Deep Neural Network Architecture Optimization for Edge Computing Based on Evolutionary Algorithms

Li Wang [1], Xiuming Cheng [2,*]
[1]School of Information and Electronics Engineering, Jiangsu Vocational Institute of Architectural Technology, Xuzhou 221116, Jiangsu, China
[2]School of General Courses, Jiangsu Vocational Institute of Architectural Technology, Xuzhou 221116, Jiangsu, China
E-mail: xiuming_cheng@hotmail.com
*Corresponding author

Keywords: vehicular edge computing (VEC), edge server placement, network condition adaptation, synergistic fibroblast optimized efficient deep neural network (SFO-Eff-DNN)

Received: May 28, 2025

Vehicular Edge Computing (VEC) is a crucial component of Intelligent Transportation Systems (ITS), enabling low-latency and energy-efficient services by offloading computation to the network edge. However, optimizing system performance in such environments requires careful edge server placement, especially in dynamic vehicular contexts characterized by high mobility and unpredictability. Achieving optimal performance under the constraints of latency, energy consumption, and mobility remains a significant challenge. This research proposes a comprehensive framework for optimizing deep learning architectures in VEC, utilizing advanced evolutionary algorithms. Building on real-world vehicular mobility traces, the framework employs the Synergistic Fibroblast Optimized Efficient Deep Neural Network (SFO-Eff-DNN) to identify optimal configurations and edge server placements. The dataset includes details about task offloading under different mobility levels, the data was preprocessed using Min-Max normalization to ensure smooth learning. Among the algorithms evaluated, Synergistic Fibroblast Optimization (SFO) consistently produces well-distributed Pareto-optimal solutions and effectively handles trade-offs between competing objectives. The DNN is utilized to learn complex patterns in vehicular mobility and network conditions, which helps predict the best configurations for edge server placements. The proposed system efficiently minimizes latency and energy consumption while ensuring scalability and adaptability to real-world scenarios. Results demonstrate that SFO-Eff-DNN achieves superior convergence speed and energy efficiency, making it well-suited for time-sensitive deployments. Comparative simulations validate that this approach outperforms traditional methods, providing valuable insights for deploying efficient and robust edge intelligence architectures in next-generation intelligent transportation systems.

Povzetek: Ta raziskava se osredotoča na področje robnega računalništva v vozilih (VEC), kar je ključno za zagotavljanje nizke zakasnitve v inteligentnih transportnih sistemih. Vsebina prispevka predstavlja hibridni okvir SFO-Eff-DNN, ki združuje globoko učenje in evolucijsko optimizacijo za reševanje kompleksnega problema postavitve robnih strežnikov in prilagajanja arhitekture nevronske mreže. Glavni dosežki vključujejo rešitev večciljne optimizacijske naloge, ki uspešno minimizira zakasnitev in porabo energije v dinamičnem voznem okolju.

1 Introduction

An ITS enhances the safety of moving vehicles and hikers within the vicinity. In recent times, problems regarding road traffic safety have increased and accidents continue to occur regularly (Wan et al., 2020). Fortunately, a growing number of related technologies have been applied to the transportation industry as wireless communication and sensor technologies have developed and matured in recent years. The increased need for road efficiency and safety in intricately linked road systems has drawn a lot of attention to ITS in recent years (Boukerche et al., 2020). The exponential growth in ITS has resulted in an increased demand for responsive, energy-efficient, and intelligent processing solutions that can manage the dynamic vehicular environment (Elassy et al., 2024). VEC is a pattern that brings the cloud computing capacities closer to the network edge and is a likely solution to service demands for low-latency services, such as auto-corrective driving support, real-time traffic management, and location-based services (Alhilal et al., 2024). Connected vehicles benefit from VEC by shortening the response time of their systems and helping them save power by assigning tasks to local servers (Chougule et al.,

2024). Greater safety, dependability, efficiency in transportation, fast action and network reach make smart and sustainable driving networks possible (Talpur and Gurusamy, 2021). To minimize the time for data exchanges and energy used in vehicles, VEC allows vehicles to perform certain tasks on edge servers nearby. As a result, connected vehicles receive a much better level of service (Zaki et al., 2024). Due to their speed and patches of unpredictability, the movement of vehicles complicates VEC systems (Zhao et al., 2023). The greatest aspect to focus on is the best locations and times for edge servers so that moving vehicles can be handled efficiently (Shen et al., 2021). With many vehicles moving, topology shifts taking place and numerous demands for services, generic or manual placements are not usually enough. Similarly, managing various goals, including keeping reaction times quick, using as little energy as possible, maintaining flexibility, and scaling up, remained prominent in network research (Peyman et al., 2023). As simulation traces were used, working with many nodes and requiring some attention to used parameters, this approach might face issues when put to practical use.

Deep learning and evolutionary optimization are used in the design to choose the best locations for edge servers. Specifically, the SFO-Eff-DNN approach allows the system to recognize patterns using a DNN and search globally using an SFO algorithm. This framework processes actual data from vehicle movement to understand vehicle movements and the state of the network, as well as select the best position for the servers. The key contribution of the research is as follows.

In extremely dynamic vehicle contexts, it was best to formulate the edge server placement problem as a multi-objective optimization task that simultaneously reduces the latency and energy consumption.

To create the SFO-Eff-DNN framework, which combines biologically inspired optimization with effective deep learning to deliver scalable and flexible placement solutions.

To compare the system against traditional techniques and perform comprehensive simulations using genuine mobility datasets, showcasing notable advances in placement accuracy, energy economy, and convergence speed.

The remainder of this research is separated into the following sections: the literature review on edge server placement and the intelligent optimization techniques in VEC are reviewed. The phrasing of the problem and the system model are then given in detail, as well as the description of the proposed SFO-Eff-DNN framework. The next section will discuss the experimental settings and performance evaluations, and the results and insights will then be discussed. Lastly, the research is concluded with directions for further research.

The introduction highlights the significant importance of edge servers' placement efficiency in the VEC for improving ITS performance. The literature review reveals the weaknesses of existing methods, especially their inability to handle the dynamism of vehicular mobility effectively in the process of optimizing latency and energy consumption.

2 Related work

This section discusses the positioning of border servers within the VEC, including the traditional heuristics, deep learning (DL), evolutionary algorithms, the challenges in dynamic vehicular environments, and the recent data-driven and optimization-based developments of this space for better adaptability and performance. To fix the issue of resource assignment in cloud computing Infrastructure as a Service (IaaS), an Equilibrium Optimization (EO)-based evolutionary Recurrent Neural Network (RNN) was presented (Ebrahimi Mood et al., 2025). This model was designed to give virtual machines an optimal number of physical machines by improving how they work in general and by reducing their complexity. The simulations were faster and more reliable than the conventional ones.

The significance of edge computing topics such as selecting the right tasks for offloading, allocating resources, and ensuring good Quality-of-Service and Quality-of-Experience (Vijayakumar et al., 2021). The challenges in optimizing and scheduling were solved with models and DL techniques based on evolution. This approach helps to make better decisions and effectively manage resources in environments at the edges of a network. Yang et al., (2021) introduced a method that can manage both accuracy and the speed of neural networks on edge devices. An estimate of resource use latency created from the profiling model and the Pareto Bayesian search was driven by constraints on accuracy and latency. Without sacrificing accuracy, the inference process was 94.71% faster and the search process became 18.18% more efficient.

An energy-efficient DNN offloading was developed under deadline and budget constraints in edge-cloud environments; this optimization modeling was performed using an Enabled Hybrid Chaotic Evolutionary Algorithm Dynamic Voltage Frequency Scaling (HCEA-DVFS) (Li et al., 2024). The Archimedes Optimization and Simulated Annealing were applied for global exploration, and local search improvement based on the Genetic Algorithm (GA) chaotic strategy. Experiments proved that HCEA-DVFS decreased energy consumption by 7.93% to 19.38% relative to baseline techniques on a variety of DNN-based apps. A suitable deep learning model and a proper method for training the effective training scheme for the deep neural network (ETS-DNN) were created to allow real-time monitoring in an Internet of Medical Things (IoMT) system that used edge computing (Pustokhina et al., 2020). Optimization of the neural network with autoencoders and softmax layers was achieved by using a Hybrid Modified Water Wave Optimization (HMWWO) algorithm. Examination of simulation results indicated that ETS-DNN performed better when processing prompts and making accurate diagnoses. Table 1 demonstrates the summary of the literature review.

