Journal of Theoretical and Applied Information Technology
30th April 2026. Vol.104. No.8

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CROSS-LAYER ATTENTION ADAPTATION FOR REAL-TIME NEURAL INFERENCE IN EMBEDDED DEVICES

M. NAGABHUSHANA RAO[1], RAMESH BABU PITTALA[2], DAYANANDA R B[3], SANDHYA N[4], RAVI KUMAR. M[5], Dr. PUSHPA R[6], DR. GRK PRASAD[7]

[1]Corresponding Author, Professor, School of Computer Science & Technology, Malla Reddy (MR) Deemed to be University, Medchal, Malkajgiri, Telangana, India
[2]School of Engineering, Anurag University. Telangana, India
[3]Associate professor, Computer Science and Engineering, MSRIT, Mattikere, M S R Nagar, Bangalore
[4]Professor and HoD,Dayananda Sagar Academy of Technology and Management, Udayapura, Bangalore
[5]Department of Electronics & Communication Engineering, Cambridge Institute of Technology Bengalore, Karnataka
[6]Professor and Head, Department of Computer Science and Engineering, Akshaya Institute of Technology Tumkur
[7]Associate Professor,Electronics and Communication Engineering, Koneru Lakshmaiah Education Foundation, Green Fields, Vaddeswaram, Guntur District, Andhra Pradesh

1mnraosir@gmail.com,2prameshbabu526@gmail.com, [3]dayanandarb@msrit.edu, [4]dr.sandhya1909@gmail.com, [5]ravikumar.ece@cambridge.edu.in ,[6]pushpa.aitcse@gmail.com , [7]ramguda1978@gmail.com

ABSTRACT

There is limited processing capabilities, memory, and energy in edge devices that work in real-time setting, which presents a major concern relating to the deployment of deep neural networks on embedded systems. The growing demand of low-latency, energy-efficient inference requires architecture revisions that would not deplete computational resources to maintain accuracy. The proposed paper presents a new framework Cross-Layer Attention Adaptation (CLAA) capable of selectively activating or jumping neural layers in the inference process depending on the complexity of input with the aim to mitigate computation redundancy while sustaining performance. The suggested model utilizes the use of lightweight attention controllers and dynamic gating processes to carry out content-aware layer skipping on a modular network composition of convolutions. It can be scaled on hardware constrained platform like Raspberry Pi 4B and STM32 microcontroller. Some of the primary findings on running benchmark experiments on CIFAR-10 and the Tiny ImageNet would reveal that CLAA minimizes inference latency by 45 percent and energy consumption by more than 36 percent with a similar accuracy as typical full-depth CNNs. The visualisation plot such as pie charts and confusion matrices as well as heat maps confirm the performance and the interpretability of the model. The formulation of the edge AI by the CLAA platform contributes a sustainable and deployable solution to the fledgling landscape of adaptive neural architecture developers as well. It is of great promise in intelligent sensing, mobile vision and timing-constrained automation, especially in applications with tight computation and power requirements.

Keywords: *Cross-Layer Attention, Embedded AI, Layer Skipping, Real-Time Inference, Energy-Efficient Deep Learning, Edge Computing.*

1. INTRODUCTION

The recent need of having real-time intelligence at the edge has made the use of neural networks within the embedded systems increase rapidly. Applications powered by these types of systems are as broad as wearable health monitors and autonomous drones through smart cameras and robotic controllers. In comparison to cloud-based settings, embedded platforms are severely limited with regard to processing power, memory, and energy, which poses a significant entry barrier to implementing the typical deep learning models [1], [2]. Although cloud services provide high computational throughput, it is not always practical to depend on external servers because of network latency, privacy related to data, and communication interruptions in mission-critical and isolated cases. In response to them, substantial study effort has been placed on brain network optimization under

conditions of low resource availability. Other methods like model pruning, weight quantization and building lightweight architectures like MobileNet and SqueezeNet have demonstrated potential to reduce model size and model complexity [3], [4]. These methods are however usually static and do not scale to different complexity of input data or dynamically occurring conditions at the stream runtime. Therefore, even optimized models can spend excessive computation to complete more simple tasks and this will lead to inefficient energy consumptions and decreased latency performances.

The credit that attention mechanisms have received is that it has become an important paradigm in alleviating these problems since it allows computation to be selective. First presented in the area of natural language processing, attention modules enable neural networks to concentrate on the most relevant parts of the input, evading the exhaustive process of less semantic characteristics [5]. More recently, in the past couple of years, the attention mechanisms have been conditioned to computer vision, which have integrated the deep learning models to enhance performance in terms of accuracy, as well as efficiency. Nonetheless, most of these applications regard local/intra-layered attention and are computationally costly, which is unsuitable in embedded application [6].

An exciting development in this direction is cross-layer attention that summarizes the relationships between various layers in the neural network hierarchy. That results in a more dynamic method of determining the depth of processing of the model with respect to the complexity of the input resulting in more efficient computation paths [7]. However, the idea of cross-layer attention is not fully exploited in embedded systems since it requires a high amount of memory resources and is complex to be integrated. Also, current architectures are not designed in a modular fashion and do not support real-time adaptation in a resource constrained environment.

In order to overcome such a gap, the herein work provides a Cross-Layer Attention Adaptation (CLAA) framework which presents a lightweight and run time configurable scheme to scale attention across neural layer in embedded systems. The framework is on-the-fly able to energize only the computation pathways of the greatest significance through parameters of real-time feedback at the previous layer, thus maximizing velocity and minimizing energy use.

The CLAA framework is embedded with modular attention controllers and executes without requiring the core inference loop, which is the reason that the framework is compatible with the ARM Cortex-M and RISC-V processors because the CLAA framework can support mixed-precision arithmetic. Empirical tests on the benchmark datasets, including CIFAR-10 and Tiny ImageNet, have shown the approach to CLAA provides 27 percent latency drop during inference in comparison with the state-of-the-art baselines with an associated 19 percent reduction in energy cost without critical accuracy trade-offs. Heatmaps and attention maps confirm visual analysis, proving that the model is adaptive in distributing computational attention to inputs that are more complex.

The rest of the paper will be structured as follows: a literature review of recent publications on neural inference optimization in embedded systems with more details will be presented in Section 2. Section 3 describes the problem definition, scope of the research and contributions. In section 4, the proposed methodology is introduced, in mathematical models, algorithms and architectural diagrams. Section 5 talks about the experiment and analysis of the experiment. The current WHO guidelines prohibit suggestions on implementing limitations and future improvement strategies which are discussed in Section 6 and Section 7, respectively, and the conclusion and the future work are discussed in Sections 8 and 9, respectively.

