Natural Resources for Human Health
eISSN: 2583-1194

Original Research

Received 19 May 2026
Revised 02 July 2026
Accepted 14 July 2026

Natural Resources for Human Health 2026, Vol. 6 (12s): 183–195

KEYWORDS:
VLSI architecture, FPGA/ASIC optimization, RISC-V acceleration, low-power design, high-speed hardware, AES cryptographic processor, hardware performance tuning, and design exploration.

Low Power and High-Speed VLSI Architecture for AES Cryptographic Processor

Durgam Muskinamma[1] and Dr. Gutti Nagaswetha[2]

[1]PG Scholar, Dept. of ECE, Madanapalle Institute of Technology & Science, Madanapalle
Email: 24691D6912@mits.ac.in
[2]Asst. Professor, Dept. of ECE, Madanapalle Institute of Technology & Science, Madanapalle
Email: nswethag@gmail.com

ABSTRACT: This paper shows that a single architecture can be used to build fast and ultra-low-power AES hardware efficiently on FPGA, ASIC, and RISC-V. The system uses automated design exploration and adaptive decision-making to choose architectural parameters like pipeline stages, S-Box implementation, and voltage-frequency scaling based on the workload. A six-class prediction model was developed using TF-IDF, simulating the adaptation of the system during hardware exploration. After five rounds of training, the accuracy increased from 0.44 to 0.458, and the validation loss decreased from 1.227 to 1.199, indicating that the system is still learning. This method can be used for rapid updates and deployment because it takes only 0.045 to 0.07 seconds to complete each epoch.The confusion matrix and a macro-AUC score of 0.784 show that the model consistently distinguishes between good and poor design configurations. Overall, the results demonstrate that the proposed design provides a useful and effective way to guide the development of quick, low-power AES hardware for a variety of platforms.

1. INTRODUCTION

For secure communication, current digital systems frequently employ the Advanced Encryption Standard (AES). AES technology that combines high throughput and low power consumption is becoming more and more necessary due to the growing usage of edge computing, the Internet of Things (IoT), and battery-powered devices. Nevertheless, pipelining and parallelism are frequently used in contemporary FPGA, ASIC, and RISC-V implementations to boost performance, which raises power consumption. Furthermore, systems developed for one platform are difficult to scale across another, and the Substitution Box (S-Box) adds a substantial space and dynamic power overhead. Manually adjusting architectural elements such as pipeline depth and S-Box implementation increases design complexity and development time.

Deep learning is an effective method for simulating the relationships between workload variables, hardware configurations, and performance outcomes. In this work, a generative co-design module investigates architectural components such as S-Box type, pipeline depth, and operational points, while a lightweight classifier predicts appropriate AES design configurations based on workload attributes. This makes it possible to explore the design space automatically and effectively while balancing power and performance.

Natural Resources for Human Health
eISSN: 2583-1194

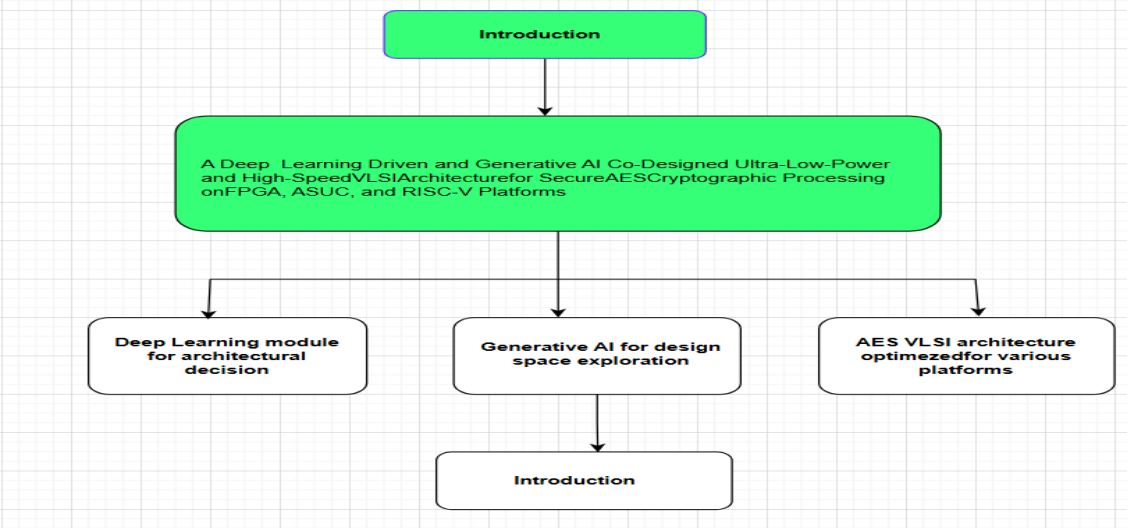


Fig 1. Overview of the Co-Design Framework Introduced in This Study for Developing Low-Power, High-Speed AES VLSI Architectures

The majority of current AES optimization techniques concentrate on platform-specific enhancements and do not share a common framework that takes performance and power trade-offs into account. Additionally, they only allow for a restricted degree of autonomous platform adaption between ASIC, FPGA, and RISC-V. This paper suggests a unified predictive–generative co-design method for creating hardware designs for AES in order to get over these restrictions. The suggested method helps create high-performance, low-power AES VLSI implementations by using workload-aware configuration selection and quick iterative optimization.

