Check for updates

ACM DIGITAL LIBRARY | Association for Computing Machinery | acm open

Latest updates: https://dl.acm.org/doi/10.1145/3772673.3772698

RESEARCH-ARTICLE

Hardware-Efficient Activation Approximation based on Error-Sensitivity Analysis for Deep Neural Networks

JUHYUK AHN, Hanyang University, Seoul, South Korea

KWANGRAE KIM, Hanyang University, Seoul, South Korea

CHANHOON KIM, Hanyang University, Seoul, South Korea

SOO-MIN RHO, Hanyang University, Seoul, South Korea

KI-SEOK CHUNG, Hanyang University, Seoul, South Korea

Open Access Support provided by:

Hanyang University

PDF Download
3772673.3772698.pdf
26 March 2026
Total Citations: 0
Total Downloads: 19

Published: 16 March 2026

Citation in BibTeX format

ACMLC 2025: 2025 7th Asia Conference on Machine Learning and Computing (ACMLC)
July 25 - 27, 2025
Hong Kong, China

https://doi.org/10.1145/3772673.3772698
ISBN: 9798400718816

Hardware-Efficient Activation Approximation based on Error-Sensitivity Analysis for Deep Neural Networks

Juhyuk Ahn
Department of Electronic Engineering
Hanyang University
Seoul, Republic of Korea
juhuyk123@hanyang.ac.kr

Kwangrae Kim
Department of Electronic Engineering
Hanyang University
Seoul, Republic of Korea
kksilver91@hanyang.ac.kr

Chanhoon Kim
Department of Electronic Engineering
Hanyang University
Seoul, Republic of Korea
kch1103@hanyang.ac.kr

Soo-Min Rho
Department of Electronic Engineering
Hanyang University
Seoul, Republic of Korea
smrho@hanyang.ac.kr

Ki-Seok Chung*
Department of Electronic Engineering
Hanyang University
Seoul, Republic of Korea
kchung@hanyang.ac.kr

Abstract

Implementing hardware units for non-linear activation functions is challenging due to their distinct characteristics. While approximation methods offer a promising solution, conventional approaches such as CORDIC and Chebyshev polynomials suffer from high latency, and piecewise linear (PWL) methods require large lookup tables (LUTs) to achieve acceptable accuracy. Furthermore, existing methods often overlook the varying impact of approximation errors across input regions.

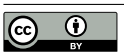
This paper proposes a hardware-efficient approximation method based on symmetric PWL approximation with error compensation guided by error-sensitivity analysis. A symmetry-aware base function is first constructed using PWL approximation. Then, only the difference between this base function and the target function is selectively compensated in high error-sensitivity regions using a lightweight error compensation module. This selective compensation enables accurate approximation across various non-linear functions using significantly fewer LUTs.

Synthesized with Synopsys Design Compiler and UMC 28nm libraries, the proposed design achieved over 90% LUT area reduction per function compared to uniform PWL, with only 1.22%p average accuracy loss. Experimental results confirm that the method delivers scalable and flexible activation function computation for resource-constrained hardware.

CCS Concepts

• **Computer systems organization** → **Embedded systems**; • **Hardware** → *Arithmetic and datapath circuits*; • **Computing methodologies** → *Artificial intelligence*.

*Corresponding author.



This work is licensed under a Creative Commons Attribution 4.0 International License.
ACMLC 2025, Hong Kong, China
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1881-6/25/07
https://doi.org/10.1145/3772673.3772698

Keywords

hardware acceleration, deep neural networks, activation function, piecewise linear approximation, error-sensitivity analysis, lookup table optimization

ACM Reference Format:
Juhyuk Ahn, Kwangrae Kim, Chanhoon Kim, Soo-Min Rho, and Ki-Seok Chung. 2025. Hardware-Efficient Activation Approximation based on Error-Sensitivity Analysis for Deep Neural Networks. In *2025 7th Asia Conference on Machine Learning and Computing (ACMLC) (ACMLC 2025), July 25–27, 2025, Hong Kong, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3772673.3772698

1 Introduction

With the emergence of artificial intelligence technology, deep neural networks (DNNs) have achieved great success in natural language processing, computer vision, etc. An activation function that provides non-linearity to a DNN is one of the keys to such success. With DNN architectures becoming increasingly complex, a wide variety of activation functions[5] have been proposed to address task-specific challenges. However, implementing a diverse set of activation functions in hardware is challenging, especially in resource-limited hardware (e.g., edge devices or neural processing units). Implementing hardware to compute multiple activation functions will require different computational structures for each function, resulting in increased hardware complexity and area overhead.

