[1]Jian Feng
[2]Xu Chen*

Optimized Meta-path extracted graph neural network for embedded computer performance evaluation model

Abstract: - The process of evaluating the effectiveness, speed, and general capabilities of a computer system that is embedded inside a larger device or system and created for a particular purpose is known as embedded computer performance evaluation. Specialized computing systems called embedded systems are built into a wide range of gadgets, including medical, industrial, , automotive and consumer electronics. In this manuscript, Optimized Meta-path extracted graph neural network for embedded computer performance evaluation model(MEGNN-EC-PEM) is proposed. Initially, the input data is obtained from real-time sensor measurements and system metrics for model training and testing. The input image is pre-processed using Orthogonal Master-slave Adaptive Notch Filter (OMANF) and it removes the noise from the collected data. Then, the pre- processed data are fed to embedded computer performance using Meta-path extracted graph neural network (MEGNN). In general, MEGNN does not express adapting optimization techniques to determine optimal parameters to assure precise embedded computer performance evaluation model. Hence, the Hunter–prey optimization algorithm (HPOA) is used to optimize Meta-path extracted graph neural network which accurately categorized embedded computer performance. Then, the proposed MEGNN-EC-PEM is implemented and the performance metrics like Modelling Accuracy, latency, Throughput, memory cost and Energy Consumption are analyzed. The performance of the MEGNN-EC-PEM approach attains 19.41%, 20.08% and 32.57% higher modelling accuracy, 22.41%, 23.08% and 24.57% lower latency and 23.01%, 23.08% and 24.07% higher throughput when analyzed through existing techniques like a state-based modelling approach for effective performance evaluation of embedded system architectures at transaction level (SMA-EPE-ESAT), Evaluating the performance of pre-trained convolutional neural network for audio classification on embedded schemes for anomaly detection in smart cities (PCNN-ACES-ADSC) and Animal behaviour classification via deep learning on embedded systems (ABC-VDL-ES) methods respectively.

Keywords: Computer performance evaluation model, embedded systems, Hunter–prey optimization algorithm, Meta-path extracted graph neural network, Orthogonal Master-slave Adaptive Notch Filter.

I. INTRODUCTION

The integration of high-performance applications and the enhancement of mobility and communication capabilities are the current trends in embedded systems design in the consumer space [1]. These features have a substantial impact on hardware resources implemented, program complexity, and system architectures [2]. Memory, specialized hardware blocks, fully programmable CPU cores, and standard interface modules are examples of the various components that make up a multicore platform. Chip technology advancements makes possible to combine more resources. As a result, massively parallel architectures organized according to application type will be used. Furthermore, network infrastructure is a practical replacement for bus-based communication to increase the scalability of such platforms [3- 5]. System architecture in this context refers to the process of determining system applications' best allocation on platform resources and configuring memory, processor, and communication resource characteristics based on functional and non-functional needs. While non-functional requirements are used to accurately adjust associated architecture's characteristics, functional requirements specify what the designer hopes to implement [6]. Cost, power consumption, and time restrictions are common non-functional criteria for embedded systems. To find viable architectures, the design space is explored in accordance with these specifications [7]. Next, the candidate architectures' performances are assessed and contrasted. Especially in the development process, accurate assessment of non-functional features and quick design space exploration have become essential to avoiding expensive design iterations and maintaining a short design time [8, 9]. System architects need particular methodologies and tools to help them create trustworthy models since system complexity is increasing and evaluation of architectural performances is necessary [10]. As mentioned earlier, while creating models for architecture performance evaluation, the Y-chart model's principles are typically adhered to. This approach integrates an application model onto a model of the platform that is being evaluated, and the resulting description is then evaluated through the use of analytical or simulation methods [11]. Analytical techniques are applied in lieu of formal analysis on architectural models. If deterministic or worst-case behaviour is a realistic assumption for the architecture under examination, then these methods work well [12-14]. Simulation techniques are based on simulating an architecture under examine and executing it in response to a predetermined set of stimuli [15].

[1]Quality Management Department, Sichuan Post and Telecommunication College, Chengdu Sichuan 610067, China
[2]Department of Engineering Technology, Meishan Vocational & Technical College, Meishan, Sichuan, 620010, China
*Corresponding Author Email: Chenxu5377@sina.com

In contrast to analytical techniques, the system model's dynamic and nondeterministic impacts need to be investigated using simulation techniques [16]. In order to compare the performances of a small set of possible architectures, as demonstrated in methods, simulation results are obtained.The definition of efficient simulation-based methods focuses on reducing modelling complexity and increasing simulation speed. There exists a clear correlation between the accuracy and speed of simulation and the amount of abstraction used to model the system architecture [17, 18]. Computational modelling and communication modelling are substantially separable and can be defined at different levels of abstraction on application and platform sides. The Transaction Level Modelling (TLM) approach is one simulation-based technique that has garnered a lot of attention lately from the scientific and industrial worlds in an effort to increase system productivity and design [19]. After that, architecture models are simulated to assess resource utilization in relation to a specific set of stimuli. Nevertheless, TLM currently lacks reference techniques that lessen modelling efforts and make it easier to create and modify performance models of system architectures.