Table 1: Related work VEC optimization methods and outcomes

Methods	Aim	Outcome	Challenge	Author/Ref.
DeepMaker Framework (Multi-objective Evolutionary Approach)	Automatically design robust DNN architectures for embedded devices	Achieved up to 26.4x compression on CIFAR-10 with only 4% accuracy loss; optimized network size and accuracy for limited resources	Designing efficient DNNs that fit resource constraints while maintaining accuracy	(Loni et al., 2020)
Internet of Things (IoT)-Defender (Modified GA)/ Deep long-short-term memory (LSTM)	To detect cyberattacks in IoT networks using an efficient, lightweight edge-based IDS	Achieved higher accuracy, superior detection rate, greater precision, false alarm rate, mIoU, and training time on BoT-IoT dataset; effective real-time deployment on Raspberry Pi devices	Addressing IoT security with limited resources, class imbalance, and low hardware security in edge computing environments	(Saheed et al., 2024)
Genetic Simulated Annealing-based Particle Swarm Optimization (GSP)	To reduce latency and energy usage in smart mobile devices by partially offloading	Achieved lower energy consumption and faster convergence compared to three baseline methods using real-life data; provided joint optimization of offloading ratio, bandwidth, and transmission power allocation	Balancing limited resources of SMDs with high communication costs and maintaining energy-efficient service	(Bi et al., 2020)
Greedy Algorithm and GA for Task Scheduling	Optimizing task scheduling in cloud-edge systems to reduce the average response time of DNN-based apps	Achieved near-optimal scheduling performance with reduced average response time; GA outperformed greedy in accuracy but required more computation time.	Reducing excessive delays during DNN task offloading to enhance the vehicle experience	(Chen et al., 2020)
Particle Swarm Optimization (PSO)	to efficiently and quickly transfer activities from resource-constrained edge devices to MEC servers in IIoT contexts	Reduced MEC server delay, balanced energy consumption, and enabled effective resource allocation compared to GA and SA methods	Designing a low-delay and energy-efficient offloading technique in a system with several vehicles and MECs	(You et al., 2021)
Differential Evolution (DE)	To maximize IoT edge computing task clustering and scheduling	Outperformed the Firefly Algorithm and PSO in reducing execution time and improving system efficiency and stability under heavyweight workloads	Clustering and scheduling tasks effectively in heterogeneous IoT edge environments	(Yousif, et al., 2024)
Greedy Algorithm + Lagrangian Dual + Adaptive Harmony Search in federated learning (FL)	To minimize the worst-case cost of FL in VEC by optimizing computation, transmission, and local model accuracy	Achieved convergence and effective trade-off between cost and fairness through dynamic vehicle selection and resource allocation optimization	Heterogeneous capabilities and data quality among vehicles; energy and time constraints in VEC	(Xiao et al., 2021)
VECMAN (Resource Selector + Energy Manager Algorithms)	To improve energy efficiency in VEC systems by managing resource sharing among EVs	Achieved 7–18% energy savings vs. local execution and ~13% vs. RSU offloading by selecting participating vehicles and optimizing sharing durations	Uncertainty in future vehicle locations; difficulty in determining optimal resource sharing and energy management	(Bahreini et al., 2021)
VaCo (Vehicle-assisted Collaborative Caching System	To enhance intelligent service deployment in VEC by using vehicles' storage for collaborative caching	VaCo effectively utilizes vehicle resources, reducing the service failure rate and cost. Real-world dataset evaluation confirms its ability to balance benefits for all.	real-time scheduling of vehicle storage; benefit evaluation under dynamic load	(Jiang et al., 2025)
HSCoNAS (Hardware-aware Evolutionary NAS Framework)	Optimize DNN architecture for accuracy and latency on edge devices	Achieved strong accuracy–latency trade-offs on ImageNet across CPU, GPU, edge	High search overhead and runtime approximation challenges	(Luo et al., 2021)
LENS (Latency-aware NAS for Edge–	Incorporate wireless communication into NAS for hierarchical systems	Improved Pareto front performance by 76.47% (energy) and 75% (latency)	Scalability issues and fixed-tier constraints	(Odema et al., 2021)

Cloud Systems)				
Federated Learning in Edge Computing (Survey)	Review implementation, taxonomy, and challenges of FL in EC	Classified FL methods, hardware constraints, and case studies; identified open issues	Synchronization delays, hardware resource limits	(Abreha et al., 2022)
RL-Dynamic (Reinforcement Learning Framework)	To optimize service placement in vehicular networks by considering mobility and dynamic service demands	Reduced delay and improved edge server utilization compared to static placement; fairness trade-offs demonstrated	Model complexity and vehicle mobility unpredictability	(Talpur and Gurusamy, 2021)

2.1 Problem statement

Optimizing resources and edge server placement in VEC as a result of high mobility, variable networks, and few resources was hard. Usually, greedy algorithms and other traditional methods do not work well in environments that change dynamically (Chen et al., 2020). PSO faces the issues of early convergence and fixation when working with multiple vehicles (You et al., 2021). DE was not suitable for clustering tasks in real time on heterogeneous edge systems due to its issues with scalability and computation (Yousif et al., 2024). Therefore, the proposed framework SFO-Eff-DNN was used to learn how devices move and decide on offloading. It minimizes delays and uses less power, all while offering adaptability, scalability, and fast convergence in changing VEC networks.

3 Methods

3.1 Architectural overview and problem formulation

The VEC would feature wireless connection, permanent edge servers, and mobility vehicles. The simulation's rise can be increased by using vehicles to carry out new missions on surrounding servers. As individuals move around and the network evolves, it is important to find these servers with practical jobs and make sure they supply energy. The problem is solved by optimizing multiple objectives, with the main variables being the location of servers and the way vehicles connect to them throughout the day.

A) Architectural components

The architecture of the VEC system consists of three main layers, such cloud, VEC, and vehicle, Cloud storage allows for convenient processing and provides a backup system. Figure 1 illustrates the architecture of VEC. The VEC layer includes a network of Roadside Units (RSUs) with edge servers, allowing local computing and rapid exchanges of data. Intelligent vehicles make up the vehicle, layer and handle task generation and offloading depending on the current network and mobility issues. Environmental sensors like Global Positioning System (GPS) and cameras in vehicles provide live data that is key for improved traffic management and safety. They enable Vehicle-to-Vehicle (V2V) and Vehicle-to-RSU (V2R) communication and were able to process or offload tasks according to resource availability. Vehicles also allow for caching of data in memory, which makes the system work more responsively.

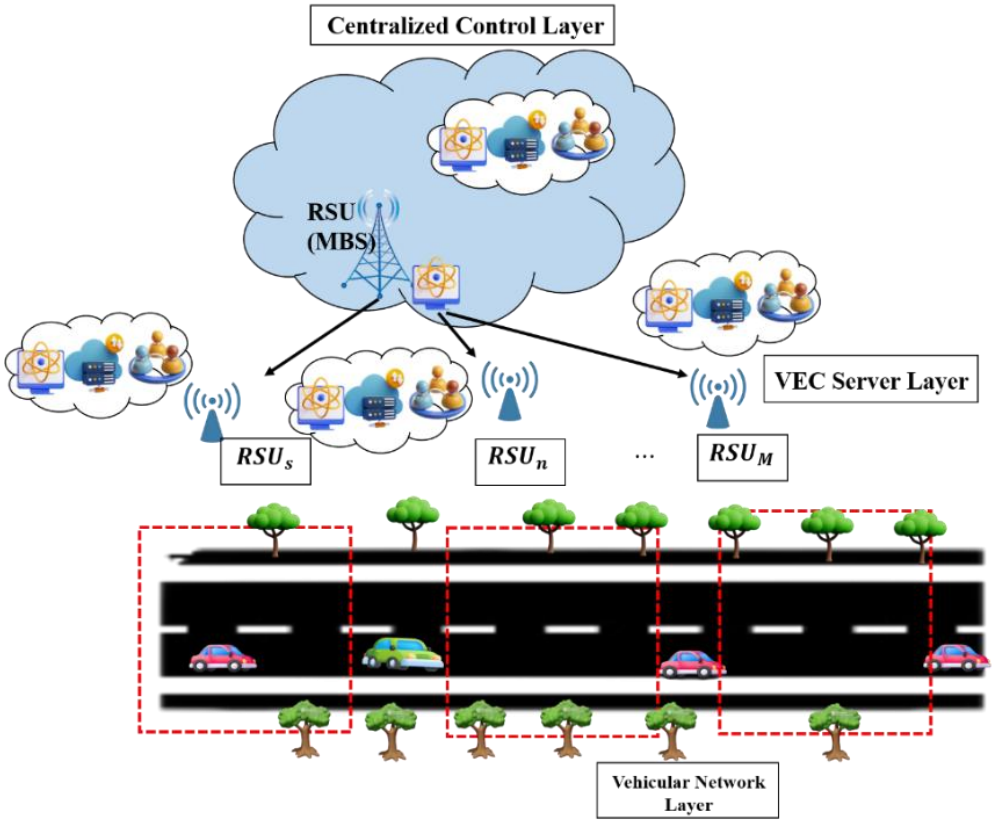

Figure 1: The architecture of the VEC

Vehicle Definition: The vehicle V defined as a six-tuple is expressed in equation (1).