To address this, researchers have explored approximation-based methods. Conventional methods, such as CORDIC[10] and Chebyshev polynomials[7], may share computational units across functions to reduce the circuit area, but suffer from high latency due to their iterative nature or high-degree polynomial evaluations. Piecewise Linear (PWL) approximation enables low-latency computation by segmented linear approximation. However, it requires either a large amount of lookup tables (LUTs) in the case of uniform segmentation or function-specific logic circuits in non-uniform segmentation. Symmetry-aware PWL optimizations effectively reduce LUT size by exploiting the symmetric property of certain activation functions. However, these optimizations can be applied only to functions that feature such symmetry. Furthermore, some of the

Table 1: Comparison of Activation Function Approximation Techniques

Approximation Method	Area	Latency	Flex.	Aware
Conventional Approaches				
CORDIC	Low	Medium	✗	✗
Chebyshev Polynomial	Medium	High	✓	✗
LUT-based PWL Approaches				
Non-uniform Segment	High	Low	✓	✗
Uniform Segment	High	Low	✓	✗
Symmetry	Low	Low	✗	✗
Proposed Method	**Low**	**Low**	✓	✓

Note: Area indicates the relative cost per function for comparable accuracy. **Flex.** indicates support for multiple activation functions. **Aware** indicates consideration of error-sensitivity for approximation.

prior works[6, 11] achieved high accuracy but require fine-tuning, which imposes additional design effort.

Further, most of these methods did not consider that approximation errors impact model accuracy unevenly across different input ranges. In the proposed method, an error compensation module is selectively activated only for input ranges where approximation errors cause a significant drop in model accuracy. To quantify this phenomenon, we define the degree to which activation approximation errors impact model accuracy as **error-sensitivity**. Table 1 demonstrates that methods without considering error-sensitivity result in inefficient approximation. They suffer from high area overhead, increased latency, or inability to support multiple activation functions simultaneously.

In this paper, we propose a novel hardware-efficient approximation method for various activation functions, with error compensation selectively applied to input intervals with high error-sensitivity. To demonstrate our approach, suppose we want to design hardware that approximates multiple non-linear function variants (e.g., the GELU-like family shown in Table 2). We call such functions the target activation functions. Our method claims that the function variants can be largely approximated by the base functions such as GELU [3] or Swish [2], and differences between them will occur only in a small number of input intervals. Instead of approximating each activation function individually, first, we approximate only the base function via symmetric PWL, and apply error compensation selectively to the input intervals where the differences cause a significant drop in model accuracy. To identify these critical input intervals, we propose an error-sensitivity analysis method. To this end, we introduce a three-phase pipeline structure as follows: **(1)** Analyze error-sensitivity via a method called *Activation Error Injection*, **(2)** Quantify the model accuracy degradation caused by replacing the target activation with the base function and evaluate the functional difference between them, and **(3)** Apply error compensation methods to the common compensation region with high error-sensitivity. Our proposed hardware architecture offers scalable and low-overhead support for a diverse set of activation functions. When synthesized using Synopsys Design Compiler with UMC 28nm target libraries, the design achieves an impressive 90% reduction in the LUT usage per function compared to the design for uniformly segmented PWL, while maintaining a minimal average accuracy degradation of only **1.22%p** when approximating 16 GELU-like activation functions.

2 Related Works & Motivation

2.1 LUT-Based Approximation Methods

Piecewise-linear (PWL) approximation via look-up tables (LUTs) is widely adopted due to its implementation efficiency. PWL algorithms approximate a nonlinear function by dividing the input space into intervals and fitting linear segments based on precomputed parameters. There are two main segmentation strategies.

- **Uniform segmentation:** The input range is split into equally sized segments, which simplifies control logic. However, maintaining high approximation accuracy often requires many segments, leading to a large number of LUTs.
- **Non-uniform segmentation:** Finer segments are granted in areas of high curvature for higher resolution. However, the control logic becomes more complex due to the non-uniform segmentation.

Exploiting Functional Symmetry: To achieve a good trade-off between implementation simplicity and the required LUT size in PWL, functional symmetry may be exploited with uniform segmentation methods. Functions such as Swish and GELU are inherently symmetrical, satisfying the property $f(x) = x + f(-x)$, which implies that the LUT needs to store the parameters only for the positive input range. The output for the negative input is then derived by mirroring, effectively reducing the LUT size by half.

