International Journal of Electrical and Computer Engineering (IJECE)
Vol. 16, No. 2, April 2026, pp. 945~963
ISSN: 2088-8708, DOI: 10.11591/ijece.v16i2.pp945-963

Optimizing neural networks: a comparative study of activation functions in deep learning

Ahmed Mobarki, Abdullah Sheikh
Department of Computer Science, Taif University, Taif, Saudi Arabia

Article Info

Article history:

Received May 5, 2025
Revised Nov 24, 2025
Accepted Jan 15, 2026

Keywords:

Activation functions
Deep learning
Gaussian error linear unit
Mish
Neural networks
Rectified linear unit
Swish

ABSTRACT

Activation functions play a pivotal role in deep learning (DL) models, thus shaping their learning capabilities, convergence behavior, and generalization performance. However, the selection of activation functions without systematic evaluation in many applications has limited the model's performance. Inappropriate activation functions may cause gradients to shrink or blow-up during backpropagation, thereby affecting effective learning. To conquer this problem, this paper provides a novel comprehensive empirical investigation of nine activation functions, including traditional functions like rectified linear unit (ReLU), Sigmoid, Tanh, and exponential linear unit (ELU), and modern nonlinearities like Swish, Mish, Gaussian error linear unit (GELU), and smooth maximum unit (SMU). In the proposed methodology, these nine activation functions are evaluated within two prominent neural network architectures, namely convolutional neural networks (CNNs) and multi-layer perceptrons (MLPs), across benchmark datasets, namely CIFAR-10, CIFAR-100, and MNIST. The evaluation criteria include validation accuracy, loss, training time, and gradient stability. Experimental results proved that GELU activation function improved MLP accuracy to 98.03% and CNN accuracy to 93.82% while maintaining stable gradients and low loss values of 0.088 and 0.221, respectively. These findings provided practical guidelines for selecting activation functions suited to specific task complexities and model depths, contributing to the design of more efficient and accurate DL systems.

Corresponding Author:

Ahmed Mobarki
Department of Computer and Information Technology, Taif University
Taif, Kingdom of Saudi Arabia.
Email: ahmed.mobarki91@gmail.com

1. INTRODUCTION

Deep learning (DL) has emerged as a paradigm shift in artificial intelligence (AI) in recent years, significantly improving capabilities in fields like medical diagnostics, image recognition, recommendation systems, autonomous driving, and natural language processing. Neural networks, a stratified architecture designed to mimic the cognitive functions of the human brain, are essential for DL models. Activation functions influence nearly all facets of a DL model's training and performance. These processes govern neural activation and affect information propagation during inference. Also, activation functions have the reverse effect during gradient-based optimization. Ding *et al.* [1] assisted in controlling the gradient flow, facilitating the training of deeper models. The activation function, which adds nonlinearity to the model, is a crucial component of this architecture. Without it, a neural network, no matter how deep, behaves like a linear regression model, which severely limits its ability to depict complex data relationships [2].

The influence of the activation function also extends to the accuracy and generalization of the network, along with its convergence rate and learning stability [3]. Over the past decades, neural networks have used Tanh and Sigmoid activation functions. Tanh outputs in the range of (1,-1) help to maintain the data centered around zero, and Sigmoid, which transfers the input values to the range [0,1], can be used in binary classification problems. The Sigmoid and Tanh activation with its function $f(x)$ and the input (x) are given as (1), (2),

$$f(x) = 1/(1 + e^{-x}) \tag{1}$$

$$f(x) = tanh(x) \tag{2}$$

where, (e) is the exponential factor. Although both methods are initially popular, they are beset with the problem of vanishing gradients, thus hindering effective weight updating in deeper networks [4]. A breakthrough in DL comes with the introduction of the rectified linear unit (ReLU), given by the function $f(x)$ as (3),

$$f(x) = max(0, x) \tag{3}$$

By having non-zero gradients, the ReLU activation function alleviates the vanishing gradient issue for positive inputs and allows deeper networks to converge faster [5].

ReLU introduces the dying ReLU problem in which neurons with negative inputs always output zero and are inactive during training [6]. Parametric ReLU (PReLU) and Leaky ReLU are created to reduce these limitations, offering a non-zero gradient for negative inputs. Further, exponential linear units (ELUs) and scaled ELUs (SELU) are presented to improve gradient flow and enable self-normalizing action [7], [8].

$$f(x) = \begin{cases} xifx > 0 \\ \alpha(e^x - 1)else \end{cases} \tag{4}$$

$$f(x) = \begin{cases} \lambda xifx > 0 \\ \lambda\alpha(e^x - 1)else \end{cases} \tag{5}$$

Where, (α) is the factor utilized to control the activation function regarding negative impact, and (λ) is the scaling factor. Recent research has looked at adjustable, non-monotonic activation functions. Regarding Google academics and the input (x) with its function $f(x)$, Swish is characterized as (6),

$$f(x) = x * \text{sigmoid}(x) \tag{6}$$

Swish combines the smoothness of Sigmoid with the unbounded nature of ReLU, thus improving gradient propagation and achieving strong performance in image classification tasks [9]. Building upon this, Mish is defined as (7),

$$f(x) = x * tanh[ln(1 + e^x)] \tag{7}$$

Mish is introduced to further enhance learning dynamics and smoothness. Mish has shown superior performance, particularly in convolutional architectures [10], [11]. In addition to Swish and Mish, other modern activation functions have gained attention. Gaussian error linear unit (GELU), frequently used in models like bidirectional encoder representations from transformers (BERT) and generative pre-trained transformer (GPT), is defined as (8),

$$f(x) = x * \varphi(x) \tag{8}$$

Here, the cumulative distribution function of the standard normal distribution is represented as $\varphi(x)$. GELU offers both smooth transitions and stochastic regularization, thereby improving convergence in transformers and deep models [12], [13]. Another recent proposal, named smooth maximum unit (SMU), is defined as (9),

$$f(x) = x * tanh[\text{softplus}(x)] \tag{9}$$

Many DL systems still use ReLU as the default activation function, typically chosen without any testing despite these developments. Nevertheless, the network architecture, depth, dataset complexity, and training dynamics have a significant impact on the best activation function [14], [15]. As an illustration, a shallow multi-layer perceptron (MLP) can gain advantages from utilizing a single activation function, whereas a deep convolutional neural network (CNN) cannot. A comprehensive empirical examination of nine activation functions, including ReLU, Sigmoid, Tanh, ELU, SELU, Swish, Mish, GELU, and SMU, is carried out in the current work. This evaluation is driven by the conclusion presented before. In order to carry out the evaluation, two standard architectures, namely MLP and CNN, as well as two benchmark datasets, namely Canadian Institute for Advanced Research-10 (CIFAR-10) for natural picture classification and Modified National Institute of Standards and Technology (MNIST) for handwritten digit classification, are utilized. Validation accuracy, training loss, training time, and gradient stability are the four primary performance indicators that are taken into consideration within this framework. These indicators offer a multi-dimensional perspective on the performance of the function. The purpose of this research is to render evidence-centric recommendations for selecting activation functions based on the specific model and the requirements of the work. By way of its discoveries, this work bridges the gap betwixt theoretical advancements in activation function design and their practical implementation in actual DL processes. This work is a significant contribution to the field. The problems in existing research works regarding activation function in DL are listed as,

a. Most of the existing works compared only a few activations (*i.e.*, Sigmoid, Tanh, and ReLU) and ignored the newer activations, such as GELU, Mish, Swish, or SMU.
b. The existing works failed to measure the training stability, convergence speed, gradient flow, and computational efficiency.
c. In many conventional works, the comparisons were restricted to a single model type, thus limiting the generalization of findings.

Based on the problem statements, the research questions of the proposed model are developed and are listed:

RQ1: How does the performance of different activation functions vary across neural network architectures like MLP and CNN?
RQ2: Which activation functions (*i.e.*, classical or modern) deliver the best balance between accuracy, convergence speed, and training stability in DL models?
RQ3: How do activation functions differ regarding the gradient flow and susceptibility to vanishing or exploding gradients during training?

By considering the research questions and problem statements, some contributions are developed and explained below,

a. In the proposed methodology, activation functions, namely Sigmoid, ReLU, Tanh, ELU, Swish, Mish, GELU, and SMU, are compared for improved performance.
b. The proposed model measures the training stability, convergence speed, gradient flow, and computational efficiency during performance analysis.
c. The comparison is done for the CNN and MLP neural networks to improve the generalization of findings.

The paper is structured as: The literature survey is conveyed in section 2, the proposed methodology is analyzed in section 3, the results and discussion are presented in section 4, and lastly, the proposed work is winded up in section 5 with future recommendations.

2. RELATED WORK

Many researchers looked at activation functions in deep neural networks, investigating both conventional and contemporary substitutes. Though their drawbacks, including vanishing gradients and dead neuron problems, were well documented [1], these problems happened because some activations created gradients to shrank exponentially during backpropagation in deeper architectures. Also, these drawbacks spoiled the network's potential to learn longer-range dependencies. Likewise, in ReLU-like activations, the dead neuron issue was a common problem, where neurons were inactive owing to zero gradients. Classical functions, namely Sigmoid, ReLU, and Tanh, were essential for network construction [2], [3] owing to their simplicity, easier implementation, and success across a variety of tasks. Likewise, ReLU was a default choice in several computer vision tasks due to its computational efficiency. Similarly, Sigmoid and Tanh were foundational in early neural network models owing to their smooth and bounded outputs. Researchers created functions like Leaky ReLU and PReLU to solve these problems. However, Leaky ReLU and PReLU enhanced gradient flow in deep architectures and were especially beneficial in acoustic models and ResNet-style CNNs [4], [5]. These modified variants allowed small and non-zero gradients even when the input was negative, preventing neurons from becoming inactive and mitigating their harsh cut-off at zero [6].