A. Problem statement and Motivation behind this Research work

The application model can simulate anticipated memory and processing capacity while taking into account different embedded system architectures is the difficult part. The intricate relationships and patterns of resource consumption prevalent in a variety of embedded designs are frequently too complicated for current models to adequately capture. In order to effectively replicate processor and memory resources across various architectures and enable accurate performance evaluation and embedded system optimization, it is necessary to establish rigorous modelling approaches.

In this paper, specific transaction level modelling approach for evaluating the hardware/software architectures' performance is provided. The proposed method is based on a general execution paradigm with minimal modelling effort. Models that have been created are utilized to simulate expected processing and memory resources in accordance with different architectures.

B. Contribution

The major contribution of this paper includes,

- This approach involves usingOptimized MEGNN for embedded computer performance evaluation model.
- A MEGNN on embedded devices to decrease inference time and enhance prediction accuracy.
- The embedded system model makes it simple for developers to refocus on the new approach models and user requirements and assessment of the effectiveness of a generalist development environment in terms of model integration with embedded systems.

Remaining manuscript is organized as: segment 2 presents literature survey, segment3 defines proposed approach, segment4 illustrates result and discussion, segment5 offers conclusion.

II. LITERATURE SURVEY

Numerous researches were presented depend on embedded computer performance evaluation model. A few works are reviewed here,

Gorospe et al. [20] have presented a generalization performance study utilizing deep learning networks in embedded systems. Here, the development of an environment built on Mbed OS and Tensor Flow Lite, which was embedded in any general-purpose device, enables the introduction of deep learning embedded systems embedded apparatus. When compared to competing commercial systems, the experiments shown here demonstrate how competitive the suggested system. Also, real-time systems was implemented without transferring all data to centralised servers thanks to the advancements in edge computing, that focus on integrating artificial intelligence as close to the client as feasible. The presented method provides higher modelling accuracy but it provides lower throughput.

Lamrini et al. [21] have suggested Analyzing the effectiveness of convolutional neural networks that have already been trained for audio classification on embedded systems in order to detect anomalies in smart cities. In the suggested case, evaluating an already-established, trained method was necessary for using Raspberry Pi (RPi) and TPU platforms other than laptops.. By precisely classifying sounds using Machine Learning (ML) classifiers that have been trained, Environmental Sound Recognition plays a important role in smart cities. Cities that have examined ambient sounds to get information and insights find the presented application useful. The presented method provides higher memory cost but it provides lower throughput.

Arablouei et al. [22] have presented use of deep learning on embedded systems to classify animal behaviour. Th the suggested method, the multilayer and a sequence of finite- and infinite-impulse-response (FIR) and IIR filters achieve feature extraction and classification at the same time. The used IIR and FIR filters were viewed as specific types of recurrent and convolutional neural network layers, respectively. They evaluate the effectiveness of the presented strategy using two real-world datasets collected from a total of eighteen grazing beef cattle using collar tags. The presented method provides higher energy consumption but it provides lower modelling accuracy.

Zu, [23] have presented deep learning parallel processing and assessment for processors with embedded system clustering architecture. Here, the convolutional neural network implementation and parallel optimization methods in the Parallels platform. The convolutional neural network's parallel optimization approach on the heterogeneous multi-core system's clustering architecture processor was provided. The next section develops the convolutional neural network system's function and examines at the high-performance Parallels platform implementation of the convolutional neural network. The presented method provides higher latency but it provides lower modelling accuracy.

Gagliardi et al. [24] have suggested an AI-Based Real-Time Classification System for Road Surface Analysis and an Embedded System for Acoustic Data Processing. The presented system was capable of assessing the quality of road infrastructure's pavement. The embedded system uses integrated artificial intelligence (AI) technologies to capture, process, and classify the wheel-road contact acoustic data in real time, enabling it to assess the general condition of the roadways. The dataset needed to train a convolutional neural network (CNN), measurements were made around Pisa using a car that was traveling at various cruising speeds. This method provides higher modelling accuracy but it provides lower memory cost.

Cornetta and Touhafi [25] have presented design and evaluation of a new machine learning framework for IoT along embedded devices. Here the most widely used machine learning algorithms are reviewed and thoroughly analyzed, with a focus on those that are better suited for use on embedded devices with limited resources. An abundance of new platforms and low-cost, high-performance embedded devices are hitting the market. Some of them were capable of being connected to external hardware accelerators for Machine Learning (ML) algorithms, or they feature internal GPUs. The presented method provides higher throughput but it provides lower modelling accuracy.

III. PROPOSED METHODOLOGY

In this proposed methodology, MEGNN-EC-PEM is discussed for identifying embedded computer performance evaluation model. The model maintains the collection of preliminary data by precisely determining the existence, location and attributes that require more processing. These phases endure major three processes like system model, preprocessing, and embedded computer performance evaluation model succeeding sectors. The proposedMEGNN-EC-PEMis represented by Figure 1.