$$\boldsymbol{V = \{V_{jd}, V_{st}, vK, G, J[\,_r]\}} \quad (1)$$

Each vehicle V is identified by its V_{jd}, can be activated or deactivated (V_{st}), has a type of task ($v_{j,}$), is located by Simulation of Urban Mobility (SUMO's) data $K = \{k_w, k_z, k_{y,} s_t\}$, is equipped with certain hardware (G), and is running several active instances of applications $J[\,_r]$.

Vehicle Hardware Specifications and Role of RSU: a vehicle's hardware specifications $G$are represented as a set in equation (2).

$$\boldsymbol{G = \{O, N[\,_r], A, T, d, e\}} \quad (2)$$

Each vehicle's hardware profile G includes processor specs (O), memory configuration $N[\,_r]$ distinguishing central processing unit (CPU)/Graphics processing unit (GPU usage, battery capacity (A), installed sensors (T), communication interfaces (d) such as Wi-Fi, Long Term Evolution (LTE), or 5G New Radio (NR), and communication frequency range (e). These parameters influence the vehicle's ability to process or offload computational tasks.

RSUs were placed along roadways that help to process and store data close to the network. RSUs were better at processing and managing data than vehicles and at storing and communicating with the internet whenever necessary. It provides quick answers to requests in maps, and videos, and controls traffic while edge servers rely on them.

Edge Server: An edge server F is defined as a three-tuple in equation (3).

$$\boldsymbol{F = \{F_{jc}, D, K\}} \quad (3)$$

The edge server is identified by a unique ID (F_{jc}) and characterized by its computational capacity (D), which includes memory, processing speed, and storage modeled similarly to vehicle hardware specifications. Its geographical location (K) is also a key attribute for optimal placement within the VEC network.

Properties of edge servers in VEC

Dynamic vehicle assignment: Vehicle assignments to clusters at any time s were independent of previous assignments, allowing the system to adapt in real-time to the high mobility and changing network topology of vehicular environments.

Dedicated edge server assignment: Each vehicular cluster was mapped indirectly to a single edge server, ensuring exclusive service per cluster. This approach minimizes resource conflicts and supports the demanding performance requirements of VEC applications.

Many-to-one vehicle-to-server mapping: Multiple vehicles can offload computational tasks to the same edge server, enabling efficient resource utilization and centralized task processing within the VEC framework.

Data from edge servers is uploaded to remote data centers, known as cloud servers, which supply large amounts of computing and storage services over a large area. Using information from vehicles and edge servers,

cloud services can manage the network from one central place and take the best actions. The combination of vehicular terminals, edge servers, and cloud infrastructure makes the VEC system both strong and capable of handling the needs of intelligent transportation management.

With the architectural components established, the server placement strategy in the proposed VEC framework can now be formally defined to optimize performance under dynamic vehicular conditions.

Edge server placement: In the VEC model, the placement of edge servers was modeled by a bipartite graph with two sets: $F$is for edge servers, while V is for client vehicles. Each server $f \in F$ comes with a defined W_{max_f} , showing its maximum vehicle capacity. Communication cost indicates how well a vehicle v works with a server e due to the effects of latency K_{Vf} and energy consumption F_{Vf} . The objective is to determine a good subset F_1 out of F and describe the mapping$\phi: V \to F_1$:, assigning each vehicle to a server to minimize both the total delay and the power used across the system.

Average latency:$K^-$is used to mean the average time taken for vehicles to communicate with edge servers while offloading their tasks. It helps to measure the effectiveness of server placement and matching vehicles to servers in the VEC framework under changing mobility conditions. It is computed as in equation (4).

$$\boldsymbol{K = \frac{1}{|V|}\sum_{v \in V} K_{vf}} \quad (4)$$

The $|\ V\ |$ denotes the total number of vehicles within the VEC network. K_{vf} represents the communication latency encountered by vehicle v during task offloading to edge server f, defined as equation (5).

$$\boldsymbol{K_{vf} = S_{receive} - S_{send}} \quad (5)$$

In this context, S_{send}indicates the timestamp when a vehicle initiates the task offloading request, while $S_{receive}$ marks the moment the vehicle receives the processed response from the edge server.

The goal of the edge server placement was to minimize the average latency K^-, ensuring efficient, low-latency communication for all vehicles within the network.

B) Model formulation

The edge server placement issue in a VEC network is defined in this section to minimize overall energy usage and delay through optimal edge server placement. The decision variables, objective functions, and constraints involved in the problem formulation are detailed below.

Consider a fixed of vehicles $V = \{v_1, v_2 \ldots v_{\mathrm{m}}\}$, a set of edge servers $F = \{f_1, f_2, \ldots, f_n\}$, anda list of possible deployment sites $J = \{j_1, j_2, \ldots, j_n\}$for placing edge servers within the network.

1) Decision Variables:

To model the edge server placement in the VEC network, define decision variables that indicate whether an edge server is deployed at a specific location and how vehicles were assigned to these servers for optimal performance. A_{vf} is a binary decision variable that indicates the connection status between vehicle v and edge server e in equation (6).

$$A_{vf} = \begin{cases} \mathbf{1}, \ \boldsymbol{if\ vehicle\ u\ is\ connected\ to\ edge\ server\ f} \\ \mathbf{0}, \boldsymbol{otherwise} \end{cases} \quad (6)$$

A_{vf} is a binary decision variable indicating the deployment status of an edge server at location j in equation (7).

$$A_{fj} = \begin{cases} 1, if\ an\ edge\ server\ is\ placed\ at\ location\ j \\ 0, otherwise \end{cases} \quad (7)$$

2) Parameters

The parameters in the formulation define the system characteristics essential for optimizing edge server placement in the VEC network. The energy consumption for a vehicle v to offload computational tasks to an edge server e is denoted as F_{vf}. In equation (8).

$$F_{uf} = \left(O_{sw} + O_{qw}\right).S_{comm} \quad (8)$$

Where $O_{sw}$is the vehicle's transmission power, $O_{qw}$is the reception power, and S_{comm} is the time taken for the communication exchange. This metric helps quantify energy efficiency in task offloading scenarios within the VEC environment. The latency experienced by a vehicle v when offloading tasks to an edge server e is denoted as K_{vf} . Equation (9) defines it as the interval of time between the sending of the offloading request and the receiving of the processed response.

$$\boldsymbol{K_{vf} = Receive\ Time - Send\ Time} \quad (9)$$

In the VEC environment, key parameters include O_f, the active power consumption of edge server f; c_{vf}, the distance between vehicle v and edge server f ; D , the maximum number of servers on the edge deployable in the network; and $Capacity_i$, the maximum number of vehicles that a server on the edge i can handle. These factors guide optimal server placement.

3) Objective Function

To minimize overall energy consumption and reduce total latency in the VEC network.

Minimize Total Energy Consumption: Total energy consumption includes the energy used by vehicles to offload tasks (F_{vf}) and the power consumed by active edge servers (O_f). The objective is to minimize the sum of vehicle offloading energy and edge server power across the network in equation (10).

$$Minimize \quad \sum_{v=1}^{N}\sum_{f=1}^{M} A_{vf}F_{vf} + \sum_{f=1}^{m}\sum_{j=1}^{m} A_{fj}o_f \tag{10}$$

Where o_f and A_{fj} indicates vehicle-to-server connections and server placements, respectively.

