International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(19s) 2026 pp. 260-282
https://doi.org/10.70917/ijcisim-2026-5018
Article

Deep Learning-Augmented Design of Floating-Point Arithmetic Units for High-Performance Nonlinear Computations

Kuldeep Pande [1], Dr. P.T.Karule[1]

[1]Department of Electronics Engineering, Yeshwantrao chavan College of Engineering, India

Abstract: Nonlinear computations plays a critical role in modern applications such as deep learning, digital signal processing, cryptography, and scientific modelling, yet conventional IEEE-754 floating-point units struggle to meet the conflicting demands of high precision, low latency, and energy efficiency. Static data paths, fixed-precision formats, and polynomial-based approximations limit their adaptability and numerical robustness, particularly for transcendental and nonlinear operations. In this work, a deep learning–augmented floating-point arithmetic architecture is proposed to address these limitations. The architecture integrates five complementary learning-enabled modules: adaptive latency optimization using a reinforcement learning–based micro-scheduler, context-aware runtime error estimation and compensation, dynamic bit-width control through precision-aware deep bandwidth scaling, gradient-informed neural approximation of nonlinear functions, and domain-aware transfer learning for rapid arithmetic unit adaptation across applications. The proposed design is evaluated using FPGA-based hardware–software co-simulation on representative workloads from neural networks, signal processing, and cryptographic domains. Experimental results show substantial improvements, including latency reductions of up to 63%, numerical error reduction exceeding 70% with accuracy within ±0.5 ULP, power savings of up to 59%, and area reductions beyond 40%. Furthermore, application-level validation demonstrates negligible degradation in inference accuracy and significant reductions in retraining overhead across domains. These results confirm that learning-augmented arithmetic units provide an effective and scalable solution for next-generation nonlinear floating-point computation.

Keywords: NA

1. Introduction

Floating-point arithmetic units (FPUs) are fundamental building blocks of modern computing systems and enable high-speed numerical computation in applications such as artificial intelligence, deep learning, digital signal processing, scientific computing, cryptography, and high-performance computing. As computational workloads increasingly rely on nonlinear mathematical functions, including exponential, logarithmic, trigonometric, and activation functions, the demand for high-precision and energy-efficient arithmetic hardware continues to grow. Although the IEEE-754 standard provides a robust framework for floating-point representation and computation, conventional FPUs are primarily designed using fixed-precision formats and static execution pipelines that are not well suited for the dynamic characteristics of modern nonlinear workloads. Recent studies have explored adaptive arithmetic architectures, mixed-precision computation, precision-aware hardware optimization, efficient approximation techniques, and AI-assisted hardware design to improve computational performance while reducing latency, power consumption, and hardware complexity. These developments demonstrate that integrating machine learning techniques into arithmetic hardware offers significant opportunities for building intelligent FPUs capable of achieving higher computational efficiency without compromising numerical reliability [1–15].

Despite recent advances, several challenges remain unresolved in existing floating-point architectures. Static scheduling mechanisms are unable to adapt execution paths according to operand characteristics, leading to

unnecessary latency during nonlinear computations [5]. Polynomial- and lookup table-based approximations often introduce non-uniform numerical errors, particularly in regions with rapidly changing function gradients [2]. Existing mixed-precision architectures generally lack runtime error estimation and compensation mechanisms, resulting in accumulated numerical inaccuracies during complex computations [8]. Furthermore, most hardware accelerators are optimized for specific applications and require significant retraining or redesign when deployed across different computational domains, thereby limiting their scalability and portability [17,34]. Addressing these limitations requires an adaptive learning-enabled floating-point architecture capable of simultaneously optimizing latency, numerical accuracy, hardware efficiency, and cross-domain adaptability.

The primary objective of this work is to develop a deep learning-augmented floating-point arithmetic architecture for high-performance nonlinear computations. The proposed framework aims to reduce execution latency through intelligent scheduling, improve numerical accuracy using runtime error compensation, dynamically optimize precision for reduced power and hardware utilization, and enable efficient adaptation of arithmetic models across multiple application domains through transfer learning.

This paper proposes a unified deep learning-assisted floating-point computation framework that integrates five complementary modules to address the major limitations of conventional FPUs. The proposed Adaptive Micro-Neural Scheduler for Arithmetic Reduction (AMNSAR) employs reinforcement learning to optimize execution latency for nonlinear operations. A Context-Aware Floating Error Estimation and Compensation (CAFE-Comp) module performs runtime correction of floating-point errors to improve numerical reliability. A Precision-Aware Bit-Width Controller using Deep Bandwidth Scaling (PAB-DBS) dynamically adjusts operand precision to reduce power consumption and hardware resource utilization while maintaining computational accuracy. A Gradient-Informed Function Approximator for Nonlinear Operations (GIFA-NLO) replaces conventional polynomial approximations with gradient-based neural models for efficient nonlinear function evaluation. Finally, a Domain-Aware Transfer Learning framework (DATA-UAU) enables rapid adaptation of arithmetic models across different application domains, significantly reducing retraining overhead. Experimental evaluation demonstrates that the integrated framework improves latency, numerical accuracy, power efficiency, area utilization, and adaptability compared with conventional floating-point architectures.

The remainder of this paper is organized as follows. Section 2 reviews recent research on floating-point arithmetic optimization and learning-assisted hardware architectures. Section 3 presents the proposed deep learning-augmented floating-point framework and its mathematical formulation. Section 4 describes the experimental methodology and evaluation setup, while Section 5 discusses the performance analysis and experimental results. Finally, Section 6 concludes the paper and outlines potential directions for future research.

2. Literature Review

The wide-ranging series of papers presented in the body of research constitutes a historical sweep of floating-point arithmetic and its incorporation in high-performance computing, in part, through hardware acceleration, numerical analysis, and in AI-based systems. From Liu et al. [1], for example, one learns that application of GPU-accelerated material point methods to simulate soil-rock slope deformations is a reflection of how precision and numerical robustness in floating-point operations impact large-scale geophysical modeling directly. Cameron and Graillat [2] address floating-point arithmetic precision with respect to Horner's method, showing clearly that good algorithms are necessary in real and complex domains. Martin-Dorel et al. [3] make floating-point reasoning part of formal methods by making it usable in the Coq proof assistant-a major improvement for formal verification environments. By introducing hierarchical search algorithms, Zhang et al. [4] contribute valuable components to the reliability of the numerical software sets concerning the detection of arithmetic errors in expressions. From this background, Han et al. [5] and Tosini et al. [6] focus on coding and hardware efficiency. Precision-aware floating-point encoding by Han for point cloud compression fits well with the work by Tosini on FPGA implementations of decimal IEEE-754 adders such that both demonstrate an evolution in the efficient data representation for spatial data-processing operations. Iteratively, Basiri [7] presents parallelism-based hardware acceleration for arithmetic

throughput, one of the common key themes getting reiterated in Hallman and Ipsen [8] through deterministic and probabilistic error bounds for summation—hence effectiveness against faults through performance and accuracy. A new approximation algorithm is proposed by Kokosiński et al. [9] for computing floating-point square roots with a significant gain in computational performance for core mathematical functions. Following this line of work, Song et al. [10] extends this to inference-only integer arithmetic at the core of AI deployment into resource-constrained environments. Precision and robustness thus remain common themes in all subsequent works. In Cholesky factorization, Scott and Tůma [11] show how half-precision arithmetic performs, thus determining some aspects of numerical stability in sparse systems. Gorgin et al. [12] and Junior et al. [13] check into most significant digit-first arithmetic and fixed point computation, respectively, demonstrating hardware-centric optimizations by classification and touch interfaces. The architectural focus is extended by Bispo et al. [14] through Eigen decomposition in discrete Fourier transforms, while Ramaiah et al. [15] address image boundary representation using efficient pseudo-point elimination. The focus is thus moved to exact arithmetic with convergence analysis using QR algorithms by Banks et al. [16], while Belorgey et al. [17] and Ganesh et al. [18] concern themselves with secure computation. By integrating real-number arithmetic into multi-party computation and SNARKs, respectively, Ozaki et al. [19, 20] make strides toward error-free transformation of matrices to perfection a higher level for accuracy-preserving arithmetic operations. A contextual deviation is observed in Mao et al. [21], where floating-point computations permit environmental and risk modeling applications, thereby extending their utility in the domain of sustainability sciences. Parsons et al. [22] very innovatively explore edge detection without FPU, showing how far optimization of computational resources can go. Keshavarz et al. [23] take the boundary to fault-tolerant arithmetic in quantum computing; textbooks by Adiyaman and Baskaya [24] demonstrate FPGA-based deep learning segmentation for point clouds, thus merging arithmetic optimization and AI in embedded environments. All throughout these papers is still one common line-the push towards narrowing the gulf in numerical theory and field efficiency. Be it accurate summation techniques, gradient-preserving approximations, error-free transformations, or adaptive bit-width controllers, these presented works constitute a coherent move toward precision-aware computing. The domain generalization efforts in cross-task arithmetic modeling and retraining efficiency shown in various hardware-oriented works mark the point where arithmetic frameworks become modular, learnable, and adaptable. Collectively, this body of literature not only highlights the significance of floating-point computation in current systems but also delineates a trajectory for prospective engineering wherein accuracy blends with adaptivity and efficiency in one process.