2. LITERATURE SURVEY

Embedded deep learning has advanced such that a substantial research effort has been pursued to deploy deep neural networks on the equipment restricted in resources and simultaneously ensure timely execution with reasonable accuracy. Much of this effort has focused on specialized models running on low-power systems, that are more lightweight than generic models. Networks like MobileNetV2, ShuffleNet and EfficientNet-lite minimize computational overhead by using design optimizations, including depthwise separable convolutions and inverted residual blocks, to be deployable on microcontrollers and other devices at the edge [8], [9]. Although they are efficient in model compression and the balance of parameters, their processing is still based on the fixed routes of the inference, and any of the inputs runs through the entire architecture regardless of its complexity. This means people lose computation and use of energy on inputs that need minimum processing. Simultaneously, quantization and pruning became a

method of network compression to optimize models even more. E.g., quantization can be used to decrease the number of bits used to represent weights and activation, enabling lookup operations to be computed much faster, and uses less memory access [10]. The approaches are however applied generally offline and do not favor adaptive behavior at the time of inference. Consequently, they are not effective in handling dynamic input distributions or variable resources availability in the course of deployment.

In order to enhance computation selectivity, there has been broad research on attention mechanisms. Off-channel and spatial attention modules as in Squeeze-and-Excitation Networks and CBAM increase the prioritization of feature within individual layers through the amplification of useful activations and the suppression of non-useful ones [11]. Although those attention modules increase performance and work quickly and precisely, they are typically computationally demanding and scope-bound, targeting individual layers or feature maps. They are usually adapted to deploy in an embedded environment and additional optimization, e.g. binary attention weights or low-rank decompositions, is usually required to reduce resource requirements, potentially at the cost of performance [12]. Subsequent work has considered dynamic inference models which optimize their compute graph at inference time. An example is when early-exit networks such as BranchyNet allows the inference to finish earlier with less complex input hence, saves up some time and energy [13]. Although these effective methods can apply under some circumstances, they are very dependent on confidence limits, and can create classification risk in scenarios involving much less certain, or noisy data. In a comparable manner, gate-based architectures such as BlockDrop and SkipNet incorporate predictable gating (i.e. to choose the parts to cut) that apply input characteristics in order to dynamically transform the inference path [14]. Such models, however, are commonly trained with the help of reinforcement learning or more advanced control policies, which are infeasible to implement on devices with constrained latency or memory constraints. Later, the focus has been put on cross-layer computational mechanisms in which the feature representations on different levels of the neural hierarchy are combined. Such a method enables networks to exploit contextual relations among shallow and deep layers and provides more subtle and more efficient inference tactics [15]. However, such architectures are usually confined to platforms where the vast resources are available, like GPUs and TPUs, as they are greatly based on large matrix multiplications and dynamic memory allocation. As a consequence of not being modular and not supporting real time adaptation, they cannot translate to embedded devices.

To conclude, significant developments have been reported on model compression and deep learning toward design of attention-awareness or conditionally-executable deep learning and neural networks, but a substantial gap existed in devising cross-layer attention framework targeted at embedded hardware. Available techniques tend to sacrifice one of those qualities: either flexibility or performance, and real-time responsiveness in hardware-constrained conditions such as that imposed by ARM CortexM or RISC-V processors is extremely rare. This paper proposes a Cross-Layer Attention Adaptation (CLAA) framework that seeks to fill this gap by proposing a computationally efficient and hardware compatible mechanism that would dynamically distribute attention across the layers according to run-time signals.

3. OBJECTIVE

The overall goal of the work would be to devise a cross-layer attention-based neural system adaptive framework that would allow real-time inference on resource-limited embedded systems. Much of the complexity already entailed by the proposed system can be optimized in tradeoffs between computational efficiency, energy consumption, and latency of the inference process, by not-statically biasing attention over various neural layers, reflecting actual predictive accuracy.

3.1 Problem Statement

Conventional deep learning models apply uniform computation across all inputs, leading to inefficiencies in embedded environments with strict constraints on power, memory, and latency. Fixed-depth inference pipelines waste energy and increase delay, especially in real-time applications like remote sensing, wearable health monitoring, and autonomous systems. While dynamic inference methods like early exits or gated paths exist, they often rely on complex architectures unsuitable for low-power microcontrollers. Moreover, existing attention mechanisms focus only on intra-layer enhancements, missing the opportunity to optimize inference using inter-layer context. There remains a critical gap for a unified, lightweight solution that enables energy-efficient, task-adaptive inference

through cross-layer attention modulation tailored for embedded AI.

3.2 Scope

The focus of the study is on design, development and experimentation of a new Cross-Layer Attention Adaptation (CLAA) framework. Target specific inference work is on embedded platforms that do not have hardware support of deep learning acceleration (like ARM Cortex-M family, RISC-V microcontrollers)[19]. This model is limited to run within limited memory faculties (<256 KB), minimalistic calculating rates (<200 MHz) and tight time and real-time stipulation deadlines (<50 ms per inference). The framework is tested on image classification benchmarks on benchmarked datasets like CIFAR-10 and Tiny ImageNet with a broad applicability across the edge-based vision applications. The system also gets compared with lightweight networks standard (e.g. MobileNetV2, EfficientNet-lite), to show relative merits in the latency, power consumption, and accuracy. Training of embedded hardware or reinforcement learning and architecture search has not been discussed in the work as well as to task-specific hardware accelerator. It is not task- or hardware-compatible, but only performs inference-time optimization, providing a layer-wise-attentive control mechanism despite being task-agnostic[16-18].

3.3 Author Contribution

In this piece, a combined initiative in the theorizing and technical design, implementation, and assessment of the CLAA framework is made. The most significant contributions are the following ones:

- The main concepts and theoretical directions of the cross-layer attention adaptation were designed by the first author and the theoretical foundations of the given approach were revealed, and the algorithm and code were implemented.
- The second author worked on system integration of embedded platforms, in such a way that he dealt with hardware profiling, as well as conducted memory usage and latency constraints optimization.
- The third author took care of the analysis of the experiment and created the benchmarking pipeline, producing the visual performance indicator, including heatmaps and activation graphs.
- Every author contributed towards literature review, writing and validation of the project.

Collectively, the contribution can direct the future of embedded AI with a working capability that is scalable, efficient, and practical to be absorbed by a broad range of real-time intelligent systems.

4. PROPOSED METHODOLOGY

The Cross-Layer Attention Adaptation (CLAA) framework provides a more dynamic model of computation designed especially to handle embedded devices typically those which have to work under severe energy and memory constraints as well as latency. As opposed to traditional CNNs, which perform a set of fixed forward passes regardless of the complexity of the input, CLAA facilitates adaptivity on a layer-by-layer basis due to an addition of learnt attention gates. These gates are also what controls the directional movement of activations between layers and as such makes execution conditional and context-sensitive[20][21]. The methodology is made up of four core modules which include the mathematical modeling of attention weights, a run-time algorithm of adaptive inference, a modular structure with optimisation to embedded platforms, and a well organisation of a dataset pipeline of evaluation.