Minimize Total Cumulative Latency: To reduce the overall communication delay experienced by vehicles when offloading tasks to edge servers. This total latency is calculated as the sum of the individual latencies K_{vf}, defined by the time difference between sending the task and receiving the response expressed in the following equation (11).

$$\boldsymbol{Minimize} \quad \boldsymbol{\sum_{f=1}^{m}\sum_{u=1}^{n} A_{vf}K_{vf}} \tag{11}$$

Where A_{vf} indicates if vehicle v offloads to server f, and K_{vf} is the latency between them.

4) Constraints

The optimization problem includes constraints to guarantee efficient deployment of edge servers and proper assignment of vehicles, ensuring that server capacities were not exceeded and system resources were utilized effectively.

Server Capacity Constraint: Each edge server has a limited capacity, restricting the number of vehicles it can serve. The total vehicles assigned to server f must not exceed its capacityD_f, ensuring balanced load distribution and preventing server overload in equation (12).

$$\boldsymbol{\sum_{u=1}^{M} A_{vf} \leq D_f \; \forall_f} \tag{12}$$

Vehicle Assignment Constraint: To ensure proper task offloading, each vehicle must be assigned to exactly one edge server. This guarantees that every vehicle connects to a single server for processing its tasks, expressed as equation (13).

$$\sum_{f=1}^{F} A_{vf} = 1 \quad \forall_v \tag{13}$$

Restrictions on Edge Server Positioning: The deployment of edge servers within the network is restricted by a maximum allowable number, denoted by D. This constraint confirms that the total quantity of placed edge servers does not exceed D, and is formulated as equation (14).

$$\sum_{f=1}^{m}\sum_{j=1}^{m} A_{fj} \leq D \tag{14}$$

Binary Constraints: The decision variables A_{vf} and A_{fj} are binary, reflecting the discrete nature of the problem. Specifically, a vehicle v is either connected to an edge server f or not, and an edge server is either deployed at location j or not. These binary constraints ensure clear and unambiguous decision-making in the edge server placement and vehicle assignment process within the VEC network in equations (15) and (16).

$$A_{uf} \in \{0,1\}\forall_{u,f} \tag{15}$$

$$A_{fj} \in \{0,1\}\forall_{f,j} \tag{16}$$

3.2 Dataset

For a Vehicular Edge Computing scenario, this 5,811-record task offloading event dataset is used to validate the effectiveness of the proposed SFO-Eff-DNN system. This dataset includes information on task arrival/completion time, processing time, network latency, energy consumption, and vehicle node mobility. The model can learn intricate mobility and network behaviors because to this dataset's capture of dynamic, real-world vehicle settings. This is in line with the framework's goal of optimizing edge server placements and deep neural network settings. It encourages scalability, responsiveness, and efficiency for real-time VEC and smart mobility by facilitating an equitable examination of latency vs. energy trade-offs.

Source:
https://www.kaggle.com/datasets/programmer3/vec-edge-server-offloading-dataset

3.3 Preprocessing Using Min-Max Normalization

To create an energy-efficient optimum structure of a deep neural network for real-time VEC activities with enhanced energy economy, reduced latency, and scalable performance, min-max normalization is applied in the preprocessing stage. The model's convergence is enhanced and a uniformly distributed collection of features is made possible for efficient decision-making for real-time VEC operations by normalizing the input parameters of delay, energy consumption, and vehicle speed between 0 and 1.The value of property B is normalized from $[min_B, max_B]$ to $[new_{min_B}, new_{max_B}]$ using equation (17), which maximizes data representation:

$$\frac{u-min_B}{max_B-min_B}\left(new_{min_B}, new_{max_B}\right) + new_{min_B} \tag{17}$$

In addition to enhancing prediction reliability and preserving a consistent data distribution, this normalization facilitates effective implementation in real-time automotive applications.

3.4 Synergistic fibroblast optimized efficient deep neural network (SFO-Eff-DNN)

The research to improve the DL architecture, the SFO-Eff-DNN suggests a hybrid intelligence framework with edge servers situated in VEC. It relies on the predictive power of Eff-DNN and integrates the ability of the SFO algorithm to adjust itself. Eff-DNN is used to figure out how vehicles move around and how the network changes, while SFO acts like fibroblast cells in real healing to search through lots of different solutions quickly. SFO helps set up Eff-DNN weights, biases, and learning rates to ensure good latency, energy consumption, and ability to scale up or down. As a result of hybridization, the system evades local optima and gradually finds the best solution. The use of real-world data for vehicles confirms that the SFO-Eff-DNN framework can quickly converge, lower the time needed for inference, and help with making energy-efficient decisions in rapidly changing VEC environments. Algorithm 1 represents the proposed SFO-Eff-DNN model working process.

Algorithm 1: SFO-Eff-DNN

$Step\ 1$: $Initialization$

$def\ setup()$:

  $M = 30$ $\#\ Population\ size$

  $N = num\_parameters()$ $\#\ Total\ Eff-DNN\ parameters$

  $max\_iter, rho, tau = 100, 0.5, 5$

  $s, k\_pq, L = 1.0, 0.8, 10.0$

  $data = load\_VEC\_data()$ $\#\ Real-world\ mobility/network\ data$

  $return\ M, N, max\_iter, rho, tau, s, k\_pq, L, data$

$Step\ 2$: $Initialize\ the\ population\ of\ solutions$

$def\ init\_population(M, N)$:

  $return\ [\{'params': rand\_vec(N), 'velocity': rand\_vec(N)\}\ for\ \_\ in\ range(M)]$

$Step\ 3$: $Train\ and\ evaluate\ the\ Eff-DNN\ model$

$def\ evaluate(params, data)$:

  $model = build\_EffDNN(params)$

  $train\_DNN(model, * data)$

  $latency, energy = evaluate\_latency\_energy(model)$

  $return\ latency + energy\ \#\ Simple\ fitness\ function\ (lower\ is\ better)$

$Step\ 4$: $Velocity\ update\ with\ feedback\ and\ local\ correction$

$def\ update\_velocity(ind, past\_pos, rho)$:

  $c = local\_correction(ind['params'])$

  $d = vector\_div(past\_pos, norm(past\_pos))$

  $return\ ind['velocity'] + (1 - rho) * c + rho * d$

$Step\ 5$: $Position\ update$

$def\ update\_position(ind, vel, s, k\_pq, L)$:

$speed = s / (k\_pq * L)$

$direction = vector\_div(vel, norm(vel))$

$return\ ind['params'] + speed * direction$

Step 6: *Main SFO − EffDNN Optimization*

$def\ optimize\_SFO\_EffDNN()$:

$M, N, T, rho, tau, s, k\_pq, L, data = setup()$

$pop, history = init\_population(M, N), []$

$for\ t\ in\ range(T)$:

$for\ ind\ in\ pop$:

$ind['fitness'] = evaluate(ind['params'], data)$

$past = pop\ if\ t < tau\ else\ pop.copy()$

$For\ i, ind\ in\ enumerate(pop)$:

$ind['velocity'] = update\_velocity(ind, past[i]['params'], rho)$

$ind['params'] = update\_position(ind, ind['velocity'], s, k\_pq, L)$

$best = min(pop, key = lambda\ x: x['fitness'])$

$history.append(best['fitness'])$

$return\ best, history$

To improve performance in dynamic Vehicular Edge Computing (VEC) settings, the SFO-Eff-DNN algorithm 1 combines the strength of Efficient Deep Neural Networks (Eff-DNN) with Synergistic Fibroblast Optimization (SFO), an optimization technique inspired by nature. Using actual traffic and network data, the algorithm initializes a population of solutions, each of which represents a set of Eff-DNN parameters, and assesses each according to latency and energy consumption. The approach is perfect for real-time intelligent transportation systems because it ensures quick convergence and improved flexibility by updating its location and velocity depending on fitness input and historical experiences.