2.2 Motivation: Limitations and Observation

Various approximation methods have been proposed to find good trade-offs among accuracy, area, latency, and flexibility. However, most existing methods have largely overlooked the impact of *error-sensitivity* on model accuracy, which is not uniform across input ranges. Therefore, analyzing error-sensitivity helps to identify critical input regions for error compensation, enabling more efficient and accurate approximations.

Moreover, activation functions such as GELU, Swish, and their variants have similar shapes, and their differences are concentrated in a small number of input intervals. This allows error compensation to be selectively applied only where it is most needed, as effectively identified by the proposed error-sensitivity analysis.

These insights motivate our design, using a symmetry-aware base function (e.g., GELU or Swish) implemented with compact hardware combined with a lightweight error compensation module for high flexibility and low overhead in approximating diverse activation functions.

3 Activation Function Approximation based on Error-Sensitivity

The key problem to solve is to identify a common compensation region where the error compensation should be applied across multiple activation functions, and the proposed method consists of three phases as follows: **(1)** Analyze error-sensitivity via a method

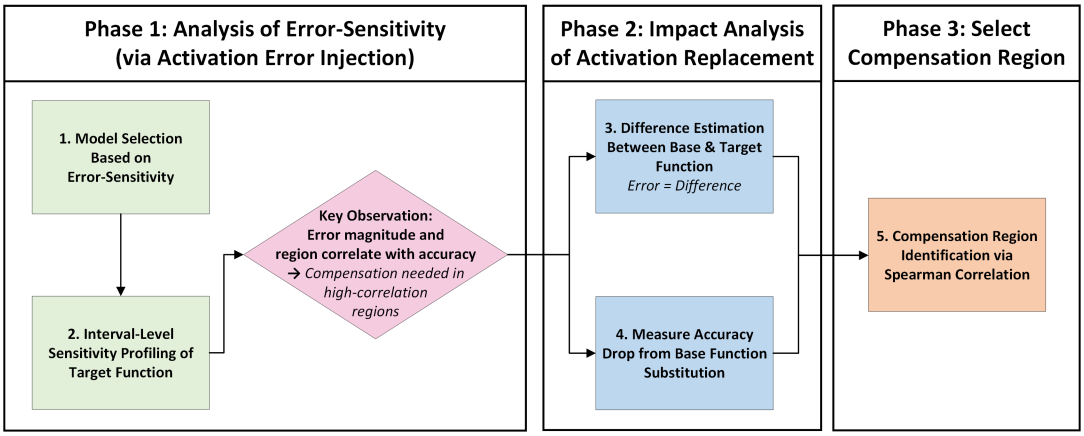
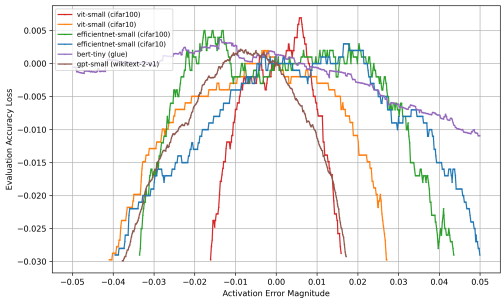
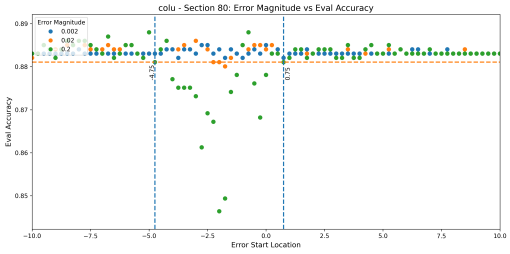


Figure 1: Overview of the three-phase compensation region identification methodology.



(a) Error-Sensitivity across various models.



(b) Accuracy degradation from interval-wise Activation Error Injection over in `colu`, one of GELU-family.

Figure 2: Impact of activation errors on model evaluation accuracy.

called *Activation Error Injection*, **(2)** Quantify the accuracy drop from replacing the target activation with the base function and the functional difference between them, and **(3)** Identify the common compensation region for error compensation by error-sensitivity analysis. The three phases are shown in Fig. 1, and they are executed in a pipelined fashion.