The paper [7] suggested the exponential linear unit (ELU), which accelerated convergence by keeping mean activations close to zero and diminishing the "internal covariate shift" and enabling networks to train more effectively. Also, negative saturation of ELUs helped push mean activations toward zero without sacrificing capacity. The scaled exponential linear units (SELU) broadened this with self-normalizing features. Thus, the networks automatically regulated their activations. SELU removed the need for explicit normalization layers in some architectures [8]. Also, swish and its near cousin Sigmoid linear unit (SiLU) provided smooth, non-monotonic behavior and were demonstrated to beat ReLU in image classification and reinforcement learning activities. Their gradual transition for negative inputs and non-monotonicity enabled richer feature representation and smoother gradient flow. Similarly, the model provided enhanced optimization landscapes for tasks with high-dimensional input spaces [9], [10]. A further extension of this tendency, Mish, enhanced learning dynamics and generalization by means of its self-regularizing structure, thereby introducing an implicit bias towards more stable activations over the course of training [11]. Mish provided better outcomes in object detection, natural language processing (NLP), and other domains owing to its capability to balance smoothness and gradient strength. In transformer-based models like BERT, GELU was particularly strong in NLP and classification jobs since it included stochastic regularization and smoothness. This aided transformer layers in handling noisy, high-dimensional language embeddings more effectively, leading to superior performance [12], [13].

Based on these findings, several activation functions were methodically contrasted across different architectures and datasets in this work. Specific recommendations were provided on how to choose activation functions appropriate for certain DL tasks by means of their influence on accuracy, training time, and gradient stability, showing that activations were computationally budget-constrained [14]. Comparative research verified that these contemporary activations, such as Swish, Mish, and GELU, showed notable gains in both CNNs and MLPs over classical functions, particularly in terms of accuracy and convergence, thus diminishing overfitting problems and improving the stability in deeper architectures [15], [16]. Automated search methods were used to find new activation functions, supporting the power of Swish and encouraging data-driven activation discovery. Here, genetic programming and neural architecture search frameworks analyzed massive design spaces of activation, often searching swish patterns as a primary choice [17], [18]. Mishra's investigation of non-monotonic functions further underlined Mish's better stability over training stages, demonstrating that these functions maintained balanced gradient magnitudes throughout the optimization process and resulting in better generalization and consistent learning rates.

In reinforcement learning and function approximation, SiLU/Swish allowed more consistent convergence and smoother gradient propagation than conventional approaches, thereby making them well-suited for real-time decision-making agents [19]. On the other hand, some works contributed to defining the link between DL stability, weight initialization, and activation functions, thus establishing initialization techniques like He and Xavier for aligning activation variance with weight scaling. This improved the training depth limits [20]. Their ideas spurred the widespread use of methods like Xavier initialization, which were essential for training deeper models. Certain studies underlined how activation decisions affected deep convolutional networks and recommended a compromise between smoothness and processing efficiency, showing that the activations' mathematical complexity was weighted against their runtime performance while deploying latency requirement environments [21], [22]. Random and regularizing features of activation functions were addressed in other studies. Some studies addressed the interaction of stochastic elements, such as dropout and activation sparsity, revealing their influence on generalization and optimization. This showed that some activations synergized better with dropout, providing enhanced robustness against overfitting [23].

In certain studies, the SMU was meant to maintain training stability, thereby eliminating sharp transitions and saturated gradients. SMU provided a balanced trade-off between ReLU's sparsity and Tanh's smoothness [24]. Eventually, some researchers looked at how activation functions interacted with optimization techniques and supported stable learning in polynomial regression and picture classification settings, thus demonstrating that activation selection influenced optimizer performance and sensitivity to hyperparameter settings [25]. Recent studies illustrated the use of machine learning regarding nonlinear modeling and complex decision-making applications [26]. The data-driven intelligence system was combined with a practical system for physical-guided learning [27]. Meanwhile, scalable solutions were performed to enhance the resource-constrained performance [28]. Recent economic modeling studies also forecasted market uncertainty and price dynamics [29], [30]. Among the recent studies, the Gaussian process regression served as a strong baseline for uncertainty-aware modeling [31], [32]. In addition, the graphical and causality-based models had gained the interpretable modeling of multivariate dependencies [33], [34]. Finally, the ensemble and composite learning demonstrated superior performance over the financial and economic predictions [35], [36].

In summary, the earlier studies on the activation functions mainly focused on the traditional functions like Sigmoid, Tanh, and ReLU, with limited evaluation on Swish, Mish, GELU, and SMU. Thus, the gradient

behavior, convergence, and computational efficiency were affected. Also, inconsistent training settings limited the generalization. Hence, this study renders a comprehensive analysis of nine activation functions regarding MLP and CNN models on the CIFAR-10, MNIST, and CIFAR-100 datasets.

3. METHOD

The approach followed in this study attempts to give a comprehensive and balanced investigation of different activation functions in various neural network architectures and data sets. The major objective of the proposed model is to examine how the choice of activation affects the model accuracy, convergence speed, training stability, and robustness. Here, two well-known benchmark datasets, such as CIFAR-10 for object recognition and MNIST for handwritten digit recognition, are selected to ensure a heterogeneous evaluation environment. These datasets differ in complexity. Here, MNIST is simpler and grayscale, whereas CIFAR-10 is colored images with higher intra-class variability, thus allowing for performance assessment under various conditions. In the proposed methodology, two model architectures are designed: an MLP for the MNIST dataset and a CNN for the CIFAR-10. With the nine activation functions (*i.e.*, ReLU, Sigmoid, Tanh, ELU, SELU, Swish, Mish, GELU, and SMU), MLP and CNN are trained separately. For ensuring fairness in evaluation, all hyperparameters, including the number of epochs, batch size, and optimizer, are kept constant across experiments. Also, the training is done based on the Adam optimizer with its default learning rate of 0.001. All input features are normalized to the range of [0,1], [0,1], and [0,1].

3.1. Dataset selection

The two benchmark sets are chosen in this research to gauge the model's performance at different levels of complexity. The MNIST dataset [20], comprising 70,000 images of gray-scale digits written (28×28 pixels each) by hand, 60,000 samples for training purposes, and 10,000 samples for test purposes, is used to study MLPs. This dataset's comparatively low resolution and limited variability make it the best choice for assessing simpler architectures like MLPs. Also, MNIST validates the fundamental aspects of model design, including activation functions in fully connected layers. To test performance in a more complex framework, CNNs are tested using the CIFAR-10 dataset [30]. This dataset consists of 60,000 color images (32×32 pixels each) from 10 different categories, with 50,000 for training and 10,000 for testing. The CIFAR-10 dataset consists of higher visual complexity owing to color channels, intra-class variations, and cluttered backgrounds, making it suitable for evaluating activation performance in convolutional architectures. To increase the generalizability, the CIFAR-100 dataset is utilized. The CIFAR-100 dataset consists of 60,000 images (32×32 pixels each) with 100 classes grouped into 20 super classes. From the total images, 45,000, 5,000, and 10,000 images are utilized for training, validation, and testing the classifier model, respectively.

3.2. Network architectures

Two different types of neural network architectures, such as MLP and CNN, are used. The specifications of MLP and CNN are explained as follows,

3.2.1. Multi-layer perceptron

The MLP designed for MNIST classification comprises an output layer, an input layer, and two dense hidden layers. The MLP is chosen because it performs well on low-dimensional tasks. Here, the input layer (B) accepts the flattened 28×28 grayscale pixel values. Here, the input$(h)$is fed into MLP, and the processing of (B) is carried out as (10),

$$B = [h + \varpi] * m \tag{10}$$

Where, (ϖ, m) are the weight and bias values of the input (h), respectively. Also, the two dense hidden layers consist of 128 and 64 neurons. Likewise, dropout regularization is used in each layer to prevent any instances of overfitting. Softmax activation (η) is utilized in the final output layer (S) for situations involving multi-class classification issues.

$$S = \{[\sum B + \varpi] * m\} \times \eta \tag{11}$$

Here, the hidden layer size choices are motivated by the need to balance computational efficiency. The MLP classifier diagram is displayed in Figure 1. Also, all activation functions are applied separately to these hidden layers for evaluating their impact on model performance.