Efficient Deep Neural Network (Eff-DNN) The proposed optimized deep learning architecture in VEC makes use of an Eff-DNN to represent how vehicles and networks interact. An Eff-DNN architecture has an input layer, an output layer, and many hidden layers, as shown in Figure 2. The network is set up with six input layers and seven hidden layers, all containing 64 neurons to avoid overfitting. Model complexity and generalization were managed in TensorFlow by setting them as hyperparameters within the layers. It uses input about how vehicles behave and interact to determine the best positioning of the edge servers. The network uses the Rectified Linear Unit (ReLU) function to make its computations non-linear and adjust the weights it uses for learning through backpropagation. The Eff-DNN can provide quick and efficient decisions in ever-changing vehicular environments due to the backpropagation process, which keeps the cost function low. Neuron outputs were computed as follows in equation (18).

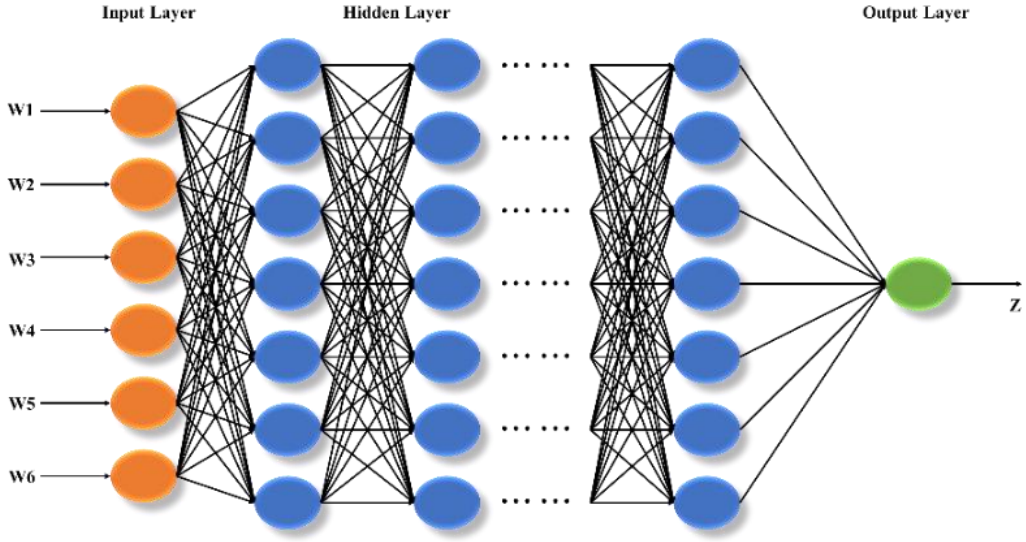

Figure 2: Architecture of Eff-DNN

$$z_r^{m+1} = \sigma(y) = \sigma\left(\sum_{j=1}^{n} \omega_{j_r}^m z_j^m + a_r^{m+1}\right) \quad (18)$$

Where σ(z) represents the activation function, and z_r^{m+1} is the output of the $r-th$neuron in the $(m+1)$-th layer. The weights among the $j-th$ neuron of layer n and the $r-th$ neuron of layer $(m+1)$ are labeled $\omega_{j_r}^m$, and a_r^{m+1}represents the bias term for linear transformations. While training, the loss function compares the predicted outcomes with the desired ones. The model finds the best values for ω and a by minimizing the loss, making the network predict more accurately. The Eff-DNN's loss function is explained in equation (19).

$$f(\theta) = -\frac{1}{m}\sum_m \sum_r s_{mr} \log z_{mr} \quad (19)$$

Where s_{mr} represents the actual value of the $r-th$ sample's $m-th$element, z_{mr} denotes the predicted value for the same element, and θ represents the collection of parameters including weights ω and biases a. Here, M is the total quantity of samples. To reduce overfitting, a dropout mechanism is employed that randomly disables neurons during training, effectively disrupting the network structure and promoting generalization. Furthermore, the proposed method enhances the conventional gradient descent by dynamically adapting the learning rate for improved convergence. The optimization of the parameter set θ is formally defined as equations (20) and (21).

$$\begin{cases} n_s = \beta_1 n_{s-1} + (1-\beta_1)h_s \\ U_s = \beta_2 u_{s-1} + (1-\beta_2)h_s^2 \\ h_s = \nabla_\theta F(\theta_{s-1}) \\ \hat{n}_s = \frac{n_s}{1-\beta_1^s} \\ \hat{U}_s = \frac{u_s}{1-\beta_2^s} \\ \theta_s = \theta_{s-1} - \propto \frac{\hat{n}_s}{\sqrt{\hat{u}_s+\varepsilon}} \end{cases} \quad (20)$$

$$\propto = \propto_0 \beta_3^{\frac{\frac{epoch-num}{M}}{batch-size}} \quad (21)$$

Where $U_s$represents the weighted average of exponentially the squared gradients, while h_s denotes the gradient of the parameters at time s, $n_s$captures the average movement of the gradient, and $\propto_0$is the initial learning rate. The corrected versions of these estimates were denoted by $\hat{U}_s$and $\hat{n}_s$, which improve optimization accuracy. Exponential decay rates β_1 ,$\beta_{2,}and\ \beta_3$ are used to stabilize updates. Additionally, parameters such as batch size (epbatch−size) and current training iterations (ochnum) influence convergence behavior. The improved DNN supports dual operational modes, RDL-1 for normal conditions and RDL-2 for power swing detection, ensuring adaptive command generation aligned with dynamic vehicular network scenarios.

Synergistic fibroblast optimization (SFO)

SFO is modeled after migratory fibroblast cells that heal tissue by responding to the extracellular matrix (ECM). Every solution searches the solution space by varying its position and velocity about diffusion and fitness. This bio-inspired method allows for greater flexibility and avoids local minima, making it appropriate for optimizing neural networks and edge server placement in dynamic VEC settings.

A model based on the adaptive actions of fibroblast cells used in repairing tissues. SFO works on tuning how deep neural networks are set up and arranging edge servers in dynamically changing virtual edge clouds. Much as fibroblasts respond to the extracellular matrix (ECM), SFO looks for solutions in many different ways. Ongoing testing and evaluation of fitness ensure the best solutions use both energy and time efficiently. For this reason, this approach ensures flexibility in the way transportation systems are managed.

The process of biomechanical analysis was strengthened each time by paying attention to interactions with the ECM. As it runs, the program tests different combinations of settings, much like fibroblasts, to improve its outcome. The simulated cells disperse and travel to the most promising areas to avoid getting caught in local minima. Depending on the speed and distribution of the particles, the algorithm updates its next action using the information and trends it has gathered. As a result, the process can handle the trade-offs between speed, performance, and movement better in VEC networks.

Initialization: Within the N -dimensional solution space, initialize a population of physical activity movements $f_{i,}$, where$i = 1,2, \dots, M,$. Each movement is assigned a random position ( )and velocity (v_i). Key parameters such as the diffusion coefficient ρ and movement speed $s$are established.

Fitness Evaluation: For each candidate solution f_i in the N-dimensional space, the fitness function $e(f_i)$ is evaluated iteratively to assess the quality of each movement. This process aims to identify the optimal solution (maximum or minimum) within the evolving search region. Based on the fitness outcomes, the position (b_i) and velocity (v_i) of each movement were updated accordingly using the update rules given by Equations (22) and (23), enabling the algorithm to adaptively explore the solution space.

$$v_i^{(t+1)} = v_i^{(t)} + (1-\rho)c\left(f_i^{(t)}\right) + \rho^* \frac{f_i(t-\tau)}{||f_i(t-\tau)||} \quad (22)$$

Where $t$is the current iteration, τ is the time delay, and the diffusion coefficient ρ is set to 0.5.

$$b_i^{(t+1)} = b_i^{(t)} + s^* \frac{v_i^{(t+1)}}{\left\|v_i^{(t+1)}\right\|} \quad (23)$$

The movement speed t is defined as $s = \frac{s}{k_{pq}L}$, where "k_{pq}" represents the baseline movement rate and L denotes the movement length. The SFO-Eff-DNN hybrid model optimizes edge server placement in dynamic VEC environments by combining adaptive search with deep learning. It efficiently predicts optimal configurations, improves convergence speed, and reduces latency and energy use, making it ideal for real-time intelligent transportation systems.

4 Results and discussion

The experimental setup uses an Intel i7 CPU. Simulations were conducted in Python with TensorFlow and the Veins platform using Cologne traffic traces. The dataset was split using an 80:20 ratio, where 80% was used for training the SFO-Eff-DNN model and 20% was reserved for testing to evaluate performance and generalization.