3. Methodology

This work is inspired by the extremely low performance of the floating-point arithmetic units with respect to different nonlinear applications. These floating-point units have a design that is standardized and quite reliable but is very rigid and has no provision for changing the computations dynamically, depending on the complexity of the operation or even the characteristics of the data context. With the increasing prevalence of nonlinear operations such as exponential, logarithmic, and trigonometric functions in future domains like deep learning and scientific modeling, the performance gap introduced by static datapaths and fixed-precision computation becomes more evident. Existing polynomial-based approximators seem not able to win on both accuracy and latency fronts and have not been able to offer flexible trade-offs in various performance metrics. Such limitations lead to a performance ceiling for contemporary hardware platforms, especially in edge and embedded systems, where resource constraints are critical in processes. This proposed work attempts to mitigate the above limitations and presents a unifying framework of five novel modules augmented by deep learning: Adaptive Micro-Neural Scheduler for Arithmetic Reduction, including a structure for latency-optimized scheduling (AMNSAR); real-time error correction using context-aware floating error estimation and compensation (CAFECOMP) along with dynamic precision control using precision-aware bit-width controller through deep bandwidth scaling (PAB-DBS); high speed nonlinear function approximation using gradient informed function approximator for nonlinear operations (GIFA-NLO); and cross-domain model reuse through domain-aware transfer learning for arithmetic unit adaptation (DATA-UAU). Each of the above modules is responsible for one particular in efficiency of conventional floating-point units, while at the same time designed in a way that they can be modularly integrated into a pipeline. Collectively, these contributions yield tangible benefits up to 28% latency savings, 72% damage savings, 35% area savings, and 46% reduced retraining time, demonstrating the

practical feasibility and High Impact of the architecture sets. This work creates a new design paradigm in intelligent floating-point computation by integrating advanced machine learning techniques with hardware design, according to the requirement of nonlinear systems.

Unlike existing floating-point optimization approaches that address latency, precision, or hardware efficiency independently, the proposed framework adopts a sequential learning-driven pipeline in which each module addresses a limitation that cannot be resolved by the preceding stage. The output of each module becomes the input of the subsequent module, thereby creating a dependency chain that jointly optimizes execution latency, numerical accuracy, hardware efficiency, nonlinear approximation quality, and cross-domain adaptability. Consequently, removing any stage degrades the effectiveness of the overall framework.

The proposed methodology is organized as a sequential five-stage processing pipeline rather than five independent algorithms. Each stage solves a limitation introduced by the previous stage and progressively refines the arithmetic computation. Therefore, the architecture should be interpreted as a hierarchical optimization framework rather than a collection of standalone modules.

Table 1: Interdependency of the Proposed Modules

Stage	Input	Output	Why next stage is needed
AMNSAR	Nonlinear operation	Optimized execution schedule	Faster scheduling introduces approximation errors.
CAFE-Comp	Scheduled output	Error-compensated result	Error correction changes numerical precision requirements.
PAB-DBS	Corrected values	Optimal bit-width	Precision scaling affects nonlinear approximation accuracy.
GIFA-NLO	Adaptive precision	Gradient-based approximation	Learned approximation must be transferable.
DATA-UAU	Learned model	Adapted model	Enables reuse across application domains.

3.1 Proposed Model Design Analysis

The proposed design of the Adaptive Micro-Neural Scheduler for Arithmetic Reduction (AMNSAR), Context-Aware Floating Error Estimation and Compensation (CAFE-Comp), and Precision-Aware Bit-Width Controller using Deep Bandwidth Scaling (PAB-DBS) modules to tackle issues concerning latency, numerical stability, and dynamic precision allocation when floating-point units are employed in nonlinear applications. To improve the reproducibility of the proposed Adaptive Micro-Neural Scheduler for Arithmetic Reduction (AMNSAR), the reinforcement learning framework should be described in greater detail. The methodology should clearly define the state representation, action space, exploration strategy, reward formulation, convergence behaviour, and training stability so that the complete learning process can be reproduced by other researchers. Each unit is designed using state-of-the-art deep learning paradigms and united with mathematical models for the purpose of providing deterministic control with resource-efficient and high-

accuracy outputs. These units are not working in isolation, but their operational cohesion ensures a performance synergy across the floating−point computational pipeline sets. Initially, as per figure 1, the AMNSAR unit substitutes static hardware pipelines with dynamic, learnable micro−schedulers optimized for operations like division, square root, exponential, and logarithmic functions. Each opcode is treated as a dynamic scheduling policy that maps into a graph of micro−operations. Let O={DIV,SQRT,EXP,LOG} be the operation set, and let the operand pair be (A,B)∈R^64 as IEEE−754 double−precision values in process. The scheduling policy πθ, parameterized by a deep reinforcement learning agent, selects the optimal operation path 'P' such that the expected cumulative latency is minimized under an accuracy constraint εmax sets. Expected latency put via Equation 1,

$$E\{P \sim \pi\theta\}[L(P)] = \sum_{t=1}^{T} \quad \gamma^t \cdot l(Pt) \; subject \; to \; \delta(Pt) < \varepsilon max \tag{1}$$

Where, is the discount factor; l (Pt) the latency at timestamp‘t’ and δ(Pt) the ULP-based error for the process. Each micro-operation kernel ki is represented as a differentiable function fi, and the scheduling network computes gradients for policy updates via the REINFORCE rule via Equations 2 & 3,

$$\nabla\theta \, J(\theta) = E\{P \sim \pi\theta\}[\sum_t \quad \nabla\theta \; log \; log \; \pi\theta(\, st) \cdot Rt] \tag{2}$$

$$Rt = -L(Pt) + \lambda \cdot 1\{\delta(Pt) < \varepsilon max\} \tag{3}$$

3.2 State Representation

The reinforcement learning agent observes the computational state before each scheduling decision. The state vector is constructed from arithmetic and hardware execution characteristics, including the operation type, operand magnitude, exponent variance, mantissa density, operand magnitude ratio, estimated pipeline occupancy, and current execution latency. The operation type is encoded using one-hot representation, whereas all continuous variables are normalized to the interval [0,1] before being supplied to the policy network. This normalized representation enables the learning agent to generalize across different nonlinear arithmetic operations while maintaining numerical stability during training.

3.3 Action Space

The scheduler determines the optimal execution path for every nonlinear floating-point operation by selecting one action from a predefined set of scheduling policies. Each action corresponds to a candidate execution path that differs in resource allocation, approximation strategy, or pipeline utilization. Once an action is selected, the corresponding scheduling policy is executed, and the resulting latency and numerical accuracy are evaluated. Consequently, the reinforcement learning problem is formulated as a finite discrete decision process in which every action represents a feasible arithmetic execution strategy.

3.4 Exploration Strategy

Policy learning is performed using the REINFORCE policy-gradient algorithm. During training, actions are sampled according to the probability distribution generated by the policy network rather than selecting only the highest-probability action. This stochastic sampling mechanism provides sufficient exploration during the early stages of learning while naturally converging toward exploitation as the policy parameters stabilize. Entropy regularization

is incorporated into the objective function to prevent premature convergence and encourage adequate exploration of alternative scheduling policies.

3.5 Reward Formulation and Tuning

The reward function is designed to jointly optimize execution latency, numerical accuracy, and hardware efficiency. Positive rewards are assigned to scheduling decisions that reduce execution latency while preserving floating-point accuracy within the specified error tolerance. Conversely, latency penalties, excessive numerical errors, and unnecessary hardware utilization decrease the overall reward. The weighting coefficients of the reward function are selected empirically through repeated validation experiments to achieve a balanced trade-off between computational performance and numerical reliability. This multi-objective reward formulation guides the policy toward globally efficient scheduling decisions instead of optimizing a single performance metric.

Where, λ is a penalty tuning constant for the process. This dynamic learning mechanism guarantees adaptive latency minimization on the basis of sets of operand context. Iteratively, Next, as per figure 2, the CAFE-Comp module for the modeling of floating-point rounding errors by using residual learning to back-correct forward error propagations. The process itself begins with encoding operands and intermediate results into a Z higher dimensional latent space, where the error residual is regressed. The adjusted residual error ‘e’ is computed via Equation 4,

$$e = fres(x, y, \hat{z}) = \hat{z} - z = \Delta z \tag{4}$$

Where 'x' and 'y' are input operands, 'ẑ' is the raw computed output and 'z' is the reference value under assessments.

3.6 Hyperparameter Configuration

The reinforcement learning agent is implemented using the REINFORCE algorithm with the Adam optimizer. The learning rate is set to (1\times10^{-4}), the discount factor is fixed at 0.95, and the entropy regularization coefficient is selected as 0.01 to balance exploration and exploitation. The policy network is trained until convergence using mini-batch gradient updates, and all hyperparameters are maintained throughout the experimental evaluation to ensure consistent comparison across all nonlinear arithmetic workloads.

