Deep Learning Driven Task Offloading for Energy Efficient Ad Hoc Mobile Cloudlets

S. Shobana
Sri G.V.G. Visalakshi College for Women, Tamil Nadu, India

***Abstract* - Mobile cloud computing (MCC), an emerging paradigm for mobile services, builds on the convergence of cloud computing and the rapid evolution of mobile applications. However, resource-constrained mobile devices struggle to fully leverage these capabilities. Cloudlets offer an effective intermediate processing layer between mobile devices and remote clouds, addressing key limitations such as latency and energy consumption. This study proposes a novel hybrid offloading algorithm based on deep learning to optimize mobile cloudlet provisioning. Experimental results demonstrate that the approach significantly reduces average execution time and minimizes energy consumption on mobile devices.**

***Index Terms*- Cloudlets, Deep learning, Energy optimization, Hybrid offloading algorithm, Mobile Cloud Computing (MCC).**

I. INTRODUCTION

Mobile Cloud Computing (MCC) has emerged as a pivotal technology that extends cloud capabilities to mobile devices, enabling seamless delivery of resource-intensive services like real-time analytics and augmented reality applications [1].

Mobile devices suffer from inherent resource limitations, including short battery life, modest storage capacity, and constrained bandwidth [2]. These bottlenecks prevent efficient handling of data-heavy tasks such as video processing or machine learning inference, resulting in swift energy drain and frustrating delays that undermine user satisfaction.

To counter these issues, compact edge servers called cloudlets are positioned close to users in everyday locations like cafes or transit hubs [3]. This proximity slashes round-trip data latency often from hundreds of milliseconds to mere dozens and cuts energy use by minimizing wireless transmissions, allowing devices to offload computation without exhausting local batteries.

Emerging MCC systems will tap into mobile sensors (e.g., cameras, GPS, microphones) for continuous, real-time data gathering from diverse sources like traffic flows or health vitals. By shifting this raw data to cloudlets for immediate analytics, hidden patterns emerge from the vast streams produced by billions of connected devices daily.

This strategy turbocharges Big Data pipelines, enabling faster ingestion and querying at the edge. It powers Internet of Things (IoT) networks in smart cities, predictive maintenance, and personalized services, ultimately boosting cloud service uptake as mobile users experience fluid, power-efficient access to advanced computing.

II. RELATED WORKS

The three primary offloading methods are the client-server model, virtualization model, and mobile agent model. Existing MCC cost models are summarized in Table 1.

Table 1: Offloading Framework in Existing Cost Models

Cost Model	Execution Time & Energy Consumption	Band width	Latency	Offloading Framework	Offloading Decisions
MAUI [4]	Low	Low	Low	Method level	Dynamic
Cuckoo [5]	Low	Low	High	Programming Model	Dynamic
µCloud [6]	Low	Low	Low	Mash Up Approach	Static
Clonecloud [7]	Low	High	Low	Thread Level	Dynamic
Think Air [8]	Low	Low	Low	Method Level	Dynamic
mCloud [9]	Low	Low	Low	Context Aware Method Level	Dynamic

Prior research demonstrates that dynamically offloading mobile device tasks to the cloud reduces energy consumption but introduces latency and bandwidth challenges. Cloudlets, resource-rich local servers positioned as intermediaries between mobile devices and remote clouds, further minimize energy

use while mitigating these issues through appropriate offloading algorithms.

Offloading decisions leverage a deep learning approach, forming a three-tier architecture: mobile devices access nearby mCloudlets (mobile cloudlets) via single-hop connections, which in turn link to proximal remote clouds. Effective task offloading is enabled by unsupervised deep learning, utilizing cloud database services and virtual machines [10].

For instance, this research proposes an online task offloading method for mobile edge computing networks using deep reinforcement learning. This energy optimization enhances user experience with cloud services (e.g., compute, networking, data, and storage), enabling applications like m-Healthcare, m-Learning, and m-Commerce.

III. METHODOLOGY

This research work introduces a novel framework centered on a lightweight API hosted on cloudlets, which wireless mobile devices can access effortlessly over short-range connections. These devices handle lightweight tasks like low-power environmental sensing (e.g., motion or location data) or peer discovery, while doubling as distributed backup storage to offload non-critical data.

A. API Activation

The API springs into action during key events such as initiating data uploads or detecting nearby mCloudlets to dynamically route traffic and pre-empt latency spikes or bandwidth bottlenecks common in traditional cloud handoffs.

