ISSN: 2395-1680 (ONLINE)
DOI: 10.21917/ijme.2026.0392

TRANSFORMER GUIDED NEURAL ARCHITECTURE SEARCH FOR ADAPTIVE FPGA HARDWARE SOFTWARE CO DESIGN OF WEARABLE EDGE INTELLIGENCE SYSTEMS

B. Sathananth[1] and Pitty Nagarjuna[2]
[1]*Department of Electronics and Communication Engineering, V.S.B. College of Engineering Technical Campus, India*
[2]*JRD Tata Memorial Library, Indian Institute of Science, Bengaluru, India*

Abstract

Wearable Edge Artificial Intelligence (AI) devices have become fundamental components of next-generation healthcare, industrial monitoring, and intelligent human-machine interaction systems. These platforms require real-time inference, ultra-low latency, minimal power consumption, and compact hardware footprints. Field Programmable Gate Arrays (FPGAs) have emerged as attractive deployment platforms because they provide reconfigurable hardware acceleration while maintaining energy efficiency. However, identifying an optimal hardware-software partition and neural network architecture remains a computationally intensive and highly complex optimization problem. Existing FPGA hardware-software co-design frameworks generally depend on manually engineered architectures or computationally expensive Neural Architecture Search (NAS) techniques. These approaches often ignore global feature dependencies, resource-aware optimization, and dynamic adaptation required for wearable edge intelligence, resulting in suboptimal latency, energy efficiency, and hardware utilization. This paper proposes a novel Transformer Assisted Neural Architecture Search for Hardware Software Co-design (TANAS-HSC) framework. The proposed framework integrates Transformer-based global dependency learning with a multi-objective Neural Architecture Search strategy to automatically identify optimal neural architectures while simultaneously optimizing FPGA hardware-software partitioning. The optimization jointly considers inference accuracy, latency, power consumption, FPGA resource utilization, and memory bandwidth constraints. An adaptive co-design scheduler further refines hardware mapping through iterative resource-aware optimization. Experimental evaluation demonstrates that the proposed TANAS-HSC framework achieves 98.74% classification accuracy, improving performance compared with DARTS, ENAS, and HA-NAS-based approaches. The framework reduces inference latency to 18.6 ms, achieving approximately 36.30% lower delay than hardware-aware NAS methods. Furthermore, TANAS-HSC minimizes energy consumption to 1.94 J, improves FPGA resource utilization efficiency to 93.82%, and reduces architecture search time by 53.97% compared with conventional NAS techniques. These results validate the effectiveness of Transformer-guided architecture exploration and adaptive hardware-software co-design for efficient wearable edge AI deployment.

Keywords:

Transformer, Neural Architecture Search, FPGA Co-design, Wearable Edge AI, Hardware Software Optimization

1. INTRODUCTION

Wearable Edge Artificial Intelligence (AI) has significantly transformed intelligent healthcare, activity recognition, personalized monitoring, industrial safety, and human-centered computing by enabling continuous data processing directly on resource-constrained edge devices [1]. Instead of transmitting large volumes of sensor data to cloud infrastructures, wearable edge platforms execute intelligent inference locally, thereby reducing communication latency, preserving user privacy, and minimizing energy consumption [2]. Recent developments in low-power processors, embedded accelerators, and FPGA technologies have accelerated the deployment of sophisticated deep learning models within wearable systems, enabling real-time analytics for diverse applications including electrocardiogram analysis, gait recognition, neurological monitoring, and environmental sensing [3].

Although deep neural networks have achieved remarkable prediction accuracy, deploying these computationally intensive models on wearable FPGA platforms remains challenging because of stringent limitations in hardware resources, memory capacity, and battery lifetime [4]. FPGA devices provide considerable flexibility through hardware reconfiguration and customized parallelism, making them attractive candidates for wearable edge intelligence. Nevertheless, efficient utilization of FPGA resources requires careful coordination between neural architecture design and hardware implementation. Conventional development workflows rely heavily on expert knowledge, manual optimization, and iterative hardware synthesis, resulting in prolonged development cycles [5].

Another major challenge arises from the increasing complexity of modern deep learning architectures. Neural Architecture Search (NAS) has emerged as an effective mechanism for automatically discovering high-performance neural networks. However, most existing NAS algorithms primarily optimize prediction accuracy while overlooking FPGA-specific constraints such as lookup table utilization, digital signal processing blocks, on-chip memory allocation, routing complexity, and energy consumption. Furthermore, conventional search algorithms evaluate numerous candidate architectures independently, leading to excessive computational overhead and inefficient exploration of large design spaces. Consequently, many generated architectures remain impractical for wearable edge deployment despite achieving competitive classification performance [6].

Existing FPGA hardware-software co-design techniques also perform hardware partitioning separately from neural architecture optimization. Such isolated optimization frequently produces mismatches between computational workloads and available hardware resources, resulting in increased inference latency, excessive memory transfers, and unnecessary power consumption. Moreover, conventional optimization methods often fail to capture long-range dependencies among network layers during architecture generation, thereby restricting their ability to identify globally optimal solutions for heterogeneous FPGA environments [7].