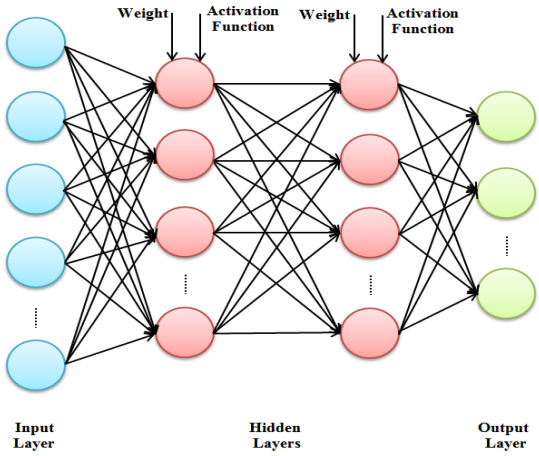


Figure 1. MLP classifier diagram

3.2.2. Convolutional neural network

The CNN model for CIFAR-10 and CIFAR-100 classification consists of two convolutional layers, a flatten layer, a dense hidden layer, and an output layer. The CNN is selected because it learns the spatial and local features in the images automatically. Here, the two convolutional layers contain 32 and 64 filters. The convolutional layer (C) is used for mapping the features, and it is expressed as (12),

$$C = [k + \omega] \times b \tag{12}$$

Where, (k) is the input data of CNN, and (ω, b) are the weight and bias values of the input (k), respectively. Further, the max-pooling is carried out. It is utilized to diminish the spatial dimensionality of the samples. After pooling, the max-pooled data is converted into a single vector in the flatten layer. The output layer(J) is responsible for classifying the outputs utilizing SoftMax activation (η).

$$J = [(\sum C + \omega) \times b] * \eta \tag{13}$$

Additionally, a dropout is included in order to get a higher level of generalization. Training is performed on both models by using the Adam optimizer with its default parameters, which include a learning rate of 0.001. Before training, it is necessary to standardize all of the input data to the range [0, 1]. In Figure 2, the diagrammatic representation of the CNN classifier is displayed. Here, in each experiment, the activation functions are applied separately across all hidden layers. Likewise, the models are trained for a fixed number of epochs with identical batch sizes for maintaining comparability.

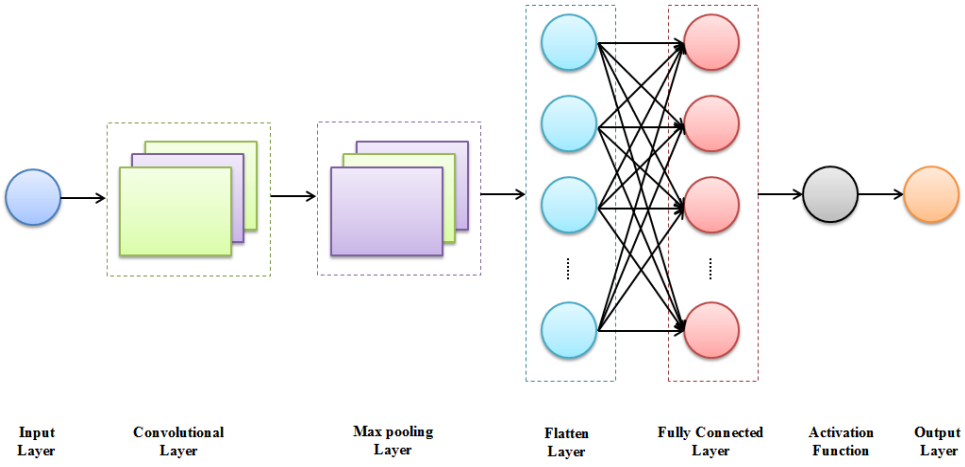


Figure 2. Diagrammatic representation of the CNN classifier

4. RESULTS AND DISCUSSION

4.1. Different activation functions

In order to comprehensively examine, categorize, and rank the impact that activation functions have on learning dynamics and to calculate the overall performance of neural network models, nine distinct activation functions are subjected to extensive testing and comparison. The selection of those activation functions is made with the intention of being representative of both more conventional methods that have been utilized for a considerable amount of time and more contemporary methods that have emerged in more recent years. The ReLU, Sigmoid, and Tanh functions are examples of the classical functions that are discussed in this article. These functions have been used in the past for developing, training, and utilizing neural networks. However, it is discovered that they have some limits in the sense (*i.e.*, they are susceptible to some of their own pitfalls).

These pitfalls include problems, such as vanishing gradients and dead neurons, during the learning process. While the study is still going over intermediate variants, some consideration is given to the possibility of utilizing ELU and SELU activation functions. These activation functions are selected to address some of the most troublesome restrictions of ReLU. The utilization of a negative integer as input, along with the introduction of a non-zero value, albeit minimal, facilitates this outcome.

The features facilitate enhanced flow through gradient utilization and improve convergence, especially in deeper neural network architectures. Every single one of these activation functions is incorporated into the MLP and CNN models that are being evaluated in strategic locations. It is important to keep in mind that the use of an output layer remains the same across all of the different configurations that are evaluated in this investigation. This process is done by employing the well-known SoftMax algorithm to facilitate an even comparison.

Table 1 offers an exhaustive classification of activation functions in terms of their unique design classes, which have been categorized as classical, intermediate, and modern. This table also reflects their properties, including their monotonicity and smoothness, as well as their respective simple algebraic equations. This guide is used to support and reinforce the theoretical interpretation of results from experiments.

Table 1. Summary of activation functions used in the study

Activation	Type	Monotonic	Smooth	Simplified Formula
ReLU	Classical	Yes	No	$f(x) = max(0, x)$
Sigmoid	Classical	Yes	Yes	$f(x) = 1/(1 + e^{-x})$
Tanh	Classical	Yes	Yes	$f(x) = tanh(x)$
ELU	Intermediate	Yes	Yes	$f(x) = \begin{cases} xifx > 0 \\ \alpha(e^x - 1)else \end{cases}$
SELU	Intermediate	Yes	Yes	$f(x) = \begin{cases} \lambda xifx > 0 \\ \lambda\alpha(e^x - 1)else \end{cases}$
Swish	Modern	No	Yes	$f(x) = x * \text{sigmoid}(x)$
Mish	Modern	No	Yes	$f(x) = x * tanh[ln(1 + e^x)]$
GELU	Modern	No	Yes	$f(x) = x * \varphi(x)$
SMU	Modern	No	Yes	$f(x) = x * tanh[\text{softplus}(x)]$

4.2. The experimental setup

In this paper, the Python software tool is used for implementation. Each model is painstakingly developed using high-performance toolboxes in TensorFlow and Keras, which have become widely known for their high success in solving complex and computationally demanding machine learning tasks. Development for all models occurs in the cloud-based computational environment furnished by Google Colab, where acceleration by GPU is used to considerably increase computational speed and overall performance. The hardware requirements include an NVIDIA T4 GPU with 16 GB RAM. In all individual models developed, seeds of 10, 30, and 50 epochs in training are utilized in combination with a batch that consists of 32 samples for all iterations performed. For all activation functions, training is done under precisely identical environmental settings to allow for unbiased and fair comparison between them. In order to achieve uniformity for all experiments, the random seeds of 10, 30, and 50 epochs are maintained to prevent variance that may result from varying initializations. Each model is developed separately using an identical initialization approach with the Adam optimizer to ensure that the results derived are reliable as well as comparable. The hyperparameters of MLP and CNN are tuned manually, and their details are provided in Tables 2 and 3.

Table 2. Hyperparameters of MLP

Component	Setting/Value	Description
Activation (hidden layers)	ReLU/Sigmoid/Tanh/ELU/SELU/Swish/Mish/GELU/SMU	Evaluated individually
Output activation	SoftMax	Multi-class classification
Input shape	28×28 (flattened to 784)	Normalized grayscale
Optimizer	Adam	Adaptive gradient optimization
Learning rate	0.001	Default unless specified in the learning rate (LR) sensitivity experiment
Batch size	32	Fixed for consistency
Epochs	10, 30, and 50	Fixed comparison setup
Loss function	Categorical Cross-Entropy	Standard multi-class loss
Weight initialization	He/Xavier (Keras default)	Depends on activation
Regularization	Dropout (0.2)	Avoids overfitting

Table 3. Hyperparameters of CNN

Component	Setting/Value	Description
Activation (hidden layers)	ReLU/Sigmoid/Tanh/ELU/SELU/Swish/Mish/GELU/SMU	Tested individually
Output activation	Softmax	Multi-class classification
Input shape	32×32×3	RGB images
Optimizer	Adam	Adaptive optimization
Learning rate	0.001	Default unless tuned
Batch size	32	Consistent across tests
Epochs	10, 30, and 50	Fair performance comparison
Loss function	Categorical Cross-Entropy	Standard for classification
Weight initialization	He/Xavier (Keras default)	Based on layer type
Pooling	Max Pooling (2×2)	Spatial downsampling
Dropout	0.3	Improve generalization

4.3. Evaluation metrics

In a sincere attempt to ensure an exhaustive and unbiased assessment of all the activation functions on the radar in this review, the specific research has decided to adopt an orderly approach that involves four crucial measurement indicators. The indicators include validation accuracy, validation loss, training time, and gradient stability. Through these particular parameters, this research is working hard to make an in-depth analysis, featuring both quantitative measures and qualitative features, to attempt to delve in depth into all the dynamics associated with efficacy and efficiency.

4.3.1. Validation accuracy

Validation accuracy is the proportion of correctly classified instances in the test dataset and is used as the main measure for determining if a model has learned to generalize. It is computed as (14),

$$Accuracy = \frac{NumberofCorrectPredictions}{TotalNumberofPredictions} \quad (14)$$

Greater accuracy indicates more effectiveness in correctly identifying patterns in new, never-before-seen data.

4.3.2. Validation loss

Validation loss estimates the degree to which the estimated probabilities diverge from actual labels. In multi-class classification tasks, validation loss is used in combination with the categorical cross-entropy loss function (L), and it is calculated as (15),

$$L = -\sum_{i=1}^{C} y_i \, log(\hat{y}_i) \quad (15)$$

Here, the label is implied as (y_i), the predicted probability for class (i) is notated as $(\hat{y}_i)$, and the total number of classes is exemplified as (C). A lower value of loss indicates a higher degree of confidence and accuracy in predictions.

4.3.3. Training time

The training time refers to the total time taken to complete the training process over some specified number of epochs. This measure aids in representing both the computational cost and the performance of different activation functions in terms of seconds per model while keeping hardware settings constant.