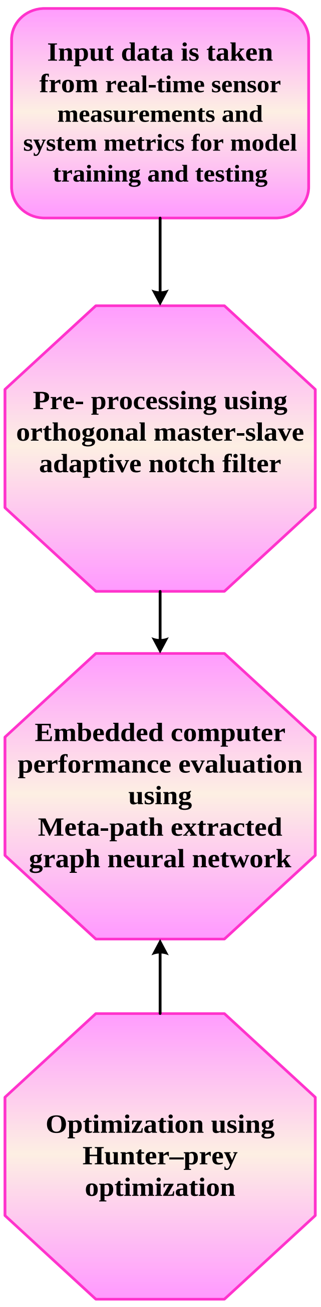
Input data is taken from real-time sensor measurements and system metrics for model training and testing
Pre- processing using orthogonal master-slave adaptive notch filter
Embedded computer performance evaluation using Meta-path extracted graph neural network
Optimization using Hunter–prey optimization

Figure1: Block Diagram for proposed MEGNN-EC-PEM Method

A. Data Acquisition

In this section, data are takenfrom real-time sensor measurements and system metrics for model training and testing [26] dataset. The quantitative evaluations and parameters are used to analyse the effectiveness and performance of machine learning models during the training and testing phases are referred to as measurements and system metrics for model training and testing. These measures help assess a model's ability to generalize to new, unknown data and how well it learns from the data during training.

B. Considered Modelling Approach for Performance Evaluation of System Model

This section's modelling method aims to develop models for the purpose of assessing the resources that make up system designs. It is noted that a thorough description of all system functionalities is not necessary for a system architecture model to exist. The architecture model in the approach under consideration integrates the structural description of the system application with the non- functional features pertinent to the hardware and software resources under consideration. Resources are used in sequences of processing delays interspersed between transferred transactions.

C. Pre-processing using Orthogonal Master-Slave Adaptive Notch Filter

In this section, OMANF[27] technique is utilized which used to remove the noise from the collected input data. When compared to non-adaptive filters, adaptive filters are meant to deliver efficient filtering with embedded system model. This can be useful in real-time applications or settings when evaluating embedded system is constrained. Notch filters are especially good at rejecting interference from certain frequencies. The OMANF adaptive nature may allow it to respond to changes in interference characteristics. From the above discussion, that for orthogonalmaster-slave ANF, to obtain the accurate tracking results, the evaluating system adaptation is important. Thus it is given in equation (1),

$$\omega = -\frac{\gamma}{2\xi} x_1\left(\bar{x}_1 + \hat{\omega}^2 x_1\right) \tag{1}$$

Where, ω denotes the input data; ξ denotes the response speed of filter; $\hat{\omega}$ denotes the activities that compose system model; $\bar{x}_1$ implies set of input relation and x_1 denotes as state variable. The system adaptation is implemented when it gets close to the specified embedded performance. Thus it is given in equation (2).

$$\omega = -\frac{\gamma}{2\xi} x_1^2\left(\hat{\omega}^2 - \omega_1^2\right) \tag{2}$$

Here, ω is denotes input data; x_1 denotes as state variable; ξ denotes the response speed of Filter; γ is denotes degrades the performance; $\hat{\omega}^2$ denotes fundamental embedded evaluating system and ω_1^2 is denotes the length of the system model. The system convergence is accompanied by the evaluated computer model, which degrades the performance of computer model estimation. Thus it is given in equation (3),

$$x_1^2 = \frac{B^2}{2\omega_1^2}\left(1 + \cos(2(\omega_1 s + \theta_1))\right) \tag{3}$$

Here, B denotes as amplitude; θ is denotes the initial phase; s is denotes the time period of the input data.; ω_1^2 is denotes the linkage of the system model. This system models, enable early performance assessment, and facilitate quick design space. Thus it is given in equation (4)

$$\omega = -\frac{\gamma}{2\xi\omega_1}\left[B^2\left(\hat{\omega} - \omega_1\right) + B^2\left(\hat{\omega} - \omega_1\right)\cos(2(\omega_1 s + \theta_1))\right] \tag{4}$$

Here, ω is denotes flux linkage; γ denotes degrades the performance; ξ denotes the response speed of filter; ω_1 denotes the length of the system model; B denotes as time consumption of the system model; $\hat{\omega}$ denotes set of activities that compose system model and s denotes the time period of input data. Evaluating software and hardware designs and managing the growing complexity of embedded systems. After, the embedded system is correspondingly removing the noise is using given in equation (5)

$$\omega = -\gamma\left(\hat{\omega} - \sqrt{\bar{x}_{1\alpha}^2 + \bar{x}_{1\beta}^2} \Big/ \sqrt{x_{1\alpha^2} + x_{1\beta^2}}\right) \tag{5}$$

Where, ω is denotes flux linkage; γ is denotes degrades the performance; $\hat{\omega}$ denotes set of activities that compose the system model; x_1 denotes as state variable.Finally, the noise is removed from the collected dataset. Then, the pre-processing data are fed to MEGNN.