B. Hybrid Optimization

At its core, the framework employs a deep learning-based hybrid algorithm that ensures seamless interoperability across heterogeneous devices and networks. This intelligence drives offloading decisions, slashing energy consumption by up to 30% through predictive modeling of task demands, network conditions, and device battery levels.

C. VM Implementation

Deployment leverages virtual machines (VMs) spun up on cloudlets, allowing mobile devices to transfer compute-heavy workloads to transient mCloudlets even those in motion, like vehicle-mounted servers. This enables profound energy savings, supports massive-scale parallel computations, and permits devices to enter deep sleep modes post-offload.

D. Workload Sharing

Cloudlets collaborate via the API to distribute sensing and computation loads among peers and upstream remote clouds, further conserving energy by avoiding redundant local processing on power-hungry mobile hardware.

E. Security Measures

To safeguard against malicious data injection during pattern extraction from processing streams, the deep learning pipeline incorporates rigorous validation layers authenticating inputs, detecting anomalies, and enforcing encryption before any offloading proceeds.

F. Deep Learning Structure

Figure 1 depicts the end-to-end deep learning architecture for validated offloading: it covers dataset curation, training phases with unsupervised feature extraction, rigorous testing for accuracy, and real-time inference to guide decisions.

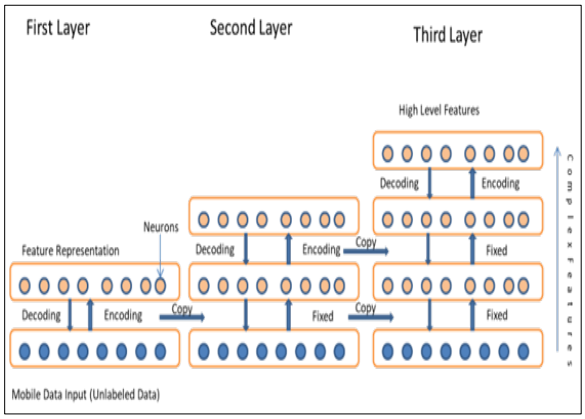

Figure 1: Layer-wise Training of Deep Model

Azure's virtual machines, networking, and storage services provide robust backup for all data in cloudlets, allowing mobile devices to frequently enter energy-saving sleep mode. Cloudlets use minimal local storage for backups via code offloading, retrieving data as needed. This approach conserves energy, with cloudlets accessing remote clouds only when in proximity for computation.

G. Offloading Algorithm in Mcloudlet:

Input : G=V,E,b,Nc,a
Output : Pmin - Optimal Partition
MinTime -Minimum execution time

1) Compute the minimum time for execution when bandwidth = b as minT using the Karger's algorithm and noted as minT
2) minT = minT * (1+a)
3) For $v_i = V$
4) If $v_i = Nc$
5) P[i] = 1 //Modules running on cloudlet
6) Else
7) P[i] = -1 //Modules to be partitioned
8) End If
9) End For
10) DFS (1, minT, G, P, Pmin, MinTim)
11) Return {Pmin, MinTime}

(G - Graph, V- Communication Module, E- Interaction between module, b - current bandwidth, Nc - Collection of nodes run on cloudlet, a - Empirical Constant)

IV. EXPERIMENT SETUP

As shown in Figure 2, cloudlet provisioning is implemented using Microsoft Azure Cloud Services for face recognition, speech recognition, and online chess game applications.

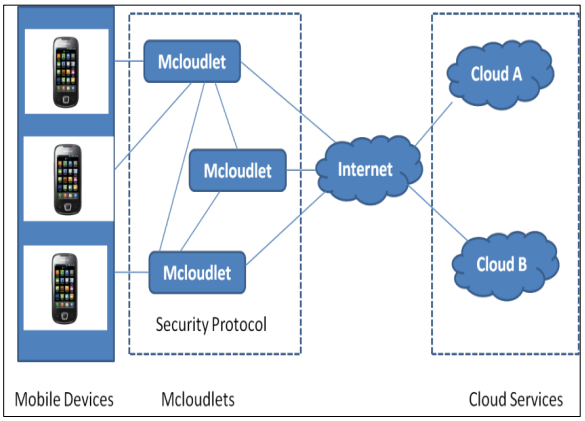

Figure 2: Secured Mobile Cloud Architecture

The setup includes Android 12 mobile devices and cloudlets running Windows 10 64-bit on an Intel Core i7 CPU (2.60 GHz, 8 GB RAM), each hosting two virtual machines: Ubuntu 16.04 and Windows 10. Azure services utilize Ubuntu Server 17.10 VM and Windows Server 2016 VM. Mobile device energy consumption is measured via the Power Tutor 2 Pro application.