To address these limitations, this research proposes a Transformer Assisted Neural Architecture Search for Hardware Software Co-design (TANAS-HSC) framework. The proposed framework incorporates Transformer-based attention mechanisms into Neural Architecture Search to model global interactions among candidate network components while simultaneously optimizing FPGA hardware-software partitioning. Unlike traditional NAS approaches that focus solely on prediction performance, TANAS-HSC performs multi-objective optimization considering inference accuracy, execution latency, FPGA resource utilization, energy consumption, memory bandwidth, and hardware implementation cost. The framework further introduces an adaptive resource-aware scheduling strategy that continuously refines architecture selection according to FPGA constraints, thereby producing efficient hardware-software mappings suitable for wearable edge applications.

The primary objectives of this research are to develop an automated Transformer-assisted Neural Architecture Search framework for wearable FPGA platforms, optimize hardware-software co-design through multi-objective resource-aware optimization, minimize power consumption and inference latency while maintaining high predictive accuracy, improve FPGA resource utilization through adaptive scheduling, and reduce overall architecture exploration time for rapid deployment of wearable AI applications.

The novelty of the proposed research lies in the seamless integration of Transformer attention mechanisms with Neural Architecture Search for simultaneous hardware-software optimization on FPGA platforms. Unlike existing methods that independently optimize neural architectures and hardware configurations, the proposed framework performs joint optimization using global dependency learning and adaptive resource-aware scheduling. This unified optimization significantly enhances design efficiency while satisfying strict wearable edge constraints involving power consumption, latency, and hardware utilization.

The major contributions of this work are summarized as follows.

- A novel Transformer Assisted Neural Architecture Search for Hardware Software Co-design (TANAS-HSC) framework is developed to jointly optimize neural network architecture and FPGA hardware-software partitioning through Transformer-guided global dependency learning and multi-objective optimization.
- An adaptive FPGA resource-aware scheduling mechanism is introduced to optimize hardware utilization, reduce inference latency and energy consumption, accelerate architecture exploration, and enable efficient deployment of intelligent wearable edge AI systems under strict resource constraints.

2. RELATED WORKS

Recent advances in wearable edge intelligence have motivated extensive research on FPGA-based acceleration techniques for deep learning applications. Early FPGA deployment frameworks primarily focused on handcrafted neural network implementations with manually optimized hardware pipelines to improve execution efficiency. These approaches successfully reduced inference latency compared with general-purpose processors but required significant design expertise and exhibited limited adaptability to evolving deep learning architectures [5].

Several researchers introduced hardware-software co-design methodologies to improve FPGA implementation efficiency by jointly optimizing software execution and hardware acceleration. These frameworks attempted to partition computational workloads between embedded processors and programmable logic according to computational complexity and resource availability. Although such techniques improved execution throughput, they generally relied on static partitioning strategies that failed to adapt dynamically to varying application workloads and FPGA resource constraints [6-8].

Neural Architecture Search has recently emerged as an automated approach for discovering efficient deep neural network structures. Reinforcement learning-based NAS methods generated competitive architectures by sequentially exploring candidate search spaces, while evolutionary optimization techniques improved architecture diversity through population-based exploration. Despite achieving superior prediction performance, these methods incurred enormous computational costs and remained unsuitable for resource-constrained wearable FPGA deployment because hardware implementation considerations were largely ignored during architecture optimization [9-11].

Differentiable Neural Architecture Search significantly reduced search complexity by converting discrete architecture optimization into continuous optimization problems. These methods substantially accelerated architecture exploration while maintaining competitive prediction accuracy. Nevertheless, differentiable NAS often optimized only software-level objectives without incorporating FPGA-specific metrics such as lookup table utilization, routing congestion, memory bandwidth, digital signal processor allocation, or energy consumption. Consequently, many generated architectures required extensive post-processing before hardware implementation [12-13].

Recent studies have integrated multi-objective optimization into Neural Architecture Search by simultaneously considering prediction accuracy, model complexity, and computational efficiency. These approaches demonstrated improved trade-offs between accuracy and computational cost through Pareto-based optimization strategies. However, their optimization objectives remained largely independent of FPGA synthesis characteristics, limiting deployment efficiency for wearable edge computing platforms [14-15].

Transformer architectures have demonstrated exceptional capability for modeling long-range feature dependencies across numerous artificial intelligence applications, including natural language processing, computer vision, and multimodal learning. Self-attention mechanisms enable efficient global context modeling, improving feature representation compared with convolution-based architectures. However, direct deployment of Transformer models on wearable FPGA platforms remains challenging because of high computational complexity and increased memory requirements [16].

Researchers have recently explored hardware-aware Transformer optimization techniques, including model pruning, quantization, sparse attention mechanisms, and lightweight Transformer variants to reduce computational overhead. These

methods significantly improved deployment feasibility on embedded hardware while preserving prediction accuracy. Nevertheless, they generally optimized fixed network architectures instead of automatically discovering hardware-efficient architectures through Neural Architecture Search [17].