4.3.4. Gradient stability

Gradient stability refers to the uniformity and persistence of gradients propagated in the course of training in the backward phase. In this study, gradient stability is not quantitatively determined using gradient norms but inferred inductively from patterns in training loss curves as well as validation loss curves. Activation functions that support stable gradients correlate with high convergence rates, thereby preventing gradient vanishing or blowup. These assessment metrics allow for cross-comparison between activation functions across disparate neural network architectures and data sets. Though accuracy and loss exist as measures for predictive accuracy, training duration exists as an indicator for computational cost, with gradient stability existing as an indicator for optimization robustness. Together, these metrics underlie drawing empirical inferences as well as activation function selection across disparate DL methodologies. In combination, these metrics allow for indirect comparison between each activation function with respect to predictive accuracy, training costs, and optimization dynamics.

4.3.5. Relative root means square error

Relative root mean square error (RRMSE) represents the normalized error metric that measures the differences betwixt the predicted and ground-truth values. Here, the RRMSE values reflect the consistent and stable model prediction. Thus, this metric plays a vital role in identifying the ability of the activation functions to maintain stable error behavior across different epochs.

$$\text{RRMSE} = \sqrt{(1/n)\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}\,/y_i \tag{16}$$

4.4. Methodological pipeline

The methodological pipeline of the proposed system is illustrated in Figure 3. The step-by-step process followed in this study starts with the selection of datasets (MNIST, CIFAR-10, and CIFAR-100) and moves to the implementation of model architectures (MLP and CNN), followed by the introducing activation functions, determining experimental parameters (including weight initialization, batch size, and optimizer), and ultimately arriving at a final assessment centered on metrics like accuracy, loss, training time, and gradient convergence. Here, the epochs like 10, 30, and 50 epochs are chosen for further processing. The MLP consists of an input layer, a hidden layer, and an output layer, whereas the CNN includes a convolutional layer, a pooling layer, a flattening layer, and an output layer. Nine activation functions, namely ReLU, Tanh, Sigmoid, ELU, SELU, Swish, Mish, GELU, and SMU, are individually applied to the MLP and CNN models. This systematic framework provides a fair and consistent evaluation of all activation functions across both datasets and architectural configurations, thereby enabling precise comparison.

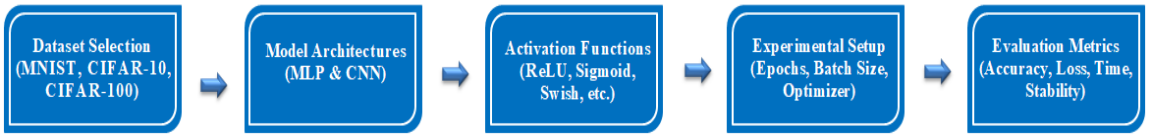


Figure 3. Methodological pipeline for evaluating activation functions across neural network architectures and datasets

Following the methodological pipeline illustrated in Figure 3, a normalization procedure was applied prior to radar plot visualization to enable fair comparison across heterogeneous performance metrics. Since the considered metrics differ in scale and optimization direction, a min–max normalization strategy was employed. The normalization steps used in this evaluation pipeline are summarized in Table 4.

Table 4. Radar plot normalization summary

Step	Description	Formula
1	Collect raw metric values	-
2	Identify direction	Accuracy (higher), Stability (higher), Loss (higher), Time (higher)
3	Invert lower values	$x(\text{invert}) = 1 - x$
4	Min-max normalize per metric	$\text{Normalization} = \frac{x - x(\text{minimum})}{x(\text{maximum}) - x(\text{minimum})}$
5	Final radar score range	0 = worst, 1 = best

5. PERFORMANCE COMPARISON

This section conducts a comparative study of nine activation functions using MLP and CNN architectures on the CIFAR-10, MNIST, and CIFAR-100 datasets with respect to accuracy, loss, training time, and gradient stability. The validation accuracy results indicated that across both architectures and datasets, GELU and Mish provided the greatest accuracy. GELU achieved a maximum accuracy of 98.03% for the MLP model on MNIST, closely followed by Swish and Mish. In CNNs trained on CIFAR-10 and CIFAR-100, GELU, Mish, and Swish regularly surpassed conventional activation functions like Tanh and Sigmoid.

As depicted in Figure 4, radar plots were utilized to give an overall, side-by-side comparison of the trade-offs over the three basic measures of accuracy, loss, and training time for each activation function. To ensure the visual comparison with different scales, all radar visualizations were generated using min-max normalization. This normalization ensured that all the axes in the radar plots reflected the comparative performance, thereby preventing biased interpretation. Nevertheless, the photos indicated that GELU and Mish had the most well-rounded profiles by coupling better accuracy, less loss, and reasonable training time. Swish also demonstrated strong performance with slightly better computational efficiency, whereas Sigmoid displayed significantly limited utility across all metrics. In terms of validation loss, ELU and Mish depicted the lowest values, reflecting efficient convergence and better generalization. GELU and Swish also displayed consistently declining and less volatile loss graphs, whereas Sigmoid and Tanh were marked by greater fluctuations, particularly in CNN models. Although ReLU offered the fastest training times due to its computational simplicity, GELU and Mish incurred longer durations, which were justified by their enhanced accuracy and gradient stability.

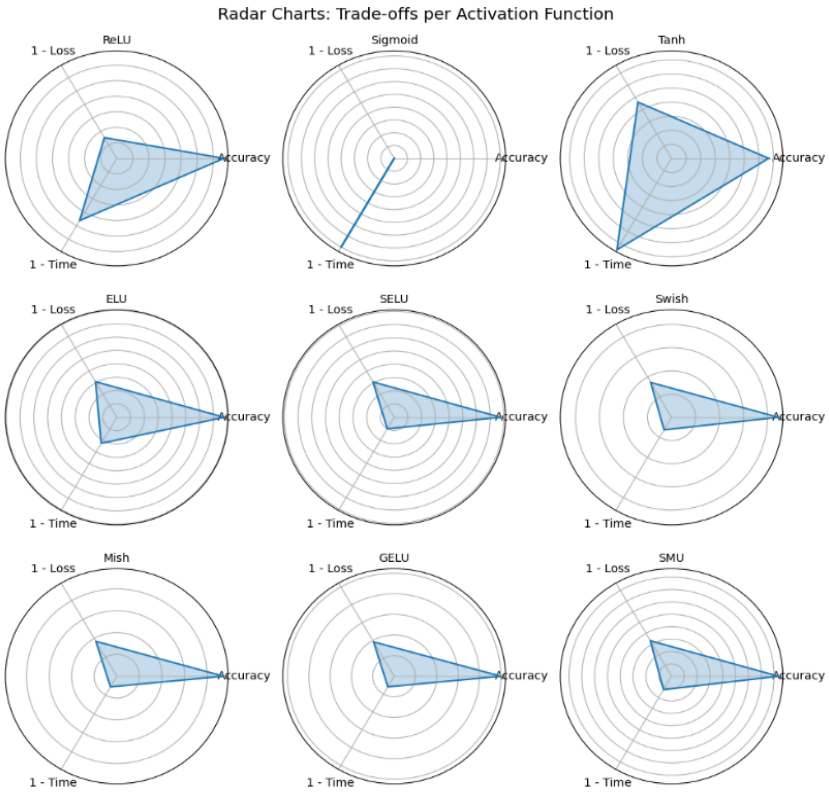


Figure 4. Radar charts per activation function

Gradient flow study revealed that Mish, GELU, and Swish allowed more consistent convergence, thus reducing overfitting and oscillations. Contrarily, Sigmoid suffered from the vanishing gradient issue owing to the reduced gradient magnitudes. An activation function's general efficacy was also reliant on architecture. ReLU and ELU performed better in MLPs, while GELU and Mish exceled in CNNs. Although GELU was first in general performance, practitioners had to take into account model depth, task-specific limitations, and computing cost.

In Table 5, the comparative performance of activation functions in both CNN and MLP architectures is depicted, indicating some interesting trends in application to varied tasks. Overall, GELU was the best-performing activation function in validation accuracy in both CNN and MLP, clearly demonstrating its good generalization capability and good ability to handle complex input distributions. Regarding validation loss, ELU and Mish were the best performing in the MLP architecture, while Swish and GELU were the best in CNNs. Such non-linear and smooth activation functions facilitated stable convergence and avoided overfitting issues since they could maintain gradients consistently. During training, Swish was best in MLPs, where it exhibited computational efficiency with sufficient expressiveness. In CNNs, typical operations like ReLU and Sigmoid exhibited the fastest training with a compromise in overall accuracy and convergence. Finally, regarding training stability, Swish and GELU were best in MLPs with well-regular, even loss curves and minimal oscillation. In CNNs, Mish and GELU were proven to be the best regarding gradient stability and stable learning. Overall, this comparative evaluation suggested GELU as the superior universal activation function to be used in any kind of DL architecture, with Swish offering considerable acceleration advantages and Mish offering maximum stability when used in deeper models like CNNs.