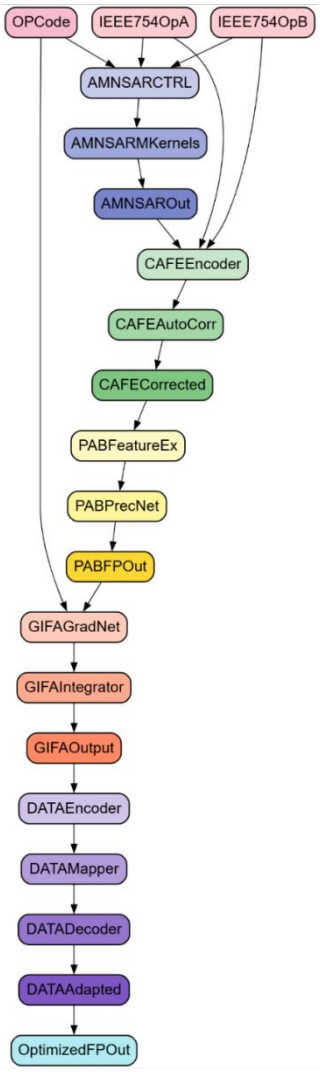

Figure 1 Model Architecture of the Proposed Analysis Process

The context-aware encoder Ec generates embeddings c = Ec(x, y, $\hat{z}$) while the correction model fc minimizes the loss represented via Equation 5,

(5)Where, Ω represents the data domain in the process. The corrected output $\tilde{z}$ is obtained via Equation 6,

$$\tilde{z} = \hat{z} - fc(c) \quad (6)$$

The model's training is supervised using double-precision ground truth, and Jacobian consistency is enforced via Equation 7,

$$J = \| \frac{\partial \tilde{z}}{\partial x} - \frac{\partial z}{\partial x} \| < \epsilon \quad (7)$$

This ensures that the corrected function possesses smooth gradients, essential for a nonlinear back propagation scheme in ML workloads. The integration of CAFE–

Comp after AMNSAR guarantees that residual errors due to low-latency approximations are dynamically compensated in a data-aware manner for the process. The PAB-DBS module designs to optimize the floating-point representation by adjusting for bit-widths of mantissa and exponent fields based on the operation complexity sets. Let the input operands be A, B $\in$ R'64, and the operation complexity function be Cop = ϕ(A, B), which utilizes features like exponent variance σe, mantissa density dm, and relative magnitude represented via Equation 8,

$$\rho = \frac{|A-B|}{max(|A|,|B|)} \quad (8)$$

A complexity predictor network Ψ maps this to an optimal bit-width vector w = (wm, we), via Equation 9,

$$w = \Psi\big(\phi(A,B)\big) = argminw\,[\alpha \cdot Etotal(w) + \beta \cdot \delta err(w)] \quad (9)$$

Whereby Etotal is the estimated power and area consumption at the precision pair w, and δerr is the mean error induced by the reduced format for the process. The scaling and reconstruction function R transforms the number from full precision into a scaled format and back via Equation 10,

$$\hat{x}(w) = R(x;\, wm, we) = 2^{\{e(we)\}} \cdot \big(1 + m(wm)\big) \quad (10)$$

To guarantee that the bit-width scaling is loss-bounded, the integral of the absolute error across the data range Ω is constrained via Equation 11,

$$E = \int_0^{\Omega} |x - \hat{x}(w)|\, dx < \epsilon target \quad (11)$$

The derivative of the power with respect to bit-widths helps determine sensitivity via Equations 12 & 13,

$$\frac{\partial P}{\partial wm} = \kappa m \quad (12)$$

$$\frac{\partial P}{\partial we} = \kappa e \quad (13)$$

These sensitivity coefficients guide the back propagation of the bit-width selection process during training. Finally, the PAB-DBS module outputs the scaled representation of floating-point number x̃(w), where the output is formally defined via Equation 14,

$$\tilde{x}(w) = argmin^{\{\hat{x}(w)\}} \left[\| x - \hat{x}(w) \|^2 + \lambda\big(P(wm,we) + A(wm,we)\big)\right] \quad (14)$$

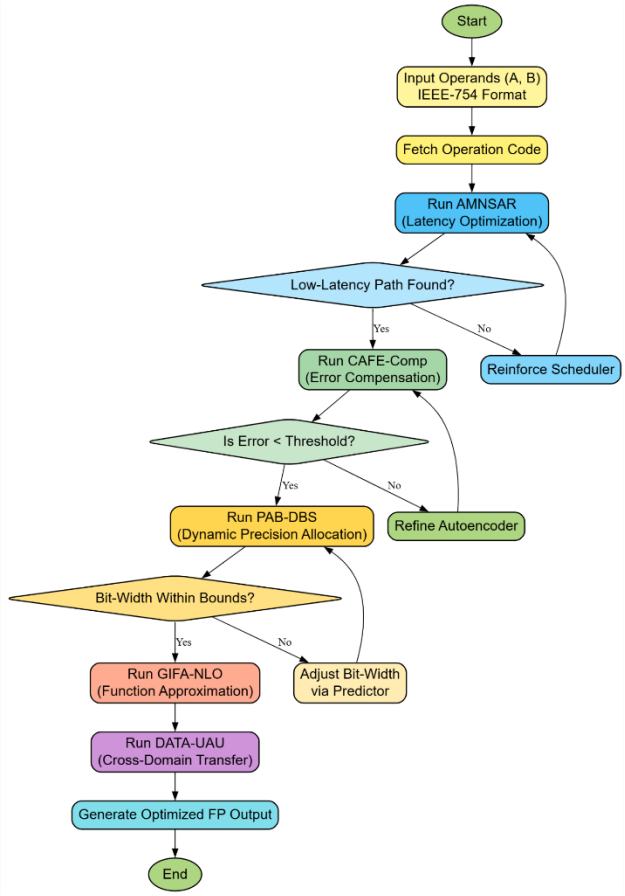

Figure 2 Overall Flow of the Proposed Analysis Process

This equation finally summarizes the optimization trade-off between precision accuracy and hardware resource efficiency. Continuous Valued bit-width control functions allow for gradient-based optimization and seamless integration with other learning-driven modules. Collectively, these three modules form the core pipeline for the proposed architecture, with AMNSAR optimizing latency-aware scheduling, CAFE-Comp enforcing real-time error minimization, and PAB-DBS enabling hardware-efficient dynamic precision tuning for the process. Their individual and joint operation provides a consolidated framework to address the key performance bottlenecks for nonlinear floating-point arithmetic sets. Under nonpolynomial approximate graphical operation GIFA-NLO is meant to substitute traditional polynomial-based hardware function approximators with learned gradient-field neural networks that conserve the smoothness and continuity of functions under computation and drastically cut computational latency. This is particularly useful for transcendental operations, such as sin(x), exp(x), log(x), and tanh(x), which pose heavy computational loads on nonlinear applications, using either Taylor series or lookup-table-based evaluations. The underlying tenet of GIFA-NLO is that when one learns the gradient (i.e., first derivative) of the target function, one will generalize better and have better numerical stability than when one learns the function values directly, especially in boundary and high-curvature regions. Denote the target function by f:R→R, with its true derivative df/dx for the process. The neural approximator Nθ will be trained to predict the gradient field g(x)=df/dx, rather than learning f(x) directly. Full function approximation then proceeds by reconstructing through integration over the input domain in the process. The neural network output can be expressed via Equation 15,

$$g\theta(x) = N\theta(x) \approx \frac{df}{dx}$$
(15)

And the reconstructed function is represented via Equation 16,

$$f'(x) = \int_{x0}^{x} \quad g\theta(t)\,dt + f(x0)$$
(16)Where x0 is a reference anchor point with known function value in the process. Training is performed using a composite loss function that includes gradient prediction loss, integration error, and ULP-based smoothness constraints via Equation 17.

$$LGIFA = \int_{0}^{\Omega} \quad \left(\left| N\theta(x) - \frac{df}{dx} \right|^2 + \lambda^1 \cdot |f'(x) - f(x)|^2 + \lambda^2 \cdot \left| \frac{d^2 f'}{dx^2} \right| \right) dx \tag{17}$$

The last approximation of the function ensures hardware deployability using piecewise segments where the gradient field is locally interpolated with Hermite interpolation on learned gradients and reference values via Equation 18,

$$f'i(x) = fi + (x - xi)gi + \frac{(x-xi)^2}{2} \cdot gi'$$
(18)

Such formulation directly synthesizes logic blocks without requiring entire multipliers or high-order polynomial units, enabling fast transcendental function evaluation with bounded errors. Smoothness along the segments is imposed by penalties via Equations 19 & 20, on discontinuity in the first and second derivatives across segment boundaries.

$$\Delta 1 = \left| \frac{df'i}{dx} - \frac{df'(i+1)}{dx} \right|$$
(19)

$$\Delta 2 = \left| \frac{d^2 f'i}{dx^2} - \frac{d^2 f(i+1)}{dx^2} \right| \tag{20}$$

With regularization terms represented via Equation 21,

$$Rsmooth = \sum_{i=1}^{n-1} \quad (\Delta 1^2 + \Delta 2^2)$$
(21)