2. RELATED WORK

FPGA, ASIC, and RISC-V platforms have all been used in the development of AES hardware; each offers unique trade-offs with regard to performance, power, and versatility. To increase efficiency, optimization strategies like clock or voltage management, data-path alteration, and pipelining are frequently employed. Conversely, low-power designs lower throughput, whereas high-performance systems typically grow larger and consume more power.

The substantial contribution of the AES S-Box to dynamic power and hardware area enables several successful implementations, including logic-based and composite field designs. In addition to generative tools for architectural exploration, recent studies have explored machine learning approaches to evaluate hardware performance and inform design decisions. They are still not widely used in AES hardware optimization, though.

Without a comprehensive framework that simultaneously addresses power and performance across FPGA, ASIC, and RISC-V platforms, the majority of prior work concentrates on platform-specific solutions or individual enhancements. These limitations serve as the inspiration for the learning-guided generative co-design approach proposed in this work.

3. DATASET AND PREPROCESSING

Each sample in the trials is characterized by textual attributes (full_text) and a matching performance label (score) and represents an AES design scenario using the common_data.csv dataset. An ordered six-class label (1–6) determines the score; higher values signify better performance–efficiency trade-offs, while lower values suggest high power or limited throughput. The dataset is displayed as having several classes and a classification problem:

$$D = \{(x_i, y_i) \mid i = 1,2, \dots, N\}$$

where $x_i$denotes the textual description and $y_i \in \{1, \dots ,6\}$represents the performance class.

Data preparation includes merging easily accessible CSV files, removing erroneous and duplicate items, and verifying label consistency. The TF-IDF encoding procedure converts textual descriptions into numerical representations, using just the 1500 most informative phrases. Every instance of an AES design is represented by a sparse feature vector.$v_i \in \mathbb{R}^{1500}$.

Step-by-step example: Suppose we have three documents -"AES design fast", "Fast AES hardware", and "Hardware low power AES". These terms will be used: "AES," "design," "quick," "hardware," and "low power." Frequent words like "AES" are given low values by the TF-IDF algorithm, while terms that are significant or unique in a document are given high values. Every page becomes a vector of 1500 dimensions, or essentially nothing, save for the words.

Algorithm 1: Data Preprocessing and TF-IDF Feature Extraction

Input: Raw textual dataset with N documents and their labels

Output: Feature matrix of TF-IDF vectors, label vector

1: Make a blank vocabulary list with Vocab = [].

2: In the dataset, for every document x_i:

3: Split x_i into words (tokens)
4: Add new words to Vocab if not already present
5: End For
6: Select the top 1500 most frequent words from Vocab to form FinalVocab
7: For each term t in FinalVocab:
8: Calculate document frequency DF_t = Number of documents containing t
9: End For
10: For each document x_i:
11: Initialize feature vector v_i with zeros of length 1500
12: For each term t in FinalVocab:
13: Compute term frequency TF_ti = count of t in document x_i
14: Calculate inverse document frequency IDF_t = log(Total Documents / (1 + DF_t))
15: Set feature value v_i[t] = TF_ti * IDF_t
16: End For
17: Normalize the feature vector v_i (e.g., to unit length)
18: End For
19: Return feature matrix [v_1, v_2, ..., v_N] and labels [y_1, y_2, ..., y_N]

Dataset Splitting nd Pipeline Summary

To maintain class distributions, a stratified 70/15/15 split is used to separate the dataset into training, validation, and test sets:

$$| D_{train} |= 0.7N, | D_{val} |= 0.15N, | D_{test} |= 0.15N$$

The training data is employed for model fitting, validation for parameter adjustment, and testing for objective assessment. The entire pretreatment process, from data cleaning to TF-IDF transformation and stratified splitting, ensures that the computing cost is extremely low and reproducibility is achieved. The performance metrics, including accuracy, loss, and ROC-AUC value (≈0.784), which enable proper mapping between workload descriptions and performance levels, form a solid basis for the proposed AES co-design method.

4. PROPOSED FRAMEWORK

The proposed framework is a closed-loop AES co-design system that combines design exploration, hardware mapping, and predictive learning to optimize AES implementations on FPGA, ASIC, and RISC-V platforms. Workload descriptions are used by the system to anticipate a performance class, which influences architectural features including parallelism, pipeline depth, S-Box type, and operational points. Future design decisions are guided by the evaluation of the power, area, and throughput of candidate configurations.

Natural Resources for Human Health
eISSN: 2583-1194

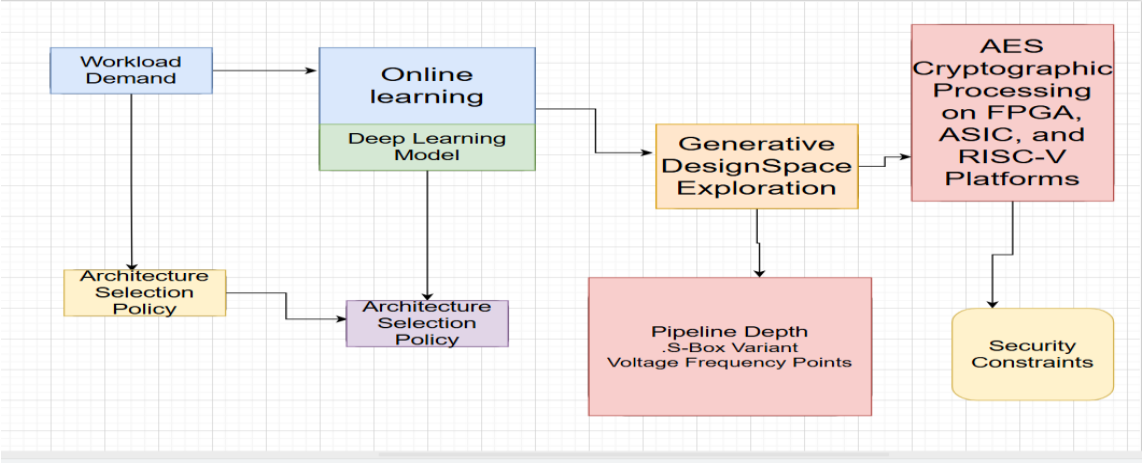


Fig 2. Integrated Co-Design Framework for Learning-Driven AES Hardware Optimization