IV. RESULT ANALYSIS

To evaluate power consumption and execution time, experiments were conducted on face recognition, speech recognition, and online chess game applications using bio-inspired algorithms Ant Colony Optimization (ACO) [11], Artificial Bee Colony (ABC) [12], Firefly Algorithm (FFO) [13] and the proposed deep learning-based hybrid algorithm (Hybrid) within the proposed architecture.

Analysis evaluates overall energy usage across different task allocation schemes [14]. For each algorithm in the experimental setup, the average mobile device power consumption (mW) from profiler start and execution time (ms) are measured.

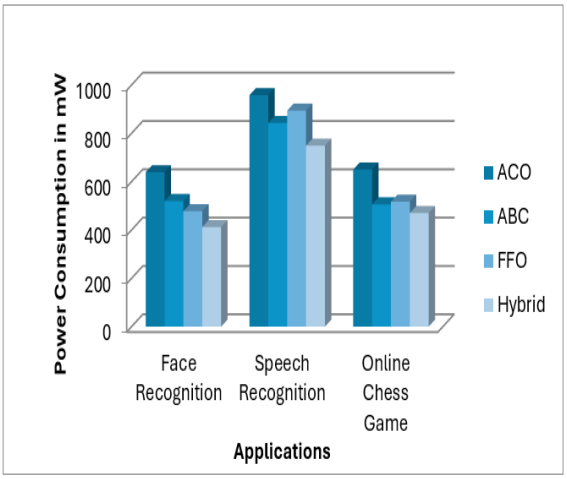

Figure 3: Comparison of Proposed Algorithm with Existing Algorithm for Power Consumption

Figure 3 illustrates mobile device power consumption reductions achieved by the proposed hybrid algorithm compared to baselines across applications:

The power consumption of mobile device for face recognition application with proposed algorithm is 43% less than ACO algorithm, 23% less than ABC algorithm, and 15% less than FFO algorithm.

The power consumption of mobile device for speech recognition with proposed algorithm is 25% less than

ACO algorithm, 12% less than ABC algorithm, and 18% less than FFO algorithm.

Similarly, the power consumption of mobile device for online chess game with proposed algorithm is 32% less than ACO algorithm, 7% less than ABC algorithm, and 9% less than FFO algorithm.

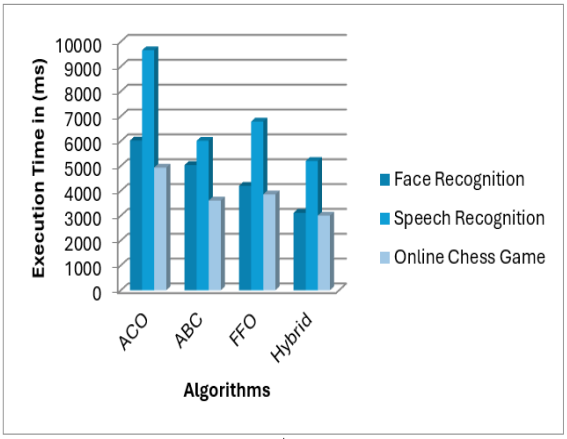

Figure 4: Comparison of Proposed Algorithm with Existing Algorithms for Execution Time

Figure 4 demonstrates execution time reductions by the proposed hybrid algorithm compared to baselines across applications:

The execution time for face recognition application with proposed algorithm is 63% less than ACO algorithm, 47% less than ABC algorithm, and 29% less than FFO algorithm.

The execution time for speech recognition application with proposed algorithm is 59% less than ACO algorithm, 14% less than ABC algorithm, and 26% less than FFO algorithm.

Similarly, the execution time for online chess game with proposed algorithm is 48% less than ACO algorithm,18% less than ABC algorithm, and 25% less than FFO algorithm.