3.1 Phase 1: Error-Sensitivity Analysis

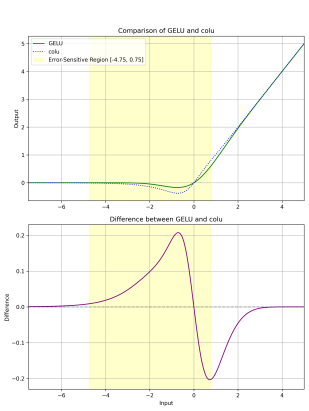
Model Selection Based on Error-Sensitivity: To estimate the error-sensitivity, we inject errors into the activation outputs during inference. We call this step *Activation Error Injection*. In Fig. 2a, the x-axis represents the magnitude of the injected error, while the y-axis shows the corresponding accuracy loss. A large accuracy drop (i.e., a more negative y-value) when the x-value is close to zero indicates that the model is highly sensitive to small errors, reflecting high error-sensitivity. Among the models that we evaluated, `ViT-Small`[1] on `CIFAR-100`[4] exhibited the highest error-sensitivity. Accordingly, this model was selected as the test case and was used in all subsequent experiments.

Interval-Level Sensitivity Profiling: To analyze the error-sensitivity of GELU-like activation functions, we also applied Activation Error Injection by uniformly dividing the input interval $[-10, 10]$ and injecting errors of various magnitudes into their outputs. Fig. 2b presents the results for `colu`[9], one of GELU-like functions. The orange line indicates the baseline accuracy, and a bigger deviation from the baseline indicates a larger degradation. With the largest error magnitude of 0.2, the accuracy degradation occurs mainly at the interval $[-4.75, 0.75]$. Moreover, some sub-intervals within this interval exhibit a significantly larger impact on accuracy. This tendency is similarly observed for all GELU-like functions, and the interval $[-3, 1]$ is identified as the critical input interval with the most significant impact on accuracy.

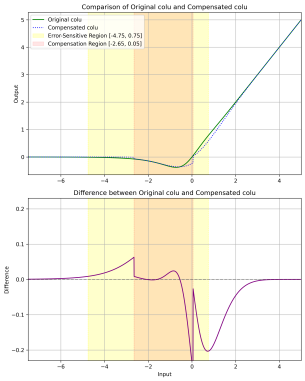
Key Observation: According to the error-sensitivity via *Activation Error Injection*, the model accuracy depends on both the magnitude of the error and the input intervals. Therefore, error compensation efforts should be applied to the intervals with high error-sensitivity, where even small errors result in significant accuracy degradation.

3.2 Phase 2: Estimating the Impact of Activation Replacement

Based on the observation from Phase 1, compensation should be applied to high error-sensitive intervals. This implies that the target function can be approximated by the base function approximation for input ranges with low error-sensitivity, and we need to focus only on the input range with high error-sensitivity for difference



(a) Approximation error between GELU and colu.



(b) Comparison of original and compensated colu based on GELU.

Figure 3: Visualization of approximation errors for colu.

compensation. To identify these high error-sensitive regions, it is necessary to measure the error magnitude and the corresponding accuracy drop when replacing the target GELU-like functions with the base function.

Difference Estimation Between Base & Target Functions: When the base functions are used instead of other GELU-like functions, the differences due to the replacement must be appropriately compensated. Such errors can be easily calculated as the difference between the two functions, as illustrated in Fig. 3a. The yellow region indicates the error-sensitive interval of colu identified in Phase 1. Since intervals outside this area have minor effects on accuracy, compensation is not essential.

Measure Accuracy Drop from Base Function Substitution: We measured the accuracy degradation of the error-sensitive model when each GELU-like activation function was replaced with the base functions (GELU, Swish). The results are presented in Table 2. This accuracy degradation arises from the approximation errors previously identified in the error calculation step.

3.3 Phase 3: Compensation Region Selection

Within the identified range $[-3, 1]$ in Phase 2, we perform a hierarchical greedy search to determine the interval where the compensation will be commonly beneficial for the GELU-like family. We call this interval *common compensation region*. This searching method recursively explores candidate intervals, progressively refining the search granularity by a factor of 1/10 at each of the four hierarchical levels, enabling increasingly detailed examination.

At each level, the search focuses on the region with the highest Spearman correlation[8] value, which then becomes the candidate for a finer-grained search and the correlation evaluation in the subsequent level. The correlation values are computed separately based on three criteria: GELU-only, Swish-only, and their average. Finally, among these candidates, the one selected based on the criterion that achieves the highest accuracy is chosen. The detailed results are presented in Section 5.1. The orange region in Fig. 3b represents the final selected common compensation region. In this selected

Table 2: Accuracy drop when replacing original activation with GELU or Swish.