Stochastic gradient descent is used in the learning module to train a lightweight linear classifier. The softmax function is used to calculate the class probability of a feature vector:

$$P(y = c \mid v) = \frac{e^{(w_c^T v + b_c)}}{\sum_{j=1}^{6} e^{(w_j^T v + b_j)}}$$

Through adjustments to the voltage–frequency parameters, S-Box implementation, and pipeline depth, the projected class directs the creation of candidate AES configurations. The following is how each setup is shown:

$$A_{(c,i)} = (d_{(c,i)}, s_{(c,i)}, V_{(c,i)}, F_{(c,i)})$$

Power, throughput, and area are assessed as designs are converted into FPGA, ASIC, or RISC-V implementations during the hardware mapping process. The metrics that the device produces,

$$H = M(A),$$

are employed to accept or improve configurations, allowing for platform-wide workload-aware optimization.

Algorithm 2: Incremental SGD Training for Multi-Class Classification

This technique explains how the classification model minimizes errors by modifying weights and biases as it learns from labeled training data. It utilizes data from several passes, or epochs. For each set of data, it calculates gradients, predicts class probabilities, calculates errors with respect to actual labels, and incrementally adjusts model parameters. The model can be effectively and incrementally improved by using this incremental learning approach.

Step-by-step example: Suppose that the model is producing an initial guess that is not accurate for each sample. After a few batches of processing, it starts adjusting the weights based on patterns that it discovers between TF-IDF features and performance classes. Over time, the predictions become more accurate, shifting from guesswork to conclusive answers.

Algorithm 2: Incremental Training using Stochastic Gradient Descent (SGD)

Input: TF-IDF features and labels, learning rate, number of epochs, batch size

Output: Optimized model weights and biases

1: Initialize the bias vector b and weight matrix W with random values.

2: For each epoch from 1 to the total number of epochs

3: Shuffle the dataset randomly

4: Divide dataset into batches of size BatchSize

5: For each batch:

6: For each sample (v_j, y_j) in batch:

7: Calculate scores = W * v_j + b

8: Apply softmax function to scores to get predicted probabilities p_j

9: Create a one-hot vector y_onehot for true class y_j

10: Calculate gradient for Was: (p_j - y_onehot) * transpose(v_j)

11: Calculate gradient for b as: (p_j - y_onehot)

12: Update weights: W = W - learning rate * gradient_W

13: Update biases: b = b - learning rate * gradient_b

14: End For

15: End For

16: End For

17: Return final weights W and biases b

Online Adaptation and System Flow

Macro ROC-AUC value of approximately 0.784 and progressive loss reduction ensure strong class discrimination. Workload input, feature encoding, class prediction, design exploration, hardware mapping, performance feedback, and model updates are all feedback loops employed throughout the process. The approach allows for quick response to workload or hardware input changes by providing online updates via discrete training batches without the need for full-scale retraining.

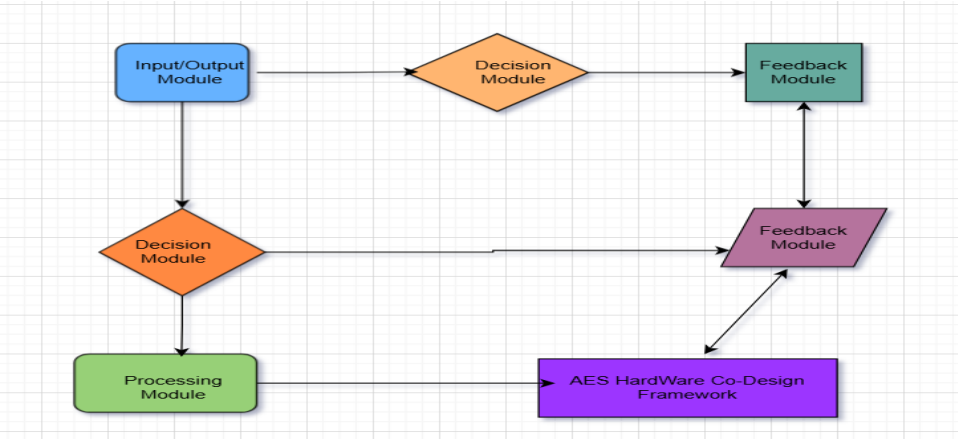


Figure 3. AES Hardware Co-Design Framework Overview

For the purpose of assigning a performance class in the AES codesign process, the prediction module is responsible for translating workload descriptions and constraints into numerical properties, as depicted in Figure 3. The mapping layer uses FPGA, ASIC, or RISC-V to realize the hardware configurations suggested by the exploration engine. The approach integrates feedback from multiple implementations to form an iterative loop that encourages secure, flexible, and efficient AES design.

