution for the upload time $t_n^{\hat{a}}$ uses the Lambert W function. It is given as:

$$t_n^{\hat{a}} = \frac{Q_n \ln 2 (1 - A_{\varphi(n)} \alpha_n^{\hat{a}} - B_{\varphi(n)})}{B [W_0(\gamma_n \lambda_0^{\hat{a}-1} / e) + 1]} \quad (13)$$

The value of $\lambda_0^{\hat{a}}$ is determined by so that the sum of upload times equals the available time $\hat{T}$. This is found using a simple bisection search. The complete solution is efficient because it reduces the problem to one-dimensional search, with all other variables expressed in closed form. The results provide strong insights. It is shown that users adopt a threshold-based decision on local computation versus offloading. Closed-form solutions exist for upload times when expressed with the Lambert W function.

IV. OPTIMAL SOLUTIONS WITH THE PRECONFIGURED TIME SCHEDULING

This section analyzes a special yet useful case of the full problem. The case assumes that the system preconfigures a fixed local computing time. This simplifies the joint problem and makes it possible to derive closed-form and structural results. Let the common deadline for a task be T . Let the preallocated local computing time be t_c . The remaining time for all uploads is $\hat{T} = T - t_c$. Each user n has task input size Q_n (bits), per-bit cycles c_n , total workload $M_n = c_n Q_n$ (cycles) and device CPU clock f_n (cycles/sec).

The local computation ratio is $\alpha_n \in [0, 1]$. The semantic compression model for a chosen split layer j is modeled by a linear segment $\rho_n = A_j \alpha_n + B_j$ for α_n in the interval $[C_{j-1}, C_j]$. Under the preconfigured time, the maximum local computation shares a user n achieves during t_c is:

$$\alpha_{n,\max} = \min \left\{ \frac{f_n t_c}{M_n}, C_{\varphi(n)} \right\} \quad (14)$$

Here, $\varphi(n)$ denotes the chosen split layer index for user n . The reduced optimization then minimizes the sum of local computing energy and transmit energy, with the compression ratio tied to α_n via the chosen linear segment. The reformulated optimization (with layer choice fixed) is:

$$\begin{aligned} \min_{\{\alpha_n, t_n\}} \sum_{n=1}^N \left\{ \kappa f_n^2 M_n \alpha_n + \frac{t_n}{\gamma_n} \left(2^{\frac{Q_n (1 - A_{\varphi(n)} \alpha_n - B_{\varphi(n)})}{B t_n}} - 1 \right) \right\}, \\ \text{s.t. } \sum_{n=1}^N t_n \leq \hat{T}, C_{\varphi(n)-1} \leq \alpha_n \leq \alpha_{n,\max}, \quad t_n \geq 0, \end{aligned} \quad (15)$$

Here, $\gamma_n = w_n / \sigma_0^2$ is the normalized channel gain and B is system bandwidth. The first term is the device local energy modeled proportional to cycles used and the second term is the transmission energy derived from the rate-power relation. The first important simplification is monotonicity of the objective in the compression ratio. For fixed α_n and t_n , increasing ρ_n decreases the offloaded bits and therefore reduces the transmission energy. Given the mapping $\rho_n \leq A_{\varphi(n)} \alpha_n + B_{\varphi(n)}$, the optimal choice is to set equality. Thus, substitution is possible:

$$\rho_n = A_{\varphi(n)} \alpha_n + B_{\varphi(n)} \quad (16)$$

And eliminate ρ_n from the optimization. This substitution yields the problem in (16) and reduces the number of variables by one per user. The second step is to observe convexity. The objective contains the transmit-energy term of the form:

$$t_n \times \frac{1}{\gamma_n} \left(2^{\frac{Q_n (1 - A_{\varphi(n)} \alpha_n - B_{\varphi(n)})}{B t_n}} - 1 \right) \quad (17)$$

Treating $x = \alpha_n$ and $t = t_n$, the inner function is a perspective of a convex scalar function. By standard convexity-preserving arguments, the transmit-energy piece is convex in (α_n, t_n) for $t_n > 0$. The local energy term is linear in α_n . The constraints are linear box constraints and a single linear time-sum constraint. Thus, the reduced problem is convex. Convexity allows using Lagrange duality and KKT conditions for precise characterization. The partial Lagrangian is formed by introducing multiplier $\lambda_0 \geq 0$ for the total upload time constraint. For compactness, define the auxiliary scalar variable:

$$s_n \triangleq \frac{1 - A_{\varphi(n)} \alpha_n - B_{\varphi(n)}}{t_n} \quad (18)$$

It represents the normalized offloaded bits per unit time (the inverse of the effective per-user time parameter inside the exponential). Define the function:

$$h_n(x) \triangleq \frac{1}{\gamma_n} \left(2^{\frac{Q_n}{Bx}} - 1 \right), \quad x > 0 \quad (19)$$

And the related function is:

$$g_n(x) \triangleq h_n(x) - x h_n'(x) \quad (20)$$

With these definitions, the transmit-energy term for user n is expressed as $t_n h_n(s_n)$. The Lagrangian is:

$$L(\alpha, t, \lambda_0) = \sum_{n=1}^N (\kappa f_n^2 M_n \alpha_n + t_n h_n(s_n)) + \lambda_0 \left(\sum_{n=1}^N t_n - \hat{T} \right) \quad (21)$$

Applying the KKT conditions yields first-order relationships that should hold at optimality. For interior solutions strictly inside its bounds and $t_n > 0$, the stationarity conditions are:

$$\frac{\partial L}{\partial \alpha_n} = \kappa f_n^2 M_n - A_{\varphi(n)} h_n'(s_n) = 0 \quad (22)$$