Original Function	Accuracy Drop When Replacing With:	
	GELU	Swish
hard_swish	0.083	0.117
loglogish	0.076	0.029
colu	0.047	0.338
eanaf	0.054	0.830
modified_silu	0.096	0.062
expexpish	0.065	0.866
mish	0.075	0.020
hardelish	0.041	0.350
elish	0.034	0.691
serf	0.072	0.058
suish	0.069	0.101
rectified_exponential_unit	0.069	0.104
tanhexp	0.072	0.075
lalu	0.037	0.253

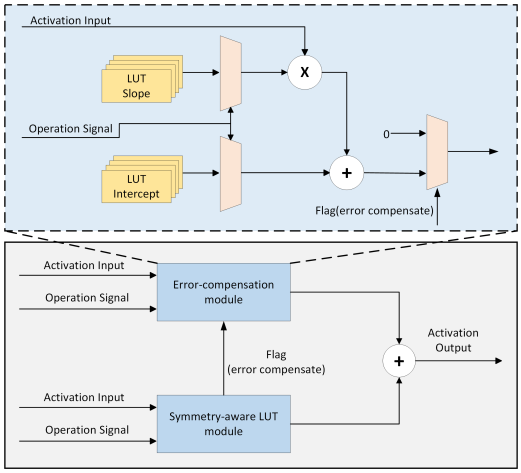


Figure 4: Proposed design with an error compensation module.

common compensation region, error compensation is applied using a simple linear operation, with the parameters chosen to minimize the mean absolute error (MAE) with respect to the original function. It can be observed that the error near -2, which was identified in Fig. 2b as the most error-sensitive region, is significantly reduced, effectively avoiding model accuracy degradation.

4 Hardware Design

We implement the compensation module based on error-sensitivity analysis as introduced in Section 3. The design consists of a symmetry-aware uniform PWL approximation unit for the base function and a lightweight error compensation module, as shown in Fig. 4. This allows efficient support for multiple activation functions while

Table 3: Comparison of Accuracy Drop for various LUT Sizes and Correlation Strategies.

Metric / Strategy	Base LUT = 8 segments			Base LUT = 16 segments			Base LUT = 32 segments		
	Swish Corr.	GELU Corr.	Average Corr.	Swish Corr.	GELU Corr.	Average Corr.	Swish Corr.	GELU Corr.	Average Corr.
Avg. Acc. Drop	0.0122	0.0139	0.0152	0.0121	0.0151	0.0173	0.0113	0.0155	0.0174
Max. Acc. Drop	0.0297	0.0575	0.0466	0.0347	0.0565	0.0446	0.0297	0.0515	0.0446

Table 4: Comparison of Area, Power, and Latency (UMC 28nm, 800 MHz, FP32 precision)

Method	Area (μm^2)	Power (mW)	Latency (cycles)	Supported Functions
CORDIC	8845.1	2.40	9	GELU
Chebyshev	5539.5	3.99	10	GELU
Proposed	10 798.6	3.83	1	GELU, `colu`

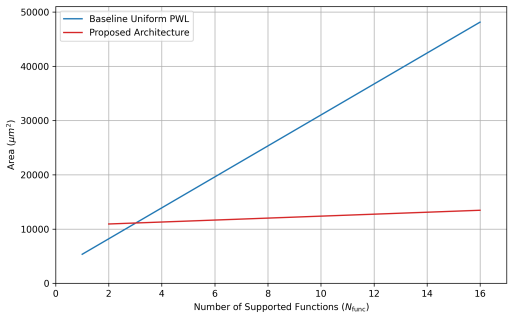


Figure 5: Comparison of area scaling.

minimizing hardware overhead. **Symmetry-aware LUT module** approximates a base function (e.g., GELU or Swish) using a compact LUT and symmetry-aware logic. An address generator detects whether the input falls within the predefined compensation region. **Error-compensation module** is triggered when compensation is needed; it retrieves precomputed parameters from a small LUT and applies a corresponding linear correction term. **Activation output** is computed as a simple sum of the outputs from the symmetry-aware LUT module and the error-compensation module.

5 Evaluation

We evaluated the proposed design in terms of: (1) accuracy preservation with error compensation, (2) hardware efficiency, and (3) scalability for multi-function support.