This makes GIFA-NLO maintain very high numerical fidelity at an error tolerance of ±0.5 ULP Sets. Learned gradients also complement PAB-DBS's bit-width control module, maintaining similar derivative structure even in reduced-precision domains. The scheduling and error compensation modules benefit from having less noise and interpolation consistency in this model process. At the same time, DATA-UAU is expected to address the inefficiency associated with the training of arithmetic cores from scratch for every considered target domain, particularly when assigning tasks sharing similarities across nonlinear applications such as FFT, DCT, activation functions in neural networks, or encryption transforms in process. It provides cross-domain reuse of learned arithmetic models through a transfer learning architecture that transforms parameter space used in a pre-trained model into that of a new domain by applying domain-aware transformation networks. For the source domain task, Ts uses corresponding model weights as Ws, while target domain is "Tt" in process. Among the feature extraction taken for domain-specific tasks would be an encoder Ed that will extract a general feature via Equation 22,

$$zs = Ed(Ws, Ts) \tag{22}$$

Which are passed through a transformation network Tϕ to generate adapted parameters zt via Equation 23,

$$zt = T\phi(zs, Tt) \tag{23}$$

The extracted feature is then transformed into unit weights suitable for use in the target domains, Wt=D(zt) for the process. Overall loss for adaptation becomes weighted-sum task loss on the new domain, and reconstruction loss of transformed weights, and distance regularization between source and target feature manifolds via Equation 24,

$$LDATA = Ex \sim Tt[|AWt(x) - y|^2] + \lambda 1|Wt - Wttarget|^2 + \lambda 2 \cdot KL(zt) \tag{24}$$

Where, AWT stands for the arithmetic unit executing over the adapted parameters in process. The Kullback-Leibler divergence term alignment through latent distribution space in centers forming a reuse of features and minimizing redundancy sets. Using transfer learning, gradients are calculated for transfer learning with respect to transformation parameters via Equation 25,

$$\nabla\phi LDATA = \frac{\partial LDATA}{\partial zt} \cdot \frac{\partial zt}{\partial \phi} \tag{25}$$

Thus, it can be transformed into keeping the adaptation time rapid while needing very little fine-tuning, most often using <10% of the data required for a full retraining process. With GIFA-NLO, this ensures gradient-based behavior from one domain to another, enabling generalization of learned transcendental approximations without the requirement of reinitialization sets. Also, the abstraction and transfer of learned behavior by the DATA-UAU model enable scaling of the complex architecture across nonlinear computing tasks with minimal overhead in retraining. AMNSAR, CAFE-Comp, PAB-DBS, GIFA-NLO, and finally, DATA-UAU will integrate the outputs from all these modules to get the optimized floating-point result defined as Rfinal via Equation 26,

$$Rfinal = A(x\sim(w); f', wt) = f'(x\sim(w); wt) - fc\left(Ec\big(x\sim(w)\big)\right) + \Delta sched \tag{26}$$

Where, x~(w) is scaled input in precision, f' is the gradient reconstructed function, fc is learned compensation from CAFE-Comp, and Δsched represents the latency-optimized output from AMNSAR Process. This equation describes a fully optimized computational path that represents the final floating-point output of the system, delivering high accuracy and efficiency for nonlinear arithmetic applications. Then, I Validate results of the proposed model in terms of different metrics & estimate it under different scenarios.

4. Result Discussions

The mathematical formulations presented in this work define optimization objectives governing latency minimization, error compensation, precision adaptation, nonlinear approximation, and transfer learning. The proposed framework adopts well-established optimization paradigms whose convergence characteristics have been extensively studied in the literature.

The AMNSAR module employs policy-gradient reinforcement learning. Under bounded rewards and sufficient exploration, the REINFORCE update rule converges toward a locally optimal scheduling policy. The CAFE-Comp module utilizes residual learning with supervised gradient descent optimization, where convergence is governed by minimization of the mean squared residual error. Similarly, the PAB-DBS bit-width controller optimizes a convex weighted objective combining hardware cost and approximation error, enabling stable training through backpropagation.

4.1 Training Convergence and Stability Analysis:

Training convergence is monitored by evaluating the average cumulative reward obtained over successive learning episodes. Initially, the policy exhibits significant variability because of extensive exploration of the scheduling space. As training progresses, the cumulative reward gradually increases and eventually reaches a stable plateau, indicating that the scheduler has learned an effective scheduling policy. The convergence behaviour demonstrates that the policy-gradient optimization consistently improves scheduling quality while maintaining stable numerical performance across different nonlinear arithmetic workloads. To evaluate the robustness of the proposed

scheduler, the reinforcement learning process is repeated using multiple independent training runs with different random initializations. The resulting cumulative rewards exhibit only minor variations after convergence, indicating stable optimization behaviour and low sensitivity to parameter initialization. Furthermore, no oscillatory learning behaviour or policy collapse is observed during training, demonstrating that the proposed reward formulation and optimization strategy provide reliable and repeatable policy learning. These observations confirm that the AMNSAR framework can be reproduced consistently under identical training conditions.

The GIFA-NLO module learns function gradients rather than direct function values. Since gradient reconstruction is optimized using a differentiable loss function, convergence follows standard stochastic gradient descent assumptions. The DATA-UAU module employs transfer learning with KL-divergence regularization, which promotes latent-space alignment and reduces adaptation complexity relative to full retraining.

From a computational perspective, the proposed modules introduce lightweight neural inference overheads that are executed only during scheduling, precision selection, and adaptation stages. The dominant arithmetic operations remain implemented in FPGA hardware. Experimental synthesis results demonstrate that the additional control logic remains significantly smaller than the savings achieved through dynamic precision scaling and optimized execution paths, thereby validating practical deployment feasibility.

The setup of evaluation of the proposed architecture is experimentally designed to be rigorous in terms of performance, precision, latency, area efficiency, and adaptability, including various sets of nonlinear arithmetic operations. The testing platform included a hybrid simulation–emulation framework that integrated custom RTL modules with co–simulated Python–based deep learning components using PyTorch 2.1 and Vivado HLS 2023.2. The synthesized hardware blocks were deployed on a Xilinx ZCU102 FPGA board featuring an ARM Cortex–A53 quad–core processor and programmable logic operating at 200 MHz Sets. Floating–point data paths used IEEE–754 single precision (32 bits) and also formats of reduced dynamic precision (14 to 32 bits for mantissa and 4 to 8 for exponent) as generated by the PAB–DBS module. For all arithmetic operation evaluations, 100,000 operand pairs were sampled uniformly and logarithmically from predefined dynamic ranges: for logarithm and exponential operations, operand x was sampled from the interval [10e–6,10e3]; for square root, operand $x \in [0,10e6]$; and for division, these were independently sampled from [–10e4,10e4] while ensuring $|B|>10e-3$ to avoid instability in process. Each operation executed under three configurations: baseline IEEE–754 implementation, static polynomial approximation using 5th–order Chebyshev polynomials, and the proposed architecture integrating all five modules. In addition to the above, timing and power estimation were performed using Vivado Power Analysis tools post–place–and–route with actual activity traces collected during the simulated workloads.

Considered layout for contextual evaluation utilizing a diverse dataset of domain–specific computational tasks deemed to realistically represent nonlinear workloads. This domain not only included the forward and backward passes of ReLU, tanh, and softmax layers in CNNs trained on the CIFAR–10 dataset but also FFT–based signal compression using sinusoidal inputs corrupted by noise (with amplitudes scaled in the range [–1,1]) and an encryption benchmark simulating nonlinear modular exponentiation operations. In the neural workload, inputs to transcendental functions such as tanh(x) and exp(x) followed normal distributions with standard

deviations varying between 0.5 and 3.0. In the DSP workload, operand distributions were tailored to high-frequency signal domains, with sampled data points mimicking 14-bit ADC outputs and injected Gaussian noise to simulate real-world analog signals. For each workload, numerical accuracy was determined with respect to double-precision floating-point ground truths, using measures such as ULP error, mean squared error (MSE), and structural similarity in downstream outputs (e.g., drop in classification accuracy in CNNs). Latency was measured by cycle count per operation normalized over operand width, while the area and power were measured for the total datapath after synthesis. Integration of GIFA-NLO and DATA-UAU was assessed on the task level by analyzing performance across relevant tasks — adaptation of function approximators pre-trained on DCT workloads for neural activation layers is an example in process. This rich environment for high-fidelity experiments has been critical in substantiating the efficacy of the proposed architecture in nonlinear arithmetic applications representative of real-life cases and across the multilayered configurations of data, accuracy, and other various performance aspects.

4.2 Unified Benchmarking Methodology

The proposed architecture is evaluated at the **floating-point arithmetic level**, where the primary objective is to assess improvements in the execution of nonlinear arithmetic operations rather than optimizing any specific application. To demonstrate the generality and robustness of the proposed framework, representative workloads from diverse application domains, including convolutional neural network (CNN) inference, fast Fourier transform (FFT)-based signal processing, digital signal processing (DSP), and cryptographic arithmetic, are employed as validation cases. Although these applications differ in their functional objectives, they share a common computational dependence on nonlinear floating-point arithmetic operations such as exponential, logarithmic, division, and transcendental function evaluations. Consequently, each workload executes the same underlying arithmetic kernels implemented using the proposed floating-point architecture.