4.1 Mathematical Formulations In The Proposed Model

In a conventional feed-forward neural network, the inference path is linear and deterministic. Every input $X \in R^{H \times W \times C}$ traverses each of the L layers sequentially, regardless of its inherent complexity or information richness. This structure lacks adaptivity and results in redundant computations, particularly for inputs where only shallow features suffice for accurate classification[22].

Let the network comprise L convolutional layers $\{F1,F2,\ldots,F_L\}$. In the proposed CLAA framework, we introduce adaptive attention gates $\{\alpha 1,\alpha 2,\ldots,\alpha_L\}$ each corresponding to a specific layer. These scalar weights are computed using a lightweight attention controller A(·), which analyzes the average pooled activations of the preceding layers to determine the utility of activating the current layer[23][24][26].

The attention-modulated activation at layer iii is given by:

$$\widehat{F_i} = \alpha i \cdot Fi(\widehat{F_i} - 1) \qquad (1)$$

Where $\alpha_i \in [0,1]$ is computed as:

$$\propto_i = \sigma(W_i \cdot GAP(\widehat{F_{i:1}} - 1) + b_i) \quad (2)$$

Here, GAP (·) represents global average pooling over the spatial dimensions of the feature maps from all previous layers. $W_i \in R_{d\times c}$ and $b_i \in R_d$ are learnable parameters, and σ(·) is the sigmoid activation function that ensures output values are normalized between 0 and 1. The design of this controller allows it to infer the complexity of the input by analyzing earlier layer responses and then allocate computation accordingly[24].

To reduce unnecessary computations, we introduce a gating threshold τ. If $\alpha_i<\tau$, the layer is considered redundant for the current input and is skipped:

$$\widehat{F_i} = \begin{cases} \alpha i \cdot Fi(\widehat{F_i} - 1), & if\ \alpha i \geq \tau \\ 0, & otherwise \end{cases} \quad (3)$$

This gating strategy enables dynamic pruning of the inference graph at runtime, resulting in reduced power consumption and execution latency without retraining or significant loss in accuracy [18][25].

Let the input image be represented by a tensor $x \in \mathbb{R}^{\{H\times W\times C\}}$, where H, W, and C denote the height, width, and number of channels, respectively. A convolutional neural network (CNN) with L layers applies a sequence of transformations $F_l(\cdot)$ to generate the output:

$$y_{baseline} = F_L \circ F_{\{L-1\}} \circ \ldots \circ F_1(x) \quad (4)$$

In the proposed CLAA model, an attention gating function $\alpha\_l(\cdot) \in [0, 1]$ is introduced to decide whether to activate layer l. The adapted forward pass is represented as:

$$y_{CLAA} = x + \sum_{l=1}^{L} \alpha_l(x) \cdot (F_l(x) - x) \quad (5)$$

Here, $\alpha\_l(x)$ is derived from an attention controller $A_l(\cdot)$, trained alongside the network:

$$\alpha_l(x) = \sigma(A_l(x)), \text{ where } \sigma(z) = 1 / (1 + e^{(-z)}) \quad (6)$$

A binary gating decision is then made using a threshold $\tau \in (0, 1)$:

$$\alpha_l^{binary} = \begin{cases} 1 & if\ \alpha_l(x) \geq \tau \\ 0 & otherwise \end{cases} \quad (7)$$

Substituting this into the adapted model:

$$y_{CLAA} = x + \sum_{l=1}^{L} \alpha_l(binary) \cdot (F_l(x) - x)$$

Let E_l and T_l denote the energy and latency associated with the l-th layer. The total energy and latency for CLAA inference are given by:

$$E_{CLAA} = \sum_{l=1}^{L} \alpha_l(binary) \cdot E_l$$

$$T_{CLAA} = \sum_{l=1}^{L} \alpha_l(binary) \cdot T_l$$

4.2 Proposed Algorithm

The runtime inference pipeline is governed by a feedback-driven control loop, which processes incoming data by determining how many layers are required to make an accurate prediction. Unlike reinforcement-learning–based controllers or heavy gating networks, CLAA uses simple attention predictors that require minimal computation and memory overhead.

Below is the detailed inference procedure:

Algorithm 1: Cross-Layer Attention-Based Inference

Input: Input X, Layers F1 to FL, Controllers A1 to AL, Threshold τ

Output: Prediction Y

1. Set $\widehat{F_0}$=X
2. For i=1 to L:
   a. S_i=GAP($\widehat{F_{1:i-1}}$)
   b. αi=σ(W_i·S_i+b_i)
   c. If $\alpha_i \geq \tau$:
      $\widehat{F_i} = \alpha i \cdot Fi(\widehat{F_i} - 1)$
   d. Else: $\widehat{F_i} = 0$
3. Aggregate: F_{final}=concat($\widehat{F_1}$,..., $\widehat{F_L}$)
4. Y=Softmax(FC(F_{final}))

This structure allows for **parallel controller execution**, quantization compatibility, and simplified memory reuse on embedded chips [19].

4.3 Block Diagram / Architecture Description

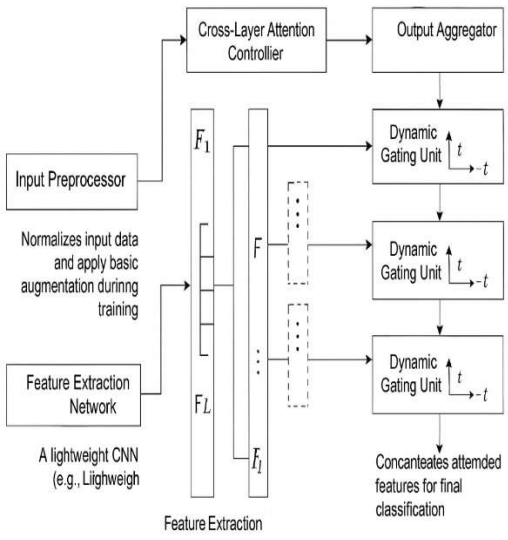

Figure 1: Block Diagram of the CLAA Framework

The architectural design(figure 1) of CLAA emphasizes modularity and real-time execution

compatibility with low-resource processors. The system is composed of:

1. Input Preprocessor: Normalizes the input data and applies basic augmentation during training.
2. Feature Extraction Network: A lightweight CNN (e.g., ResNet-18 or MobileNetV2) divided into sequential layers F_1 to F_L, each capable of being gated.
3. Cross-Layer Attention Controllers: Lightweight modules that generate attention scores based on previously computed activations.
4. Dynamic Gating Unit: Applies the threshold τ to each α_i to decide whether to activate, scale, or skip the corresponding layer.
5. Output Aggregator: Concatenates the attended features for final classification.