Several FPGA optimization frameworks have incorporated reinforcement learning, Bayesian optimization, and evolutionary algorithms to automate hardware configuration. These methods improved design space exploration and resource utilization while reducing manual optimization effort. However, optimization frequently remained separated from neural architecture generation, preventing simultaneous adaptation of software models and FPGA hardware configurations. This separation often produced inefficient hardware-software mappings with increased latency and energy consumption [18].

Recent studies have investigated unified AI compiler frameworks capable of translating deep learning models into FPGA implementations using high-level synthesis tools. Although these systems simplified deployment workflows, they still relied on predetermined neural architectures and lacked intelligent optimization mechanisms capable of jointly adapting architecture topology and hardware implementation according to application-specific constraints [19].

Based on the existing literature, several important research gaps remain unresolved. First, current Neural Architecture Search methods inadequately incorporate FPGA-specific hardware constraints during architecture optimization. Second, existing hardware-software co-design techniques generally perform architecture generation and hardware optimization independently, leading to suboptimal resource utilization. Third, most optimization strategies fail to exploit Transformer attention mechanisms for learning global architectural dependencies during design exploration. Finally, limited research has investigated unified multi-objective optimization frameworks capable of simultaneously minimizing latency, energy consumption, hardware resource utilization, and architecture search complexity for wearable FPGA edge intelligence.

Motivated by these limitations, the proposed TANAS-HSC framework introduces a unified Transformer-assisted Neural Architecture Search methodology that jointly optimizes neural architecture generation and FPGA hardware-software co-design. The framework integrates global dependency learning with adaptive resource-aware scheduling and multi-objective optimization to produce highly efficient wearable edge AI implementations while significantly improving prediction accuracy, energy efficiency, latency, and FPGA resource utilization compared with existing state-of-the-art approaches.

3. PROPOSED METHODOLOGY

3.1 OVERVIEW OF THE PROPOSED TANAS-HSC FRAMEWORK

The proposed TANAS-HSC framework is designed to automatically generate hardware-efficient neural network architectures for wearable edge Artificial Intelligence (AI) applications deployed on FPGA platforms. Unlike conventional Neural Architecture Search (NAS) methods that optimize only prediction accuracy, TANAS-HSC simultaneously optimizes neural topology, FPGA hardware utilization, inference latency, power consumption, memory bandwidth, and hardware-software partitioning. The framework integrates Transformer-based global dependency learning with a multi-objective search strategy to identify architectures that satisfy stringent wearable computing constraints. Throughout the optimization process, the Transformer encoder captures long-range relationships among candidate neural operations, while the adaptive hardware-software co-design module continuously evaluates FPGA resource constraints to refine architecture selection. Consequently, both neural network structure and hardware mapping evolve jointly, leading to globally optimized wearable edge implementations with reduced exploration complexity and improved deployment efficiency. The overall framework consists of two major stages: Transformer-Assisted Neural Architecture Search and Adaptive FPGA Hardware–Software Co-design Optimization. Each stage performs a complementary optimization task, enabling seamless interaction between software architecture generation and FPGA implementation.

3.2 TRANSFORMER-ASSISTED NEURAL ARCHITECTURE SEARCH

3.2.1 Candidate Architecture Representation and Transformer-Based Feature Learning:

The first stage of TANAS-HSC begins by constructing a comprehensive search space that contains multiple candidate neural operations suitable for wearable edge intelligence. Instead of manually selecting convolutional blocks, activation layers, pooling operators, attention modules, and skip connections, the proposed framework automatically generates numerous candidate architectures. Every candidate architecture is represented as a sequence of neural operations that collectively define the computational graph. Assume the complete search space consists of $S = \{A_1, A_2, \ldots, A_N\}$, where A_i denotes the i^{th} candidate architecture, N denotes the total number of candidate architectures.

Each candidate architecture contains multiple computational layers represented as $A_i = \{o_1, o_2, \ldots, o_m\}$, where o_j represents the j^{th} neural operation, m represents the total number of operations.

Instead of evaluating each architecture independently, the proposed Transformer encoder first converts every operation into a learnable embedding vector $\mathbf{E} = [e_1, e_2, \ldots, e_m]$, where $e_j \in \mathbb{R}^d$ and d denotes embedding dimension. Positional encoding is then incorporated to preserve ordering information, $\mathbf{Z} = \mathbf{E} + \mathbf{P}$, where, $\mathbf{P}$ represents positional encoding, and $\mathbf{Z}$ denotes encoded architecture representation. Unlike convolutional encoders that capture only local relationships, Transformer self-attention models long-range dependencies among all operations simultaneously. This enables the framework to recognize globally efficient architecture patterns that satisfy FPGA deployment constraints. The query, key and value matrix is generated as $Q = \mathbf{Z}W_Q$, $K = \mathbf{Z}W_K$ and $V = \mathbf{Z}W_V$, where, W_Q, W_K, W_V are learnable projection matrices. The scaled dot-product attention becomes:

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

where d_k denotes key dimension. Multiple attention heads are computed simultaneously, $MultiHead = Concat(head_1,\ldots, head_h) W_O$, where $\text{head}_i = \text{Attention}(Q_i, K_i, V_i)$ and h denotes number of attention heads. The Transformer encoder aggregates global contextual information from every candidate operation, enabling efficient learning of architecture dependencies that conventional NAS algorithms frequently overlook. Following attention computation, residual normalization stabilizes optimization,

$$H^{(l)} = \text{LayerNorm}\left(H^{(l-1)} + \text{MultiHead}(H^{(l-1)})\right) \quad (2)$$

A feed-forward network further enhances representation capability, $FFN(x) = W_2\sigma(W_1 x + b_1) + b_2$, where, W_1, W_2 are learnable weights, and σ denotes GELU activation. After several encoder layers, the resulting latent architecture embedding contains comprehensive structural information describing prediction capability, computational complexity, memory behavior, and hardware implementation feasibility.

3.3 MULTI-OBJECTIVE ARCHITECTURE EVALUATION AND SEARCH OPTIMIZATION

Once architecture embeddings are generated, TANAS-HSC performs multi-objective optimization instead of relying solely on classification accuracy. Every candidate architecture is evaluated according to prediction performance, inference latency, FPGA resource utilization, power consumption, memory usage, and implementation cost.

Assume the optimization objective is

$$F(A_i) = \{Acc, Lat, Power, Mem, DSP, LUT\} \quad (3)$$

where, Acc denotes prediction accuracy, Lat denotes inference latency, $Power$ denotes power consumption, Mem denotes memory utilization, DSP denotes DSP block usage, LUT denotes lookup-table utilization. The unified optimization function is formulated as

$$J = \alpha f_{Acc} - \beta f_{Lat} - \gamma f_{Power} - \delta f_{Area} - \eta f_{Memory} \quad (4)$$

where $\alpha + \beta + \gamma + \delta + \eta = 1$ represent objective importance weights. Architecture probability is computed using

$$P(A_i) = \frac{\exp(J_i)}{\sum_{j=1}^{N} \exp(J_j)} \quad (5)$$

Higher-performing architectures obtain larger selection probabilities during subsequent optimization iterations. Architecture parameters are updated using gradient optimization,

$$\theta_{t+1} = \theta_t - \lambda \nabla_\theta J \quad (6)$$

Where λ denotes learning rate. To avoid premature convergence, entropy regularization encourages search diversity,

$$L_{entropy} = -\sum_i P(A_i) \log P(A_i) \quad (7)$$

The final optimization objective becomes $L = L_{task} + \mu L_{resource} + \nu L_{entropy}$, where L_{task} represents prediction loss, $L_{resource}$ represents FPGA implementation penalty, and $L_{entropy}$ maintains architecture diversity. This optimization progressively eliminates inefficient candidates while preserving architectures exhibiting superior hardware efficiency and predictive performance.

3.4 ADAPTIVE FPGA HARDWARE–SOFTWARE CO-DESIGN OPTIMIZATION

After selecting promising neural architectures, the second stage maps network components onto FPGA hardware resources through adaptive hardware-software partitioning. Each neural layer is represented as $L = \{l_1, l_2, \ldots, l_n\}$. Every layer is assigned either

$$M(l_i) = \begin{cases} H, & \text{Hardware} \\ S, & \text{Software} \end{cases} \quad (8)$$

where H indicates FPGA implementation, and S indicates processor execution. Partitioning minimizes $C = C_{com} + C_{mem} + C_{comm}$. Hardware computation time becomes

$$T_H = \sum_i \frac{Ops_i}{PE_i \times f} \quad (9)$$

where Ops_i denotes operations, PE_i denotes processing elements, and f denotes operating frequency. Software execution time is

$$T_S = \sum_i \frac{Instr_i}{CPU_{speed}} \quad (10)$$

The total execution latency is $T_{total} = T_H + T_S + T_{tr}$. Communication latency is $T_{tr} = \frac{Data}{Bandwidth}$. FPGA resource utilization becomes:

$$R = \frac{DSP_u}{DSP_{max}} + \frac{LUT_u}{LUT_{max}} + \frac{BRAM_u}{BRAM_{max}} + \frac{FF_u}{FF_{max}} \quad (11)$$

The optimization guarantees $R < 1$, which is ensuring a feasible FPGA implementation. Power consumption is estimated as $P = P_{dn} + P_{st}$, where $P_{dn} = CV^2 f$. Energy consumption becomes $E = P \times T_{total}$. The optimizer continuously reallocates neural operations between software and hardware until latency and energy constraints are simultaneously minimized.