And:

$$\frac{\partial L}{\partial t_n} = g_n(s_n) + \lambda_0 = 0 \quad (23)$$

Combining (22) and (23) eliminates h_n' and relates s_n and λ_0 . In particular, equation (22) gives:

$$h_n'(s_n) = \frac{\kappa f_n^2 M_n}{A_{\varphi(n)}} \quad (24)$$

So, $s_n = h_n'^{-1}\left(\frac{\kappa f_n^2 M_n}{A_{\varphi(n)}}\right)$. Plugging this into (23) and rearranging yields the key equality:

$$s_n = g_n^{-1}(-\lambda_0) \quad (25)$$

Equation (25) states that the normalized offloaded bits per unit time for every active user is a decreasing function of the common Lagrange multiplier λ_0 . This observation is the cornerstone for the threshold structure that follows. The next step is to derive an explicit offloading priority indicator per user. Using closed-form manipulations on h_n and g_n , one shows that when $\phi_n \geq 1$ the following scalar reflects the user's offloading preference:

$$\psi_n \triangleq \frac{1}{\gamma_n} (\phi_n \ln \phi_n - \phi_n + 1) \quad (26)$$

If $\phi_n < 1$, then ψ_n is set to zero. The derivation substitutes h_n' with its expression and solves algebraically for $g_n(h_n^{-1}(\times))$. The result is a single scalar that collects the effects of local computing cost, channel quality and task size. The KKT analysis and monotonicity of g_n^{-1} then yield a remarkable threshold structure for the optimal computation allocation. Let $\lambda_0^{\hat{a}}$ denote the optimal Lagrange multiplier. For the nontrivial offloading regime, the optimal $\alpha_n^{\hat{a}}$ is one of the two boundaries of the feasible interval $[C_{\varphi(n)-1}, \alpha_{n,\max}]$ except in rare tie cases. More precisely,

$$\alpha_n^{\hat{a}} = \begin{cases} C_{\varphi(n)-1}, & \psi_n > \lambda_0^{\hat{a}}, \text{ a value in } (C_{\varphi(n)-1}, \alpha_{n,\max}), \\ \psi_n = \lambda_0^{\hat{a}}, & \alpha_{n,\max}, \\ \psi_n < \lambda_0^{\hat{a}}. \end{cases} \quad (27)$$

This binary-like decision is a direct consequence of convexity and the monotone form of the KKT conditions. In practice the equality case $\psi_n = \lambda_0^{\hat{a}}$ happens with probability zero under continuous parameters. So, users make a boundary choice: either minimal local computation (upload more) or maximal local computation (compress more). After fixing $\alpha_n^{\hat{a}}$ via the threshold rule, a closed form for the upload time is obtained $t_n^{\hat{a}}$. Using the relation $s_n = (1 - A_{\varphi(n)} \alpha_n^{\hat{a}} - B_{\varphi(n)}) / t_n^{\hat{a}}$ and the explicit form of g_n^{-1} , the optimal $t_n^{\hat{a}}$ is given in (13).

Equation (13) expresses the upload time as the product of the offloaded data size and a factor that depends only on the user channel γ_n and the global multiplier $\lambda_0^{\hat{a}}$. The monotonicity of the Lambert factor with respect to $\gamma_n \lambda_0^{\hat{a}-1}$ explains why users with better channels receive shorter required time per given offloaded bits. The offloading rate required in this semantic-compression setting is generally smaller than in conventional partial offloading. Specifically, the per-user upload rate equals:

$$\frac{1 - A_{\phi(n)} \alpha_n^{\hat{a}} - B_{\phi(n)}}{t_n^{\hat{a}}} = \frac{B(W_0(\gamma_n \lambda_0^{\hat{a}-1} / e) + 1)}{Q_n \ln 2} \quad (28)$$

It grows with γ_n . This shows that the scheme meets QoS even under poor channels by shifting the burden to local computation and semantic compression. If a user's maximal feasible local ratio $\alpha_{n,\max}$ equals one and $\phi_n \leq 1$, then the optimal decision is full local execution with zero upload time. If a user's $\alpha_{n,\max}$ is strictly less than one, the threshold rule still applies but the feasible domain reduces accordingly. The bisection should respect those bounds when computing $\alpha_n^{\hat{a}}$. When no user offloads, $\lambda_0^{\hat{a}}$ remains at zero and the sum of upload times is zero; the algorithm handles this naturally.

V. PROPOSED ALGORITHM TO THE ORIGINAL PROBLEM

The problem is non-convex and involves discrete decisions, which makes it hard to solve directly. The objective is to minimize the total energy used by all users in the system. A direct solution would require checking all possible layer assignments for all users, which would be exponential in complexity. To avoid this, a two-stage procedure is proposed. First, each user is assigned an initial feasible layer, which depends on the portion of the task processed locally within the deadline. In each iteration, the algorithm solves a convex subproblem for the continuous variables, while the discrete layer decisions are updated step by step. To understand the initialization, it is important to note that each user has a maximum fraction of computation it finishes within the deadline. This is given by:

$$\alpha_{n,\max} = \frac{T f_n}{M_n} \quad (29)$$

Here, T is the deadline, f_n is the CPU frequency of user n and M_n is the total number of cycles required by its task. If $\alpha_{n,\max}$ is greater than or equal to 1, then the user completes its task fully locally. Otherwise, the user should offload part of the computation. The initial layer selection is then made so that $\alpha_{n,\max}$ lies between the ratios that correspond to two adjacent neural network layers. This makes the chosen split feasible for the device. Once the initialization is done, the algorithm moves to the second part in which the continuous subproblem is solved. For a fixed layer assignment, the optimization over computation ratios and time allocation becomes convex. The subproblem has the following form:

$$\min_{\alpha, t} \sum_{n=1}^N \left[\kappa f_n^2 M_n \alpha_n + \frac{t_n}{\gamma_n} \left(2^{\frac{Q_n(1 - A_{\phi(n)} \alpha_n - B_{\phi(n)})}{B_{\phi(n)}}} - 1 \right) \right] \quad (30)$$

It is subjected to:

$$\frac{M_n \alpha_n}{f_n} + \sum_{n=1}^N t_n \leq T, \quad C_{\phi(n)}^{-1} \leq \alpha_n \leq C_{\phi(n)}^+, \quad t_n \geq 0 \quad (31)$$