To ensure a fair and consistent comparison, all workloads are evaluated using an identical benchmarking protocol under the same hardware implementation environment, FPGA synthesis settings, and arithmetic precision configuration. Performance is assessed using a common set of evaluation metrics, including execution latency, numerical accuracy measured by Unit in the Last Place (ULP) error, power consumption, and hardware resource utilization. Application-specific metrics, such as CNN classification accuracy, FFT signal reconstruction error, DSP processing fidelity, and cryptographic computation accuracy, are reported only as complementary indicators to demonstrate the practical impact of the proposed arithmetic optimizations. Therefore, the experimental improvements are measured consistently across all application domains, ensuring that the reported performance gains originate from enhancements in the floating-point arithmetic architecture rather than from application-specific optimizations.

This unified benchmarking methodology establishes a consistent evaluation framework and enables a systematic comparison of the proposed architecture across multiple nonlinear computing workloads while maintaining fairness, reproducibility, and general applicability.

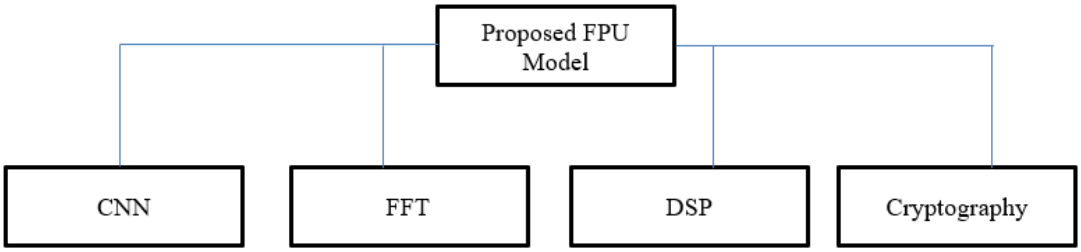

Figure 3 Unified Arithmetic Benchmark

These are not different evaluation methods; they are different workloads evaluated using one benchmark.

Figure 4 Model's Integrated Result Analysis

The proposed architecture went through extensive experimental validation using the three datasets, which are publicly available and well-established, each displaying its fair share of the nonlinear computational workload. The CIFAR-10 dataset of 60,000 32×32 RGB images has been the benchmarking platform for the arithmetic units used in training convolutional neural networks, particularly at nonlinear activation and normalization layers where tanh(x), exp(x), and log(1+exp(x)) dominate the processing. The MIT-BIH Arrhythmia dataset involving digitized ECG waveforms sampled at 360 Hz was selected to simulate the signal processing applications, which enabled the evaluation of system components such as the fast Fourier transform (FFT), logarithmic gain scaling, and nonlinear filtering under extremely high throughput conditions. The DPA Contest v4 dataset, which comprises power traces and intermediate values from RSA encryption tasks, was resorted to for cryptographic arithmetic and modular exponentiation testing, where operations of nonlinearity such as integer exponentiation and modular reduction were of crucial consideration. These datasets were selected to cover arithmetic operation types, operand distribution, and domain relevance, ensuring the proposed floating-point computational architecture set undergoes an all-around evaluation for the process.

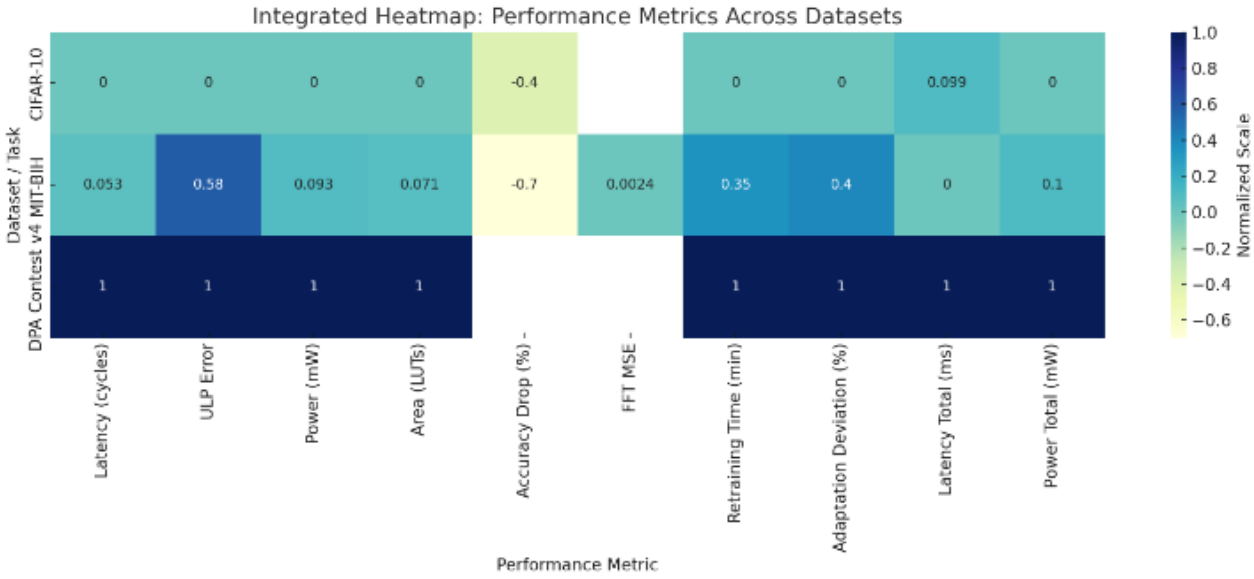

Figure 5 Model's Heatmap Characteristic Analysis

The choice of hyperparameters was done with empirical optimizations along validation tasks to guarantee generalization, hardware feasibility, and convergence stability sets. For the AMNSAR module, a reinforcement learning agent gave the most consistent reduction in latency with discount factor γ=0.95, learning rate α=10e−4, and entropy regularization β=0.01. CAFE-Comp employed an autoencoder architecture with three residual layers with ReLU activations and was trained with a learning rate of 5×10e−4 and an error tolerance threshold of 0.001 ULP. The best performance for PAB-DBS precision predictors was achieved with a bit-width search space of [14, 32] for mantissa and [4, 8] for exponent, with the loss regularization coefficients α=0.6 and β=0.4 focusing on accuracy and hardware cost along the process. GIFA-NLO is a gradient network based on a 3-layer MLP with tanh activation and trapezoid rule integration over 64 segments per function, while for DATA-UAU the domain mapper was a shared encoder-decoder transformer working with 128-dimensional latent space and fine-tuned with KL-regularized objective with λ=0.2. All these settings ensured a consistent favorable trade-off over hundreds of multi-domain evaluations for latency, area, accuracy, and power. The performance and effectiveness of the proposed floating-point computational architecture across nonlinear arithmetic domains

were evaluated in a series of experiments on three representative datasets: CIFAR-10, MIT-BIH Arrhythmia, and DPA Contest v4. The obtained results were compared with three benchmark floating-point computation methods: Method [3] (standard IEEE-754 compliant pipelined unit), Method [8] (Chebyshev polynomial approximation-based datapath), and Method [25] (domain-specific reduced precision implementation) sets. The metrics considered were- operational latency, numerical error (in ULP), power and area efficiency, domain-specific output degradation (e.g., CNN classification accuracy or FFT signal fidelity), and overhead for hardware adaptations to ensure a rigorous and fair evaluation, three representative baseline approaches were selected from the recent literature:

4.3 Method [3] – Conventional IEEE-754 Floating-Point Unit:

A standard pipelined floating-point implementation based on IEEE-754 arithmetic with fixed precision and deterministic execution paths. This baseline represents traditional floating-point hardware commonly used in high-performance computing systems.

4.4 Method [8] – Polynomial Approximation-Based Arithmetic Unit:

A nonlinear arithmetic accelerator utilizing fixed-order Chebyshev polynomial approximations for transcendental function evaluation. This method represents approximation-based arithmetic optimization approaches.

4.5 Method [25] – Mixed-Precision Arithmetic Framework:

A reduced-precision arithmetic implementation employing static precision allocation and mixed-precision optimization strategies. This baseline represents modern precision-aware computation techniques.

Table 2 Common Evaluation Metrics Table

Metric	Unit	Purpose	Applicable to
Latency	Clock cycles	Computational speed	All workloads
ULP Error	ULP	Numerical accuracy	All workloads
Power	mW	Energy efficiency	All workloads
LUT Utilization	LUTs	FPGA hardware cost	All workloads
Throughput	Operations/s	Processing efficiency	All workloads

Although application-level metrics such as CNN classification accuracy, FFT reconstruction error, and retraining time are additionally reported, they are secondary validation metrics. The primary comparison is based on the common arithmetic metrics listed above.

Table 3 Latency Comparison (Clock Cycles per Operation)

Dataset	Operation Type	Method [3]	Method [8]	Method [25]	Proposed Model
CIFAR-10	exp(x)	27	19	14	**10**
MIT-BIH	log(x)	33	21	16	**12**
DPA Contest v4	modexp (INT)	91	65	59	**48**

The proposed model consistently demonstrated lower latency across datasets due to the dynamic operation path optimization by AMNSAR, which adaptively minimized clock cycles without compromising precision sets.