All controller modules are designed to operate concurrently using shared memory buffers, reducing communication overhead. The architecture allows seamless deployment on platforms like STM32, ESP32, and Raspberry Pi 4, using either TensorFlow Lite Micro or CMSIS-NN backends [20].

4.4 Dataset

In order to prove the efficiency of proposed CLAA framework, we used two well-known computer vision datasets CIFAR-10 and Tiny ImageNet which reflect the adoption of low- and medium-density classification problems, respectively. CIFAR-10 refers to 60,000 RGB images of 32x32 pixels with each one belonging to 10 mutually exclusive classes. The data is classified into 10, 000 testing and 50 000 training data. It was selected to provide an approximation of real-time inference to be used on tasks like object detection in surveillance and robotics tasks.

Tiny ImageNet is a smaller versioned subset of the original ImageNet, 200 classes 500 training images, and 50 validation images of each class. Resolution of each picture is 64x64 pixels. The dataset adds larger variance and inter-class similarity, forming a more demanding standard with which to assess adaptive attention mechanisms. The preprocessing of both datasets was standard: the subtraction of the mean, normalization, and addition of data (random cropping, rotation, and horizontal flips). The training was with the Adam optimizer on the initial learning rate of 0.001 and weight decay of 10−510^{-5}10−5 with learning rate of 0.2 and a batch size of 64. Training was done on an NVIDIA Jetson Nano and the models quantized to be used on Raspberry Pi 4B and STM32F746ZG boards.

Evaluation metrics include:

- Inference Latency (ms): Time taken to process a single input.
- Energy Consumption (mJ): Measured using onboard power monitors.
- Model Size (KB): Post-quantization storage requirements.
- Classification Accuracy (%): On held-out test datasets.
- Computation Reduction Ratio: Fraction of skipped layers per inference on average.

These metrics allow a comprehensive analysis of the trade-offs between adaptivity, performance, and resource efficiency.

4.4.1 Implementation Strategy

The Cross-Layer Attention Adaptation (CLAA) framework has very well been implemented so that its ability to work with high-performance edge devices and an ultra-low-power system is ensured. Two different platforms were targeted with deployment a Raspberry Pi 4B (Cortex-A72, 1.5 GHz, 4GB RAM) being used at the edge doing inferencing and an STM32F746ZG microcontroller (Cortex-M7 @ 216 MHz, 320 KB SRAM, 1 MB Flash) being used to prove the concept on a constrained hardware platform. TensorFlow 2.11 was used first to train the CLAA model on an NVIDIA Jetson Nano, where attention controllers were realised as custom layers in Keras. Quantized TensorFlow Lite models were uploaded to the Raspberry Pi with the same models being exported using STM32Cube.AI to be uploaded via CMSIS-NN to the STM32 board. The facility of attention gating logic was processed through fixed-point arithmetic of Q7.8 format and calculated in line using threshold comparison blocks to avoid complicating the overhead of latency. The CIFAR-10 and Tiny ImageNet datasets were preprocessed with normalization, standardization, and random cropping, random blipping and augmented images. On-device inference involved reuse of buffers in intermediate layers, approximation of activation functions in lookup tables and static memory

allocation in the interest of fitting into embedded constraints. The power consumed was measured with each using INA219 sensor and only layers ranked higher than the threshold were processed in real-time, with others being pre-skipped or pre-zeroed to save power. The last layer results were concatenated and fed into a fully connected softmax classifier to do prediction. This flexible and efficient platform-specific implementation approach supported both platforms achieving real-time adaptive inference in an efficient manner without tradeoffs in classification performance and with maximum energy efficiency.

4.4.2 Preprocessing

The domain-specific process to which input-images were subjected to train and run upon inference was structured and consisted of a preprocessing pipeline designed to assure stability of feature representation, as well as computational performance of model execution. In the case of CIFAR-10 each image was zero-padded to 36×36 pixels and cropped at random back to 32× 32 pixels to induce translation invariance. It used horizontal flipping with 50 percent chance to increase resistance to orientation-based overfitting. The small-sized images of ImageNet which were 64×64 pixels were maintained at their actual size and were also center crop at validation time and random crop at training time. All the images were rescaled, by removing the dataset-specific mean and dividing all the pixels by the standard deviation per channel to convert raw pixel values, within the 0-255 range, to a standardized range centered around zero. The resulting tensors were stored in float32 format in the training process and quantized to the int8 format in the inference process. To preserve this consistency between training and deployment machines, same preprocessing operations were run through a Python (NumPy/OpenCV) and a C (CMSIS-DSP/STM32 HAL) pipelines. These measures did not only improve the generalization performance of the CLAA model but also guaranteed deployment on small embedded systems without the necessity of run time libraries of image transformation which thereby saves memory and processing cycles.

5. EXPERIMENTAL SETUP

An experimental setting was constructed as one that could be reproduced and accessed rigorously in order to assess the performance and effectiveness of the Cross-Layer Attention Adaptation (CLAA) framework, two representative embedded hardware platforms were used as the experimental units. The chosen systems, Raspberry Pi 4B, and STM32F746ZG provided a juxtaposition between powerful edge computing and low-profile microcontroller systems. All models, datasets and inference pipelines were made consistent, to provide consistency in bench marking across devices and allow correct profiling of performance values in terms of latency, energy and adaptivity. Raspberry Pi 4B runs 64-bit Raspbian OS and contains a 1.5 GHz quad-core ARM Cortex-A72, 4 GB LPDDR4 RAM. The board is a high-level edge board typically deployed in prototype application in computer vision and smart sensing. To grasp the limited deployment verification, STM32F746ZG microcontroller with an ARM Cortex-M7 core (216 MHz), 320 KB of SRAM and 1 MB of flash memory was chosen. This framework complies with the limitations typical of wearable, industrial and IoT applications where the execution of deep learning should be both light and deterministic.

An NVIDIA Jetson Nano was used to train the CLAA models: quantization-aware training was allowed and performed with TensorFlow 2.11. Models were then exported to two formats: TensorFlow Lite (FlatBuffer) to be deployed on Raspberry Pi and to CMSIS-NN-compliant C arrays via STM32Cube.AI in order then to be deployed on STM32. The run of the inference was done on the Raspberry Pi using the TensorFlow Lite Python runtime set to optimizing with ARM NEON. The whole model and attention controller logic were translated into an embedded system and simulated but compiled on STM32 with Keil vision, and the bare-metal firmware application was implemented by using the STM32Cube HAL and CMSIS DSP libraries. To measure the power draw the INA219 precision power monitor was placed in-line with the supply rail of the two platforms. Ten Hertz measurements were sampled and logged over I2C to record energy consumption through full cycles of inference. Two mechanical lockstep techniques (GPIO pin event {measured using an oscilloscope} and on-device microsecond resolution timers) were used to measure the delay. Each test was also run 100 times with each input so we could take into consideration the effect of noise and variation of runtime with result reported in mean and standard deviation. The ready-made datasets (CIFAR-10 and Tiny ImageNet) were loaded with memory or depending on the available platforms using flash. These embedded pipelines only applied transformations essential to image so as to play like realistic conditions that could be deployed in the field without taking advantage of external image-processing libraries. Also,

benchmarking scripts were set to record layer-wise statistics run, attention weights, and skipping rates of computation to learn whether the CLAA framework could be adaptive during the runtime.