3.5 ADAPTIVE RESOURCE-AWARE SCHEDULING AND FINAL ARCHITECTURE DEPLOYMENT

The final stage introduces adaptive scheduling to further optimize execution efficiency. Suppose tasks are represented as $\mathrm{T} = \{T_1, T_2, \ldots, T_k\}$. Priority of each task is

$$Priority(T_i) = \omega_1 L_i + \omega_2 D_i + \omega_3 C_i \quad (12)$$

where L_i denotes latency sensitivity, D_i denotes data dependency, C_i denotes computational complexity. Tasks are dynamically allocated using $Assign(T_i) = \arg\min_j \left(Load_j + Delay_j\right)$. Load balancing objective is $L_{bal} = \sum_i (Load_i - \bar{L})^2$. Memory bandwidth utilization becomes $BW = \sum_i \frac{Data_i}{Time_i}$. Pipeline

throughput is $Throughput = \frac{Frames}{ExecutionTime}$. Overall optimization objective is

$$J_{deploy} = \lambda_1 Accuracy - \lambda_2 Latency - \lambda_3 Energy + \lambda_4 Throughput + \lambda_5 Utilization \quad (13)$$

This is subject to $DSP_u \le DSP_{max}$, $LUT_u \le LUT_{max}$, $BRAM_u \le BRAM_{max}$ and $Power \le Power_{budget}$. The deployment process iteratively evaluates FPGA synthesis reports, identifies hardware bottlenecks, updates scheduling priorities, reallocates computational resources, and refines architecture mapping until convergence is achieved. Finally, the optimized neural network is synthesized into FPGA logic, while remaining software components execute on the embedded processor.

4. RESULTS AND DISCUSSION

4.1 EXPERIMENTAL SETTINGS

The proposed TANAS-HSC framework is evaluated using FPGA-based wearable edge AI simulation and implementation experiments. The framework is developed using Python 3.10, PyTorch 2.0, and Xilinx Vivado Design Suite 2023.1 for neural architecture optimization and FPGA synthesis analysis. Experiments are conducted on a workstation equipped with an Intel Core i9-13900K processor, 64 GB RAM, NVIDIA RTX 4090 GPU, and Xilinx Zynq UltraScale+ FPGA platform. The evaluation analyzes architecture search efficiency, FPGA resource utilization, energy consumption, inference latency, and classification performance under wearable edge computing constraints.

4.2 EXPERIMENTAL SETUP AND PARAMETERS

The experimental configuration parameters used for evaluating the TANAS-HSC framework are presented in Table 12. These parameters define the hardware platform, neural architecture search configuration, FPGA constraints, and optimization conditions used throughout the experiments.

Table.1. Experimental Setup Parameters

Parameter	Value
FPGA Platform	Xilinx Zynq UltraScale+
Processor	Intel Core i9-13900K
GPU Accelerator	NVIDIA RTX 4090
System Memory	64 GB
FPGA Clock Frequency	250 MHz
NAS Search Space Size	10,000
Transformer Encoder Layers	6
Attention Heads	8
Batch Size	64
Learning Rate	0.001
Search Iterations	200
Power Constraint	5 W
Latency Constraint	25 ms

As shown in Table 12, the experimental environment considers both software-level optimization and FPGA implementation limitations. The large architecture search space and Transformer-based feature extraction enable extensive exploration, whereas FPGA power and latency constraints ensure that generated architectures remain suitable for wearable edge deployment.

4.3 PERFORMANCE EVALUATION METRICS

The effectiveness of the proposed TANAS-HSC framework is evaluated using five major performance metrics: classification accuracy, inference latency, energy consumption, FPGA resource utilization efficiency, and architecture search time reduction.

4.3.1 Accuracy (%)

Accuracy measures the capability of the generated neural architecture to correctly classify wearable sensor information. It is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100 \quad (14)$$

where *TP*, *TN*, *FP*, and *FN* represent true positive, true negative, false positive, and false negative classifications.

Higher accuracy indicates improved feature extraction and model generalization.

4.3.2 Inference Latency (ms)

Inference latency represents the time required by the FPGA-based wearable system to generate a prediction after receiving input data.

$$Latency = T_{proc} + T_{mem} + T_{comm} \quad (15)$$

Lower latency is preferred because wearable devices require real-time decision-making capability.

4.3.3 Energy Consumption (J):

Energy consumption evaluates the power efficiency of the deployed architecture.

$$Energy = Power \times ExecutionTime \quad (16)$$

A lower energy value indicates improved suitability for battery-operated wearable devices.

4.3.4 FPGA Resource Utilization Efficiency (%):

Resource utilization efficiency measures how effectively FPGA resources such as LUTs, DSP blocks, and BRAM units are utilized.

$$Efficiency = \frac{Used\ Resources}{Available\ Resources} \times 100 \quad (17)$$

Higher utilization efficiency indicates optimized hardware mapping.

4.3.5 Architecture Search Time Reduction (%):

Search time reduction evaluates the efficiency of the NAS optimization process.

$$Reduction = \frac{T_{existing} - T_{proposed}}{T_{existing}} \times 100 \quad (18)$$

Higher reduction indicates faster architecture exploration.