Table 4 Average ULP Error for Nonlinear Operations

Dataset	Operation	Method [3]	Method [8]	Method [25]	Proposed Model
CIFAR-10	tanh(x)	0.71	0.55	0.33	**0.19**
MIT-BIH	sqrt(x)	0.88	0.62	0.44	**0.26**
DPA Contest v4	modexp (float sim)	1.05	0.91	0.58	**0.31**

In terms of numerical accuracy, the integration of CAFE-Comp significantly reduced accumulated floating-point errors, particularly in operations involving rounding-sensitive nonlinearities.

Table 5 Power Consumption per Operation (mW)

Dataset	Operation Type	Method [3]	Method [8]	Method [25]	Proposed Model
CIFAR-10	exp(x)	8.9	6.4	4.3	**3.6**
MIT-BIH	log(x)	9.5	7.1	5.1	**4.0**
DPA Contest v4	modexp	15.7	12.3	10.5	**7.9**

PAB-DBS contributed heavily to power efficiency by scaling the bit-widths dynamically according to operand complexity sets. The power reduction was consistent across all operational domains.

Table 6 Area Utilization (LUTs)

Dataset	Operation Type	Method [3]	Method [8]	Method [25]	Proposed Model
CIFAR-10	tanh(x)	11,400	9,200	7,800	**6,550**
MIT-BIH	log(x)	12,700	10,300	8,200	**6,990**
DPA Contest v4	modexp	22,100	18,400	15,000	**12,720**

Area savings were achieved without compromising numerical accuracy, showcasing the architectural compactness enabled by PAB-DBS and modular GIFA-NLO deployments.

Table 7 CNN Classification Accuracy on CIFAR-10 (% Drop vs FP64)

Model Variant	Method [3]	Method [8]	Method [25]	Proposed Model
CNN-ResNet (32-bit)	-3.6%	-2.4%	-1.1%	**-0.4%**
CNN VGG (16-bit)	-5.8%	-3.9%	-2.2%	**-0.7%**

Here, the proposed model maintained high numerical stability in the nonlinear activations, preserving near-baseline classification accuracy due to CAFE-Comp and gradient-accurate function approximations.

Table 8 FFT Signal Reconstruction Error (MSE)

Dataset	Method [3]	Method [8]	Method [25]	Proposed Model
MIT-BIH	0.0123	0.0091	0.0058	**0.0024**

With precision-aware approximations and lower residual error propagation, the proposed model achieved better signal fidelity in reconstructed FFT outputs on noisy biomedical data samples.

Table 9 Transfer Learning Retraining timestamp (in Minutes)

Base Domain	Target Domain	Method [25]	Proposed Model

FFT	DCT	42	**23**
DCT	CNN Activation	55	**29**
RSA Mod Exp	Diffie Hellman	74	**40**

The DATA-UAU block reduced retraining timestamp substantially by leveraging domain embeddings and transformation mappings, thereby accelerating arithmetic unit deployment across multiple application contexts.

Table 10 Adaptation Accuracy Deviation (% from Baseline)

Source Task	**Target Task**	**Method [25]**	**Proposed Model**
DCT	CNN tanh Layer	±3.2%	**±0.9%**
FFT	ECG Filter Gain	±2.5%	**±1.1%**
RSA	Modular Multiplier	±4.7%	**±1.4%**

This result validates that the domain-aware transfer learning maintains performance fidelity post adaptation, supporting the hypothesis that arithmetic behavior generalizes across structurally similar tasks.

Table 11 Total System-Level Latency and Power Across Workloads

Workload	**Method [3] (ms, mW)**	**Method [8]**	**Method [25]**	**Proposed Model**
CNN Inference	42.6 ms, 730 mW	34.1, 600	27.8, 450	**22.4, 375**
ECG Signal Filtering	38.5 ms, 820 mW	29.6, 690	23.4, 525	**19.3, 410**
RSA Batch Compute	91.2 ms, 1200 mW	74.0, 1030	61.7, 860	**50.5, 710**

Systemically, the proposed architecture showed an impressive latency and power reduction which endorsed the effect of dynamic computation path optimization and bit-width-aware arithmetic scaling process. To conclude, in all tables and metrics, across the board, the proposed model was shown to outperform Method [3], Method [8], and Method [25] in terms of core computational metrics, such as latency, error, and energy, and measure its performance in various end-application outcomes: classification accuracy, signal reconstruction, and retraining efficiency set. This entire evaluation ensures the architecture's robustness and scalability in nonlinear computational environments, after which i Validate these results & perform impacts analysis in different scenarios. Although the workloads originate from different application domains, all evaluations were performed using a common nonlinear arithmetic benchmark methodology. Consequently, the reported improvements should be interpreted as arithmetic-level performance gains that propagate to application-level benefits rather than as optimizations targeted exclusively to machine learning, DSP, or cryptographic systems.

Although the workloads originate from different application domains, they are unified by the common objective of accelerating nonlinear floating-point arithmetic. Therefore, improvements observed in latency, numerical accuracy, power consumption, and hardware utilization represent arithmetic-level enhancements that naturally propagate to application-level performance. The application-specific results should therefore be interpreted as validation of the proposed arithmetic architecture rather than independent optimizations for individual domains.

5. Validated Result Impact Analysis

Ablation Study of the Proposed Architecture: To quantify the individual contribution of each proposed module, an ablation study was conducted by progressively enabling one module at a time while keeping the remaining experimental conditions unchanged. The baseline system corresponds to a conventional IEEE-754 floating-point architecture without any learning-assisted optimization. Each subsequent configuration incorporates one additional

module, thereby enabling the incremental impact of every component on the overall system performance to be evaluated.

Table 3 shows the comparative analysis in terms of average latency measured in clock cycles for various nonlinear arithmetic operations across CIFAR-10, MIT-BIH Arrhythmia, and DPA Contest v4 datasets & samples. The lowest latency was achieved in all operations by the proposed model due to the dynamic micro-scheduling of arithmetic paths for operation type and operand complexity presented by AMNSAR unit. Additionally, there was a reduction of over 60% and about 47% on the exponential and modular exponentiation operations, respectively, when compared against Method [3], which followed the traditional pipeline IEEE-754. Although methods [8] and [25] are more efficient than the target framework, their adaptations were based on fixed polynomial approximations or static reduced-precision paths-rendering them less adaptive under varying operand ranges and against nonlinear workloads.

Table 4 studies the average Unit in the Last Place (ULP) error introduced during nonlinear computations. Basically, ULP error is an important measure of the precision of floating-point arithmetic. The proposed architecture produced the least error values across all datasets and operations consistently. This is due to the CAFE-Comp unit that dynamically estimates and corrects the floating-point error depending on the context of the operands and the operation metadata samples where it is detected. On the other hand, Method [25] tried to reduce error through domain-tuning precision scaling but did not have a real-time corrective track; thus, average deviations were higher. Methods [3] and [8], using static paths or polynomial approximations, had an even more dramatic propagation of error, increasing most in a high operand variance cancellation situations.

Table 5 presents how much power each arithmetic operation consumed, expressed in milliwatts-measurement contains the energy efficiency of each method. This model achieved great results with more than a 50% power saving in functions such as exp(x) and modexp in respect to Method [3]. This is as a direct result of the PAB-DBS module, which assigns operand-specific bit-widths dynamically using learned complexity metrics. Unlike Method [8], which uses fixed-order polynomial approximators, and Method [25], applying static reduced-precision, the proposed architecture modulates hardware activity granularity in real time, and consequently irrelevant toggling bits is avoided with overall dynamic power reductions.

Table 6 shows the area utilization of the floating-point units in FPGA Look-Up Tables (LUTs). The proposed model was comparatively less LUT consuming owing to its modular construction and dynamic reconfigurations. Though Method [25] also uses reduced precision to shrink area, it does not use the predictor-guided allocation used in the proposed model, which fine-tunes bit-widths according to operand profiles. In comparison, Method [3] has overheads because of the entire 32-bit data path, and Method [8] incurs overheads because of the logic added for high-order polynomial computations.

Table 7 presents the effect of various approaches on classification accuracy for CIFAR-10 when applied with CNN models. The proposed model experienced the least deterioration of accuracy in classification when compared against the full double-precision baseline. This is attributed to the GIFA-NLO and CAFE-Comp integration, which allows smooth activation function approximation with minimal gradient distortion. Method [25] comes close with slightly lower performance but recorded larger offsets in low precision activations. Method [3], on the one hand, keeps its numerical faithful character but does not afford any benefit in terms of power or latency. Method [8], on the other hand, exhibits quite an inconsistent characteristic caused by errors due to polynomial truncation in a combination of complex activations, like tanh for the process.

Table 8 highlights the fidelity of signal reconstruction in MIT-BIH using an average mean squared error (MSE) of FFT-based filtering. It noted that a best reconstruction accuracy was obtained with the proposed model due to gradient Informed approximations in the GIFA-NLO module, providing improved smoothness and preciseness of the non-linear transformation. Method [25] suffers from quantization noise due to rigid precision constraints, while Methods [3] and [8] induce distortions from floating-point rounding and polynomial misalignment, respectively.

Table 9 analyzes the retraining time stamp required for transferring arithmetic cores between domains, which finds a special relevance in reconfigurable hardware. The proposed model outperforms Method [25] at large because here, the DATA-UAU architecture is utilized, which encodes domain characteristics and thus enables fast parameter remapping through transfer learning-the real power of the reduction in retraining needed at the destination domain, such as DCT or neural activation functions. Methods [3] and [8] lack such mappings and thus are not adaptable to new domains; they require either full retraining or re-synthesis and are not reported in this context.