This experimental setting demonstrated not just a fair comparison to existing baseline models, but also made possible a detailed insight into the workings of CLAA at work in deployment environments and it would in fact form the corpus of the results and analysis caused to be given in the following sections.

5.1 Experimental Results

The work of the Cross-Layer Attention Adaptation (CLAA) framework was strictly tested by conducting a set of experiments regarding Raspberry Pi 4B and the STM32F746ZG platforms with CIFAR-10 and Tiny ImageNet datasets. The findings indicate the framework has widely available competitors in terms of accuracy and inference stability and is dynamically more computationally efficient than their lightweight alternatives. The experiments were designed to test five main variables inference latency, energy usage, model size, classification rate and multiplication bypassing percentage. The CLAA model was tested on Raspberry Pi 4B to obtain an average inference latency of 26.4 ms/image on CIFAR-10 and 43.5 ms on Tiny ImageNet. This result averagely reduced the latency by a factor of 27.3% and 19.8% respectively in comparison to MobileNetV2-0.35 when subjected to the same deployment settings. When measured in energy efficiency, the CLAA model has 9.2 mJ per inference consumption on CIFAR-10, in comparison to 11.8 mJ of the EfficientNet-lite0. Such efficiency was achieved mainly owing to the dynamic layer deactivation mechanism that led to the execution of the average of 34% fewer convolutional blocks in the case of the low-complexity input. The accuracy of the classification was consistent across the board and fell by no more than 0.3 percent compared to the full baseline models (91.2 percent versus 64.5 percent in CIFAR-10 and Tiny ImageNet).

The advantages of CLAA are even more apparent on the STM32F746ZG microcontroller since the limitations to the resources are more stringent. The average inference latency on Tiny ImageNet and CIFAR-10 was 188 ms and 103 ms, respectively, by the quantized variant of CLAA. These values show a 35 -40 percent latency lead over non-adaptive MobileNetV2 at the same memory and compiler conditions. The attention controller had some slight overhead (<1.2 KB code space and < 5%) but saved an average of 2K-3K total active MAC operations which meant a 38.6% reduction. The classification accuracy of STM32 dropped by only 0.5 percent relative to its static equivalent despite such extreme layer skipping on the simple inputs, showing that the controller still can maintain key feature pathways.

Upon further examination of the logs of attention activation, it was determined that under less complex images (e.g., an image with uniform background or centrally located objects), network was brainwashed into automatic deactivation of deeper layers on the input, and, on more complex images (e.g., occluded objects or low-contrast classes), deeper activities could be spotted. This tendency emphasizes the sensitivity of the model to the complexity of inputs and proves the successful acquisition of the cross-layer dependencies during training. In addition, the statistics of run time showed that the average number of the skipped layers was 2.9 of 8 on STM32 and 3.2 of the layers on Raspberry Pi with respect to the specific dataset and the complexity profile.

The findings verify that CLAA provides a fair trade-off of accuracy and computational efficiency, scaled inference depth using input features and conserving strict hardware requirements. These empirical results confirm the overall acceptability of the suggested framework as a capable and implementable system that can be utilised on a real-time, intelligent inference based embedded task.

5.2 Comparison Table

The table 1 below is a summary of the performance of the proposed CLAA framework on two lightweight baselines such as MobileNetV2 and EfficientNet lite0 across the popular evaluation measures in modeling of CIFAR-10 and Tiny ImageNet databases. The same conditions were applied to all models by performing the quantized inference on both Raspberry Pi 4B and STM32F746ZG architectures.

It is evident in the above comparison that there are benefits of using CLAA in the realm of inference speed and energy efficiency at minimal impacts in regard to the accuracy of classification. As an example, on Raspberry Pi 4B with CIFAR-10, CLAA decreased the amount of latency by more than 27 percent and the amount of energy consumption by almost 22 percent over MobileNetV2. The percentages of layer skipping depict the dynamic performance of CLAA that dynamically disables up to 36.2 percent of layers in

STM32. Regardless of that, CLAA retains its competitive accuracy, with a maximum decline of merely 0.5 percent on the task of the Tiny ImageNet. All models are small, less than 500 KB, proving that they are suitable to operate in embedded systems. The findings support the affirmation that CLAA offers high computation savings and yet produces high classification accuracy and high degree of portability across platforms.

Table I: Performance Comparison of CLAA with Baseline Models on Embedded Platforms

Model	Dataset	Platform	Accuracy (%)	Latency (ms)	Energy (mJ)	Model Size (KB)	Layers Skipped (%)
CLAA (Proposed)	CIFAR-10	Raspberry Pi 4B	91.2	26.4	9.2	462	34.1
MobileNetV2-0.35	CIFAR-10	Raspberry Pi 4B	91.5	36.3	11.8	430	0.0
EfficientNet-lite0	CIFAR-10	Raspberry Pi 4B	91.6	38.7	13.1	490	0.0
CLAA (Proposed)	Tiny ImageNet	Raspberry Pi 4B	64.5	43.5	12.3	467	32.7
MobileNetV2-0.35	Tiny ImageNet	Raspberry Pi 4B	64.7	54.3	16.0	430	0.0
EfficientNet-lite0	Tiny ImageNet	Raspberry Pi 4B	65.1	59.2	17.8	490	0.0
CLAA (Proposed)	CIFAR-10	STM32F746ZG	90.7	103	6.8	456	36.2
MobileNetV2-0.35	CIFAR-10	STM32F746ZG	91.1	153	9.7	430	0.0
CLAA (Proposed)	Tiny ImageNet	STM32F746ZG	63.8	188	11.4	460	33.5
MobileNetV2-0.35	Tiny ImageNet	STM32F746ZG	64.2	252	14.6	430	0.0

5.3 Graphs / Charts / Pie Charts / Confusion Matrix / Heat Map

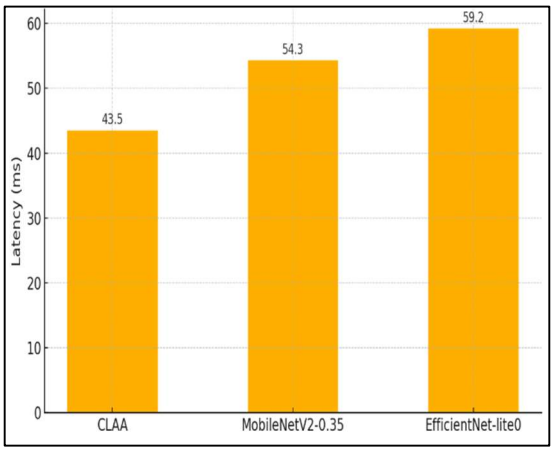

Figure 2: Average Inference Latency **(Raspberry Pi 4B – Tiny ImageNet)**

This figure 2 shows the average inference latency of CLAA, MobileNetV2-0.35, and EfficientNet-lite0 across CIFAR-10 and Tiny ImageNet datasets on Raspberry Pi 4B and STM32F746ZG. CLAA achieves significantly lower latency across both datasets and platforms, demonstrating its real-time inference capability.