D. Embedded computer performance evaluation model using Meta-path extracted graph neural network

In this section, embedded computer performance evaluation model using MEGNN [28] is discussed.A particular type of MEGNN is made to deal with heterogeneous graphs, in which nodes and edges might have various kinds and meanings. In order to overcome this difficulty, MEGNNs include meta-paths, which are preset lists of edge and node kinds that outline the ideal connections between nodes in the graph. These meta-paths offer an organized method of the network while capturing the performance evaluation between various node kinds. Without taking heterogeneity into account, apply MEGNN directly to heterogeneous graphs. Thus it is given in equation (6)

$$M^{(l)} = agg^{(l)}(H^{(l-1)}; A^{(l)}) \tag{6}$$

Here, $M^{(l)}$ denotes meta- path graph; $agg^{(l)}$ is denotes the aggregation function; $H^{(l-1)}$ is denotes the hidden function and $A^{(l)}$ signifies adjacency matrix. The model benefits from the meta-path based graphs, which explicitly capture data of high-order neighbours that infer meta-path semantics. This is given in equation (7)

$$H^{(l)} = update^{(l)}(H^{(l-1)}, M^{(l)}) \tag{7}$$

Here, $H^{(l)}$ represents the hidden feature of nodes in l^{th} neural network; $update^{(l)}$ denotes the combine functions of node layer; $H^{(l-1)}$ is denotes the hidden function and $M^{(l)}$ denotes the general graph neural network. The initial features in 0^{th} layer are taken as attributes node. Thus it is given in equation (8)

$$H^{(0)} = X \tag{8}$$

Here, $H^{(0)}$ is denotes the initial value of hidden function and X is denotes the attributes matrix. To illustrate the connection between MEGNN and meta-path based connectivity for the simplicity. Thus it is given in equation (9).

$$agg^{(l)}(H^{(l-1)}; A^{(l)} = \hat{A}^{(l)} H^{(l-1)} W^{(l)}) \tag{9}$$

Here, $agg^{(l)}$ is denotes the aggregation function; $H^{(l-1)}$ is denotes the hidden function; $A^{(l)}$ denotes the adjacency matrix; $\hat{A}^{(l)}$ denotes the symmetric normalized adjacency matrix and $W^{(l)}$ signifies linear transformation in the l^{th} layer. The execution model is based on a particular computing technique designed to increase the transaction level models' simulation speed and the embedded computer performance evaluating model is given in equation (10)

$$update^{(l)}(H^{(l-1)}, M^{(l)} = \sigma(M^{(l)}) \tag{10}$$

Here, $update^{(l)}$ is denotes the combine functions of node layer; $H^{(l-1)}$ is denotes the hidden function; $M^{(l)}$ denotes the general graph neural network and σ denotes the non- linear activation function.Finally, MEGNN accurately evaluated the embedded computer performance. Here, HPOA is employed to enhance the MEGNN, for tuning the weight and bias parameter of MEGNN.

E. Optimization using Hunter–prey optimization Algorithm

The proposed HPOA is utilized to enhance weights parameters $M^{(l)}$ of proposed MEGNN [29].The HPOA is a metaheuristic optimization method inspired by the pursuit-evasion dynamics between predators (hunters) and their prey in particular. Like many other optimization algorithms, this one looks for the optimal solutions to problems, often within complex search domains. In nature, animals communicate with one another in a variety of ways. Predator and prey behavior may interact in any of these ways. The hunter-prey cycle, one of the most important discoveries in population biology, has generated a great deal of debate among ecologists. There are many different ways for animals to hunt.

1) Stepwise Procedure for HPOA

Here, step by step procedure for obtaining appropriate MEGNN values using HPOA is described here. To creates a uniformly distributed population for enhancing the ideal MEGNN parameters. The entire step process is presented in below,

Step 1: Initialization

Initial population of HPOA is, initially generated by randomness. Then the initialization is derived in equation (11).

$$x_i = rand(1,d).*(ub - lb) + lb \tag{11}$$

Here, x_i denotes the hunter position or prey; lb denotes the minimum value for the hunter variables; ub denotes the maximum value for the hunter variables and d denotes the number of dimensions.

Step 2: Random generation

The input weight parameter $M^{(l)}$ developed randomness via HPOA method.

Step 3: Fitness function

Initialized value creates random solution. It is calculated by optimizing parameter. Then the formula is derived in equation (12)

$$Fitness\,Function = optimizing\ M^{(l)} \tag{12}$$

Step4: Exploration Phase

During the exploration stage, the well-thought-out technique and enhanced random nature investigates various search space area. Typically, the exploration phase is followed by the exploitation stage. At this point, the algorithm looks for good solutions and refines the search process by circling options. The lower and upper boundary of the hunter position is give different type of dimensions. Thus it is given in equation (13)

$$\begin{aligned} lb &= [lb_1, lb_2, \ldots.., lb_d], \\ ub &= [ub_1, ub_2, \ldots.., ub_d] \end{aligned} \tag{13}$$

Here, lb signifies the lower boundary of hunting position , ub denotes the upper boundary of hunting position. The values and orders of the numbers are random and change with each iteration. Thus it is given in equation (14)

$$C = 1 - it\left(\frac{0.98}{MaxIt}\right) \tag{14}$$

Here, C denotes the balance parameter among exploration phase of the course of iterations; it denotes the current iteration value , $MaxIt$ denotes the maximal iteration and Fig 2 shows the corresponding flowchart.