The SFO-Eff-DNN model includes ReLU-activated layers and dropout, optimized via SFO. Performance was evaluated based on latency, energy use, and server placement accuracy. Key simulation parameters with values aligned to realistic VEC scenarios are presented in Table 2.

Table 2: Key simulation parameters for the SFO-Eff-DNN VEC Framework

Parameter	***Value***
Simulation area	$1500\,m \times 1500\,m$
Simulation time	$200s, 300s, 400s$
Number of edge serve	8
Transmission power	$25\,mW, 30\,mW, 35\,mW$
RSU antenna height	$5\,m$
Receiver sensitivity	$-100\,dBm$
Message size	$100\,bits$
Message frequency	$2\,Hz$
Data rate	$10\,Mbps, 20\,Mbps, 30\,M$
Vehicle speed range	$0 - 100\,km/h$
Edge server CPU capa	$3.5\,GHz$
Edge server memory	$32\ \ GB$

4.1 Offloading ratio

Using time on the x plane and the percentage of tasks offloaded from the vehicle to edge servers on the y plane, Figure 3 shows the offloading ratio (%) in the VEC system over 10 minutes. Starting at 75%, the offloading ratio steadily rises to 89%, reflecting an increasing reliance on edge computation. This upward trend is attributed to enhanced network conditions, adaptive optimization by the SFO-Eff-DNN framework for energy efficiency, or the growing complexity of vehicular tasks that necessitate edge processing. Tracking this metric is crucial in the

research context, as a higher offloading ratio signifies more efficient utilization of edge resources, which directly contributes to lowering vehicle energy consumption and accelerating task processing, thereby improving overall system performance in dynamic ITS environments.

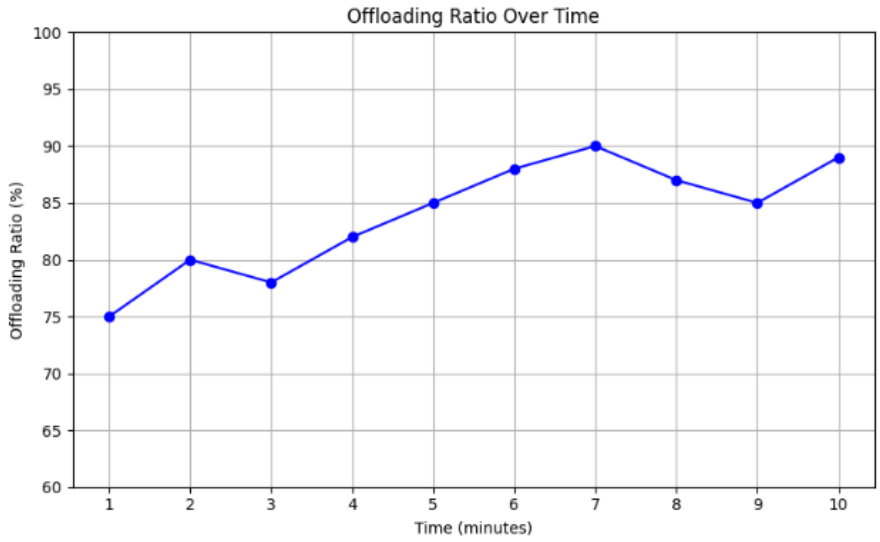

Figure 3: Offloading ratio over time

4.2 SFO-Eff-DNN Pareto Front in VEC

In VEC, the Pareto front for the suggested SFO-Eff-DNN illustrates the relationship between latency and energy use. Figure 4 illustrates that with latency increasing from 50 ms to 70 ms, the energy consumed decreases from about 70 J to 40 J, showing an inverse relationship. All points on the curve are Pareto-optimal, as enhancing one factor would cause a drop in the other. Because of the model's diversity, it is possible to choose configurations for specific needs, such as real-time applications or limited-power cases, proving its effectiveness and adaptability.

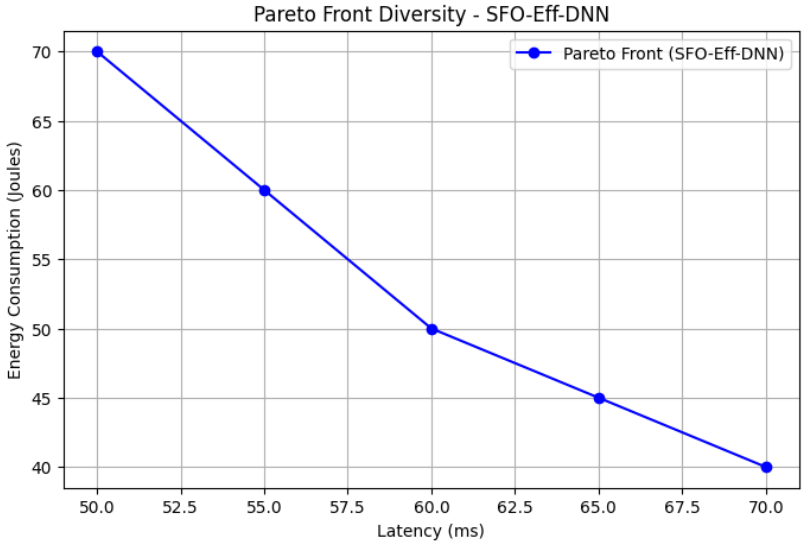

Figure 4: Pareto front diversity of SFO-Eff-DNN in VEC

4.3 Convergence Behavior of SFO-Eff-DNN

Figures 5 (a) and (b) illustrate the convergence behavior of the SFO-Eff-DNN algorithm over 100 optimization iterations for energy consumption and latency. In Figure (a), the minimum energy consumption (blue line) rapidly drops from approximately 0.34 to 0.29 within the first 10 iterations and then stabilizes, indicating that the algorithm quickly identifies energy-efficient configurations. The average energy consumption (green dashed line) also follows a similar decreasing trend, gradually converging toward the minimum, which reflects the population's collective improvement. Similarly, in Figure (b), during the first iterations, the latency drops rapidly and then becomes more stable at a much lower

level. The average latency also decreases and stabilizes around the same value, highlighting consistent performance improvement across the solution space. Overall, these trends confirm that SFO-Eff-DNN achieves efficient and simultaneous convergence toward optimal energy and latency trade-offs.

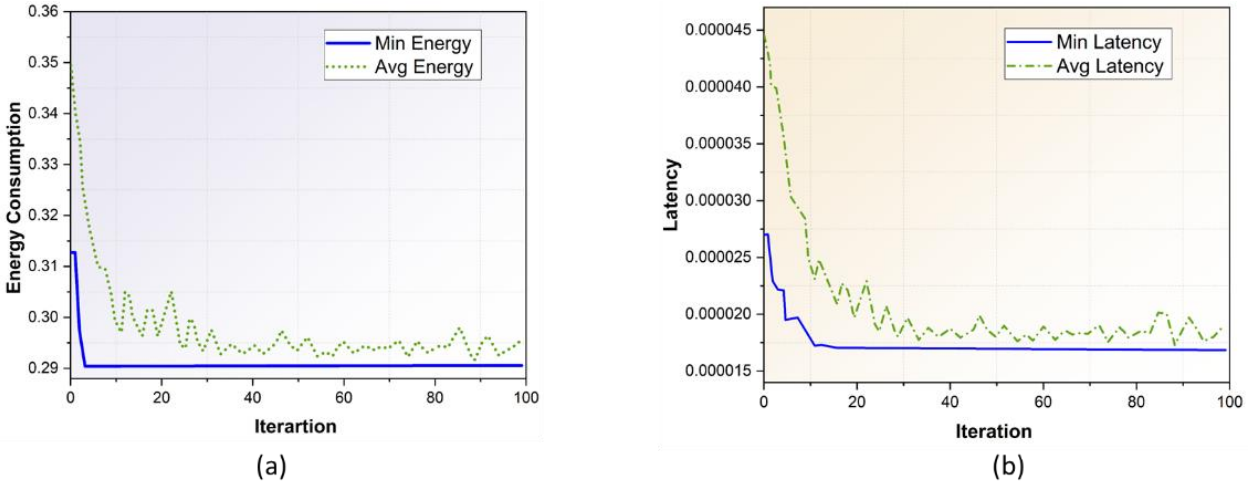

Figure 5: Convergence Behavior of SFO-Eff-DNN (a) energy conception and (b) latency

4.4 Performance analysis

A comparison of several optimization techniques based on their energy consumption and latency performance in vehicular edge computing scenarios is shown in Table 3. Among the evaluated techniques, Particle Swarm Optimization (PSO) (Surayya et al., 2025), Teaching–Learning-Based Optimization (TLBO) (Surayya et al., 2025), and Ant Colony Optimization (ACO) (Surayya et al., 2025), the proposed SFO-Eff-DNN method demonstrates the energy consumption and the latency. This highlights the superior efficiency and responsiveness of the SFO-Eff-DNN framework, making it highly suitable for real-time, energy-aware edge deployments in dynamic vehicular environments. Figure 6 demonstrates the results of the performance analysis.