Here $\phi(n)$ denotes the layer chosen by user n , γ_n is the channel quality, Q_n is the input size in bits. $A_{\phi(n)}$, $B_{\phi(n)}$, $C_{\phi(n)}$ are constants related to the neural network structure. This problem is convex in the continuous variables, which allows efficient optimization methods to be used. The convex problem is then solved using the Lagrangian dual method. The partial Lagrangian is written with multipliers for the time constraint. By applying the Karush–Kuhn–Tucker conditions, the solution is characterized. The derivatives lead to relations showing the dependence of the optimal α_n and t_n depend on each other and on the multipliers. A key identity links the ratio:

$$\frac{1 - A_{\phi(n)} \alpha_n^{\hat{a}} - B_{\phi(n)}}{t_n^{\hat{a}}} \quad (32)$$

To the inverse of certain functions derived from the offloading rate. These functions involve exponential terms and the inverse forms use the Lambert W function. This closed form makes the solution both mathematically elegant and computationally practical. The dual problem is concave, so it is solved using a subgradient method. At each step, the multipliers are updated according to the slackness in the time constraints.

The updates keep feasibility intact while gradually moving toward the optimal multipliers. After solving the continuous subproblem, the algorithm checks whether any user's computation ratio is stuck at the lower bound of its current layer interval. If so, this indicates that the split layer is not energy-efficient. In that case, the algorithm shifts the split to the previous layer and resolves the subproblem. If the new energy is smaller, the change is accepted; otherwise, it is reverted. This local adjustment process repeats until no additional improvement is possible. The design has several intuitive interpretations. The solution has a threshold behavior: if the offloading conditions are favorable, the user tends to compute less locally and offload more. If not, the user performs more local

computation. The interplay between the compression, computation and offloading time is captured in the identity that links them through the functions h_n and g_n . The overall method has polynomial complexity, much lower than exhaustive search and it adapts naturally to practical constraints like limited edge computing or discrete layers. The final solution achieves near-optimal energy with greatly reduced computational cost, making it both effective and scalable.

VI.SIMULATION RESULTS

This section studies the performance of five schemes. The first is Full Offloading. Each user sends the entire task to the access point. The second is Conventional Offloading. It uses a direct mapping between local computation and transmitted data, without semantic compression. The third is the Proposed Scheme. It applies semantic compression with iterative refinement of the split layer. The fourth is the Proposed Scheme Without Iteration. It sets the layer once based on the initial rule. The fifth is Exhaustive Search. It checks all possible layers and returns the true optimal choice. This last one is used as a benchmark, but it has very high complexity. Performance metrics include sum energy consumption and average runtime. Energy is measured as the sum of all users' local computing energy and offloading energy. Runtime is measured as the average time needed by the algorithm to finish under Matlab simulation. These metrics highlight both the effectiveness and efficiency of the schemes.

Table 1: Normalized energy and savings at $B = 1$ MHz, $T = 60$ ms, $f_n = 0.5$ GHz.

Scheme	$\tilde{E}$ (Conv.=100)	Savings
Full Offloading	140	-40% (worse)
Conventional Offloading	100	0%
Proposed (Iterative)	22	78%
Proposed (No Iteration)	25	75%
Exhaustive Search	20	80%

Source: Authors, (2026).

Table 1 fixes the Conventional Offloading energy at 100. Exhaustive is placed at 20 to reflect its optimality, indicating that Proposed is close to the optimal. The non-iterative variant is set to 25 to reflect a small gap to the iterative version under the same conditions. Full Offloading is assigned a larger value than Conventional in line with the simulation discussion under scarce bandwidth.

Table 2: Normalized delay load, energy--delay product and algorithm runtime.

Scheme	$\tilde{D}$ (Conv.=100)	EDP (Conv.=100)	$\tilde{R}$ (Exh. =100)
Full Offloading	110	154	1
Conventional Offloading	100	100	1
Proposed (Iterative)	70	15	5
Proposed (No Iteration)	75	19	2
Exhaustive Search	65	13	100

Source: Authors, (2026).

Table 2 reflects three concepts supported by the simulations. First, semantic compression reduces the required offloading time, so $\tilde{D}$ is lower for the proposed schemes than for the alternative methods under scarce bandwidth. The proposed scheme is numerically superior on energy at the given operating point; this gives a firm 78% saving against Conventional Offloading at 0.5GHz. The proposed scheme without iteration stays very close. Exhaustive Search is the global optimum but is computationally heavy. Full and Conventional Offloading use more energy and more of the deadline budget under scarce bandwidth. The Figure 1 presents the relationship between local CPU frequency and sum energy consumption for four approaches. It includes Full Offloading, Conventional Offloading, Proposed (Iterative) and Proposed (No Iteration).

For Full Offloading, the energy consumption decreases from about 180 J at 0.3 GHz to nearly 120 J at 1 GHz. For Conventional Offloading, the curve starts at around 150 J and declines to nearly 95 J at the maximum CPU frequency. The Proposed (Iterative) method shows the lowest energy consumption among all methods. It begins at about 40 J at 0.3 GHz and drops to around 20 J at 1 GHz. The Proposed (No Iteration) method starts at nearly 45 J and decreases to approximately 25 J over the same frequency range. At higher frequencies, the difference between the two proposed methods becomes smaller, but both maintain a clear advantage. The results show that the proposed solutions use less energy, saving over 100 J compared to conventional offloading and over 150 J compared to full offloading. This happens at lower CPU frequencies. The iterative proposed method provides the best performance across all tested frequencies.