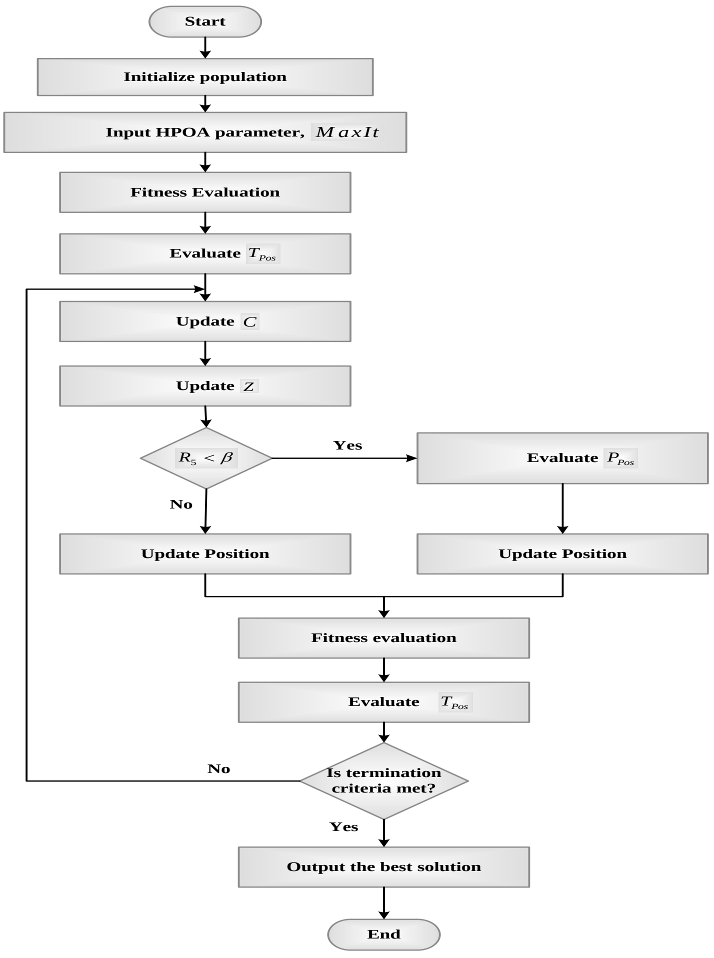

Figure2: Flowchart of HPOA optimizing for MEGNN

Step5: Exploitation phase for optimizing $M^{(l)}$

According to previously acquired information, hunters focus their search efforts on areas of the search space where they perceive prey (ideal solutions) to be located. This entails pointing hunters in the direction of regions with improved solution features or fitness values.By effectively utilizing the data present in the search space, Hunter-Prey Optimization Algorithms seek to iteratively refine and improve solutions, ultimately resulting in the identification of the optimal or nearly optimal solution for the given optimization problem. The prey distance of every search agents from the mean position is monitored. Thus it is given in equation (15)

$$\mu = \frac{1}{n}\sum_{i=1}^{n}\vec{x}_i \tag{15}$$

Here, μ denotes the average position of the prey hunting; n denotes the count of hunting position and $\vec{x}_i$ denotes objective function. The hunting scenario, when the hunter takes prey, the prey dies, and the next time, the hunter moves to the new pray. To solve the problem using the equation (16)

$$kbest = round(C \times N) \tag{16}$$

Here, $kbest$ denotes the average hunting position; C denotes balance parameter among exploitation phase of the course of iterations , N denotes the number of search agent.

Step 6: Termination

The weight parameter value of generator $M^{(l)}$ Meta-path extracted graph neural network is optimized by utilizing HPOA; and it will repeat step 3 till it obtains its halting criteria $x_i = x_i + 1$ is met. Then MEGNN-EC-PEM effectively assesses the quality of embedded computer performance evaluation model higher modelling accuracy, lessening latency with error

IV. RESULT AND DISCUSSION

The experimental results of MEGNN-EC-PEM are discussed. The simulation is implemented in a specific modelling framework based on the System C language. Attained result of MEGNN-EC-PEM method is analyzed with existing like SMA-EPE-ESAT [20], PCNN-ACES-ADSC [21] and ABC-VDL-ES [22] respectively.

A. Performance measures

Performance of proposed method is examined utilizing Modelling Accuracy, latency, Throughput, memory cost and Energy Consumption performance metrics.

1) Modelling Accuracy

Modelling accuracy is the degree to which a mathematical or computer model faithfully simulates the system or phenomenon it is intended to represent or predict in the actual world. It's a metric for how well the model's predictions match actual behaviour or outcomes from studies or empirical data. Thus it is given in eqn (17)

$$Modelling\,Accuracy = \frac{(TP + TN)}{(TP + FP + TN + FN)} \tag{17}$$

Here, TP represents True Positive; TN represents True Negative; FP represents False Positive and FN represents False Negative.

2) Latency

It is the interval of time that passes among the beginning of a process and its eventual completion or execution. Thus it is given in equation (18)

$$Latency = End\,Time - StartTime \tag{18}$$

Here, $End\,Time$ is denotes the time when the process or operation completes and $StartTime$ is denotes the time when the process or operation begins.

3)Throughput

Throughput is derived with the help of equation (19),

$$Tp = \frac{TotalNo.of\ packets\,delivered * p_s}{T} \tag{19}$$

Here p_S means the size of the packet.