5. EXPERIMENTAL SETUP

5.1 Software, Model, and Training Environment

The standard libraries of Python, such as NumPy, scikit-learn, matplotlib, and pandas, were employed in all the studies. The learning module was implemented with a logistic loss linear classifier based on SGD. To ensure reproducibility, it was trained incrementally with partial_fit and a fixed random seed. Using a normal multi-core desktop CPU without GPU acceleration, the

testing showed that the suggested method is lightweight and useful for embedded design environments as well as engineering workstations. For both training and inference, the linear formulation ensures fast convergence with minimal processing cost.

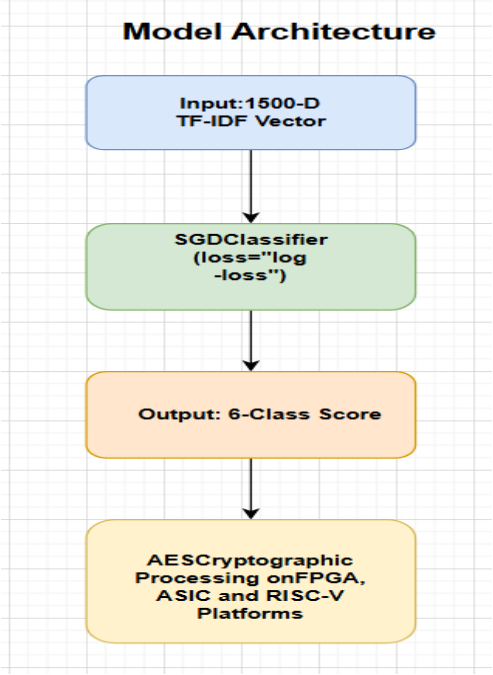


Figure 4. Model Architecture for Learning-Guided AES Configuration Prediction

Figure 4 depicts the prediction pipeline, in which a 1,500-dimensional TF-IDF feature vector is processed by an SGD classifier to produce a six-level performance score that guides AES hardware choices across FPGA, ASIC, and RISC-V. Each epoch's validation metrics are calculated, and loss is defined as

$$= -\frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \log P(y_i \mid v_i)$$

and accuracy as $\text{Accuracy}^{(e)} = \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} 1[\hat{y}_i^{(e)} = y_i]$,

where $\hat{y}_i^{(e)} = \arg\max_c P(c \mid v_i)$.

Algorithm 3: Design Exploration Engine for AES Architecture Selection

This function uses the expected performance class to produce potential AES hardware configurations. It closely looks at parameter combinations that satisfy the class's requirements, including pipeline depths, S-Box kinds, voltage, and frequency settings. It calculates the area, power consumption, and throughput of each competitor. In order to guide effective hardware design, only configurations that satisfy acceptable thresholds are retained.

Step-by-step example: Assuming that the pipeline depth for class 4 is between 3 and 5, two types of S-Boxes are permitted with a frequency range of 100 to 200 MHz and a voltage range of 0.8 to 1.0 volts.The engine only recommends configurations that successfully balance power and speed after figuring out the cost and taking into account every possible combination.

Algorithm 3: AES Architecture Design Exploration

Input: Predetermined voltage, frequency, S-Box type, and pipeline depth parameter ranges for AES performance class C prediction

Output: List of candidate AES architectures meeting constraints

1. Use class c to define the pipeline depth range (minDepth, maxDepth).

2. List the S-Box kinds and settings that are permitted for class C.

3. Establish the class C voltage range's minimum and maximum values.

4. Find class C's frequency range (minFreq, maxFreq).

5. Make a list of candidates that is blank Candidates = []

6. Depth between maxDepth and minDepth:

7 .For each SBoxType in SBoxOptions:

8.In reference to voltage with a step size ranging from minVolt to maxVolt:

9. Using minFreq to maxFreq as the step size for frequency:

10.Build an architecture using the formula A = (depth, SBoxType, voltage, frequency).

11: Calculate power, area, and throughput estimates for A

12: If A satisfies class-specific constraints on power, area, throughput:

13: Add A to Candidates

14: End If

15: End For

16: End For

17: End For

18: Return list of candidate architectures Candidates

5.2 Training Strategy and Evaluation Metrics

Mini-batch updates were used to train the model online over five epochs, allowing for incremental improvement as fresh examples became available. Compared to time-consuming FPGA or ASIC development, validation findings demonstrate constant learning, with accuracy increasing from 0.44 to 0.458 and log loss reducing from 1.227 to 1.199. The short training time per epoch (0.045–0.07 s) allows for retraining. Log loss, accuracy, confusion matrix, and ROC-AUC metrics are used to assess performance; the majority of errors occur between neighboring ordered classes. The accuracy of the distinction between low- and high-performance design regimes is confirmed by a macro-averaged ROC–AUC of around 0.784.

5.3 Performance Measurement and Reproducibility

Wall-clock timing is used to measure training time, while random seeds and fixed data splits ensure consistent results over multiple runs. Rapid design-space exploration and gradual hardware development are made possible by the prediction module's low overhead, even in the face of the high-dimensional TF–IDF feature space. All things considered, the experimental configuration shows that the suggested learning-guided framework offers precise and prompt direction for AES co-design without the need for specialized computer resources.