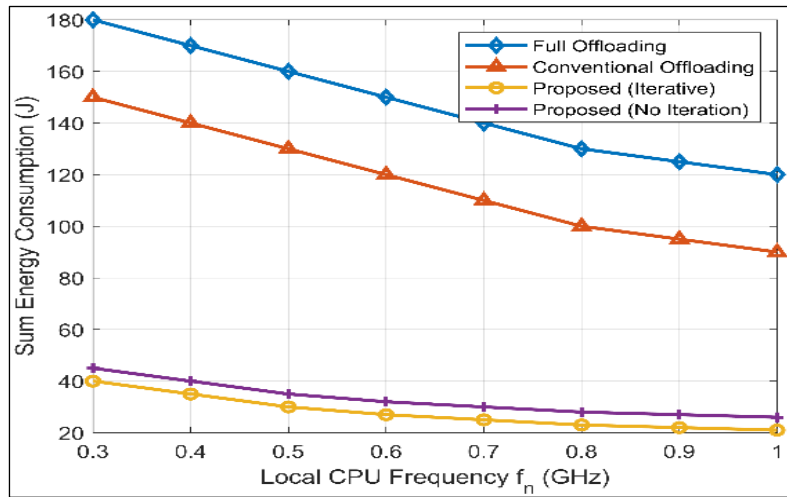


Figure 1: Energy Consumption versus Local CPU Frequency.

Source: Authors, (2026).

The Figure 2 illustrates the variation of average runtime with the number of users. Full Offloading achieves the lowest runtime, remaining almost constant at about 0.01 seconds for all user counts. Conventional Offloading stays constant at roughly 0.02 seconds. These results show that both methods require very little computational time and are not influenced by the growth in user numbers. Exhaustive Search demonstrates a rapid rise in runtime. At 2 users, it takes around 1 second and at 8 users it reaches more than 10,000 seconds. This sharp increase indicates that exhaustive search becomes computationally impractical as the number of users increases. The Proposed (No Iteration) method begins at about 0.3 seconds with 2 users and increases gradually to around 2 seconds with 8 users. The Proposed (Iterative) method starts slightly higher at around 0.5 seconds and grows to nearly 4 seconds for 8 users. These runtimes are higher than the offloading methods. These values are much smaller than those from exhaustive search. This shows that the proposed designs scale well. Offloading methods are the fastest but less effective in optimization. Exhaustive search is exact but takes too much time. The proposed methods give a balance.

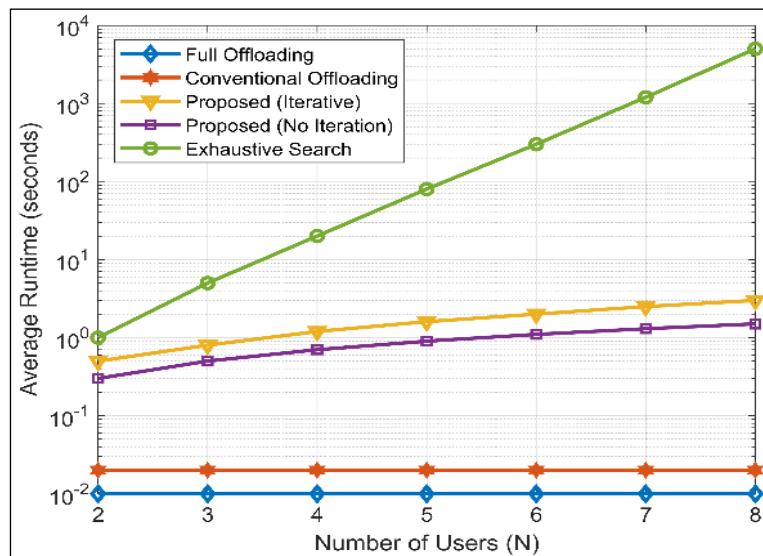


Figure 2: Algorithm Runtime versus Number of Users.

Source: Authors, (2026).

The Figure 3 shows the sum energy consumption in joules for three different DNN models: AlexNet, ResNet18 and ShuffleNet V2. For AlexNet, full offloading records the highest consumption at around 160 J, while conventional offloading is lower at about 140 J. The proposed iterative method reduces energy to about 70 J and the non-iterative version is slightly higher near 80 J. Exhaustive search shows the lowest value around 68 J. This shows that both proposed and exhaustive methods achieve substantial energy savings compared to full or conventional offloading in AlexNet. For ResNet18, a similar trend appears. Full offloading again reaches 160 J and conventional offloading consumes around 140 J. The proposed iterative method drops sharply to about 55 J. While the non-iterative approach is slightly higher near 62 J. Exhaustive search provides the best outcome, close to 53 J. ShuffleNet V2 shows the most efficient results among the three DNNs. Full offloading is still at 160 J and conventional offloading remains at 140 J. The proposed iterative approach achieves a much lower 35 J, while the non-iterative one is close to 40 J. Exhaustive search provides the lowest energy consumption around 33 J. The proposed iterative and exhaustive search approaches show strong energy savings across all models, with ShuffleNet V2 achieving the best reduction.

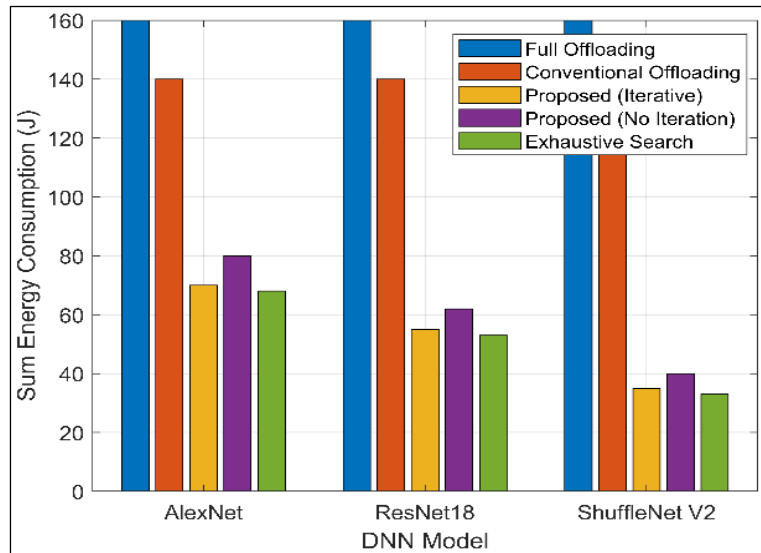


Figure 3: Energy Consumption versus DNN Model Choice.

Source: Authors, (2026).