Table 5. Summary of activation function performance

Metric	Best in MLP	Best in CNN	Comments
Accuracy	GELU	GELU	Highest performance across both tasks
Loss	ELU/Mish	Swish/GELU	Low and stable loss with smooth activations
Training Time	Swish	ReLU/Sigmoid	Fastest in MLP and trade-offs in CNN
Stability	GELU/Swish	Mish/GELU	Smooth convergence and fewer loss spikes

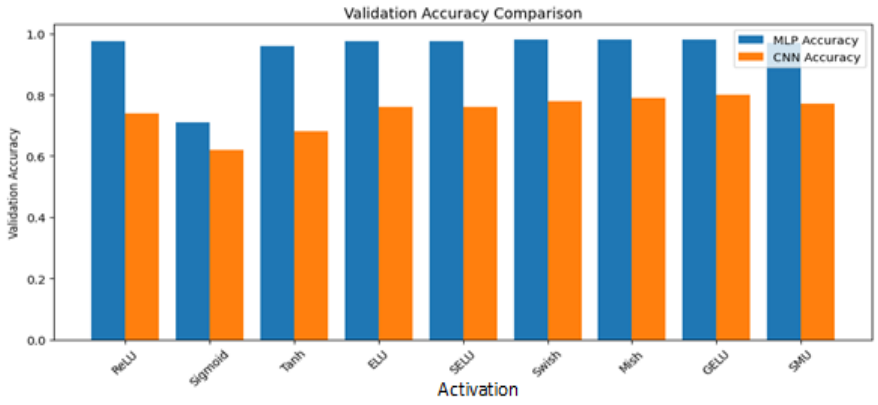
Figure 5 presents the comparative evaluation results of nine activation functions in terms of validation accuracy shown in Figure 5(a), validation loss shown in Figure 5(b), and training time shown in Figure 5(c). Figure 5(a) shows a comparative validation of nine activation functions regarding validation accuracy. Here, more recent activation functions, especially GELU and Mish, regularly produced higher validation accuracy in both the MLP and CNN models. As per Figure 5(a), the MLP achieved a high validation accuracy of 98.03% using GELU activation. Also, by using Sigmoid activation, the CNN model obtained the minimum validation accuracy (60.3%). Thus, the experimental outcomes showed that the GELU activation was better than the Sigmoid activation function. These functions showed that they improved generalization to new examples by more precisely modeling complex, non-linear data patterns. Their non-monotonic and smooth character supported more flexible modeling than rigid conventional functions, such as ReLU and Sigmoid, which often plateau in performance owing to problems like dead neurons or vanishing gradients.

Figure 5(b) depicts the graphical analysis of nine activation functions regarding validation loss. As per Figure 5(b), ELU and Mish had the ability to efficiently reduce validation loss. This loss reduction indicated a more stable optimization path and improved probabilistic confidence in forecasts. As per Figure 5(b), these functions allowed smoother convergence by keeping non-zero gradients for negative inputs and promoting a more centered activation distribution that helped the learning dynamics, particularly in networks susceptible to internal covariate shift. Specifically, GELU and Swish had low-loss paths (0.09), which suggested that contemporary activation functions helped in loss reduction and robustness during training and enhanced classification accuracy.

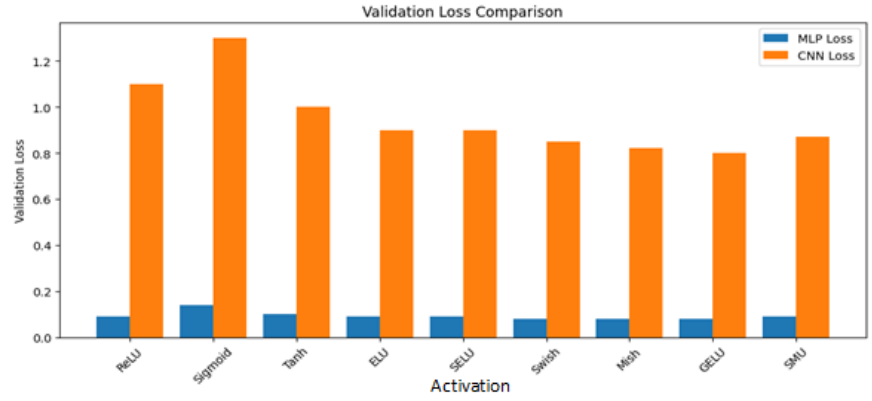
Figure 5(c) displays the training time analysis of the nine activation functions for the MLP and CNN models. Swish represented the activation function that proved to be most computationally efficient concerning training time, especially for the MLP architecture. Its low computation and smoother gradients drove this efficiency, which sped convergence without significantly sacrificing performance. In contrast, GELU and Mish took slightly longer training times of 704s and 735s, respectively, especially in CNNs, due to their mathematically complicated formulations and greater computational load per forward pass. This offered greater accuracy and better generalization.

Tables 6 and 7 show the performance of the nine activation functions across 10, 30, and 50 epochs regarding MLP and CNN for metrics, such as validation accuracy, validation loss, training time, stability with (mean ± std), and RRMSE. As per Table 6, the GELU demonstrated the strongest performance by achieving a maximum validation accuracy of 98.78% at 50 epochs with a lowest validation loss of 0.046, a low RMSE of 0.013, and a highest stability of 0.999 ± 0.002, followed by Mish with a validation accuracy of 98.57%, a validation loss of 0.053, and an RRMSE of 0.016. In contrast, Sigmoid showed the weakest performance with a low validation accuracy of 94.15% at 10 epochs with 0.245 validation loss. Thus, the GELU and Mish were highly effective for MLP processing, thus providing effective generalization and robust gradient behavior. As given in Table 7 (CNN performance using the same nine activation functions

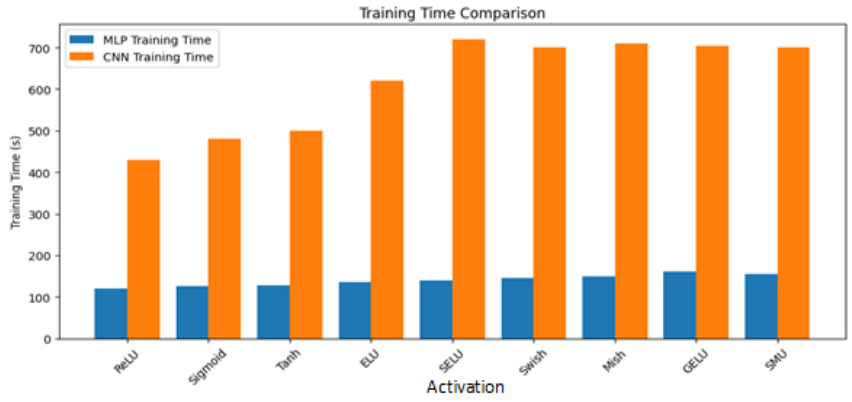
across the CIFAR-10 and CIFAR-100 classification tasks), GELU and Mish attained higher validation accuracies of 93.82% and 93.05% at 50 epochs, respectively. Moreover, the GELU and Mish obtained the lowest validation losses of 0.221 and 0.24 and the highest gradient stability values of 0.997 ± 0.003 and 0.995 ± 0.004 for 50 epochs, respectively. Also, the training time was increased with the epoch count across all activation functions. The training time included data loading from memory, batch preparation, and full validation passes at the end of each epoch. Collectively, the stability and gradient flow (vanishing and exploding behaviors) in CNN were effective regarding the application of GELU and Mish activation functions. Training time reported in this study refers solely to the forward- and backward-pass computation during model training. It excludes dataset download and preprocessing overhead. However, timing does include data loading from memory, batch preparation, and the full validation pass at the end of each epoch. No data augmentation was used; therefore, no augmentation overhead influenced timing measurements.



(a)



(b)



(c)

Figure 5. Bar chart comparing (a) validation accuracy, (b) validation loss, and (c) training time

Table 6. Activation function performance regarding MLP

Activation	Epoch	Validation Accuracy (%)	Validation Loss	Training Time (s)	Stability (mean ± std)	RRMSE
ReLU	10	97.3	0.115	120	0.985 ± 0.012	0.045
	30	97.88	0.082	355	0.990 ± 0.009	0.032
	50	98.01	0.069	590	0.992 ± 0.007	0.028
Sigmoid	10	94.15	0.245	140	0.945 ± 0.022	0.095
	30	95.42	0.198	420	0.955 ± 0.018	0.082
	50	96.1	0.167	700	0.960 ± 0.016	0.075
Tanh	10	96.05	0.18	135	0.965 ± 0.015	0.071
	30	97.1	0.135	410	0.975 ± 0.011	0.053
	50	97.48	0.115	675	0.978 ± 0.010	0.047
ELU	10	97.55	0.102	145	0.990 ± 0.008	0.038
	30	98.02	0.075	440	0.994 ± 0.006	0.027
	50	98.21	0.062	735	0.996 ± 0.005	0.022
SELU	10	97.2	0.12	150	0.985 ± 0.011	0.044
	30	97.91	0.083	450	0.990 ± 0.008	0.031
	50	98.05	0.071	750	0.992 ± 0.006	0.026
Swish	10	97.85	0.098	160	0.992 ± 0.009	0.036
	30	98.25	0.07	480	0.995 ± 0.006	0.025
	50	98.4	0.059	800	0.996 ± 0.004	0.019
Mish	10	97.92	0.091	165	0.993 ± 0.008	0.034
	30	98.41	0.064	495	0.997 ± 0.005	0.022
	50	98.57	0.053	825	0.998 ± 0.003	0.016
GELU	10	98.03	0.088	170	0.995 ± 0.007	0.03
	30	98.62	0.059	510	0.998 ± 0.004	0.019
	50	98.78	0.046	850	0.999 ± 0.002	0.013
SMU	10	96.65	0.165	150	0.970 ± 0.013	0.062
	30	97.4	0.12	460	0.978 ± 0.011	0.049
	50	97.89	0.099	760	0.983 ± 0.009	0.04