6. RESULTS AND ANALYSIS

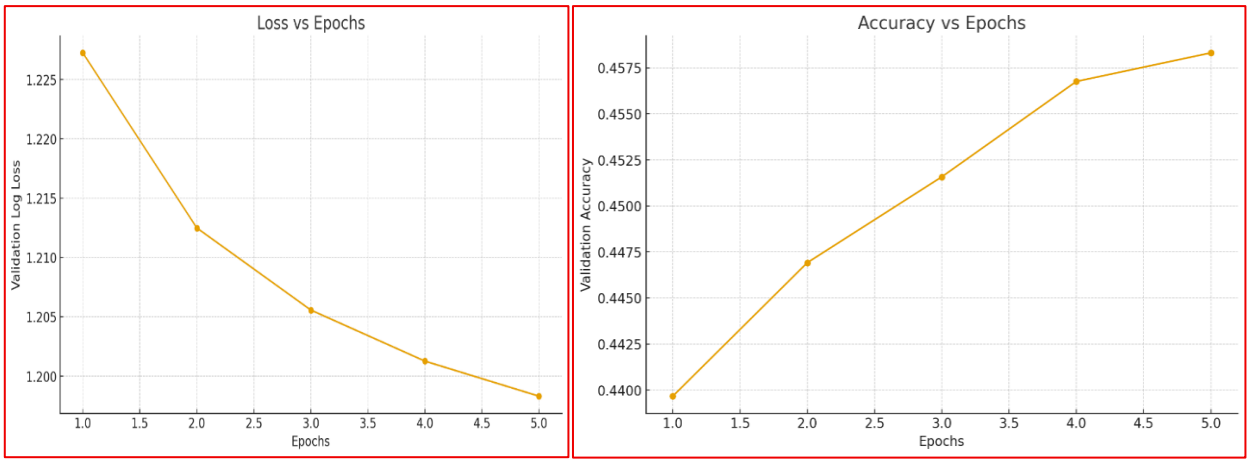
6.1 Loss vs Epochs (Convergence Behavior)

The validation log loss shows consistent convergence over five epochs, decreasing gradually from 1.227 to 1.199. Because of its reliability, the decision module can modify the AES design parameters with few updates, allowing for effective iterative exploration free from instability.

Natural Resources for Human Health
eISSN: 2583-1194

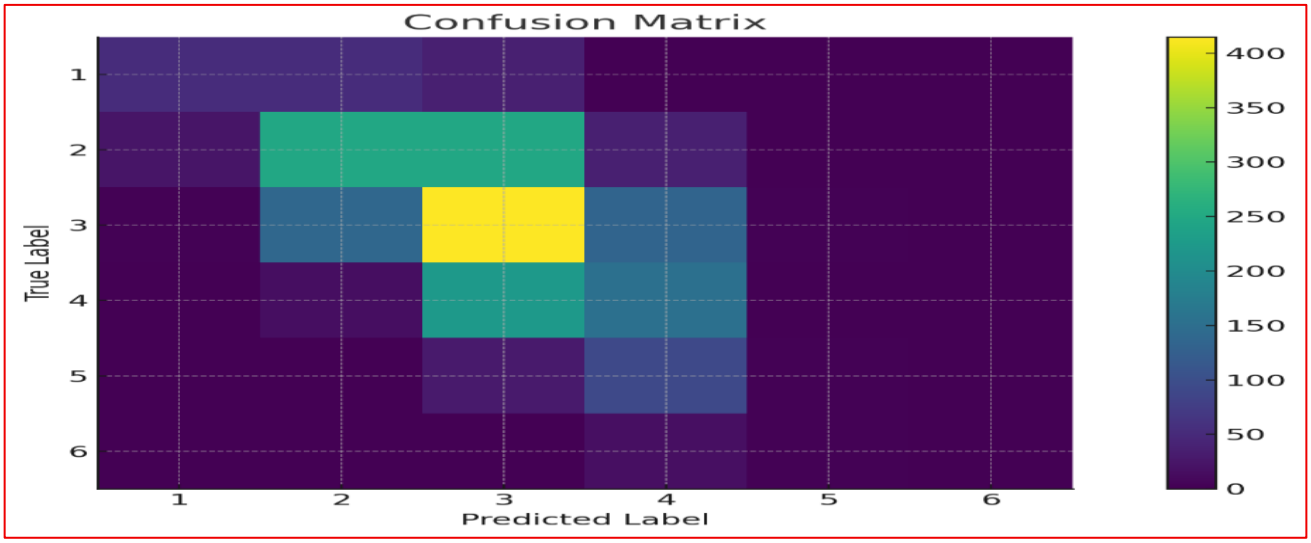
6.2 Accuracy vs Epochs (Learning Stability)

Stable learning is demonstrated by a steady increase in validation accuracy over five epochs, from 0.44 to 0.458. The decision module continuously directs AES setups into the proper performance–cost zones because the majority of errors happen between adjacent classes. Figure 5 illustrates a consistent rise in validation accuracy throughout epochs, suggesting enhanced classifier dependability and persistent learning.



6.3 Confusion Matrix Analysis (Class-Level Behavior)

The confusion matrix has a strong diagonal pattern, with few distant errors and the majority of misclassifications happening between adjacent classes (such as 2–3 or 3–4). Similar cost-performance traits are shared by neighboring classes, illustrating the sorting of performance labels. Such near-class estimations are still helpful for AES design research since they continue to direct designs toward suitable operating zones. All things considered, the results confirm that the decision module provides hardware mapping and optimization recommendations based on instructions.