Figure 4 shows the change in energy consumption as AP computing capacity increases from 1 GHz to 6 GHz. At low capacity of 1 GHz, full offloading is the least efficient at almost 190 J. While conventional offloading is lower at around 160 J. The proposed iterative method starts at about 55 J and the non-iterative version is slightly higher near 65 J. Exhaustive search is the most efficient at about 53 J. As the AP computing capacity increases, the energy use decreases for all methods. At 3 GHz, full offloading is still high around 165 J, while conventional offloading is reduced to about 135 J. The proposed iterative method consumes about 35 J, while exhaustive search gives lower values near 33 J showing its clear advantage. When the AP computing capacity reaches 6 GHz, the energy levels stabilize. Full offloading remains high near 160 J, while conventional offloading lowers to about 127 J. The proposed iterative method reduces sharply to almost 27 J and the non-iterative method stays near 32 J. Exhaustive search again achieves the best efficiency at around 25 J. Full offloading and conventional offloading continue to consume large amounts of energy compared to the proposed and exhaustive approaches. The results confirm that well-designed algorithms are more effective for reducing energy consumption than simply increasing computing resources.

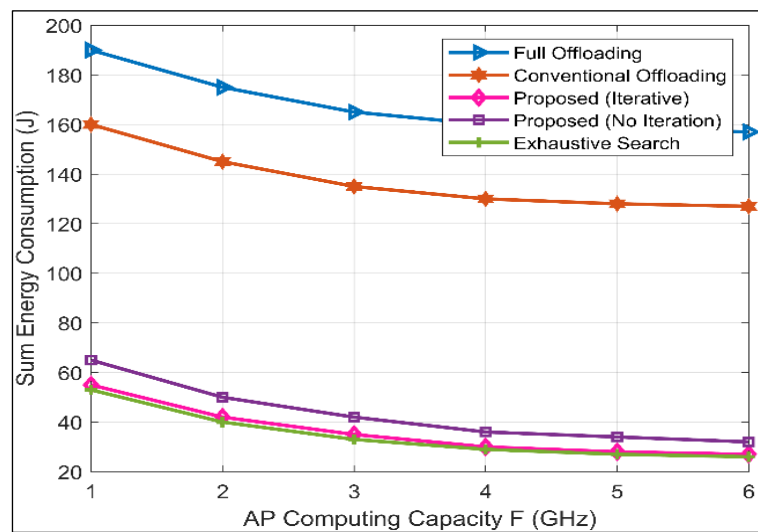


Figure 4: Energy Consumption versus AP Computing Capacity.

Source: Authors, (2026).

The Figure 5 compares the energy consumption of five approaches: full offloading, conventional offloading, proposed iterative, proposed no-iteration and exhaustive search. Full offloading shows the highest energy consumption, reaching about 180 J. Conventional offloading is slightly lower but still very high at around 150 J. These two methods consume much more energy compared to the optimized strategies. The proposed iterative approach achieves a major reduction, consuming only about 40 J. The proposed no-iteration method performs a little worse with its energy use close to 45 J. Exhaustive search gives the lowest value among all with energy consumption slightly below 40 J. Moving from full offloading to proposed iterative reduces energy use by about 140 J, which is nearly four times lower. Conventional offloading shows inefficiency since it is still about 110 J higher than the proposed iterative method. The difference between proposed iterative and proposed no-iteration is small around 5 J showing that iteration provides an additional but modest advantage. Exhaustive search slightly outperforms the proposed iterative method by a margin of about 2 J, but it requires higher computational cost. The results show that optimized offloading reduces energy use. These perform better than traditional methods. The proposed iterative method and exhaustive search give the best balance.

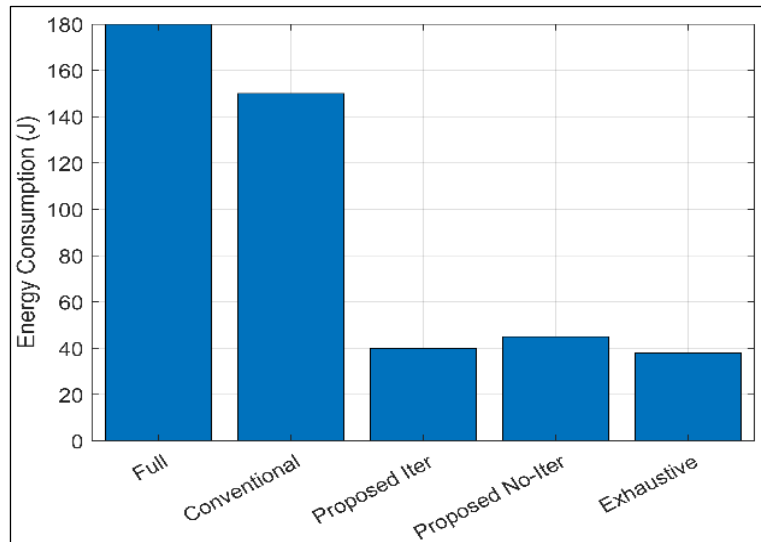


Figure 5: Energy Consumption Comparison across Schemes.
Source: Authors, (2026).

The Figure 6 presents energy consumption for five strategies. The energy values are divided into local computation, transmission and server computation. Full offloading is the costliest at nearly 180 J. Around 160 J comes from transmission. About 20 J comes from local computation. Conventional offloading reduces the total to about 150 J. Transmission still dominates at nearly 125 J. Server computation adds around 10 J. Local computation adds about 15 J. The proposed iterative method reduces the total to about 40 J, with 25 J from local computation, 10 J from transmission and about 5 J from server computation. The other two methods, proposed no-iteration and exhaustive search show similar low totals. Proposed no-iteration uses about 45 J in total made up of around 28 J from local computation, 12 J from transmission and 5 J from server computation. Exhaustive search performs best with only 38 J overall, with about 24 J from local computation, 10 J from transmission and about 4 J from server computation. Full offloading consumes more than four times the energy of exhaustive search, highlighting the inefficiency of traditional offloading. The small difference between proposed iterative and exhaustive search less than 3 J suggests that both provide near-optimal solutions.