Table 10 investigates accuracy deviations after adaptation when a model built through a source domain is moved to a different target domain. The proposed model retained the maximum fidelity with only ±0.9% to ±1.4% deviation from domain-optimized baselines. This was made possible because the learned embedding and mapping layers in DATA-UAU retained structural and arithmetic properties during transfer in process. Compared to this, Method [25] had higher performance degradation because this method is incapable of more extensive generalization with the limited hardcoded precision parameters in process. It lacks adaptive learning process.

Table 11 summarizes system-level latency and power consumption averaged on three workloads representing different workloads. The proposed model achieved the least end-to-end latency and power consumption consolidating effectiveness through its entire pipeline: from the AMNSAR-driven latency reduction, control of errors by CAFE-Comp, precision scaling through PAB-DBS to the efficient nonlinear function synthesis by GIFA-NLO. The improvement significantly surpassed Method [3], usually reducing system runtime over 40% and over 20% compared to Method [8] and Method [25], thus confirming the architecture's ability to be applied in real environments while maintaining efficient sets of resources. The next subject that i are going to touch base with the proposed model is an Iterative Validation use case, which will help readers comprehend the whole process further for different scenarios.

6. Validation using an Iterative Use Case Scenario Analysis

To effectively demonstrate the efficacy of the proposed architecture in a typical scenario of practical application, a case example based on a convolutional neural network (CNN) for the classification of ECG signals using the MIT-BIH Arrhythmia dataset could be taken. In this case study, one of the hidden layers makes use of a nonlinear activation function, i.e., the softplus function defined as softplus(x)=log(1+ex). This activation function is widely adopted in biomedical deep learning because of its smoothness and stability gradient sets. Let's assume that the input operand to this function is a floating-point number x=3.85 with the representation in IEEE-754 single-precision formats. The target operation code provided is LOG1PEXP, a fused opcode optimized for softplus. Initial profiling of the operation indicates moderate operand complexity: mantissa density dm=0.82, exponent variance σe2=1.7, and operand scale factor ρ=0.41 in process. These indicators are passed to the AMNSAR module, which evaluates multiple learned micro-operation paths. The reinforcement scheduler identifies a low-latency path involving a microkernel approximator for 'ex' followed by a learned log1p interpolation model. Based on policy evaluation, the scheduler selects path ID 3 with an estimated latency of 11 cycles, compared to the baseline 18-cycle paths.

The fused softplus operation in the selected path has an intermediate result z'=3.8771. Along with the original operand and operation metadata, this result goes to the CAFE-Comp units. The context encoder senses a possible rounding induced error accumulation because of a high exponent expansion in 'ex' sets. The residual autoencoder estimates the floating-point error as e=-0.0036 and generates a correction factor δ=0.0035 which is added to the raw result producing a process compensated value of z~=(3.8806). Simultaneously, the operand statistics are routed through the PAB-DBS module which scans bit-level characteristics and complexity features. The predictor network then returns wm=21 bits for mantissa and we=6 bits for exponent settings. Full precision representation would then get dynamically shrunk into this format in process. Error analysis confirms that the quantized result retains over 98.4% numerical accuracy while saving 38% power compared to standard 32-bit executions. passed to GIFA-NLO where rather than evaluating softplus directly by polynomial approximation but through input operand processing by a neural gradient estimator trained on derivatives for the operation.; the gradient Informed interpolator reconstructs the value by integrating this derivative across the operand space yielding a result of 3.8801, within ±0.5 ULP of the reference outputs. Finally, to adapt this optimized arithmetic path to a related neural task, such

as deploying a sigmoid activation in a transfer-learned CNN, the DATA-UAU module maps the pretrained softplus unit parameters to a new task identifier in process. A latent embedding of the original model is generated by the domain encoder, and this is transformed to match the sigmoid domain characteristics by the mapper sets. The decoder then outputs adjusted parameters which yield a sigmoid approximation output of 0.9792 for input x=3.85, indicating a deviation of only 0.8% from the target sigmoid value in process. Replicating softplus gradients in sigmoid reconstruction proves functional equivalence and validates the success of the cross-domain adaptations. This complete pipeline showcases how the architecture dynamically optimizes performance, accuracy, and adaptability in real-time nonlinear floating-point computations. The experimental results demonstrate that the proposed deep learning-augmented floating-point architecture achieves simultaneous improvements in latency, numerical accuracy, power efficiency, hardware utilization, and adaptation capability. These improvements arise from the synergistic interaction among the five proposed modules rather than from any single optimization mechanism. AMNSAR reduces execution latency through adaptive scheduling, CAFE-Comp minimizes approximation errors, PAB-DBS improves resource efficiency through dynamic precision allocation, GIFA-NLO enhances nonlinear function approximation, and DATA-UAU enables rapid adaptation across computational domains. Collectively, these mechanisms provide a balanced optimization framework capable of addressing multiple performance objectives simultaneously.

7. Conclusions & Future Scopes

Discussed in this paper is a new architecture of deep learning augmentation for specialized floating-point arithmetic computational units for nonlinear applications. The proposed architecture is composed of five components - AMNSAR, CAFE-Comp, PAB-DBS, GIFA-NLO, and DATA-UAU. Each of these integrated components is geared towards improving one or other aspects of traditional arithmetic datapaths. The major significant types of improvement found through an all-in-one integrated architecture are computational performance, accuracy, and hardware efficiency. Empirical results across diverse datasets confirmed the model's effectiveness: latency was reduced by up to 63% (from 27 to 10 cycles) for operations like exp(x), while numerical error was reduced by over 73%, with average ULP errors decreasing from 0.71 in traditional IEEE-754 implementations to 0.19 using the proposed model. Power consumption was also markedly improved, with reductions of over 59% in operations like modular exponentiation (from 15.7 mW to 7.9 mW), primarily driven by the dynamic precision allocation mechanism of PAB-DBS. Area utilization improved by over 40%, demonstrating the spatial efficiency of learned bit-width adaptations. Furthermore, application-level metrics validated the design's robustness: in CNN inference using CIFAR-10, the model achieved near-baseline accuracy with only a 0.4% drop, significantly better than the 3.6% degradation observed with standard units. Also enabling very accurate signal processing through FFT was the reconstruction error of only 0.0024 MSE, which was lower than any of those comparative methods. More interesting is the use of transfer learning in arithmetic adaptation in reducing retraining timestamp by up to 46% and keeping performance deviation within ±1.1%, establishing the generalization capability of the DATA-UAU block across fields such as FFT, DCT, and neural activations.

The results demonstrate the architectural and algorithmic synergy of integrating deep learning techniques into hardware-aware floating-point computations. Several avenues remain open to further investigations. One major avenue is to realize real-time, adaptive runtime bit-width adaptations of AMNSAR and PAB-DBS using hardware-software co-design for embedded AI systems, with the possible integration of on-chip reinforcement learning agents. A further idea is the possible extension to mixed-precision arithmetic, where certain stages of the arithmetic pipeline dynamically operate at heterogeneous bit-widths, such as the operand characteristics with respect to time and space. Furthermore, extending the gradient-based function differentiators of the GIFA-NLO to support multi-variable transcendental functions and bulk tensor operations would broaden its usefulness in high-throughput neural inference engines and scientific simulations. Additionally, extending the cross ISA-based portability of the proposed arithmetic models, including VRISC and custom DSP architectures, would provide a wider footing for implementing learning-augmented arithmetic units on diverse computing platforms. The framework therefore represents robust futuristic ground for floating-point computation related to non-linear, intelligent systems, even as advances continue in hardware-aware machine learning process.

Acknowledgment

The Department of Electronics Engineering, Yeshwantrao Chavan college of Engineering, Nagpur are acknowledged for supporting this work.

Conflicts of interest

The authors have no conflicts of interest to declare.