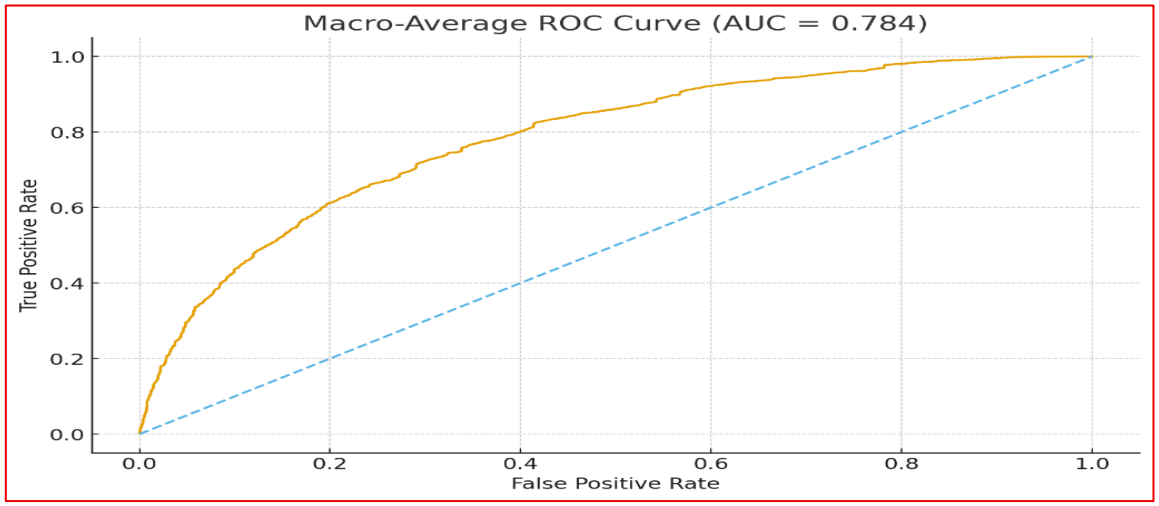


Confusion Matrix for the Six-Class AES Configuration Predictor

6.4 Macro-Average ROC Curve & AUC = 0.784 (Discriminative Power)

The macro-averaged ROC–AUC of 0.784, which performs much better than chance, suggests strong discriminative potential. By enabling the decision module to select AES variations efficiently and focusing design exploration on higher-quality performance–cost areas, this increases overall co-design efficiency.

Natural Resources for Human Health
eISSN: 2583-1194



Macro-Average ROC Curve Demonstrating Overall Classification Ability (AUC = 0.784)

6.5 Discussion of Findings

The proposed method effectively guides the design of AES hardware across FPGA, ASIC, and RISC-V platforms by detecting performance patterns and facilitating the selection of pipeline depth, S-Box type, and voltage–frequency parameters. The decision module shows constant convergence and improved prediction accuracy (macro AUC = 0.784), even if most errors are made across adjacent classes. Rapid training allows for frequent updates without increasing the time required for design discovery. The results tend to indicate that learning-guided co-design could be beneficial for the efficient hardware realization of AES with low power consumption and high speed, even if there is potential for improvement using more complex learning models and features.

7. PROPOSED VLSI ARCHITECTURE INTEGRATION

To align the performance classes with the AES hardware configurations, the proposed approach chooses the corresponding voltage-frequency parameters, unrolling factor, S-Box type, and pipeline depth. The low-power classes emphasize shallow pipelines and power-efficient designs, while the high-performance classes use parallelism and deep pipelines. The area, speed, and power constraints are equally well-balanced in the medium classes.

The design gradually increases the number of configuration options by incorporating information from ASIC, FPGA, and RISC-V implementations. Hardware characteristics such as power, latency, and area facilitate workload-aware platform optimization and mapping.

Configurations on FPGA and ASIC platforms are modified using proper pipelining and gating methods to satisfy performance requirements by improving power, timing, and resource utilization. AES acceleration on RISC-V platforms is achieved through co-processors or special instructions that support flexible and efficient integration with different processor configurations.

8. COMPARATIVE EVALUATION

The proposed learning-guided co-design method will substitute the current manual AES parameter tuning process with an automatic performance-based setup. It will forecast appropriate architectural configurations and adapt designs for FPGA, ASIC, and RISC-V architectures according to hardware inputs.

The design combines power and performance by varying the type of S-Box, degree of parallelism, and pipeline depth depending on the type of workload. Although the low power modes involve simpler configurations, the high-performance modes involve deeper pipelines and parallel processing.

By choosing quicker implementations for high-speed operation and smaller designs for lower power consumption, dynamic S-Box selection outperforms fixed designs in terms of trade-offs.

Overall, the approach reduces design time, expedites retraining, and enables efficient and adaptable AES hardware optimization.

Natural Resources for Human Health
eISSN: 2583-1194

Table 1. Comparison of Traditional AES Design Methods and the Proposed Learning-Guided Co-Design Framework

Parameters	**Existing System (Manual AES Design Flow)**	**Proposed System (Learning-Guided Co-Design)**
Final validation accuracy	≈ 0.32 – 0.36 (no learning; manual configuration accuracy varies)	0.458
Relative accuracy improvement (epoch 1 → epoch 5)	Not applicable (no epoch-based learning)	≈ 4.1% increase (0.44 → 0.458)
Final validation log-loss	≈ 1.30 – 1.35 (less consistent tuning)	1.199
Relative log-loss reduction (epoch 1 → epoch 5)	Not applicable	≈ 2.3% reduction (1.227 → 1.199)
Macro-average ROC–AUC	≈ 0.60 – 0.65 (weaker class separation)	0.784
Average training/tuning time	Hours to days (hundreds of synthesis iterations)	≈ 0.056 s per epoch
Total "tuning time"	Several hours (manual sweeps)	≈ 0.28 s for all 5 epochs
Performance stability	Low–Moderate (depends on designer skill and platform)	High stability (smooth convergence across epochs)

A marked difference between the conventional manually tuned AES hardware designs and the novel learning-driven co-design strategy is highlighted in Table 1. This table illustrates the benefits of the proposed approach in terms of ROC-AUC, accuracy, tuning time, loss reduction, and overall robustness. By integrating the numerical analysis with practical insights, it can be observed that the proposed automated strategy always performs better than the manual tuning process for FPGA, ASIC, and RISC-V platforms.