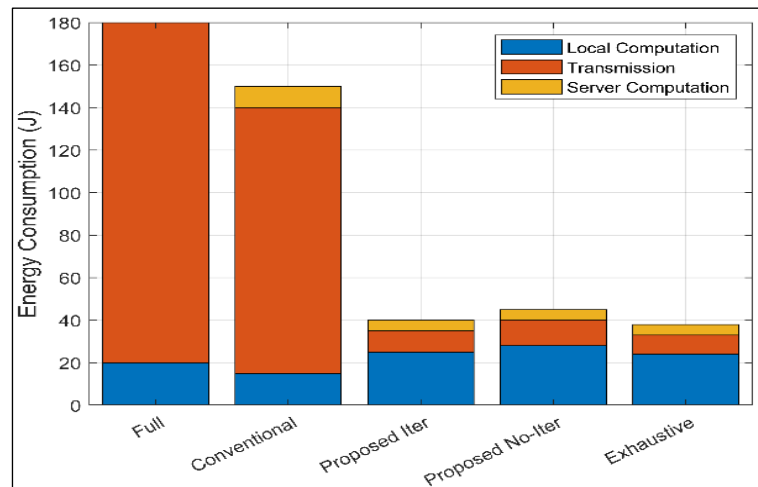


Figure 6: Energy Breakdown by Scheme.
Source: Authors, (2026).

The Figure 7 shows the relationship between energy consumption in joules and average delay. At an average delay of 30 ms, full offloading records the highest energy use at around 225 J, while conventional offloading is slightly lower near 180 J. The proposed iterative method starts at about 70 J and the proposed no-iterative version is close to 80 J. As delay increases to 50 ms, the energy consumption of all methods decreases. Full offloading drops to about 185 J and conventional offloading falls to around 150 J. The proposed iterative method reduces to nearly 40 J, while the proposed no-iterative method reaches about 48 J. At longer delays like 80 ms, the downward trend continues. Full offloading still consumes the most energy at about 155 J, while conventional offloading is reduced to around 120 J. The proposed iterative method decreases to about 27 J, while the proposed non-iterative approach stays slightly higher near 30 J. The proposed iterative method keeps the lowest energy use. This shows that well-designed algorithms are more efficient than traditional methods. The performance gap becomes smaller as delay limits are relaxed.

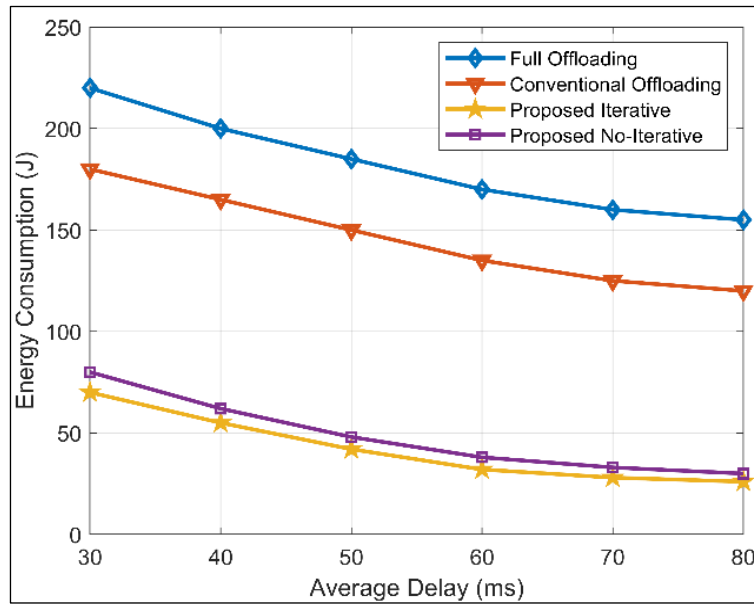


Figure 7: Energy-Delay Tradeoff Curve.

Source: Authors, (2026).

The Figure 8 shows delay satisfaction levels for different computation strategies. Full offloading performs the worst with only about 40 percent satisfaction, indicating that most tasks fail to meet the required delay limits. Conventional offloading does better, reaching around 60 percent satisfaction but it is still not efficient enough. The proposed iterative method shows a big jump, reaching almost 95 percent, which indicates that nearly all tasks meet delay expectations under this approach. The proposed no-iteration method achieves close to 90 percent, slightly lower than the iterative one but still a strong result. Exhaustive search gives the best outcome, achieving around 98 percent satisfaction, which makes it the most reliable method in this comparison. Full offloading and conventional offloading show poor performance, with satisfaction rates well below 70 percent. The proposed methods and exhaustive search reach values above 90 percent, proving that optimized strategies handle delays much better. Exhaustive search has a slight edge over the proposed iterative method but the difference is small around 3 percent. The figure shows that traditional offloading does not meet delay needs well. Optimized methods handle delay more effectively. Intelligent task allocation gives very high satisfaction. This makes them better for real-time applications.

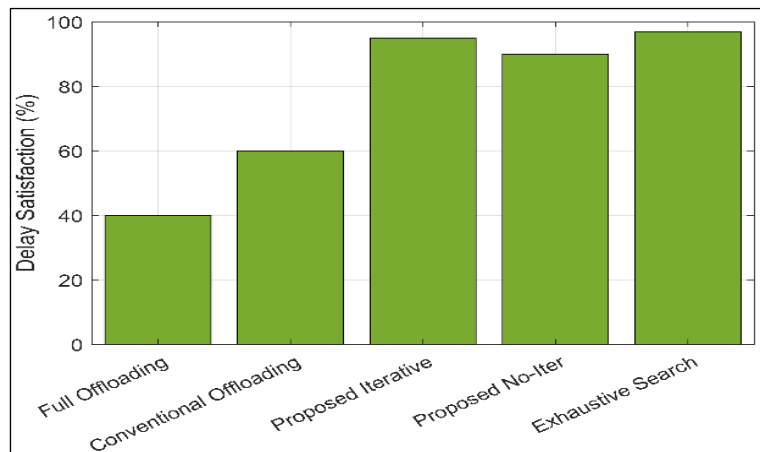


Figure 8: Comparison of Delay Satisfaction across Schemes.