Table 3: Comparison of optimization methods by energy consumption and latency

Methods	Energy Consumption (J)	Latency (S)
PSO (Surayya et al., 2025)	0.3535	40 µs
TLBO (Surayya et al., 2025)	0.3546	40 µs
ACO (Surayya et al., 2025)	0.3517	60µs
SFO-Eff-DNN (Proposed)	0.3480	30 µs

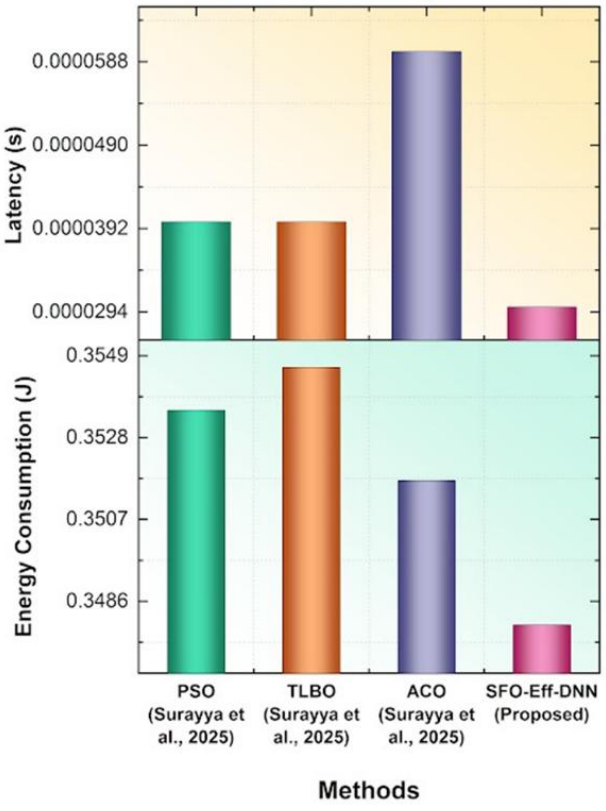

Figure 6: Comparison methods by energy consumption and latency

Analyzing different optimization methods for their energy consumption and latency when used in VEC. SFO-Eff-DNN shows better results than other models by using the least amount of energy (0.3480 J) and having the shortest latency (30 µs). Here, microseconds (µs) are used,

since 1 μs is a millionth of a second, which is needed to ensure fast response times vital in real-time VEC systems. For energy usage, PSO and TLBO lead with 0.3535 J and 0.3546 J, respectively, but both have a latency of 40 μs, while ACO uses 0.3517 J with the highest latency of 60 μs. The results demonstrate that SFO-Eff-DNN offers better results in real-time, energy-sensitive VEC applications.

A comparison of task drop rates for various placement techniques in dynamic VEC situations is shown in Table 4 and Figure 7. In comparison to the generic method's 2.90% (Khamari et al., 2022) dropped task rate, the suggested SFO-Eff-DNN model performs better, attaining a dropped task rate of just 1.83% (Proposed). In latency-sensitive, high-mobility edge computing systems, this research demonstrates how well the SFO-Eff-DNN optimises server workload allocation and lowers service denial.

Table 4: Comparison of task dropped rate between placement strategies in VEC environments

Placement strategies	Dropped Tasks (%)
generic method (Khamari et al., 2022)	2.90%
SFO-Eff-DNN (Proposed)	1.83%

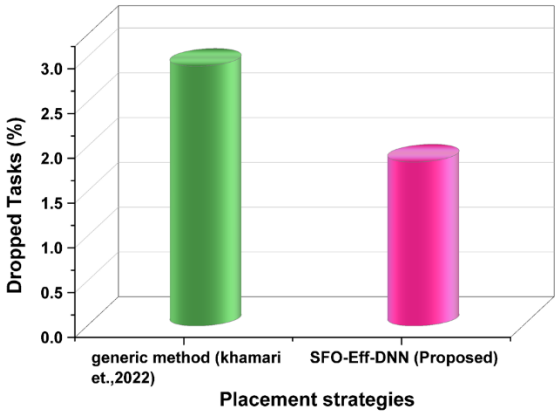

Figure 7: Comparison of Dropped Task Rates for Generic Method and SFO-Eff-DNN

4.5 Discussion

By optimizing the placement of edge servers and DL networks, the SFO-Eff-DNN in VEC reduces latency and conserves energy. The technique has some problems with responding to changes in vehicles and adapting to sudden network changes in VEC settings (Bi et al., 2020). While VECMAN saves energy by sharing resources among electric vehicles, it is difficult for it to accurately predict where vehicles are and to schedule them in situations that are constantly changing (Bahreini et al., 2021). As both PSO and TLBO (Surayya et al., 2025) prioritize low energy over low latency, they may not respond fast enough when ultra-low latency is necessary. ACO (Surayya et al., 2025) can distribute solutions equally, but its slow execution means it is not suitable when time is critical. A PSO, TLBO, and ACO lead with low energy of 0.3535 J, 0.3546 J, and 0.3517 J. Using the SFO-Eff-DNN model, energy costs and latency can be cut down at the same time, compared to older versions. Compared to the generic method's 2.90% dropped task rate (Khamari et al., 2022), the SFO-Eff-DNN's dropped task rate was only 1.83%, indicating its resilience in workload balancing and edge resource utilisation in dynamic vehicular situations. Due to advanced techniques and deep learning, the system reacts to updates in vehicles and can quickly and accurately configure servers for VEC applications.

The computational load brought on by the hybridization of deep learning and evolutionary optimization constitutes one of the key issues, especially during the early phases of training and adaption. Despite its potential for convergence efficiency, iterative optimization can be resource-hungry on edge nodes with constrained computing capacity. Another problem is the system's scalability in high-density vehicle networks. While the model works well for simulations of intermediate scale, more study is needed to determine how it responds and operates in large, real-time vehicular systems with hundreds of nodes. These limitations highlight the significance of future studies that focus on distributed training practices and lightweight optimization versions that can sustain performance without increasing compute demands in practical applications.

5 Conclusion

VEC is a pattern that encourages cloud computing capabilities closer to the network edge services needed for low-latency services, such as auto-corrective driving support, real-time traffic management, and location-based applications. The proposed SFO-Eff-DNN framework is used to optimize deep learning for VEC using modern evolutionary algorithms. To deal with the problem of placing servers at the edge of wireless networks in vehicles, both Synergistic Fibroblast Optimization and deep neural networks were used. It makes use of real travel data to manage how quickly it responds and how much energy it uses, adjusts to any changes in the network, and provides quick results. The data from experiments reveals that SFO-Eff-DNN works with 30 μs latency, 0.3480 J energy consumption, and only 1.83% dropped tasks, making it well-suited for speedy and efficient smart transportation. It strongly supports and adapts to the new directions being taken in VEC deployments. Using simulated movement and experimentation usually does not reflect real-world events or problems, meaning their practical use may not be as effective.

Future scope

Future research should integrate real-time traffic incident data and 5G network slicing to further enhance adaptability. Extending the framework with federated learning for

privacy-preserving model updates across distributed vehicles, and exploring hybrid optimizers that combine SFO with reinforcement learning could improve robustness against unforeseen network disruptions and accelerate convergence in large-scale, heterogeneous VEC deployments.