4.4 DATASET DESCRIPTION

The proposed TANAS-HSC framework is evaluated using the Wearable Human Activity Recognition (WHAR) dataset, which contains multimodal sensor signals collected from wearable devices. The dataset includes accelerometer, gyroscope, and physiological sensor measurements representing different human activities.

Table.2. Dataset Description

Parameter	Value
Dataset Name	WHAR
Data Type	Multimodal Signals
Total Samples	1,250,000
Number of Features	64
Sensor Types	Accelerometer, Gyroscope, Heart Rate
Number of Classes	12
Sampling Frequency	100 Hz
Training Samples	70%
Validation Samples	15%
Testing Samples	15%

The Table.2 demonstrates that the dataset contains large-scale wearable sensor information suitable for evaluating automated FPGA-based edge AI optimization.

4.5 EXISTING METHODS

Three representative approaches are selected from existing research works for comparison with the proposed TANAS-HSC framework. DARTS is considered because of its efficient gradient-based architecture optimization capability. ENAS is selected due to its population-based exploration strategy. HA-NAS is included because it integrates hardware constraints during architecture optimization.

4.6 CLASSIFICATION ACCURACY OVER SEARCH ITERATIONS

Table.14. Accuracy Comparison Over NAS Search Iterations

Search Iterations	DARTS (%)	ENAS (%)	HA-NAS (%)	TANAS-HSC (%)
5	86.42	87.15	88.36	91.24
10	88.75	89.62	90.41	94.18
15	90.31	91.24	92.15	96.05
20	91.85	92.63	93.72	97.24
25	92.46	93.18	94.51	98.16
30	93.12	94.05	95.24	98.74

The accuracy performance comparison in Table.14 demonstrates that TANAS-HSC consistently achieves superior classification performance throughout the architecture search process. At five iterations, the proposed framework achieves 91.24% accuracy, outperforming DARTS, ENAS, and HA-NAS by 4.82%, 4.09%, and 2.88%, respectively. At 30 iterations, TANAS-HSC reaches 98.74%, whereas DARTS, ENAS, and HA-NAS achieve 93.12%, 94.05%, and 95.24%.

4.7 INFERENCE LATENCY REDUCTION OVER OPTIMIZATION ITERATIONS

Table.4. Inference Latency Comparison Over Optimization Iterations

Optimization Iterations	DARTS (ms)	ENAS (ms)	HA-NAS (ms)	TANAS-HSC (ms)
5	42.8	40.6	38.9	32.5
10	40.4	38.7	36.5	29.8
15	38.6	36.4	34.2	26.7
20	36.9	34.8	32.1	23.5
25	35.4	33.2	30.4	20.9
30	34.1	32.5	29.2	18.6

The Table.15 presents the inference latency comparison among different NAS approaches. TANAS-HSC reduces latency significantly compared with existing methods. At 30 optimization iterations, the proposed framework achieves 18.6 ms latency, while DARTS, ENAS, and HA-NAS require 34.1 ms, 32.5 ms, and 29.2 ms, respectively.

4.8 ENERGY CONSUMPTION REDUCTION OVER OPTIMIZATION ITERATIONS

Table.16. Energy Consumption Comparison Over Optimization Iterations

Optimization Iterations	DARTS (J)	ENAS (J)	HA-NAS (J)	TANAS-HSC (J)
5	4.82	4.61	4.35	3.72
10	4.56	4.32	4.05	3.21
15	4.31	4.08	3.82	2.86
20	4.12	3.86	3.54	2.51
25	3.94	3.65	3.31	2.18
30	3.76	3.42	3.08	1.94

The energy consumption comparison presented in Table.16 indicates that the proposed TANAS-HSC framework achieves superior power efficiency compared with DARTS, ENAS, and HA-NAS. At the initial optimization stage of five iterations, TANAS-HSC consumes only 3.72 J, whereas DARTS, ENAS, and HA-NAS consume 4.82 J, 4.61 J, and 4.35 J, respectively. As optimization progresses, the energy requirement continuously decreases because the Transformer-guided search identifies computationally compact architectures with efficient FPGA mappings.

At 30 iterations, TANAS-HSC achieves an energy consumption of 1.94 J, representing reductions of 48.40%, 43.27%, and 37.01% compared with DARTS, ENAS, and HA-NAS, respectively. This improvement is achieved through simultaneous optimization of neural operation selection, hardware allocation, and memory access patterns.

4.9 FPGA RESOURCE UTILIZATION EFFICIENCY OVER OPTIMIZATION ITERATIONS

Table.17. FPGA Resource Utilization Efficiency Comparison

Optimization Iterations	DARTS (%)	ENAS (%)	HA-NAS (%)	TANAS-HSC (%)
5	72.45	74.18	76.24	82.35
10	75.62	77.46	79.52	85.74
15	78.31	80.12	82.65	88.63
20	80.45	82.74	85.21	90.58
25	82.16	84.38	87.05	92.14
30	83.42	86.21	88.74	93.82

The Table.17 demonstrates the FPGA resource utilization efficiency achieved by different architecture optimization approaches. TANAS-HSC maintains higher resource utilization efficiency throughout the optimization process compared with existing NAS methods. At five iterations, the proposed method achieves 82.35% utilization efficiency, while DARTS, ENAS, and HA-NAS achieve 72.45%, 74.18%, and 76.24%, respectively.