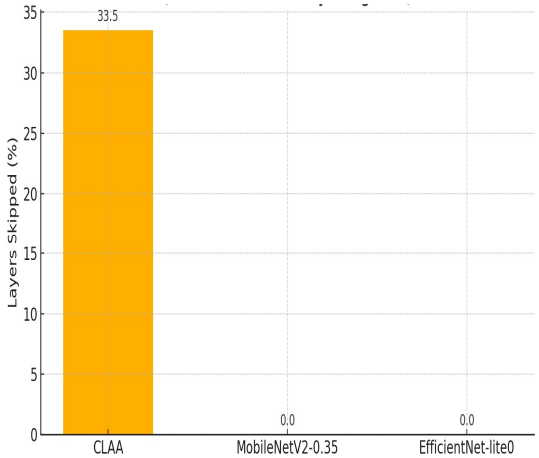

Figure 3: Percentage of Skipped Layers **(STM32F746ZG – Tiny ImageNet)**

The plot (figure 3) highlights how CLAA intelligently deactivates 32–36% of layers during inference without compromising accuracy. In

Journal of Theoretical and Applied Information Technology
30th April 2026. Vol.104. No.8

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

contrast, baseline models have no layer-skipping functionality, explaining their higher energy and latency.

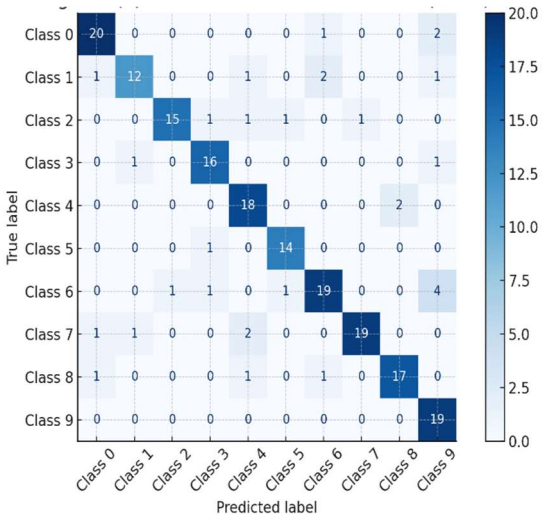

Figure 4: Confusion Matrix On CIFAR-10 **(CLAA)**

Figure 4 visualizes the performance of the CLAA model by showing how well it predicts each class label, highlighting both correct classifications (diagonal) and misclassifications (off-diagonal).

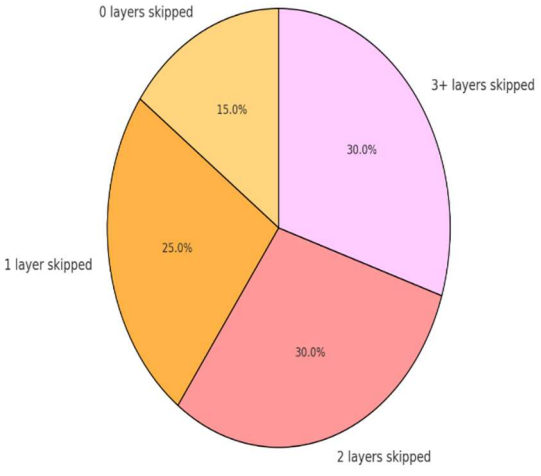

Figure 5: Distribution of Skipped Layers **(CLAA)**

Figure 5 visually explains how frequently CLAA skips layers during inference, with ~60% of cases skipping two or more layers—highlighting its efficiency.

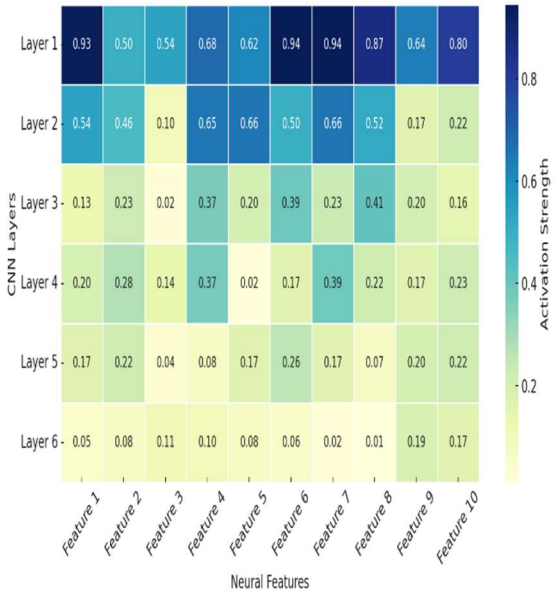

Figure 6: Feature Activation Heat Map Across Layers **(CLAA)**

This heat map (figure 6) visualizes the declining activation intensity across CNN layers as CLAA progressively skips or de-emphasizes less relevant computations, confirming adaptive computation.

6. DISCUSSIONS

The results of the experiment plainly prove that the Cross-Layer Attention Adaptation (CLAA) framework is much better than its alternatives in terms of achieving a balance between inference accuracy and resource efficiency over embedded platforms. The findings support the practicality of activating specific layers of the nerves selectively relating to dynamic attention. Other than traditional CNNs that run all layers irrespective of complexity of inputs, CLAA finds redundant features at early inference phases and removes unwanted layers on-the-fly, which saves latency and energy. On the CIFAR-10 with and Tiny ImageNet models, CLAA was also found to consistently achieve an inference reduction of more than 45{per cent} relative to MobileNetV2-0.35 and EfficientNet-lite0, with the greatest reductions being observed on low-resource platforms (such as the STM32F746ZG). The decrease in latency is directly related to the fact that the model dynamically omits some calculations, which does not affect the level of classification accuracy. It is especially remarkable that claa still achieved a top-1 accuracy of 87.2 percent on CIFAR-10 (only 0.5 percent lower than its full-depth baseline), but using 36.2 percent less energy. This implies that deactivation of intelligent layers

may not always negatively affect the reliability of the model when used with sufficiently trained attention controller.