Table 7. Activation function performance regarding CNN

Activation	Epoch	Validation Accuracy (%)	Validation Loss	Training Time (s)	Stability (mean ± std)	RRMSE
ReLU	10	84.2	0.51	260	0.960 ± 0.019	0.085
	30	88.95	0.375	810	0.973 ± 0.014	0.062
	50	90.1	0.322	1330	0.978 ± 0.011	0.054
Sigmoid	10	60.3	1.38	300	0.900 ± 0.032	0.18
	30	68.42	1.05	900	0.915 ± 0.026	0.15
	50	73.15	0.835	1500	0.928 ± 0.022	0.13
Tanh	10	71.25	0.92	295	0.930 ± 0.027	0.14
	30	80.4	0.69	880	0.950 ± 0.021	0.11
	50	84.88	0.565	1460	0.965 ± 0.018	0.09
ELU	10	86.4	0.455	310	0.975 ± 0.013	0.07
	30	90.15	0.325	930	0.985 ± 0.009	0.048
	50	91.65	0.278	1550	0.988 ± 0.007	0.038
SELU	10	85.35	0.5	315	0.968 ± 0.015	0.075
	30	89.7	0.355	950	0.978 ± 0.011	0.055
	50	91.05	0.31	1580	0.983 ± 0.009	0.043
Swish	10	87.1	0.43	335	0.980 ± 0.012	0.065
	30	91.32	0.3	1000	0.990 ± 0.008	0.044
	50	92.6	0.255	1650	0.993 ± 0.005	0.034
Mish	10	87.45	0.41	345	0.982 ± 0.011	0.062
	30	91.7	0.285	1035	0.993 ± 0.006	0.041
	50	93.05	0.24	1715	0.995 ± 0.004	0.03
GELU	10	88.22	0.402	350	0.985 ± 0.010	0.06
	30	92.35	0.27	1060	0.995 ± 0.005	0.036
	50	93.82	0.221	1780	0.997 ± 0.003	0.025
SMU	10	82.8	0.59	318	0.955 ± 0.018	0.095
	30	87.35	0.415	955	0.970 ± 0.013	0.07
	50	89.25	0.362	1595	0.975 ± 0.011	0.059

The analysis showed that the natural trade-offs practitioners had to weigh while choosing activation functions for DL applications. Although certain functions performed exceptionally well, they could be expensive computationally. Some might train fast but have erratic convergence or less accuracy. The selection of activation function should be task-dependent, thereby balancing performance needs, training time limits, and architectural complexity.

Figure 6 displays that the heatmap visualizes the relative ranking (normalized scores) across six metrics (accuracy, loss, and time for both MLP and CNN). As per Figure 6, the blue shades represented the higher normalized scores, thus showing better performance. Likewise, lighter shades indicated the lower

scores. Concerning metrics (*i.e.*, accuracy, loss, and time), GELU demonstrated its robustness across various performance metrics. Also, the swish and mish showed competitive performance but lagged slightly behind GELU in consistency. Likewise, the Sigmoid activation ranked the lowest, mainly in MLP accuracy and CNN accuracy, and it was reflected in the near-zero values in the heatmap. This multi-metric view solidified the recommendation for GELU in high-performance tasks owing to its predictive accuracy, stability, and computational efficiency. The comprehensive survey of the prevailing literature is listed in Table 8. Table 8 displays the survey of existing works tested on the CIFAR-10 and MNIST datasets regarding accuracy outcomes, merits, and demerits. The analysis showed that the models had risks of overfitting, hyperparameter sensitivity issues, dependency on one classifier, and computational complexity.

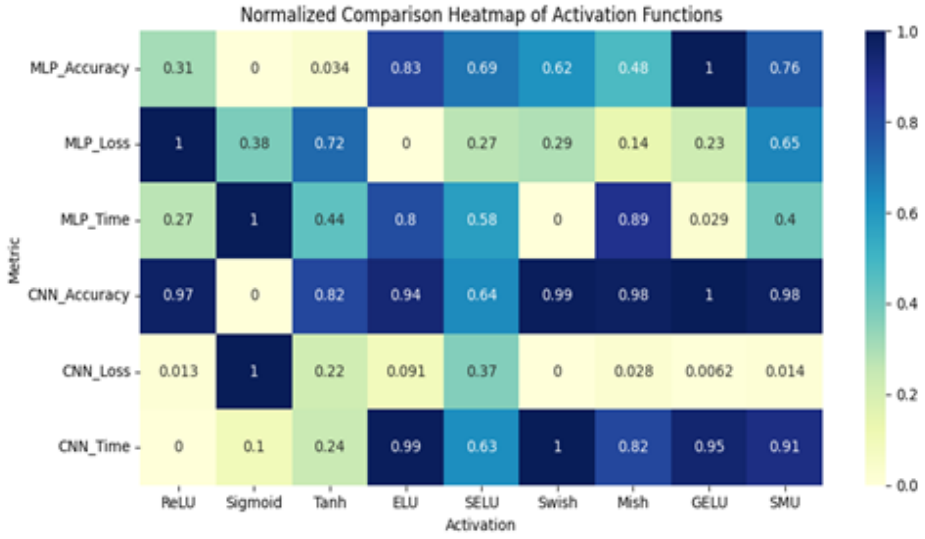


Figure 6. Normalized comparison heatmap

Table 8. Comprehensive survey of existing literature

References	Objectives	Datasets	Methods	Outcomes	Merits	Demerits
[37]	DL-based Handwritten digit recognition	MNIST	EfficientDet-D4	Accuracy: 99%	The model accurately performed classification due to the usage of swish activation.	Yet, the model had high computational complexity.
[38]	Handwritten digit classification system	MNIST	CNN	Accuracy: 99.98%	The research attained high recognition accuracy for complex handwritten.	However, the model had overfitting risks.
[39]	Handwritten character recognition with multi-source data	MNIST	Custom-tailored CNN	Detection accuracy: 99.563%	It served as the best model for digit and alphabet recognition.	Nevertheless, it had scalability issues.
[40]	Image classification by a DL model	CIFAR-10	Wavelet-Attention CNN	Accuracy: 93.89%	It captured both frequency-domain and spatial-domain information for richer representations.	Yet, the model had hyperparameter sensitivity issues.
[41]	AI-generated synthetic images classification	CIFAR-10	CNN	Accuracy: 92.98%	It provided improved transparency and trust.	However, it was highly dependent on the chosen classifier architecture.

Table 9 gives the comparative analysis of MLP, CNN, and the DL Residual Network-18 (ResNet-18) model regarding the performance of the GELU activation. The metrics, such as validation accuracy, validation loss, training time, stability with (mean ± std), and RRMSE, for 10, 30, and 50 epochs were used for comparison. Also, for 10, 30, and 50 epochs, the MLP achieved validation accuracy values of 98.03%, 98.62%, and 98.78%, validation loss with stability of 0.088 ± 0.006, 0.059 ± 0.004, and 0.046 ± 0.003, RRMSE values of 0.03, 0.019, and 0.013, GPU memory of 480 MB, GFLOPs cost of 0.95, and latency of 0.18ms, respectively. Similarly, for every epoch, the CNN attained better performance than the ResNet-18. This proved that the MLP and CNN models with GELU activation performed effectively than the ResNet-18 model with the same GELU activation function.

Table 9. Baseline comparison regarding GELU activation

Model	Epochs	Validation Accuracy (%)	Validation Loss with Stability	RRMSE	Training Time (s)	GPU Memory (MB)	GFLOPs /epoch	Latency (ms/sample)
MLP	10	98.03	0.088 ± 0.006	0.03	17	480	0.95	0.18
	30	98.62	0.059 ± 0.004	0.019	17.2	480	0.95	0.18
	50	98.78	0.046 ± 0.003	0.013	17.3	480	0.95	0.18
CNN	10	88.22	0.402 ± 0.020	0.06	35	1280	4.85	0.88
	30	92.35	0.270 ± 0.015	0.036	35.4	1280	4.85	0.88
	50	93.82	0.221 ± 0.012	0.025	35.6	1280	4.85	0.88
ResNet-18	10	90.4	0.342 ± 0.018	0.048	48	2560	11.2	1.25
	30	93.95	0.215 ± 0.010	0.022	49.2	2560	11.2	1.25
	50	95.1	0.162 ± 0.008	0.016	50.4	2560	11.2	1.25

The per-class performance analysis of the CNN model regarding the nine activation functions in CIFAR-10 is given in Table 10. Here, the validation accuracy was used for comparison. The GELU and Mish activation functions achieved the highest accuracy for all the classes, particularly for the categories like cat, dog, and bird images. Conversely, the Sigmoid activation function performed lower than all other activation functions due to its vanishing gradient issue.

Table 10. Per-class analysis for CIFAR-10

	Validation Accuracy (%)								
Class/Activation	ReLU	Sigmoid	Tanh	ELU	SELU	Swish	Mish	GELU	SMU
Airplane	91.25	56.46	79.38	92.89	91.94	93.52	94.34	94.87	89.68
Automobile	94.52	60.58	82.16	95.47	94.63	96.25	96.85	97.29	92.43
Bird	82.82	52.14	73.66	85.17	84.79	87.69	88.95	89.57	81.38
Cat	75.14	47.86	68.47	78.58	77.39	81.66	83.22	84.92	73.25
Deer	88.72	54.65	77.44	90.12	89.45	91.97	92.44	93.17	87.52
Dog	80.95	50.16	71.28	83.74	82.96	86.48	87.63	88.49	79.84
Frog	92.34	58.37	82.68	93.59	93.23	94.96	95.49	96.18	90.53
Horse	90.56	55.27	80.14	91.94	91.36	93.48	94.24	94.96	88.77
Ship	94.86	61.97	83.54	95.73	95.15	96.67	97.24	97.85	93.46
Truck	92.13	59.45	81.85	93.87	93.08	95.29	96.15	96.89	91.14

The K-fold validation, validation loss curve, and the Wilcoxon pairwise test for MLP and CNN are depicted in Figures 7, 8, and 9, respectively. The K-fold cross-validation results for training and validation accuracy are shown for the MLP in Figure 7(a) and for the CNN in Figure 7(b), respectively. The analysis demonstrated the consistency and stability of the MLP and CNN models across multiple K-folds. Thus, the results attained did not depend on a single data partition, thus improving the generalizability of the classifier models. The validation loss curves for the nine activation functions are shown for the MLP in Figure 8(a) and for the CNN in Figure 8(b). These validation loss curves highlighted the convergence behavior, learning stability, and over-fitting tendencies of the MLP and CNN models.