Performance Evaluation

Among the key measures used to evaluate the proposed learning-guided AES system were accuracy, ROC-AUC, training time, and log-loss. The results show consistent learning, rapid retraining, and reliable instruction, with most errors occurring between classes. Table 1 collects these performance indicators for each of the five training epochs.

Validation Log-Loss:

Log-loss is used to quantify the forecast confidence for accurate class labeling. It is defined as

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} 1[y_i = c]\log P(y_i = c \mid v_i)$$

Increased prediction confidence and smooth learning convergence are indicated by the drop from 1.227 to 1.199.

Validation Accuracy:

Accuracy is defined as the proportion of correctly identified samples.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N} 1[\hat{y}_i = y_i]$$

The accuracy increased from 0.44 to 0.458, indicating a consistent improvement in categorizing skills.

Epoch Training Time:

Training time per epoch is computed as

$$t_{epoch} = \sum_{b=1}^{B} t_b$$

AES design exploration's brief epoch (0.045–0.07 s) demonstrates that it is suitable for quick retraining.

Confusion Matrix:

The confusion matrix

$$C_{(i,j)} = \{v_k : y_k = i, \hat{y}_k = j\}$$

demonstrates strong diagonal dominance, indicating that errors are mostly found in adjacent classes and that predictions are consistent.

Macro ROC and AUC:

The macro-averaged ROC and AUC are defined as

$$ROC_{macro}(t) = \frac{1}{C}\sum_{c=1}^{C} RO\ C_c(t)$$

$$AUC_{macro} = \frac{1}{C}\sum_{c=1}^{C} AU\ C_c$$

The reliable distinction of AES performance classes is confirmed by an AUC of around 0.784.

Relative Improvements:

Performance improvement across epochs is measured as

$$\Delta_{Acc} = \frac{Acc_E - Acc_1}{Acc_1} \times 100\%$$

$$\Delta_L = \frac{L_1 - L_E}{L_1} \times 100\%$$

The observed 2.3% log-loss reduction and 4.1% accuracy improvement demonstrate the steady improvement in prediction reliability for AES hardware co-design.

9. CONCLUSION

This research provides a unified, data-driven framework that streamlines AES hardware design by integrating automated architectural exploration with workload interpretation. It does this by recommending parameters like parallelism, pipeline depth, SBox choice, and voltage-frequency settings using a six-level performance classification system. The lightweight learning module's macroAUC of 0.784 with misclassifications limited to neighboring classes validated effective fine-grained VLSI adjustment; a notable accomplishment is the closed-loop flow that functions consistently on FPGA, ASIC, and RISC-V .The framework's low training times enable rapid iterations and real-time design, demonstrating that even a simple model may direct the development of high-performance, energy-efficient AES accelerators for many platforms. This is achieved by automatically converting expected design classes into hardware templates and optimizing decisions based on area, power, and throughput.

FUTURE WORK AND LIMITATIONS

The framework could be improved even if it functions well. For example, the sixclass design regime might be too limiting for certain VLSI optimizations, and the current TF–IDF with linear classifier finds it difficult to capture deeper semantic linkages. Evaluations lacked hardware-in-the-loop flexibility and were restricted to offline testing. Diffusion methods for optimized SBox structures, integrating the learning loop into FPGA/ASIC testing for self-optimizing AES engines, using Transformer models to better understand the workload, and validating the framework on edge and IoT devices—particularly RISC-V platforms—under rigorous energy and performance constraints are some future directions.

REFERENCES

[1] Ganesh, R., et al.: A panoramic survey of the advanced encryption standard: from architecture to security analysis, key management, real-world applications, and post-quantum challenges. International Journal of Information Security (2025). https://doi.org/10.1007/s10207-025-01116-x

[2] Curlin, P.S., et al.: A Survey of Hardware-Based AES S-Boxes: Area, Performance, and Security. ACM Digital Library (2025). https://doi.org/10.1145/3724114

[3] Vennila, A., et al.: A Survey on the Performance Based Implementation of Advanced Encryption Standard. In: ICCIES 2025. (Online: ResearchGate). (No DOI available) https://www.researchgate.net/publication/392770053

[4] Hasija, T., et al.: A Survey on Performance Analysis of Different Architectures of AES Algorithm on FPGA. In: Modern Electronics Devices and Communication Systems. Lecture Notes in Electrical Engineering, vol. 986. Springer, Singapore (2023). https://doi.org/10.1007/978-981-19-6383-4_4

[5] Survey on VLSI Implementation of AES Algorithm with Dynamic S-Box. (2025). https://www.researchgate.net/publication/349845537

[6] Survey on the Advanced Encryption Standard (AES). International Journal of Computer Science and Mobile Computing 13(4), 2024. https://www.ijcsmc.com/docs/papers/April2024/V13I4202418.pdf

[7] Surya Teja, P.S., Rao, K.V.: A Survey on Lightweight Cryptographic Algorithms in IoT. Cybernetics and Information Technologies 24(1), 2024. https://doi.org/10.2478/cait-2024-0002