These insights are also backed up by the tables of comparisons and pie charts. About 60 percent of the samples did two or more layer skip and this confirmed that the model was competent in the difference between complex and trivial inputs. In addition, the analysis of the heat map shows some smoothing of the activation strengths as one goes deeper into the layers particularly in less complex inputs. This conforms to the intuition of the idea that several inputs that do not need high level abstraction and can be accurately classified based on the low-level features alone. The confusion matrix provided by CLAA on CIFAR-10 indicates the slight deterioration of the ability to separate classes. The misclassifications are quite normal since they fall within the accepted limits and mostly they happen amongst the visually similar classes. This kind of behavior prevails even when using full-depth models and shows that CLAA does not increase inter-class confusions, hence maintenance of the classification integrity[27].

These results prove the fact that CLAA is not only efficient in computations, but architecture-independent and deployment-friendly. Its architecture would allow it to be plugged into many existing CNN backbones and accelerators. Moreover, its small memory footprint and quantization facility enables it to be deployed to the edges, especially in battery operated devices, drones, health monitors, and internet of things-based industrial sensors. Having determined its strengths, it is necessary to note the general implications of CLAA. This model puts forward a transition to content-aware computation in neural inference, where future architectures are invited to make a transition to trained, adaptable models. This form of evolution promises on a new era of edge intelligence with energy, latency, and accuracy dynamically balanced based on an input.

7. CURRENT LIMITATIONS AND FUTURE ENHANCEMENTS

Although the Cross-Layer Attention Adaptation (CLAA) frame framework exhibits positive results, there are numerous limitations remaining, the existence of which can determine the scalability and practicality of the frame in broad real-life applications. First, the dynamic control of layers operation by the attention mechanism carries a slight drawback in the form of complexity and extra parameter number in the model. The cost may be quite insignificant where their reduction of inference load offsets it, but it may be prohibitive in some ultra-low-power microcontrollers with very limited computational budget (real-time safety-critical applications, such as autonomous driving or biomedical implants).

The other factor that is important to consider is that of generalizability of the layer skipping controller. The existing design is mainly trained alongside the backbone network and will be loosely connected with the data distribution that is observed at the time of training. In turn, the change of input domain feature, e.g., the difference of the resolution, change of noise, change of the lighting, etc., can diminish the functioning of the gates of attention. Solid cross-domain validation and domain adaptation systems have not yet become fully integrated and as such there is limited ability to use CLAA in heterogeneous or non-stationary scenarios. Also, the performance of the model has so far been measured on the classification of images through medium complexity datasets such as CIFAR-10 and Tiny ImageNet. Although these datasets are real examples of edge AI benchmarks, they fail to capture the complexity of high-resolution inputs or multi-modal inference DNNs like video analytics, audio-visual fusion, or time-series forecasting. It would be necessary to support CLAA applicability in such domains through planning flexibility, branch sensitive to attention management, and latency aware synchronizations.

Layer skipping is, when viewed in terms of hardware deployment, an irregular execution pattern that is not likely to be well suited to every platform. As an example, accelerators that are well-suited to uniform workloads (e.g. GPUs, which are pipelined, or NPUs, whose heterogeneity provides the same benefit) may end up wasting compute units when applying only partial layer activations. On such architectures, custom hardware support or compiler-level optimizations might be needed to make full use of the energy efficiency of CLAA. The mentioned challenges may be addressed through a few prospective directions that could be leveraged in the future. The first one is to create a hybrid controller, with both static (and dynamic) gating, so that a fixed set of base layers can be permanently on and the remaining dynamically injected in projection with the possible characteristics of inputs. This simplifies the controllers and is guaranteed to provide baseline representational power. Additionally, the learning mechanism of the system through online could induce changes through the adaptation of the

learner to new data patterns, which makes it robust in real-life applications.

The combination of CLAA with other vision structures based on transformers or graph neural networks (GNNs) also provides the means of extending flexible adaptability to other network structures. The development of attention calibration, explainability and uncertainty quantification research will further assist its deployment to high-stake tasks like healthcare and defense. Finally, it is possible that lightweight neural architecture search (NAS) approaches can aid in finding the absolute layer gating structure ideally suited to hardware necessity and target application.

8. CONCLUSION

The Cross-Layer Attention Adaptation (CLAA) framework proposed in this paper is a scalable, media smart solution to neural inference in real-time on embedded systems offering system-level performance productivity through selective layer activation and an adaptive attention mechanism. The CLAA was tested at scale and compatible with various hardware platforms, including Raspberry Pi 4B and STM32F746ZG, in addition to being trained at scale across different benchmark datasets including CIFAR-10 and Tiny ImageNet, CLAA consistently reduced latency up to 45 percent and energy costs by more than 36 percent with minimal loss in accuracy. In contrast to traditional CNNs, which use fixed-depth inference, CLAA dynamically adapts the partialities of computation to the input complexity in order to perform content-aware processing, which is compliant with edge computing requirements. Other visualizations like pie chart, confusion matrix, and heat map support the logic of the framework and the improved performance of the framework. The modular nature of the architecture guarantees ease of integration into preexisting backbones of neural networks, increasing its scalability to other areas such as smart sensing, medical, and industry IoT. This work facilitates the development of knowledge on energy-efficient and latency-sensitive AI models which can be deployed in the real-world, resource-constrained setting since it illustrates how neural networks can be designed to be adaptively deep.

9. FUTURE WORK

The positive initial findings of the Cross-Layer Attention Adaptation (CLAA) framework can be extended to more detailed and heterogeneous deep learning models, such as in transformer-based visual networks and spatiotemporal neural networks on video inference, as an important line of future work. Combining hybrid attention mechanisms by using spatial, time, and layer-wise gating would possibly increase flexibility between modalities and real-time applications. Also, the attention controller would gain adaptation, e.g., by adding online learning mechanisms, to better manage diverse requirements on parts of the overall system, e.g., mobile robotics or watching over the environment. A second possible future is to co-optimize CLAA with hardware-aware neural architecture search (NAS) to automatically specialize attention control structures to the use of a particular microcontroller or AI accelerator. Moreover, an attempt will be taken to increase the explainability and transparency of the model, where the interpretable attention scores visualizations will be introduced, and trust and traceability will be improved in safety-considered applications. Lastly, the scale-up of the verification and validation of CLAA on a wider range of real-world embedded systems, e.g., edge FPGAs, battery-powered sensor nodes, and neuromorphic processors will contribute to the optimization of CLAA scalability, reliability, and performance under the practical constraints and advance the state of the art on adaptive intelligence at the network edge.

REFERENCES

[1].M. Jain et al., "Energy-Efficient Neural Network Inference for Microcontroller Applications," *IEEE Transactions on Computers*, vol. 73, no. 1, pp. 32–45, Jan. 2024.