4) Memory cost

The amount of memory space used by a system, process, or application is referred to as memory cost. The amount of variables, the size and complexity of the data structures utilized, and the general software design all have an impact on memory usage. Thus it is given in equation (20)

$$MC = V + S + O \tag{20}$$

Here, MC is denotes the memory cost; V is denotes the size of variable; S is denotes the size of data structure and O is denotes the overhead of the memory cost.

5) Energy Consumption

Energy consumption is the quantity of energy used by a system, piece of machinery, or process over a given length of time. Evaluating computing and electronics energy consumption is critical to understanding device power requirements, operating costs, and environmental impact. Thus it is given in equation (21)

$$E_{consumptio} = \sum_{i\in mi}\left(\left((1+\alpha)P_{t\times i} + Pc\right)\frac{L}{W\times b_i} + E_{tr}\right) + \sum_{j\in ti} P_i \frac{C}{F_i} \tag{21}$$

Where, $P_{t\times i}$ denotes transmission power; Pc denotes circuit consumption power; E_{tr} represented as mode transition energy and P_i denoted as power level task.

B. Performance analysis

Figure 3 to 7 determines the simulation outcomes of MEGNN-EC-PEM technique. The MEGNN-EC-PEM is analysed to the existing SMA-EPE-ESAT, PCNN-ACES-ADSC and ABC-VDL-ES Respectively.

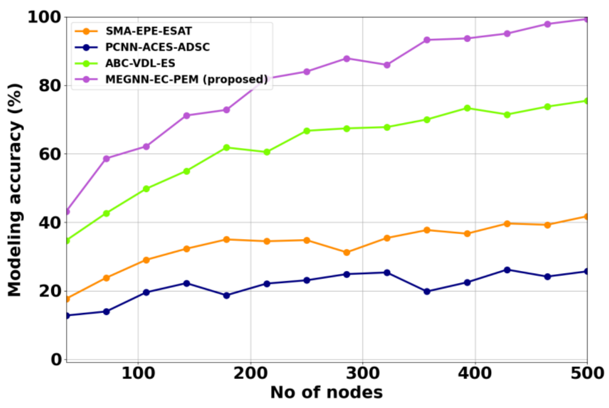

Figure3: Performance analysis of Modelling Accuracy

Figure 3 depicts Modelling Accuracy analysis. The MEGNN-EC-PEM attains 19.41%, 20.08% and 32.57% higher modelling accuracy at number of node taken for 100; 17.12%, 23.41% and 33.90% higher modelling accuracy at number of node taken for 300; 24.33%, 25.36% and 26.95% higher modelling accuracy at number of node taken for 500; which is analyzed with SMA-EPE-ESAT, PCNN-ACES-ADSC and ABC-VDL-ES methods respectively.

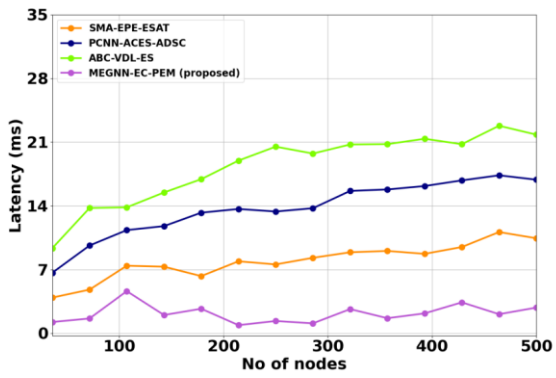

Figure4: Performance analysis of Latency

Figure 4 depicts Latency analysis. The MEGNN-EC-PEM attains 22.41%, 23.08% and 24.57% lower latency at number of node taken for 100; 25.12%, 26.41% and 27.90% lower latency at number of node taken for 300; 24.33%, 26.36% and 27.95% lower latency at number of node taken for 500; which is analyzed with SMA-EPE-ESAT, PCNN-ACES-ADSC and ABC-VDL-ES methods respectively.

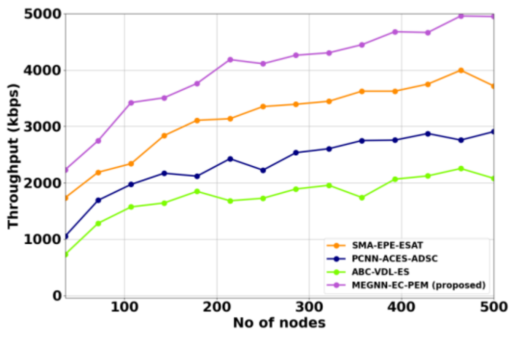

Figure5: Performance analysis of Throughput

Figure 5 depicts Throughput analysis. The MEGNN-EC-PEM attains 23.01%, 23.08% and 24.07% higher throughput at number of node taken for 100; 26.52%, 27.41% and 27.30% higher throughput at number of node taken for 300; 25.33%, 26.36% and 27.95% higher throughput at number of node taken for 500; which is analyzed with SMA-EPE-ESAT, PCNN-ACES-ADSC and ABC-VDL-ES methods respectively.