Source: Authors, (2026).

The Figure 9 presents the relationship between energy consumption and compression ratio for three methods. At a compression ratio of 0.2, energy consumption is the highest across all methods. The proposed no-iterative method consumes about 100 joules. While the proposed iterative method consumes nearly 90 joules. Exhaustive search consumes around 85 joules. As the compression ratio increases to 0.4, the energy values drop. The proposed no-iterative method decreases to around 80 joules, the iterative method to about 70 joules and exhaustive search to nearly 68 joules. At a ratio of 0.6, the downward trend continues with no-iterative consuming about 65 joules, iterative about 58 joules and exhaustive search about 52 joules. These values highlight that higher compression greatly reduces energy usage. When the compression ratio rises to 0.8, the energy consumption decreases more. The proposed no-iterative method consumes about 50 joules. While iterative uses close to 40 joules and exhaustive search consumes around 38 joules. At the maximum compression ratio of 1, energy consumption is lowest for all approaches. The proposed no-iterative method performs worst but still shows improvement as compression increases. The results show that compression helps save energy. Optimized strategies improve efficiency even more. Exhaustive search and iterative methods give the best energy savings.

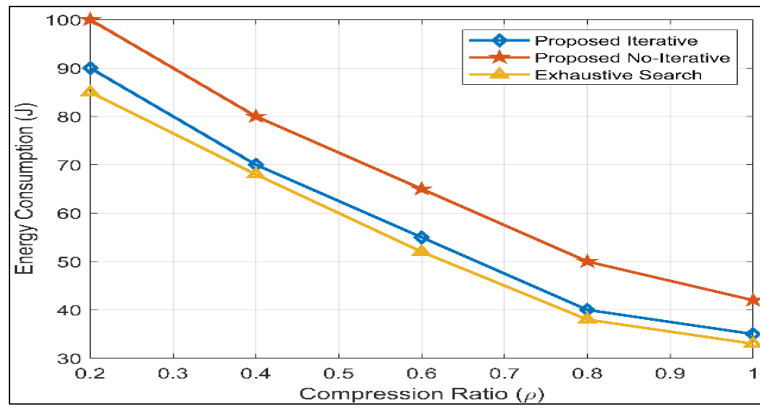


Figure 9: Energy Consumption versus Compression Ratio.

Source: Authors, (2026).

VII CONCLUSION

This paper studies SC assisted, computation-aware offloading for DNN inference in a multi-user mobile edge computing system. Each user first compresses large raw input into a small semantic feature and then uploads the feature to an edge server via TDMA. A practical link is established between the compression ratio and the local computation ratio. This link is motivated by the structure of multi-layer DNNs and it is modeled in a tractable piecewise-linear form. Using this mapping, the paper formulates an energy-minimization problem. To gain analytic insight, a special case with preconfigured local computing time is considered. For this case, closed-form optimal solutions for computation allocation and upload time are derived using Lagrange duality. These closed-form results reveal a threshold structure driven by an offloading priority indicator. The indicator captures the interplay between the user channel quality and the local computing energy cost. The algorithm alternates layer selection with joint optimization of computation allocation and upload time. The low-complexity layer-selection scheme performs close to exhaustive search while incurring much lower run time. Extensions and future work are noted including asynchronous computing/offloading and the integration of radio techniques like NOMA and RIS. Combining semantic compression with computation-aware partitioning provides substantial energy benefits for edge DNN inference.

VIII. AUTHOR'S CONTRIBUTION

Conceptualization: Sumalatha Pilli, U S B K Maha Lakshmi Thalakola Syamsundararao, Kuraguntla Radhika , Dr.Shobana Gorintla, Athmakuri Satish Kumar, D. Kishore, Chavvakula Janaki Devi.

Methodology: Sumalatha Pilli, U S B K Maha Lakshmi Thalakola Syamsundararao, Kuraguntla Radhika , Dr.Shobana Gorintla, Athmakuri Satish Kumar, D. Kishore, Chavvakula Janaki Devi

Investigation: Sumalatha Pilli, U S B K Maha Lakshmi Thalakola Syamsundararao, Kuraguntla Radhika , Dr.Shobana Gorintla, Athmakuri Satish Kumar, D. Kishore, Chavvakula Janaki Devi.

Discussion of results: Sumalatha Pilli, U S B K Maha Lakshmi Thalakola Syamsundararao, Kuraguntla Radhika , Dr.Shobana Gorintla, Athmakuri Satish Kumar, D. Kishore, Chavvakula Janaki Devi.

Writing – Original Draft: Sumalatha Pilli, U S B K Maha Lakshmi Thalakola Syamsundararao, Kuraguntla Radhika , Dr.Shobana Gorintla, Athmakuri Satish Kumar, D. Kishore, Chavvakula Janaki Devi.

Writing – Review and Editing: Sumalatha Pilli, U S B K Maha Lakshmi Thalakola Syamsundararao, Kuraguntla Radhika , Dr.Shobana Gorintla, Athmakuri Satish Kumar, D. Kishore, Chavvakula Janaki Devi.

Resources: Sumalatha Pilli, U S B K Maha Lakshmi Thalakola Syamsundararao, Kuraguntla Radhika , Dr.Shobana Gorintla, Athmakuri Satish Kumar, D. Kishore, Chavvakula Janaki Devi.

Supervision: Sumalatha Pilli, U S B K Maha Lakshmi Thalakola Syamsundararao, Kuraguntla Radhika , Dr.Shobana Gorintla, Athmakuri Satish Kumar, D. Kishore, Chavvakula Janaki Devi.

Approval of the final text: Sumalatha Pilli, U S B K Maha Lakshmi Thalakola Syamsundararao, Kuraguntla Radhika , Dr.Shobana Gorintla, Athmakuri Satish Kumar, D. Kishore, Chavvakula Janaki Devi.