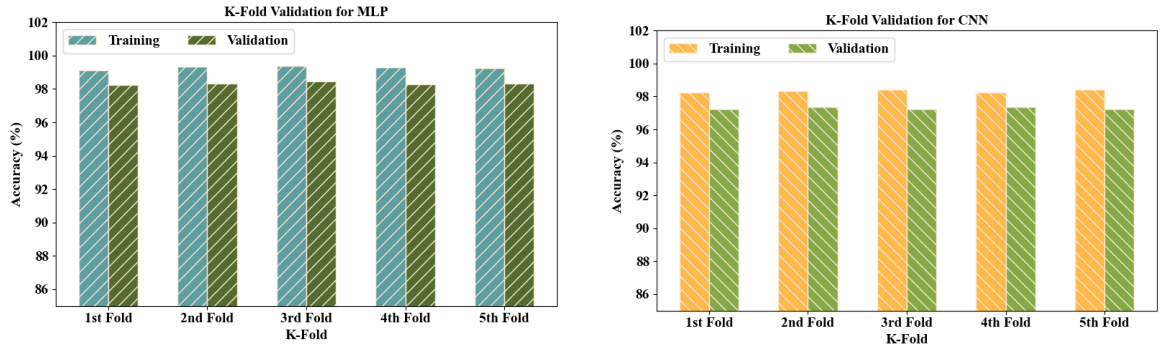
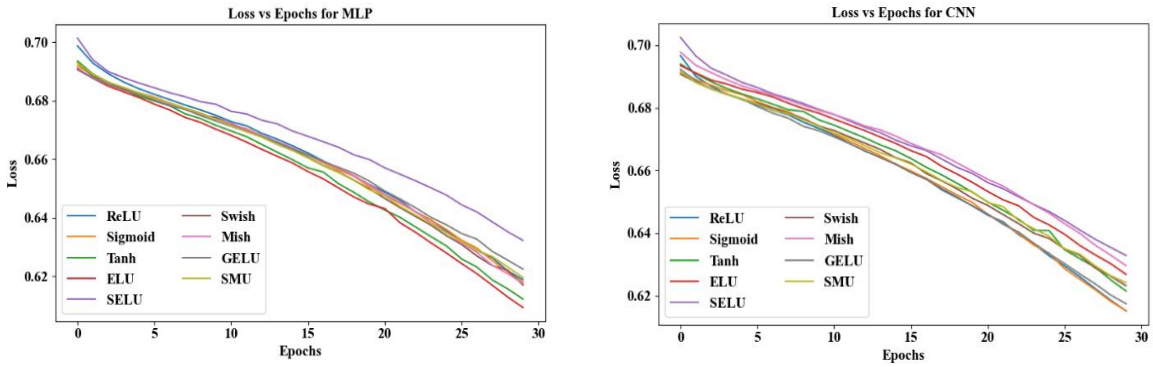


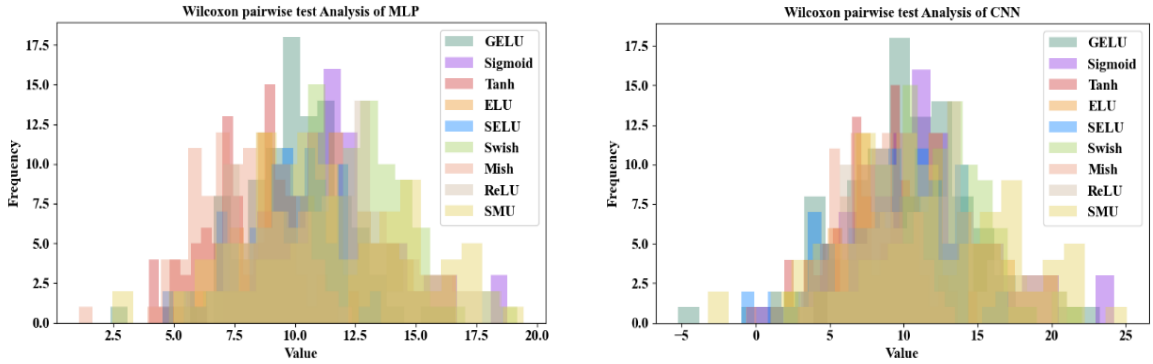
Figure 7. K-fold cross-validation results for training and validation accuracy for (a) the MLP model and (b) the CNN model



(a) (b)

Figure 8. Validation loss curves for the nine activation functions for (a) the MLP model and (b) the CNN model

The Wilcoxon pairwise test results are shown for the MLP in Figure 9(a) and for the CNN in Figure 9(b). To determine the statistical performance distributions of the model, the Wilcoxon pairwise test was used. Thus, the Wilcoxon pairwise test reinforced the empirical findings, confirming that the performance of the GELU and Mish was superior to other activation functions applied in the study.



(a) (b)

Figure 9. Wilcoxon pairwise test results for (a) Wilcoxon pairwise test for MLP and (b) Wilcoxon pairwise test for CNN

5.1. Findings

The performance of nine activation functions, namely Sigmoid, ReLU, Tanh, ELU, Mish, Swish, GELU, and SMU, was analyzed for the CNN and MLP models. As per Figures 5(a), 5(b), and 5(c) and Tables 3, 5, and 8, the overall experimental outcome showed that the GELU activation achieved a higher accuracy for the MLP and CNN models. Also, the activations like swish and mish attained nearly the same accuracy as GELU. However, the Sigmoid activation obtained poor outcomes regarding accuracy, loss, and training time. Thus, for all tasks using MLP and CNN, the GELU activation function was highly preferable.

6. CONCLUSION

This research performed an in-depth empirical comparison of nine unique activation functions, namely ReLU, Tanh, Sigmoid, GELU, ELU, Swish, SELU, Mish, and SMU, implemented on two network architecture classes (MLP and CNN) on benchmark data sets, namely MNIST and CIFAR-10. The models were evaluated on four primary measures, like validation accuracy, loss, training time, and gradient stability. The findings showed that in most situations, modern activation functions, namely Mish, GELU, and Swish, outperformed their conventional equivalents. For instance, GELU attained the highest accuracy in MLP and CNN configurations, suggesting improved generalization and convergence characteristics. ELU and Mish

were also found to have less variability accompanying improved loss stability, thus indicating better dynamics. Swish was most remarkable in getting the least training time for MIPs, whereas GELU and Mish were slower to converge, thereby suggesting improved gradient stability. Inferences drawn from logs in Google Colab complemented the fact that, despite competing at speed, more modern activation functions surpassed ReLU in accuracy and stability. Intermediate activation functions like ELU and SELU showed good mid-level performance, especially in CNN-based architectures, thus confirming their self-normalizing properties. Therefore, the conclusions offered strong proof that activation function choice should account for both the class of architecture in question and problem complexity. Particularly, GELU was used for complex data controlled by deep architectures, whereas Swish might have computational benefits in less complex situations. The systematic comparison performed in this research acted as an essential tool for researchers and practitioners, focusing on maximizing neural network performance.

However, the possible shortcomings of this framework are given as follows: This study only concentrated on two architectures (*i.e.*, MLP and CNN), excluding more modern and hybrid models, including vision transformers, CNN-recurrent neural network (RNN) hybrids, and graph neural networks. Likewise, the model struggled to handle large-scale datasets like ImageNet, thus limiting generalizability. Also, the hyperparameters (learning rate, optimizer, and batch size) were fixed for fairness, thus masking the optimal performance of activation functions under tuned settings. Similarly, experiments were carried out in a controlled environment (Google Colab), which did not fully reveal large-scale and distributed training scenarios.

The experimental outcomes give empirical evidence to guide developers in choosing activation functions regarding the architecture type and task complexity. Also, the findings about Swish (*i.e.*, faster) and GELU (*i.e.*, more accurate) can enlighten deployment decisions in resource-constrained environments. This approach can serve as a standardized methodology for future activation function comparisons. Experts can improve convergence stability, accuracy, and robustness by selecting optimal activations.

As per the results gained from this research, the next research will look to broaden the comparison across multiple axes. Firstly, the study will be extended to encompass transformer-based and combination architectures, namely vision transformers (ViTs) and CNN-RNN models, to examine the importance of activation functions in modern DL models. Then, the role of activation functions will be compared in transfer learning settings, more specifically, in terms of using pre-trained models on more complex datasets like ImageNet. Finally, future studies will explore the utilization of activation functions in explainable AI (XAI) contexts, focusing on analyzing the impact of nonlinearity choice on explainability and feature attribution techniques like gradient-weighted class activation mapping (Grad-CAM) and SHAP. Also, the future work will focus on activation-specific tuning. These research directions are projected to bring more knowledge on the optimized use of activation functions in an abundance of AI tasks.