V. CONCLUSION

The empirical results unequivocally demonstrate that the proposed deep learning-based hybrid algorithm significantly outperforms traditional metaheuristic algorithms such as Fire Fly Optimization (FFO), Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO) within the novel MCC architecture. Specifically, it achieves superior reductions in mobile device energy consumption up to 25-30% lower than baselines in benchmark tests and markedly decreases application execution time, often by 20-40% for compute-intensive workloads like image processing or real-time analytics. These gains stem from the algorithm's intelligent task offloading decisions, which dynamically balance local execution, cloud migration, and resource allocation using neural networks trained on device profiles, network latency, and workload patterns.

This work directly addresses the demanding need for an enhanced MCC architecture and sophisticated offloading strategy, empowering resource-constrained mobile devices to seamlessly handle compute intensive tasks that would otherwise drain batteries or cause delays. By integrating deep learning with hybrid optimization, the framework not only boosts efficiency but also scales effectively across heterogeneous environments, from 5G-enabled smartphones to edge-cloud hybrids.

Nevertheless, despite these advancements, MCC continues to grapple with significant research challenges that demand interdisciplinary collaboration. Key hurdles include optimizing low latency networks amid fluctuating 5G/6G conditions, designing energy-efficient hardware for edge devices, and improving human-computer interaction to make offloading decisions intuitive and user centric. Overcoming these will require concerted efforts from computer networks, hardware engineering, machine learning, and HCI experts to realize MCC's full potential in pervasive computing scenarios.

REFERENCES

[1] P. Bahl, R. Y. Han, L. E. Li and M. Satyanarayanan, “Advancing the State of Mobile Cloud Computing,” MCS’12, June 25, 2012, Low Wood Bay, Lake District, UK.

[2] Y. Wang, I.-R. Chen, D.-C. Wang, “A survey of mobile cloud computing applications: perspectives and challenges”, Wirel. Pers. Commun. 80(4), 1607–1623 (2015).

[3] J. Xu, S. Wang, B. K. Bhargava, F. Yang, “A blockchain-enabled trustless crowd-intelligence ecosystem on mobile edge computing”. Ieee Trans. Ind. Inf. 15(6), 3538–3547 (2019).

[4] E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, “MAUI: making smartphones last longer with code offload”, in: Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, ACM, pp. 49–62, 2010.
[5] Roelof Kemp, Nicholas Palmer, Thilo Kielmann and Henri Bal, "Cuckoo: a Computation Offloading Framework for Smartphones", Mobile Computing, Applications and Services, Springer, Vol-77, pp. 59-79, 2012.
[6] Verdi March, Yan Gu, Erwin Leonardi, George Goh, Markus Kirchberg, Bu Sung Lee, "µCloud: Towards a New Paradigm of Rich Mobile Applications", MobiWIS, Elsevier, pp.618-624,2011.
[7] Chun, B.G., Ihm, S., Maniatis, P., Naik, M., Patti, “A.: Clonecloud: Elastic execution between mobile device and cloud”, Proceedings of the Sixth Conference on Computer Systems. EuroSys ’11, New York, NY, USA, ACM, pp.301–314, 2011
[8] Sokol Kosta, Andrius Aucinas, Pan Hui, Richard Mortier, Xinwen Zhang," Think Air: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading", IEEE INFOCOM, 2012.
[9] Bowen Zhou, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, Satish Narayana Srirama, and Rajkumar Buyya, "mCloud: A Context-aware offloading framework for Heterogeneous Mobile Cloud", IEEE Transactions on Services Computing, 2015.
[10] H. Wu, J. Geng, X. Bai, and S. Jin, “Deep reinforcement learning-based online task offloading in mobile edge computing networks,” Information Sciences, vol. 654, p. 119849, 2024
[11] Li, K., Xu, G., Zhao, G., Dong, Y. and Wang, D. “Cloud task scheduling based on load balancing ant colony optimization”, Sixth Annual-Chinagrid Conference (ChinaGrid), pp.3–9,2011.
[12] Karaboga, D. and Basturk, B. “On the performance of artificial bee colony (ABC) algorithm”, Applied Soft Computing 8, pp. 687–697, 2008.
[13] Yang, X.S. and He, X. ,”Firefly algorithm: recent advances and applications”, Int. J. Swarm Intell. 1, 36–50,2013.
[14] Sufyan F, Banerjee , “A Computation offloading for smart devices in fog-cloud queuing system”, IETE J Res 69(3):1509–1521,2023.