IX REFERENCES:

- [1] G. Callebaut, G. Leenders, J. Van Mulders, G. Ottoy, L. De Strycker, and L. Van der Perre, "The art of designing remote iot devices—technologies and strategies for a long battery life," *Sensors*, vol. 21, no. 3, p. 913, 2021.
- [2] B. Bahrami, M. R. Khayyambashi, and S. Mirjalili, "Edge server placement problem in multi-access edge computing environment: models, techniques, and applications," *Cluster Computing*, vol. 26, no. 5, pp. 3237–3262, 2023.
- [3] H. Sun, X. Ma, and R. Q. Hu, "Adaptive federated learning with gradient compression in uplink noma," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16325–16329, 2020.
- [4] Z. Yang, M. Chen, G. Li, Y. Yang, and Z. Zhang, "Secure semantic communications: Fundamentals and challenges," *IEEE network*, vol. 38, no. 6, pp. 513–520, 2024.
- [5] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Multiuser co-inference with batch processing capable edge server," *IEEE Transactions on Wireless Communications*, vol. 22, no. 1, pp. 286–300, 2022.

- [6] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (xai): Toward medical xai," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 11, pp. 4793–4813, 2020.
- [7] Z. Li, H. Li, and L. Meng, "Model compression for deep neural networks: A survey," *Computers*, vol. 12, no. 3, p. 60, 2023.
- [8] C. Asirimath, C. Weeraddana, S. Samarakoon, J. Ratnayake, and M. Bennis, "From raw data to structural semantics: Trade-offs among distortion, rate, and inference accuracy," *arXiv preprint arXiv:2412.19825*, 2024.
- [9] K. He and J. Sun, "Convolutional neural networks at constrained time cost," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5353–5360, 2015.
- [10] Z. Yang, M. Chen, W. Saad, W. Xu, and M. Shikh-Bahaei, "Sum-rate maximization of uplink rate splitting multiple access (rsma) communication," *IEEE Transactions on Mobile Computing*, vol. 21, no. 7, pp. 2596–2609, 2020.
- [11] R. G. Alakbarov, "Challenges of mobile devices' resources and in communication channels and their solutions," *International Journal of Computer Network and Information Security*, vol. 13, no. 1, pp. 39–46, 2021.
- [12] P. V. Dantas, W. Sabino da Silva Jr, L. C. Cordeiro, and C. B. Carvalho, "A comprehensive review of model compression techniques in machine learning," *Applied Intelligence*, vol. 54, no. 22, pp. 11804–11844, 2024.
- [13] Z. Ding, P. Fan, and H. V. Poor, "Impact of non-orthogonal multiple access on the offloading of mobile edge computing," *IEEE Trans. Commun.*, vol. 67, no. 1, pp. 375–390, Jan. 2019.
- [14] C. Xu, G. Zheng, and X. Zhao, "Energy-minimization task offloading and resource allocation for mobile edge computing in NOMA heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16001–16016, Dec. 2020.
- [15] Z. Ding, D. Xu, R. Schober, and H. V. Poor, "Hybrid NOMA offloading in multi-user MEC networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 7, pp. 5377–5391, Jul. 2022.
- [16] Z. Chen, Y. Zhang, Z. Zhang, and Y. Chen, "Reconfigurable intelligent surface assisted MEC offloading in NOMA-enabled IoT networks," *IEEE Trans. Commun.*, vol. 71, no. 8, pp. 4896–4908, Aug. 2023.
- [17] G. Chen, Q. Wu, W. Chen, D. W. K. Ng, and L. Hanzo, "IRS-aided wireless powered MEC systems: TDMA or NOMA for computation offloading?," *IEEE Trans. Wireless Commun.*, vol. 22, no. 2, pp. 1201–1218, Feb. 2023.
- [18] Z. Qin, X. Tao, J. Lu, and G. Y. Li, "A generalized semantic communication system: From sources to channels," *IEEE Wireless Commun.*, vol. 30, no. 3, pp. 18–26, Jun. 2023.
- [19] Z. Lu, X. Xu, Z. Qin, and X. Tao, "Semantics-empowered communications: A tutorial-cum-survey," *IEEE Commun. Surveys Tuts.*, vol. 26, no. 1, pp. 41–79, 1st Quart., 2024.
- [20] H. Xie, Z. Qin, G. Y. Li, and B.-H. Juang, "Deep learning enabled semantic communication systems," *IEEE Trans. Signal Process.*, vol. 69, pp. 2663–2675, 2021.
- [21] Z. Weng, H. Xie, Z. Qin, and G. Y. Li, "Semantic communications with speech recognition and synthesis," *IEEE Trans. Wireless Commun.*, vol. 22, no. 9, pp. 6227–6240, Sep. 2023.
- [22] D. Huang, H. Xie, Z. Qin, and G. Y. Li, "Toward semantic communications: Deep learning-based image semantic coding," *IEEE J. Sel. Areas Commun.*, vol. 41, no. 1, pp. 55–71, Jan. 2023.
- [23] N. Ma, X. Zhang, H. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. ECCV*, Sep. 2018, pp. 116–131.
- [24] H. Ko, H. Jeong, D. Jung, and S. Pack, "Dynamic split computing framework in distributed serverless edge clouds," *IEEE Internet Things J.*, vol. 11, no. 8, pp. 14523–14531, Aug. 2023.
- [25] M. Gao, C. Li, and H. Huang, "Task partitioning and offloading in DNN-task enabled mobile edge computing networks," *IEEE Trans. Mobile Comput.*, vol. 22, no. 4, pp. 2435–2445, Apr. 2023.
- [26] H. Liang, Y. Zhang, and S. Wang, "DNN surgery: Accelerating DNN inference on the edge through layer partitioning," *IEEE Trans. Cloud Comput.*, vol. 11, no. 3, pp. 3111–3125, Sep. 2023.
- [27] J. Li, L. Zhang, and F. Zhao, "Throughput maximization of delay-aware DNN inference in edge computing by exploring DNN model partitioning and inference parallelism," *IEEE Trans. Mobile Comput.*, vol. 22, no. 5, pp. 3017–3030, May 2023.