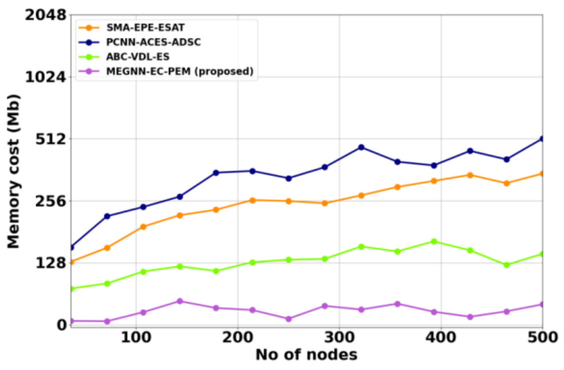

Figure6: Performance analysis of Memory cost

Figure 6 depicts Memory cost analysis. The MEGNN-EC-PEM attains 24.11%, 25.58% and 26.17% lower memory cost at number of node taken for 100; 26.82%, 27.71% and 28.30% lower memory cost at number of node taken for 300; 26.33%, 28.36% and 29.95% lower memory cost at number of node taken for 500; which is analyzed with SMA-EPE-ESAT, PCNN-ACES-ADSC and ABC-VDL-ES methods respectively.

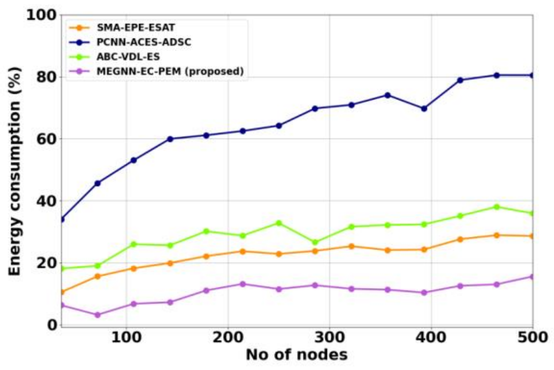

Figure7: Energy consumption analysis

Fig 7 depicts Energy consumption analysis. The MEGNN-EC-PEM attains 23.11%, 24.58% and 25.10% lower energy consumption at number of node taken for 100; 26.72%, 27.41% and 28.35% lower energy consumption at number of node taken for 300; 27.33%, 28.36% and 29.55% lower energy consumption at number of node taken for 500; which is analyzed with SMA-EPE-ESAT, PCNN-ACES-ADSC and ABC-VDL-ES methods respectively.

C. Discussion

An efficient MEGNN-EC-PEM is proposed for embedded computer performance evaluation model. Using this method, attributes pertinent to the resources utilized are incorporated into the description of activities in an embedded system described as an activity diagram; the regression functions the pre- processing using OMANF. The next embedded computer performance evaluation model using MEGNN and Optimization of MEGNN using HPOA; As a result, embedded computer model is eliminated. The throughput values of MEGNN-EC-PEM are 26.52%, 27.41% and 27.30% higher than existing methods such as SMA-EPE-ESAT, PCNN-ACES-ADSC and ABC-VDL-ES methods. Similar to this, the embedded computer model and Evaluation System is proposed 97.92% analyzed with Evaluation System is 80.42%. The proposed method MEGNN-EC-PEM has high modelling accuracy and throughput evaluation metrics than existing methods. Therefore, the comparative methods are expensive than the proposed technique. As a result, the proposed approach embedded computer performance evaluation model is more effectively and efficiently.

V. CONCLUSION

In this section, MEGNN-EC-PEM is successfully implemented. The proposed MEGNN-EC-PEM method is applied in an exact modelling framework depend on the System C language. Then the data are utilized to simulate expected processing and memory resources in accordance with different embedded system. According to the experimental results, MEGNN-EC-PEM performed better when used with the Co-training technique than when used separately regards latency, memory cost and Energy Consumption. The performance of MEGNN-EC-PEM approach attains 26.52%, 27.41% and 27.30% higher throughput, 24.11%, 25.58% and 26.17% lower memory cost and 23.11%, 24.58% and 25.10% lower energy consumption when analyzed with existing SMA-EPE-ESAT, PCNN-ACES-ADSC and ABC-VDL-ES methods respectively.

Acknowledgements

Sichuan Provincial Department of Science and Technology 2022 Soft Science Research Project (2022JDR0039) - Modernization of Big Data-Driven Vocational Education Governance System; Sichuan Provincial Department of Education 2022-2024 Vocational Education Talent Cultivation and Educational Reform Project (GZJG2022-644) - Practice and Research on the Construction of Internal Quality Assurance System in Vocational Colleges Based on School-Based Big Data. "

REFERENCE

[1] Lv, X., Su, M. & Wang, Z. (2021). Application of face recognition method under deep learning algorithm in embedded systems. *Microprocessors and Microsystems*, 104034.

[2] Kang, M., Lee, Y. & Park, M. (2020). Energy efficiency of machine learning in embedded systems using neuromorphic hardware. *Electronics*, *9*(7), 1069.

[3] Jadon, A., Varshney, A. & Ansari, M.S. (2020). Low-complexity high-performance deep learning model for real-time low-cost embedded fire detection systems. *Procedia Computer Science*, *171*, 418-426.

[4] Udendhran, R., Balamurugan, M., Suresh, A. & Varatharajan, R. (2020). Enhancing image processing architecture using deep learning for embedded vision systems. *Microprocessors and Microsystems*, *76*, 103094.

[5] Cantero, D., Esnaola-Gonzalez, I., Miguel-Alonso, J. & Jauregi, E. (2022). Benchmarking Object Detection Deep Learning Models in Embedded Devices. *Sensors*, *22*(11), 4205.