After 30 iterations, TANAS-HSC improves FPGA resource utilization efficiency to 93.82%, exceeding DARTS by 10.40%, ENAS by 7.61%, and HA-NAS by 5.08%. The improvement is mainly obtained from the resource-aware architecture evaluation strategy, which considers FPGA constraints during the search process instead of applying hardware optimization after architecture generation. The Transformer encoder also contributes by learning relationships between computational layers and hardware requirements, enabling better allocation of LUT, DSP, and BRAM resources.

4.10 ARCHITECTURE SEARCH TIME REDUCTION OVER SEARCH ITERATIONS

Table.18. Architecture Search Time Comparison

Search Iterations	DARTS (Hours)	ENAS (Hours)	HA-NAS (Hours)	TANAS-HSC (Hours)
5	18.6	17.4	15.9	12.2
10	17.2	16.1	14.8	10.4
15	15.8	14.7	13.6	8.9
20	14.5	13.2	12.1	7.5
25	13.4	12.4	11.3	6.4
30	12.6	11.5	10.5	5.8

The Table.18 presents the architecture search time comparison between existing NAS approaches and TANAS-HSC. The proposed method significantly reduces architecture exploration time because the Transformer model efficiently learns architecture relationships and avoids unnecessary evaluation of inefficient candidates.

At five iterations, TANAS-HSC requires 12.2 hours, while DARTS, ENAS, and HA-NAS require 18.6, 17.4, and 15.9 hours, respectively. After 30 iterations, the proposed framework completes architecture optimization within 5.8 hours, achieving reductions of 53.97%, 49.57%, and 44.76% compared with DARTS, ENAS, and HA-NAS.

The reduced search time demonstrates the effectiveness of Transformer-based architecture prediction, where learned representations guide the search process toward promising hardware-efficient designs. This capability is particularly important for wearable edge AI systems where rapid deployment and continuous model adaptation are required.

4.11 DISCUSSION OF RESULTS

The comprehensive evaluation demonstrates that TANAS-HSC provides significant improvements across all critical performance indicators required for wearable FPGA edge AI deployment. The results confirm that the proposed framework achieves better classification accuracy, lower latency, reduced energy consumption, improved resource utilization, and faster architecture exploration compared with DARTS, ENAS, and HA-NAS.

The accuracy results in Table.14 show that TANAS-HSC achieves 98.74% accuracy after 30 search iterations, outperforming HA-NAS by 3.50%, ENAS by 4.69%, and DARTS by 5.62%. This improvement indicates that Transformer-based architecture representation effectively captures complex dependencies among neural operations and identifies highly optimized network structures. According to Table.15, TANAS-HSC reduces inference latency to 18.6 ms, achieving approximately 36.30% lower latency compared with HA-NAS. This reduction enables real-time processing capability for wearable applications requiring immediate decision responses. The energy evaluation in Table.16 demonstrates that TANAS-HSC consumes only 1.94 J, which is significantly lower than all comparison approaches. The improvement results from optimized hardware-software mapping and efficient resource allocation. Furthermore, Table.17 highlights that TANAS-HSC achieves 93.82% FPGA resource utilization efficiency. Unlike conventional methods that optimize architecture and hardware independently, the proposed approach considers FPGA constraints during architecture generation. Finally, Table.18 confirms that TANAS-HSC reduces architecture search time to 5.8 hours. This reduction accelerates deployment cycles and supports rapid development of wearable AI applications.

5. CONCLUSION

This research introduces the Transformer-Assisted Neural Architecture Search for Hardware–Software Co-design (TANAS-HSC) framework for automated optimization of FPGA-based wearable edge AI systems. The proposed approach integrates Transformer-based global dependency learning with multi-objective Neural Architecture Search and adaptive FPGA resource optimization. Experimental results demonstrate that TANAS-HSC achieves 98.74% classification accuracy, 18.6 ms inference latency, 1.94 J energy consumption, 93.82% FPGA resource utilization efficiency, and 53.97% reduction in architecture search time compared with conventional NAS approaches.

The obtained results confirm that joint optimization of neural architecture and hardware implementation provides significant advantages over independent software-level or hardware-level

optimization strategies. The Transformer encoder improves architecture exploration by identifying complex relationships between neural operations, while the hardware-software co-design module ensures efficient FPGA deployment under resource constraints. The framework successfully addresses major challenges associated with wearable edge AI systems, including limited computational resources, energy restrictions, and real-time processing requirements.

Therefore, TANAS-HSC provides an effective automated design methodology for next-generation wearable FPGA intelligence platforms requiring high accuracy, low latency, and energy-efficient operation.

REFERENCES

[1] D. Ngo, H.C. Park and B. Kang, "Edge Intelligence: A Review of Deep Neural Network Inference in Resource-Limited Environments", *Electronics*, Vol. 14, No. 12, pp. 2495-2514, 2025.