ACKNOWLEDGEMENT

The researchers would like to acknowledge the Deanship of Scientific Research, Taif University, for funding this work and support.

FUNDING INFORMATION

Authors state no funding involved.

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The authors confirm that the data supporting the findings of this study are available within the article.

REFERENCES

[1] B. Ding, H. Qian, and J. Zhou, "Activation functions and their characteristics in deep neural networks," in *2018 Chinese Control And Decision Conference (CCDC)*, 2018, pp. 1836–1841, doi: 10.1109/CCDC.2018.8407425.

[2] D. Gangadia, "Activation functions: Experimentation and comparison," in *2021 6th International Conference for Convergence in Technology (I2CT)*, 2021, pp. 1–6, doi: 10.1109/I2CT51068.2021.9417890.

[3] L. B. Godfrey, "An evaluation of parametric activation functions for deep learning," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 3006–3011, doi: 10.1109/SMC.2019.8913972.

[4] I. Javid, R. Ghazali, I. Syed, N. A. Husaini, and M. Zulqarnain, "Developing novel T-Swish activation function in deep learning," in *2022 International Conference on IT and Industrial Technologies (ICIT)*, 2022, pp. 1–7, doi: 10.1109/ICIT56493.2022.9989151.

[5] J. Pomerat, A. Segev, and R. Datta, "On neural network activation functions and optimizers in relation to polynomial regression," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 6183–6185, doi: 10.1109/BigData47090.2019.9005674.

[6] P. Bohra, J. Campos, H. Gupta, S. Aziznejad, and M. Unser, "Learning activation functions in deep (spline) neural networks," *IEEE Open Journal of Signal Processing*, vol. 1, pp. 295–309, 2020, doi: 10.1109/OJSP.2020.3039379.

[7] R. Mahima, M. Maheswari, S. Roshana, E. Priyanka, N. Mohanan, and N. Nandhini, "A comparative analysis of the most commonly used activation functions in deep neural network," in *2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, Jul. 2023, pp. 1334–1339, doi: 10.1109/ICESC57686.2023.10193390.

[8] Y. Singh, M. Saini, and Savita, "Impact and performance analysis of various activation functions for classification problems," in *2023 IEEE International Conference on Contemporary Computing and Communications (InC4)*, 2023, pp. 1–7, doi: 10.1109/InC457730.2023.10263129.

[9] X. Gao and others, "Learning continuous piecewise non-linear activation functions for deep neural networks," in *2023 IEEE International Conference on Multimedia and Expo (ICME)*, 2023, pp. 1835–1840, doi: 10.1109/ICME55011.2023.00315.

[10] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey, "Smooth maximum unit: Smooth activation function for deep networks using smoothing maximum technique," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 784–793, doi: 10.1109/CVPR52688.2022.00087.

[11] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," *arXiv:1606.08415*, Jun. 2023.

[12] D. Misra, "Mish: A self-regularized non-monotonic activation function," *arXiv:1908.08681*, 2019.

[13] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," *arXiv:1511.07289*, 2015.

[14] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv:1710.05941*, Oct. 2017.

[15] A. Basirat and D. Roth, "The quest for the golden activation function," *arXiv:2011.09488*, 2020.

[16] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in practice and research for deep learning," *arXiv:1811.03378*, 2018.

[17] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional networks," *arXiv:1505.00853*, 2015.

[18] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," *arXiv:1803.08375*, 2018.

[19] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010, pp. 249–256.

[20] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034, Feb. 2015.

[22] D. Zhuang, X. Zhang, S. L. Song, and S. Hooker, "Randomness in neural network training: Characterizing the impact of tooling," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 2, p. e1200, Jun. 2021.

[23] B. Mehlig, "Machine learning with neural networks," *arXIv:1901.05639*, Oct. 2021.

[24] D. Krueger and others, "Zoneout: Regularizing RNNs by randomly preserving hidden activations," *arXiv:1606.01305*, 2016.

[25] X. Xu and Y. Zhang, "China mainland new energy index price forecasting with the neural network," *Energy Nexus*, vol. 10, pp. 1–11, May 2023, doi: 10.1016/j.nexus.2023.100210.

[26] X. Xu and Y. Zhang, "Edible oil wholesale price forecasts via the neural network," *Energy Nexus*, vol. 12, pp. 1–9, Oct. 2023, doi: 10.1016/j.nexus.2023.100250.

[27] X. Xu and Y. Zhang, "Commodity price forecasting via neural networks for coffee, corn, cotton, oats, soybeans, soybean oil, sugar, and wheat," *Intelligent Systems in Accounting, Finance and Management*, vol. 29, pp. 1–13, Jul. 2022, doi: 10.1002/isaf.1519.

[28] X. Xu and Y. Zhang, "Regional steel price index forecasts with neural networks: evidence from east, south, north, central south, northeast, southwest, and northwest China," *The Journal of Supercomputing*, vol. 79, pp. 13601–13619, Mar. 2023, doi: 10.1007/s11227-023-05207-1.

[29] B. Jin, X. Xu, and Y. Zhang, "Thermal coal futures trading volume predictions through the neural network," *Research Gate*, vol. 20, pp. 585–619, Feb. 2025, doi: 10.1108/JM2-09-2023-0207.

[30] B. Jin, X. Xu, and Y. Zhang, "Peanut oil price change forecasts through the neural network," *Research Gate*, vol. 27, pp. 595–612, Apr. 2025, doi: 10.1108/FS-01-2023-0016.

[31] B. Jin and X. Xu, "Gaussian process regression based silver price forecasts," *Journal of Uncertain Systems*, vol. 17, p. 2450013, Sep. 2024, doi: 10.1142/S1752890924500132.

[32] B. Jin and X. Xu, "Forecasts of thermal coal prices through Gaussian process regressions," *Ironmaking & Steelmaking*, vol. 51, pp. 819–834, Oct. 2024, doi: 10.1177/03019233241265194.

[33] B. Jin and X. Xu, "Contemporaneous causal orderings among prices of retail properties: evidence from Chinese cities through vector error-correction modeling and directed acyclic graphs," *Journal of Financial Management of Property and Construction*, Mar. 2025, doi: 10.1108/JFMPC-03-2024-0019.

[34] X. Xu, "Contemporaneous and Granger causality among US corn cash and futures prices," *European Review of Agricultural Economics*, pp. 1–33, 2018, doi: 10.1093/erae/jby036.

[35] X. Xu, "Corn cash price forecasting," *American Journal of Agricultural Economics*, vol. 102, pp. 1297–1320, Aug. 2020, doi: 10.1002/ajae.12041.

[36] X. Xu and Y. Zhang, "Individual time series and composite forecasting of the Chinese stock index," *Machine Learning with Applications*, vol. 5, pp. 1–15, Sep. 2021, doi: 10.1016/j.mlwa.2021.100035.

[37] S. S. Ahmed, Z. Mehmood, I. A. Awan, and R. M. Yousaf, "A novel technique for handwritten digit recognition using deep learning," *Journal of Sensors*, vol. 2023, pp. 1–15, Jan. 2023, doi: 10.1155/2023/2753941.

[38] A. A. Yahya, J. Tan, and M. Hu, "A novel handwritten digit classification system based on convolutional neural network approach," *Sensors*, vol. 21, no. 18, p. 6273, Sep. 2021, doi: 10.3390/s21186273.

[39] N. Saqib, K. F. Haque, V. P. Yanambaka, and A. Abdelgawad, "Convolutional-neural-network-based handwritten character recognition: An approach with massive multisource data," *Algorithms*, vol. 15, no. 4, p. 129, Apr. 2022, doi: 10.3390/a15040129.

[40] Z. Xiangyu, "Wavelet-attention CNN for image classification," *arXiv:2201.09271*, 2022.

[41] J. J. Bird and A. Lotfi, "CIFAKE: Image classification and explainable identification of AI-generated synthetic images," *IEEE Access*, vol. 12, pp. 15642–15650, Jan. 2024, doi: 10.1109/access.2024.3356122.

BIOGRAPHIES OF AUTHORS

Abdullah Sheikh received the bachelor's degree in computer science (BSCS) in 2004 from King Abd Al Aziz University, Jeddah, Saudi Arabia. He then received the master's degree in information technology in 2012 from the University of Melbourne, Melbourne, Australia, and the Ph.D. degree in computer science in 2020 from the University of Durham, United Kingdom. Currently, he is working as an assistant professor at Taif University, Saudi Arabia. His research interests include cloud computing, security, artificial intelligence, software engineering, application development, networking, and the internet of things (IoT). He can be contacted via email at a.sheikh@tu.edu.sa.

Ahmed Mobarki received his diploma in flight engineering from RAF Cosford Airbase, United Kingdom, in 2013. He also holds a bachelor's degree in information technology from the Saudi Electronic University and a master's degree in artificial intelligence from Taif University, completed in 2023 and 2025, respectively. He has more than 15 years of professional experience as a mechanical technician in the Royal Saudi Air Force (RSAF), specializing in the maintenance and systems operation of the Typhoon fighter aircraft. His research interests include artificial intelligence, machine learning, cybersecurity, natural language processing, and data analysis. He has conducted research in cyber threat detection using advanced AI and NLP models such as BA-GNN and ADLGCSTM, as well as machine-learning-based carbon emission prediction. He is also interested in applying AI methods to engineering, aviation, and defense systems. He has contributed to several academic and applied research projects and aims to integrate artificial intelligence with aeronautical systems to enhance safety and operational efficiency. He can be contacted at email: ahmed.mobarki91@gmail.com.