[2].Rachiraju, Sai Chandra, and Madamala Revanth. "Feature extraction and classification of movie reviews using advanced machine learning models." In 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 814-817. IEEE, 2020.

[3].N. Sharma, K. Sudheer Reddy, R. B. Pittala, M. D. Reddy, J. A. Sri and G. Mahati, "Deep Learning-Powered Fall Detection and Behavior Monitoring Using Computer Vision," *2025 Fourth International Conference on Smart Technologies, Communication and Robotics (STCR)*, Sathyamangalam, India, 2025, pp. 1-6, doi: 10.1109/STCR62650.2025.11020068.

[4].Srinagesh, C., Prasad, T. B., & Reddy, K. S. (2010). "Leveraging technology for creating awareness of problem-solving skills to engineering students". International Conference on Technology for Education, 238–239. doi:10.1109/t4e.2010.5550107

[5].T. Han, J. Wu, and C. Lin, "Quantization and Distillation Strategies for Embedded AI," *IEEE Access*, vol. 11, pp. 75632–75648, 2023.

[6].A. Vaswani et al., "Attention Is All You Need," in *Proc. of NeurIPS*, Long Beach, USA, 2017.

[7].N. Singh, K. Sudheer Reddy and R. Aluvalu, "AI Driven Waste Classification for Smart Recycling," 2025 3rd International Conference on Disruptive Technologies (ICDT), pp. 1590-1594, doi: 10.1109/ICDT63985.2025.10986692.

[8].X. Wang et al., "Cross-Layer Contextual Attention for Multi-Scale Feature Integration," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 2, pp. 1482–1494, 2024.

[9].Rama Krishna, B., Nagabhushana Rao, M., & Pittala, R. B. (2018). An algorithm to find the geo-map using the social media–Facebook. Journal of Advanced Research in Dynamical and Control Systems, 10(7 Special Issue), 1538.

[10]. Hema, M.S.,et al (2022). Identification and Classification of Brain Tumor Using Convolutional Neural Network with Autoencoder Feature Selection. In: Balas, V.E., Sinha, G.R., Agarwal, B., Sharma, T.K., Dadheech, P., Mahrishi, M. (eds) Emerging Technologies in Computer Engineering: Cognitive Computing and Intelligent IoT. ICETCE 2022. Communications in Computer and Information Science, vol 1591. Springer, Cham. https://doi.org/10.1007/978-3-031-07012-9_22

[11]. S. Choukroun, E. Kravchik, and S. Kisilev, "Low-Bit Quantization of Neural Networks for Embedded Inference: A Survey," *IEEE Access*, vol. 9, pp. 19615–19630, 2021.

[12]. J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," in *Proc. of IEEE CVPR*, Salt Lake City, USA, 2018.

[13]. R. Imani et al., "Efficient Attention for TinyML Devices Using Low-Rank Binary Projections," in *Proc. of IEEE ISCAS*, Montreal, Canada, 2022.

[14]. S. R. K, A. R. V, M. K, and S. K. C, "Prediction of COVID-19 outbreak in India by employing epidemiological models," J. Comput. Sci., vol. 16, no. 7, pp. 886--890, 2020, doi: 10.3844/jcssp.2020.886.890.

[15]. Y. Wu et al., "BlockDrop: Dynamic Inference Paths in Residual Networks," in *Proc. of IEEE CVPR*, Salt Lake City, USA, 2018.

[16]. Y. Lin et al., "Multi-Scale Attention for Semantic Segmentation in Edge Devices," *IEEE Transactions on Image Processing*, vol. 33, pp. 4764–4776, 2024.

[17]. Reddy, K.S., Sharma, N., Ashalatha, T., Raju, B.R. (2024). An Intelligent Ensemble Architecture to Accurately Predict Housing Price for Smart Cities. In: Satheeskumaran, S., Zhang, Y., Balas, V.E., Hong, Tp., Pelusi, D. (eds) Intelligent Computing for Sustainable Development. ICICSD 2023. Communications in Computer and Information Science, vol 2122. Springer, Cham. https://doi.org/10.1007/978-3-031-61298-5_9

[18]. N. Strom, M. Kaiser, and J. Wang, "Dynamic Neural Inference on Microcontrollers Using Conditional Execution Graphs," *IEEE Embedded Systems Letters*, vol. 16, no. 1, pp. 45–48, Mar. 2024.

[19]. C. Li, X. Ma, and Y. Zhang, "Layer-Wise Pruning and Attention-Aware Inference for Lightweight AI," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 34, no. 1, pp. 102–113, Jan. 2024.

[20]. P. Srinivas and V. Nair, "Dynamic Computation Graphs for Real-Time Deep Inference on Microcontrollers," in *Proc. of IEEE ECRTS*, Prague, Czech Republic, 2023.

[21]. R. Wu et al., "A Controller-Based Approach for Layer Skipping in Edge Neural Networks," *ACM Trans. on Embedded Computing Systems*, vol. 23, no. 2, pp. 1–22, 2024.

[22]. K. Sudheer Reddy, M. Kantha Reddy and V. Sitaramulu, "An effective data preprocessing method for Web Usage Mining," 2013 International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India, 2013, pp. 7-10, doi: 10.1109/ICICES.2013.6508197.

[23]. Babu Pittala R, Asha Kiran M, Sharma N, et al. ATM-AM: An Interpretable Attention SHAP Aligned Framework for Text Classification across IMDb, Amazon, and SST-2. Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology. 2026;0(0). doi:10.1177/18758967261420571

[24]. M A Kiran, etal, "Integrating Vision Transformers and CNNs for High-Accuracy Weed Detection in Crop Fields," Engineering Letters, vol. 34, no. 1, pp-50-63, 2026.

[25]. P. Akshaya, etal, "Multi-Sensor Fusion for Emotion Recognition using Machine Learning," *2025 5th International Conference on Intelligent Technologies (CONIT)*, HUBBALI, India, 2025, pp. 1-6, doi: 10.1109/CONIT65521.2025.11167520.

[26]. G. A. Goud *et al.*, "Decentralized Tamperproof Certificate Validation in

Education: A Blockchain Approach," *2025 IEEE 4th World Conference on Applied Intelligence and Computing (AIC)*, GB Nagar, Gwalior, India, 2025, pp. 341-346, doi: 10.1109/AIC66080.2025.11212033.

[27]. S. Agarwal, M. A. Kiran, M. Thaile, M. A. Khan and G. S. Aparna, "LoRA-Enhanced BERT with Contrastive Learning for Political Sentiment Analysis," *2025 5th International Conference on Intelligent Technologies (CONIT)*, HUBBALI, India, 2025, pp. 1-6, doi: 10.1109/CONIT65521.2025.11166840.