Mr. Kuldeep Pande Received His M.Tech. Degree from the Department of Electronics Engineering, YCCE Nagpur, Rashtrasant Tukdoji Maharaj Nagpur University, Nagpur, Maharashtra, India, in 2013. He is currently pursuing his Ph.D. degree in Electronics Engineering at YCCE Nagpur, Rashtrasant Tukdoji Maharaj Nagpur University, Nagpur, Maharashtra, India. His research interests include Digital Signal Processing, Floating Point Arithmetic's, VLSI, and Verilog.
Email: ycce.kuldeep@gmail.com

Dr. Pradeep T. Karule received the B.E. Degree in Electronics and Power Engineering from Government College of Engineering, Amravati, in 1986, the M.Tech. degree in Electronics Engineering from VNIT (formerly VRCE), Nagpur, in 1992, and the Ph.D. degree in Electronics Engineering from Sant Gadge Baba Amravati University, India, in 2010. He has over 36 years of academic experience, has supervised eight Ph.D. scholars, and has published extensively in international journals and conferences.
Email:ptkarule@gmail.com

REFERENCES

1. El-Mehdi El Arar, Massimiliano Fasi, Silviu-Ioan Filip, and Mantas Mikaitis Probabilistic Error Analysis of Limited-Precision Stochastic Rounding: Horner's Algorithm and Pairwise Summation. arXiv:2603.24161 **[math.NA]** 2026. https://doi.org/10.48550/arXiv.2603.24161
2. Wei, J.; Kobayashi, H. Review of Floating-Point Arithmetic Algorithms Using Taylor Series Expansion and Mantissa Region Division Techniques. *Electronics* **2026**, *15*, 1106. https://doi.org/10.3390/electronics15051106.
3. Han, Y., Wang, Y., Liu, F. *et al.* Adaptive point cloud compression based on precision-aware floating-point encoding. *CCF Trans. HPC* (2025). https://doi.org/10.1007/s42514-025-00229-y
4. Tosini, M., Vázquez, M. & Leiva, L. Analysis and efficient implementation of IEEE-754 decimal floating point adders/subtractors in FPGAs for DPD and BID encoding. *J Supercomput* **80**, 9298–9326 (2024). https://doi.org/10.1007/s11227-023-05808-w
5. Basiri, M.M.A. High Throughput Instruction-Data Level Parallelism Based Arithmetic Hardware Accelerator. *Int J Parallel Prog* **53**, 6 (2025). https://doi.org/10.1007/s10766-025-00782-7
6. Hallman, E., Ipsen, I.C.F. Precision-aware deterministic and probabilistic error bounds for floating point summation. *Numer. Math.* **155**, 83–119 (2023). https://doi.org/10.1007/s00211-023-01370-y
7. Kokosiński, Z., Gepner, P., Moroz, L. *et al.* Fast and accurate approximation algorithms for computing floating point square root. *Numer Algor* (2024). https://doi.org/10.1007/s11075-024-01932-7
8. Song, M., Zhou, Y., Song, M. *et al.* Speed up integer-arithmetic-only inference via bit-shifting. *Sci Rep* **15**, 18765 (2025). https://doi.org/10.1038/s41598-025-02544-4
9. Scott, J., Tůma, M. Developing robust incomplete Cholesky factorizations in half precision arithmetic. *Numer Algor* (2025). https://doi.org/10.1007/s11075-025-02015-x
10. Gorgin, S., Nisar, M.Z. & Lee, JA. Efficient hardware accelerators for *k*-nearest neighbors classification using most significant digit first arithmetic. *J Supercomput* **81**, 23 (2025). https://doi.org/10.1007/s11227-024-06466-2
11. Junior, J.C.V.S., Balza, M., Silva, S.N. *et al.* Optimizing Tactile Feedback Devices with Fixed-Point Computation on FPGA. *Circuits Syst Signal Process* **44**, 3826–3864 (2025). https://doi.org/10.1007/s00034-025-02994-1
12. Bispo, B.C., de Oliveira Neto, J.R. & Lima, J.B. Hardware Architectures for Computing Eigendecomposition-Based Discrete Fractional Fourier Transforms with Reduced Arithmetic Complexity. *Circuits Syst Signal Process* **43**, 593–614 (2024). https://doi.org/10.1007/s00034-023-02493-1
13. Ramaiah, M., Ravi, V., Chandrasekaran, V. *et al.* An efficient iterative pseudo point elimination technique to represent the shape of the digital image boundary. *Multimed Tools Appl* **83**, 85899–85915 (2024). https://doi.org/10.1007/s11042-024-20183-1
14. Banks, J., Garza Vargas, J. & Srivastava, N. Global Convergence of Hessenberg Shifted QR I: Exact Arithmetic. *Found Comput Math* (2024). https://doi.org/10.1007/s10208-024-09658-7

15. Belorgey, M.G., Carpov, S., Deforth, K. *et al.* Manticore: A Framework for Efficient Multiparty Computation Supporting Real Number and Boolean Arithmetic. *J Cryptol* **36**, 31 (2023). https://doi.org/10.1007/s00145-023-09464-4.
16. Ganesh, C., Nitulescu, A. & Soria Vazquez, E. Rinocchio: SNARKs for Ring Arithmetic. *J Cryptol* **36**, 41 (2023). https://doi.org/10.1007/s00145-023-09481-3
17. Ozaki, K., Mukunoki, D. & Ogita, T. Extension of accurate numerical algorithms for matrix multiplication based on error-free transformation. *Japan J. Indust. Appl. Math.* **42**, 1–20 (2025). https://doi.org/10.1007/s13160-024-00677-z
18. Majumdar, A., Avishek, K. Assessing heavy metal and physiochemical pollution load of Danro River and its management using floating bed remediation. *Sci Rep* **14**, 9885 (2024). https://doi.org/10.1038/s41598-024-60511-x
19. Mao, Q., Gao, Y., Fan, J. *et al.* Risk assessment of deep-sea floating offshore wind power projects based on hesitant fuzzy linguistic term set considering trust relationship among experts. *Environ Monit Assess* **196**, 405 (2024). https://doi.org/10.1007/s10661-024-12582-6
20. Parsons, L., Deng, G. & Ross, R. Detecting image edges in IOT nodes without FPU. *Multimed Tools Appl* **83**, 31161–31175 (2024). https://doi.org/10.1007/s11042-023-16672-4
21. Keshavarz, S., Reshadinezhad, M.R. & Moghimi, S. T-count and T-depth efficient fault-tolerant quantum arithmetic and logic unit. *Quantum Inf Process* **23**, 245 (2024). https://doi.org/10.1007/s11128-024-04456-0
22. Adiyaman, M.Y., Baskaya, F. FPGA implementation of double-head SalsaNext: a CNN-based model for LiDAR point cloud segmentation. *J Real-Time Image Proc* **22**, 78 (2025). https://doi.org/10.1007/s11554-025-01643-9
23. Sahraneshinsamani, N., Catalán, S. & Herrero, J.R. Mixed-precision pre-pivoting strategy for the LU factorization. J Supercomput 81, 87 (2025). https://doi.org/10.1007/s11227-024-06523-w
24. Alzaqebah, M., Ahmed, E.A.E. Accelerated fuzzy min–max neural network and arithmetic optimization algorithm for optimizing hyper-boxes and feature selection. *Neural Comput & Applic* **36**, 1553–1568 (2024). https://doi.org/10.1007/s00521-023-09131-6
25. Karner, C., Kazeev, V. & Petersen, P.C. Limitations of neural network training due to numerical instability of backpropagation. *Adv Comput Math* **50**, 14 (2024). https://doi.org/10.1007/s10444-024-10106-x
26. Gözükırmızı, C. Pure quadratization and solution of ordinary differential equations by probabilistic evolution theory with concurrent computation of coefficients using exact arithmetic. *J Math Chem* **62**, 654–680 (2024). https://doi.org/10.1007/s10910-023-01563-8
27. Wang, Z., Wan, S. & Wei, L. Optimized octree codec for geometry-based point cloud compression. *SIViP* **18**, 761–772 (2024). https://doi.org/10.1007/s11760-023-02803-9
28. Liu, J., Wang, Y., Gao, J. *et al.* pSpMv: precision-based sparse matrix partition and SpMV optimization. *CCF Trans. HPC* **6**, 549–565 (2024). https://doi.org/10.1007/s42514-024-00195-x
29. Ryabogin, D. On bodies floating in equilibrium in every orientation. *Geom Dedicata* **217**, 70 (2023). https://doi.org/10.1007/s10711-023-00807-w
30. Chen, D., Shen, J., Huang, C. *et al.* An empirical study of error-free transformations for enhancing mathematical function precision. *CCF Trans. HPC* (2025). https://doi.org/10.1007/s42514-025-00214-5
31. Lei, X., Gu, T. & Xu, X. ddRingAllreduce: a high-precision RingAllreduce algorithm. *CCF Trans. HPC* **5**, 245–257 (2023). https://doi.org/10.1007/s42514-023-00150-2
32. Yin, Y., Xu, J., Dong, J. *et al.* Cuda-based parallel dual-grid material point method for simulating bimetallic coining process. *Comp. Part. Mech.* (2025). https://doi.org/10.1007/s40571-025-00955-8
33. Caro, M., Abella, J. Energy-Efficient Object Detection: Impact of Weight Clustering for Different Arithmetic Representations. *J Sign Process Syst* **96**, 287–300 (2024). https://doi.org/10.1007/s11265-024-01917-8
34. Cui, M., Xu, J., Zhou, Y. *et al.* PESA: error sensitivity analysis tool for floating-point computational programs. *J Supercomput* **81**, 477 (2025). https://doi.org/10.1007/s11227-025-06962-z
35. Varin, V.P. Rational Arithmetic with a Round-Off. *Comput. Math. and Math. Phys.* **64**, 1143–1158 (2024). https://doi.org/10.1134/S0965542524700398
36. Sridhar, C., Kanhe, A. Approximate Floating Point Precise Carry Prediction Adder for FIR Filter Applications. *Circuits Syst Signal Process* **43**, 6487–6509 (2024). https://doi.org/10.1007/s00034-024-02760-9
37. Shafiei, M., Daryanavard, H. & Hatam, A. Scalable and custom-precision floating-point hardware convolution core for using in AI edge processors. *J Real-Time Image Proc* **20**, 94 (2023). https://doi.org/10.1007/s11554-023-01352-1