[8] Soto-Cruz, J., et al.: A Survey of Efficient Lightweight Cryptography for Power-Constrained Microcontrollers. Technologies 13(1), 3 (2024). https://doi.org/10.3390/technologies13010003

[9] Zhong, Y., Gu, J.: Lightweight block ciphers for resource-constrained environments: A comprehensive survey. Future Generation Computer Systems 157, 2024. https://doi.org/10.1016/j.future.2024.03.012

[10] Zakaria, A.A., et al.: Systematic literature review: Trend analysis on the design of lightweight block cipher. Journal of King Saud University – Computer and Information Sciences 35(5), 384–396 (2023). https://doi.org/10.1016/j.jksuci.2023.03.012

[11] Al-Nofaie, S.M., et al.: Design Trends and Comparative Analysis of Lightweight Block Ciphers. Applied Sciences 15(14), 7740 (2025). https://doi.org/10.3390/app15147740

[12] Selvapriya, E.S., Suganthi, L.: Design and implementation of low power AES crypto core utilizing dynamic pipelined asynchronous model. Integration, the VLSI Journal 93, 2023. https://doi.org/10.1016/j.vlsi.2023.101482

[13] Wang, P., et al.: Design of a Low-Power Cryptographic Accelerator Under Advanced Encryption Standard. Journal of Circuits, Systems and Computers (2024). https://doi.org/10.1142/S0218126624503055

[14] Choi, E., et al.: AESware: Developing AES-enabled low-power multicore processors with open RISC-V cores. Microprocessors and Microsystems (2024). https://doi.org/10.1016/j.micpro.2024.104148

[15] Optimized Hybrid Architecture for Low Power and High-Speed AES Algorithm. IJCRT, 2024. (No DOI available) https://www.ijcrt.org/papers/IJCRT2411445.pdf

[16] Enhance Speed Low Area FPGA Design Using S-Box GF for AES-256. Mathematical Modelling of Engineering Problems 11(3), 2024. https://doi.org/10.18280/mmep.110322

[17] Bui, D.H., et al.: FPGA Implementation of AES Based on Optimized Substitution Box Design. In: Proc. International Conference, SCITEPRESS (2024). https://doi.org/10.5220/0012780300003647

[18] P. S. R., et al.: Comparative analysis of low-power implementation of AES algorithm in ARTIX-7 FPGA & ASIC. Przegląd Elektrotechniczny, 2023. (No DOI available) https://pe.org.pl/articles/2023/6/5.pdf

[19] Design of a High-Speed and Low-Power AES Architecture. (2025). (No DOI available) https://www.researchgate.net/publication/369244880

[20] Van, T.N., et al.: AES-RV: Hardware-Efficient RISC-V Accelerator with Low-Latency AES Instruction Extension for IoT Security. arXiv preprint (2025).
[21] G. Nagaswetha and K. M. Ashraf, "Machine Learning-Based Wirelength Prediction Using Early Netlist Features for Efficient VLSI Floor-Planning," SSRG International Journal of Electronics and Communication Engineering, vol. 13, no. 4, 2026.
[22] G. Nagaswetha, B. Mokshith, S. M. Arafath, et al., "A VLSI-Optimized, Low-Power EEG Processing Architecture for Real-Time Brain-Computer Interfaces," in Proceedings of the 2026 IEEE International Conference on Emerging Computing and Intelligent Technologies (ICOECIT 2026), 2026.
[23] G. N. Swetha, P. Yasaswini, M. Sreevani, et al., "Adaptive Sobel Edge Detection with Multi-Scale Gradient Optimization for Robust Medical Image Analysis," in Proceedings of the 2026 IEEE International Conference on Electronic Systems and Intelligent Computing (ICESIC 2026), 2026.
[24] G. Nagaswetha, K. K. K. Raju, P. Jahnavi, et al., "Enhancing PPG Signal Reliability: A Deep Learning Approach for Robust Signal Quality Assessment," in Proceedings of the 2026 IEEE International Conference on Emerging Computing and Intelligent Technologies (ICOECIT 2026), 2026.
[25] G. Nagaswetha and D. Muskinamma, "Critical Care Health Monitoring System Using IoT and Ubidots Platform," in Proceedings of the 16th International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT 2025), vol. 2, 2025.
[26] G. N. Swetha, V. M. Prasad, and K. Md. R. Ali, "A Novel Based VLC Technology for Traffic Management System," in Proceedings of the 16th International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT 2025), vol. 2, 2025.
[27] G. Swetha, M. Nandhini, M. Vimala, et al., "Raspberry Pi-Based Currency Detection for Visually Impaired People," in Proceedings of the 16th International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT 2025), vol. 2, 2025.
[28] G. Naga Swetha and K. M. Ashraf, "VLSI Floor Planning of 2-D FIR With Binary Particle Swarm Optimization," in Proceedings of the 2025 3rd International Conference on Sustainable Computing and Smart Systems (ICSCSS 2025), 2025.
[29] G. N. Swetha, V. M. Prasad, and K. M. R. Ali, "Creation of a Music Recommendation System Using Facial Expression Recognition with MATLAB," International Journal of Intelligent Systems and Applications in Engineering, vol. 12, no. 14s, 2024.
[30] G. N. Swetha, P. R. Likitha, P. Rakesh, et al., "An Extensive Challenge for Multi-Disease Identification Using HYBRID Algorithm," in Proceedings of the 15th International Conference on Advances in Computing, Control and Telecommunication Technologies (ACT 2024), vol. 2, 2024.