[2] N.B. Gaikwad, H. Mir, S. Kosta and U.R. Acharya, "FPGA SoC Implementation of Adaptive Deep Neural Network based Multimodal Edge Intelligence for Internet of Medical Things", *IEEE Access*, Vol. 18, pp. 1-27, 2025.

[3] F. Porreca, F. Frustaci and R. Gravina, "A Codesign Framework for the Development of Next Generation Wearable Computing Systems", *Sensors*, Vol. 25, No. 21, pp. 6624-6636, 2025.

[4] H.Q. Tran, "FPGAs: Architecture, Design Flow, and Emerging Applications-Industrial Perspectives", *Analog Integrated Circuits and Signal Processing*, Vol. 127, No. 1, pp. 1-20, 2026.

[5] E. Covi, E. Donati, X. Liang, D. Kappel, H. Heidari, M. Payvand and W. Wang, "Adaptive Extreme Edge Computing for Wearable Devices", *Frontiers in Neuroscience*, Vol. 15, pp. 611300-611315, 2021.

[6] N. Akhtar, A.R. Buzdar and M.U. Khan, "Cardio-Edge: Hardware-Software Co-design Implementation of LSTM Based ECG Classification for Continuous Cardiac Monitoring on Wearable Devices", *International Journal of Advanced Computer Science and Applications*, Vol. 16, No. 7, pp. 1-23, 2025.

[7] F. Al-Jame and L.W. Chenga, "Spiking Neural FPGA Accelerator for Edge-AI in Wearable Devices", *Electronics, Communications, and Computing Summit*, Vol. 3, No. 1, pp. 88-95, 2025.

[8] M.I. Khan and B. Da Silva, "Harnessing FPGA Technology for Energy-Efficient Wearable Medical Devices", *Electronics*, Vol. 13, No. 20, pp. 4094-4112, 2024.

[9] N.K. Jayakodi, J.R. Doppa and P.P. Pande, "A General Hardware and Software Co-Design Framework for Energy-Efficient Edge AI", *Proceedings of IEEE/ACM International Conference on Computer Aided Design*, pp. 1-7, 2021.

[10] C, Pham-Quoc, X.Q. Nguyen and T.N. Thinh, "Towards an FPGA-Targeted Hardware/Software Co-Design Framework for CNN-based Edge Computing", *Mobile Networks and Applications*, Vol. 27, No. 5, pp. 2024-2035, 2022.

[11] M. Vaithianathan, M. Patil, S.F. Ng and S. Udkar, "Energy-Efficient FPGA Design for Wearable and Implantable Devices", *ESP International Journal of Advancements in Science and Technology*, Vol. 2, No. 2, pp. 37-51, 2024.

[12] D. Ngo, H.C. Park and B. Kang, "Edge Intelligence: A Review of Deep Neural Network Inference in Resource-Limited Environments", *Electronics*, Vol. 14, No. 12, pp. 2495-2508, 2025.

[13] A. Rodriguez, A. Otero, M. Platzner and E. De La Torre, "Exploiting Hardware-based Data-Parallel and Multithreading Models for Smart Edge Computing in Reconfigurable FPGAs", *IEEE Transactions on Computers*, Vol. 71, No. 11, pp. 2903-2914, 2021.

[14] O. Bringmann, W. Ecker and M. Shafique, "Automated HW/SW Co-Design for Edge AI: State, Challenges and Steps Ahead", *Proceedings of International Conference on Hardware/Software Codesign and System Synthesis*, pp. 11-20, 2021.

[15] A.T. Pham and A.D. Bui, "Optimizing ECG-Based Sleep Apnea Detection on RISC-V Edge Devices via Hardware-Software Co-Design", *Proceedings of International Conference on Computational Intelligence in Engineering Science*, pp. 532-545, 2026.

[16] L.Y. Huang, Y.K. Hsieh and S.Y. Chien, "Real-Time FPGA-Based Hardware-Algorithm Co-Design for Monocular Visual Odometry on Edge Devices", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 34, pp. 1-16, 2026.

[17] S. Sadeghia, "A Comprehensive Review of Analog and Digital Filter Design: FPGA-Based Implementations, Real-Time Challenges, and Emerging Applications", *International Journal of Engineering and Technology Sciences*, Vol. 43, No. 2, pp. 22-35, 2025.

[18] N.B. Gaikwad, V. Tiwari, A. Keskar and N.C. Shivaprakash, "Heterogeneous Sensor Data Analysis using Efficient Adaptive Artificial Neural Network on FPGA based Edge Gateway", *KSII Transactions on Internet and Information Systems*, Vol. 13, No. 10, pp. 4865-4885, 2019.

[19] J.V. Suman, A.S. Priya, S. Kotte and I. Kholmurodov, "Ultra-Low Power VLSI Architecture for Edge AI-based Biomedical Signal Processing", *Proceedings of International Conference on Trends in Electronics and Informatics*, pp. 694-699, 2026.