[6] Pham, M.T., Kim, J.M., & Kim, C.H. (2020). Deep learning-based bearing fault diagnosis method for embedded systems. *Sensors*, *20*(23), p.6886.

[7] Moon, H.C., Lee, H.Y., & Kim, J.G. (2020). Compression and Performance Evaluation of CNN Models on Embedded Board. *Journal of Broadcast Engineering*, *25*(2), 200-207.

[8] Marco, V.S., Taylor, B., Wang, Z., & Elkhatib, Y. (2020). Optimizing deep learning inference on embedded systems through adaptive model selection. *ACM Transactions on Embedded Computing Systems (TECS)*, *19*(1), 1-28.

[9] Triwiyanto, T., Caesarendra, W., Purnomo, M.H., Sułowicz, M., Wisana, I.D.G.H., Titisari, D., Lamidi, L. & Rismayani, R. (2022). Embedded machine learning using a multi-thread algorithm on a Raspberry Pi platform to improve prosthetic hand performance. *Micromachines*, *13*(2), 191.

[10] Knaak, C., von Eßen, J., Kröger, M., Schulze, F., Abels, P., & Gillner, A. (2021). A spatio-temporal ensemble deep learning architecture for real-time defect detection during laser welding on low power embedded computing boards. *Sensors*, *21*(12), 4205.

[11] Leone, A., Rescio, G., Caroppo, A., Siciliano, P., & Manni, A. (2023). Human Postures Recognition by Accelerometer Sensor and ML Architecture Integrated in Embedded Platforms: Benchmarking and Performance Evaluation. *Sensors*, *23*(2), 1039.

[12] Zhou, J. (2020). Real-time task scheduling and network device security for complex embedded systems based on deep learning networks. *Microprocessors and Microsystems*, *79*, 103282.

[13] Ajani, T.S., Imoize, A.L., & Atayero, A.A. (2021). An overview of machine learning within embedded and mobile devices–optimizations and applications. *Sensors*, *21*(13), 4412.

[14] Hong, S., Cho, H., & Kim, J.S. (2021). CitiusSynapse: A Deep Learning Framework for Embedded Systems. *Applied Sciences*, *11*(23), 11570.

[15] Chen, C., Liu, Y., Kumar, M., Qin, J., & Ren, Y. (2019). Energy consumption modelling using deep learning embedded semi-supervised learning. *Computers & Industrial Engineering*, *135*, 757-765.

[16] Yu, Y.S., Kim, C.G. & Hong, C.P. (2020). An Implementation of Embedded Linux system for embossed digit recognition using CNN based deep learning. *Journal of the Semiconductor & Display Technology*, *19*(2), 100-104.

[17] Saurav, S., Saini, A.K., Saini, R., & Singh, S. (2022). Deep learning inspired intelligent embedded system for haptic rendering of facial emotions to the blind. *Neural Computing and Applications*, 1-29.

[18] Mwitta, C., Rains, G.C., & Prostko, E. (2024). Evaluation of Inference Performance of Deep Learning Models for Real-Time Weed Detection in an Embedded Computer. *Sensors*, *24*(2), 514.

[19] da Silva, B., W. Happi, A., Braeken, A., & Touhafi, A. (2019). Evaluation of classical machine learning techniques towards urban sound recognition on embedded systems. *Applied Sciences*, *9*(18), 3885.

[20] Gorospe, J., Mulero, R., Arbelaitz, O., Muguerza, J., & Antón, M.Á. (2021). A generalization performance study using deep learning networks in embedded systems. *Sensors*, *21*(4), 1031.

[21] Lamrini, M., Chkouri, M.Y., & Touhafi, A. (2023).Evaluating the performance of pre-trained convolutional neural network for audio classification on embedded systems for anomaly detection in smart cities. *Sensors*, *23*(13), 6227.

[22] Arablouei, R., Wang, L., Currie, L., Yates, J., Alvarenga, F.A., & Bishop-Hurley, G.J. (2023). Animal behavior classification via deep learning on embedded systems. *Computers and Electronics in Agriculture*, *207*, 107707.

[23] Zu, Y. (2020). Deep learning parallel computing and evaluation for embedded system clustering architecture processor. *Design Automation for Embedded Systems*, *24*, 145-159.

[24] Gagliardi, A., Staderini, V., & Saponara, S. (2022). An Embedded System for Acoustic Data Processing and AI-Based Real-Time Classification for Road Surface Analysis. *IEEE Access*, *10*, 63073-63084.

[25] Cornetta, G., & Touhafi, A. (2021). Design and evaluation of a new machine learning framework for IoT and embedded devices. *Electronics*, *10*(5), 600.

[26] https://www.kaggle.com/discussions/questions-and-answers/397543

[27] Ge, Y., Yang, L., & Ma, X. (2021). A harmonic compensation method for SPMSM sensorless control based on the orthogonal master-slave adaptive notch filter. *IEEE Transactions on Power Electronics*, *36*(10), 11701-11711.

[28] Chang, Y., Chen, C., Hu, W., Zheng, Z., Zhou, X., & Chen, S. (2022). Megnn: Meta-path extracted graph neural network for heterogeneous graph representation learning. *Knowledge-Based Systems*, *235*, 107611.

[29] Naruei, I., Keynia, F., & SabbaghMolahosseini, A. (2022). Hunter–prey optimization: Algorithm and applications. *Soft Computing*, *26*(3), 1279-1314.