References

[1] Wan, S., Xu, X., Wang, T., and Gu, Z., 2020. An intelligent video analysis method for abnormal event detection in intelligent transportation systems. IEEE Transactions on Intelligent Transportation Systems, 22(7), pp.4487-4495.DOI: 10.1109/TITS.2020.3017505

[2] Boukerche, A., Tao, Y. and Sun, P., 2020. Artificial intelligence-based vehicular traffic flow prediction methods for supporting intelligent transportation systems. Computer networks, 182, p.107484.https://doi.org/10.1016/j.comnet.2020.107484

[3] Elassy, M., Al-Hattab, M., Takruri, M. and Badawi, S., 2024. Intelligent transportation systems for sustainable smart cities. Transportation Engineering, p.100252.https://doi.org/10.1016/j.treng.2024.100252

[4] Alhilal, A.Y., Finley, B., Braud, T., Su, D. and Hui, P., 2022. Street smart in 5G: Vehicular applications, communication, and computing. IEEE Access, 10, pp.105631-105656.DOI: 10.1109/ACCESS.2022.3210985

[5] Chougule, S.B., Chaudhari, B.S., Ghorpade, S.N. and Zennaro, M., 2024. Exploring computing paradigms for electric vehicles: from cloud to edge intelligence, challenges and future directions. World Electric Vehicle Journal, 15(2), p.39.https://doi.org/10.3390/wevj15020039

[6] Talpur, A. and Gurusamy, M., 2021. Drld-sp: A deep-reinforcement-learning-based dynamic service placement in edge-enabled internet of vehicles. IEEE Internet of Things Journal, 9(8), pp.6239-6251.DOI: 10.1109/JIOT.2021.3110913

[7] Zaki, A.M., Elsayed, S.A., Elgazzar, K. and Hassanein, H.S., 2024. Quality-Aware Task Offloading for Cooperative Perception in Vehicular Edge Computing. IEEE Transactions on Vehicular Technology.DOI: 10.1109/TVT.2024.3444591

[8] Zhao, L., Li, T., Zhang, E., Lin, Y., Wan, S., Hawbani, A. and Guizani, M., 2023. Adaptive swarm intelligent offloading based on digital twin-assisted prediction in VEC. IEEE Transactions on Mobile Computing, 23(8), pp.8158-8174.DOI: 10.1109/TMC.2023.3344645

[9] Shen, B., Xu, X., Qi, L., Zhang, X. and Srivastava, G., 2021. Dynamic server placement in edge computing toward the internet of vehicles. Computer Communications, 178, pp.114-123.https://doi.org/10.1016/j.comcom.2021.07.021

[10] Peyman, M., Fletcher, T., Panadero, J., Serrat, C., Xhafa, F. and Juan, A.A., 2023. Optimization of vehicular networks in smart cities: from agile optimization to learn heuristics and sim heuristics. Sensors, 23(1), p.499.https://doi.org/10.3390/s23010499

[11] Ebrahimi Mood, S., Rouhbakhsh, A. and Souri, A., 2025. Evolutionary recurrent neural network based on equilibrium optimization method for cloud-edge resource management in Internet of Things. Neural Computing and Applications, 37(6), pp.4957-4969.https://doi.org/10.1007/s00521-024-10929-1

[12] Vijayakumar, P., Rajalingam, P. and Rajeswari, S.V.K.R., 2021. Edge Computing Optimization Using Mathematical Modeling, Deep Learning Models, and Evolutionary Algorithms. Simulation and Analysis of Mathematical Methods in Real-Time Engineering Applications, pp.17-44.https://doi.org/10.1002/9781119785521.ch2

[13] Yang, Z., Zhang, S., Li, R., Li, C., Wang, M., Wang, D. and Zhang, M., 2021. Efficient resource-aware convolutional neural architecture search for edge computing with Pareto-bayesian optimization. Sensors, 21(2), p.444.https://doi.org/10.3390/s21020444

[14] Li, Z., Yu, H., Fan, G., Zhang, J. and Xu, J., 2024. Energy-efficient offloading for DNN-based applications in edge-cloud computing: A hybrid chaotic evolutionary approach. Journal of Parallel and Distributed Computing, 187, p.104850.https://doi.org/10.1016/j.jpdc.2024.104850

[15] Pustokhina, I.V., Pustokhin, D.A., Gupta, D., Khanna, A., Shankar, K. and Nguyen, G.N., 2020. An effective training scheme for deep neural networks in edge computing enabled Internet of Medical Things (IoMT) systems. IEEE Access, 8, pp.107112-107123.DOI: 10.1109/ACCESS.2020.3000322

[16] Loni, M., Sinaei, S., Zoljodi, A., Daneshtalab, M. and Sjödin, M., 2020. DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems. Microprocessors and Microsystems, 73, p.102989.https://doi.org/10.1016/j.micpro.2020.102989

[17] Saheed, Y.K., Abdulganiyu, O.H. and Ait Tchakoucht, T., 2024. Modified genetic algorithm and fine-tuned long short-term memory network for intrusion detection in the Internet of Things networks with edge capabilities. Applied Soft Computing, 155, p.111434.https://doi.org/10.1016/j.asoc.2024.111434

[18] Bi, J., Yuan, H., Duanmu, S., Zhou, M. and Abusorrah, A., 2020. Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization. IEEE Internet of Things Journal, 8(5), pp.3774-3785.DOI: 10.1109/JIOT.2020.3024223

[19] Chen, Z., Hu, J., Chen, X., Hu, J., Zheng, X. and Min, G., 2020. Computation offloading and task scheduling for DNN-based applications in cloud-edge computing. IEEE Access, 8, pp.115537-115547.DOI: 10.1109/ACCESS.2020.3004509

[20] You, Q. and Tang, B., 2021. Efficient task offloading using particle swarm optimization algorithm in edge computing for the industrial internet of things. Journal of Cloud Computing, 10, pp.1-11.https://doi.org/10.1186/s13677-021-00256-4

[21] Yousif, A., Bashir, M.B. and Ali, A., 2024. An evolutionary algorithm for task clustering and scheduling in IoT edge computing. Mathematics, 12(2), p.281.https://doi.org/10.3390/math12020281

[22] Xiao, H., Zhao, J., Pei, Q., Feng, J., Liu, L. and Shi, W., 2021. Vehicle selection and resource optimization for federated learning in vehicular edge computing. IEEE Transactions on Intelligent Transportation Systems, 23(8), pp.11073-11087.DOI: 10.1109/TITS.2021.3099597

[23] Bahreini, T., Brocanelli, M. and Grosu, D., 2021. VECMAN: A framework for energy-aware resource management in vehicular edge computing systems. IEEE Transactions on Mobile Computing.DOI: 10.1109/TMC.2021.3089338

[24] Jiang, H., Cai, J., Xiao, Z., Yang, K., Chen, H. and Liu, J., 2025. Vehicle-Assisted Service Caching for Task Offloading in Vehicular Edge Computing. IEEE Transactions on Mobile Computing.DOI: 10.1109/TMC.2025.3545444

[25] Surayya, A., Hussain, M.M., Reddy, V.D., Abdul, A. and Gazi, F., 2025. Evolutionary Algorithms for Edge Server Placement in Vehicular Edge Computing. IEEEAccess.10.1109/ACCESS.2025.3566172

[26] Luo, X., Liu, D., Huai, S. and Liu, W., 2021, February. HSCoNAS: Hardware-software co-design of efficient DNNs via neural architecture search. In 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 418-421). IEEE.https://doi.org/10.23919/DATE51398.2021.9473937

[27] Odema, M., Rashid, N., Demirel, B.U. and Al Faruque, M.A., 2021, December. LENS: Layer distribution enabled neural architecture search in edge-cloud hierarchies. In 2021 58th ACM/IEEE Design Automation Conference (DAC) (pp. 403-408). IEEE. https://doi.org/10.1109/DAC18074.2021.9586259

[28] Abreha, H.G., Hayajneh, M. and Serhani, M.A., 2022. Federated learning in edge computing: a systematic survey. Sensors, 22(2), p.450. https://doi.org/10.3390/s22020450

[29] Talpur, A. and Gurusamy, M., 2021, April. Reinforcement learning-based dynamic service placement in vehicular networks. In 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring) (pp. 1-7). IEEE. https://doi.org/10.1109/VTC2021-Spring51267.2021.9448645

[30] Khamari, S., Ahmed, T. and Mosbah, M., 2022, December. Efficient edge server placement under latency and load balancing constraints for vehicular networks. In GLOBECOM 2022-2022 IEEE Global Communications Conference (pp. 4437-4442). IEEE.https://doi.org/10.1109/GLOBECOM48099.2022.10000721