5.1 Accuracy Evaluation

The accuracy evaluation was conducted by fine-tuning each GELU-like activation on the ViT-Small model with the CIFAR-100 dataset, as specified in Section 3.2. A total of 16 functions were considered, including the 14 GELU-like variants in Table 2 and the two base functions, GELU and Swish. The function that achieves higher model accuracy was selected as the base function for each GELU-like function. We applied three correlation strategies—Swish-based, GELU-based, and the average correlation—as described in Section 3.3. Each strategy was evaluated using different LUT sizes of 8, 16, and 32 segments. Table 3 shows that, as the number of LUT segments increases, the average accuracy drop decreases. However, it does not significantly improve the maximum accuracy degradation. Moreover, the benefit in average accuracy is relatively small compared to the cost of doubling the LUT size, which is one of the key drawbacks of PWL approaches. With regard to the compensation region strategies, the Swish-based correlation consistently yields the lowest accuracy drop across all LUT sizes. The best result is achieved with a LUT size of 32 and the Swish-based compensation, showing an average accuracy drop of 0.0113 and a maximum drop of 0.0297. However, using the same strategy with only 8 segments still maintains competitive accuracy (average: 0.0122, max: 0.0297), making it the most hardware-efficient option by consuming only one-fourth of the LUT resources.

5.2 Hardware Evaluation

We synthesized three designs: CORDIC for GELU, Chebyshev for GELU, and our proposed design for both GELU and `colu`, using Synopsys Design Compiler with the UMC 28nm target libraries. All designs operate at 800MHz. All computations are carried out with 32-bit floating-point (FP32) numbers. Their characteristics are summarized in Table 4. The **CORDIC** design exhibits low power consumption (2.40mW). However, its iterative processing results in a high latency of 9 cycles for GELU, leading to a significant energy consumption. The circuit area occupies 8845.1 μm^2. The **Chebyshev** design takes up a small area with 5539.5 μm^2 mainly because it operates iteratively. Despite its compact size, its scalability is limited by the need for different parameters for each function. It consumes 3.99mW of power, with a notably high energy consumption due to the 10-cycle latency required for GELU computation. In contrast, **the proposed design** supports two functions (GELU and COLU) with a total area of 10798.6 μm^2, and achieves the best area efficiency per function. Its power consumption is 3.83mW, and all the functions are completed in a single cycle, which means that the energy consumption is significantly smaller. These benefits stem from the lightweight compensation module. The area efficiency is expected to further improve with the addition of more supported functions, as detailed discussion in Section 5.3.

5.3 Scalability

Fig. 5 illustrates the scalability of each method in terms of the circuit area as the number of target activation functions increases.

The baseline uniform PWL method incurs significant LUT overhead for each added activation function, whereas the proposed method requires only a small compensation LUT. Specifically, the proposed design takes up substantially less area than the baseline, occupying only 180 μm^2 per function compared to 2800 μm^2 for the baseline. For each additional function, more than 90% of the LUT area can be saved. This scalability is very important for effective implementation on resource-constrained hardware that needs to support multiple activation functions.

6 Conclusion

This paper proposed a hardware design that combines a module for symmetric PWL approximation with that for selective error compensation, enabling efficient multi-function activation approximations. It achieves minimal accuracy loss of only 1.22%p while reducing LUT usage by over 90%. This enables scalable and hardware-efficient deployment of deep neural networks on resource-constrained hardware platforms.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2024-00409492).

References

[1] Alexey Dosovitskiy. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).
[2] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. 2018. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks* 107 (2018), 3–11.
[3] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* (2016).
[4] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning multiple layers of features from tiny images*. Technical Report. University of Toronto.
[5] Vladimír Kunc and Jiří Kléma. 2024. Three decades of activations: A comprehensive survey of 400 activation functions for neural networks. *arXiv preprint arXiv:2402.09092* (2024).
[6] Haodong Lu, Qichang Mei, and Kun Wang. 2023. Auto-lut: Auto approximation of non-linear operations for neural networks on fpga. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.
[7] John C. Mason and David C. Handscomb. 2002. *Chebyshev Polynomials*. CRC press.
[8] Charles Spearman. 1904. The Proof and Measurement of Association Between Two Things. *The American Journal of Psychology* 15, 1 (1904), 72–101.
[9] Advait Vagerwal. 2021. Deeper learning with colu activation. *arXiv preprint arXiv:2112.12078* (2021).
[10] Jack E Volder. 1959. The CORDIC trigonometric computing technique. *IRE Transactions on electronic computers* 3 (1959), 330–334.
[11] Joonsang Yu, Junki Park, Seongmin Park, Minsoo Kim, Sihwa Lee, Dong Hyun Lee, and Jungwook Choi. 2022. NN-LUT: Neural approximation of non-linear operations for efficient transformer inference. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 577–582.