IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 March 2024, accepted 2 May 2024, date of publication 20 May 2024, date of current version 31 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3403457

RESEARCH ARTICLE

SWAG: A Novel Neural Network Architecture Leveraging Polynomial Activation Functions for Enhanced Deep Learning Efficiency

SAEID SAFAEI[1], ZEROTTI WOODS[2], KHALED RASHEED[1,3], THIAB R. TAHA[1], VAHID SAFAEI[4], JUAN B. GUTIÉRREZ[5], AND HAMID R. ARABNIA[1]

[1]Department of Computer Science, University of Georgia, Athens, GA 30602, USA
[2]Applied Physics Laboratory, Johns Hopkins University, Baltimore, MD 21218, USA
[3]Institute of Artificial Intelligence, University of Georgia, Athens, GA 30602, USA
[4]Department of Mechanical Engineering, University of Isfahan, Isfahan 8174673441, Iran
[5]Department of Mathematics, The University of Texas at San Antonio, San Antonio, TX 78249, USA

Corresponding authors: Juan B. Gutiérrez (Juan.Gutierrez3@utsa.edu)

ABSTRACT Deep learning techniques have demonstrated significant capabilities across numerous applications, with deep neural networks (DNNs) showing promising results. However, training these networks efficiently, especially when determining the most suitable nonlinear activation functions, remains a significant challenge. While the ReLU activation function has been widely adopted, other hand-designed functions have been proposed. One such approach is the trainable activation functions. This paper introduces a novel neural network design, the SWAG. In this structure, instead of evolving, activation functions consistently form a polynomial basis. Each hidden layer in this architecture comprises k sub-layers that use polynomial activation functions adjusted by a factorial coefficient, followed by a Concatenate layer and a layer employing a linear activation function. Leveraging the Stone-Weierstrass approximation theorem, we demonstrate that utilizing a diverse set of polynomial activation functions allows neural networks to retain universal approximation capabilities. The SWAG algorithm's architecture is then presented, where data normalization is emphasized, and a new optimized version of SWAG is proposed, which reduces the computational challenge of managing higher degrees of input. This optimization harnesses the Taylor series method by utilizing lower-degree terms to compute higher-degree terms efficiently. This paper thus contributes an innovative neural network architecture that optimizes polynomial activation functions, promising more efficient and robust deep learning applications.

INDEX TERMS Activation functions, factorial coefficient, neural network design, polynomial activation function.

I. INTRODUCTION

Deep learning has revolutionized many fields, enabling computational models to learn abstract representations of data using multiple processing layers [1]. Remarkable successes have been achieved by deep neural networks (DNNs) across a wide range of fields, including computer vision [6], healthcare and bio medicine [4], [5] and natural language processing [7]. This problem is further compounded by the fact that selecting the best set of nonlinear activation functions can be difficult [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Wanqing Zhao.

Several factors can influence the training process of a neural network, including weight initialization, activation functions, and network architecture [8]. Some activation functions or network architectures can cause information loss or increase the time needed to train a DNN [3], [8], [9], [10]. The ReLU activation function has gained widespread popularity due to its simplicity and effectiveness, but hand-designed activation functions have also been proposed to replace it [11], [12], [13], [14], [15], [16].

To address the challenge of selecting the best nonlinear activation functions, trainable activation functions have been proposed [3], [17]. Chung et al. [3] used a Taylor series approximation of sigmoid, tanh, and ReLU as an

initialization point for their activation functions and trained the coefficients of the approximation to optimize training. In this manuscript, we introduce a new neural network design termed SWAG. Within this structure, the activation functions across all layers are constructed from a polynomial basis, adjusted by a factorial coefficient. Contrary to the usual approach, we keep our activation functions static, ensuring they consistently form a polynomial basis. The arrangement of the hidden layers is systematic, with each layer containing k sub-layers that utilize polynomial activation functions. These are subsequently joined by a Concatenate layer and followed by a layer that employs a linear activation function.

The rest of the manuscript is organized as follows: Section II describes the mathematical foundations and architecture of SWAG, Section III presents the experiments conducted, and Section IV discusses the results and future work.

II. RELATED WORK

The evolution of neural network architectures and the diversification of activation functions have been pivotal in advancing the field of machine learning. This section delves into seminal works that have contributed to these developments.

A. ADAPTIVE SPLINE NEURAL NETWORKS (ASNNS)

The introduction of Adaptive Spline Neural Networks (ASNNs) by *Guarnieri* et al. [31] marked a significant innovation in neural network design. ASNNs utilize adaptive spline activation functions, explicitly exploiting the advantageous properties of cubic Catmull-Rom splines. This approach enhances the network's expressive power and optimizes its performance, making ASNNs particularly adept at handling complex modeling tasks with increased accuracy and efficiency.

B. CHEBYSHEV-POLYNOMIALS-BASED UNIFIED NEURAL NETWORK

Another groundbreaking contribution is the development of a Chebyshev-Polynomials-Based (CPB) unified neural network by Lee and Jeng [32]. This architecture cleverly combines feedforward and recurrent neural mechanisms through the incorporation of Chebyshev polynomials. The CPB neural network remarkably accelerates the learning process while preserving the core approximation capabilities intrinsic to traditional neural network models. This dual advantage facilitates a broader application spectrum, ranging from complex system modeling to predictive analytics.

C. ENHANCEMENTS IN GESTURE DETECTION

The realm of gesture detection witnessed notable advancements through applying Chebyshev polynomial neural networks. Research conducted by Zhiqi [33] showcased how these networks could outperform conventional Back Propagation (BP) neural networks in recognizing and interpreting dynamic gestures. By leveraging polynomial-based activation functions, this approach significantly improves the model's sensitivity and accuracy in capturing the nuances of human gestures, thereby enhancing interaction technologies.

D. ORTHONORMAL HERMITE POLYNOMIALS FOR ENHANCED ACCURACY

The innovative use of orthonormal Hermite polynomials in single-hidden-layer neural networks by Ma and Khorasani [34] represents a leap forward in accuracy enhancement. This design principle capitalizes on the precise approximation capabilities of Hermite polynomials, facilitating the construction of neural networks that can achieve superior performance levels in tasks requiring high accuracy, such as pattern recognition and data fitting.

E. LEGENDRE POLYNOMIALS IN FUNCTION APPROXIMATION

Emphasizing orthogonal functions, Yang and Tseng [35] proposed a single-layer neural network model that harnesses the power of Legendre polynomials for function approximation. This novel approach underlines the effectiveness of Legendre polynomials in enhancing the network's ability to generalize from input data, thereby improving its predictive performance across various applications.

F. TRAINABLE ACTIVATION FUNCTIONS IN IMAGE CLASSIFICATION

Zhaohe Liao introduced a novel concept where the activation function is subject to optimization alongside the neural network's weights. This approach, detailed in Trainable Activation Function in Image Classification by Liao, represents a significant departure from traditional fixed activation functions, promising improvements in model adaptability and performance [36].

G. DEEP NEURAL NETWORK USING TRAINABLE ACTIVATION FUNCTIONS

Chung, Lee, and Park further explore the concept of trainable activation functions in their work published by IEEE [3]. They demonstrate that deep neural networks equipped with activation functions that can be trained exhibit enhanced learning capabilities and generalization across various tasks. This research underscores the potential of adaptable activation mechanisms in advancing neural network designs.

These contributions emphasize the dynamic relationship between neural network architectures and activation functions. Through the exploration and integration of polynomial-based designs along with inventive methods such as trainable activation functions, researchers have notably pushed the limits of neural networks, opening avenues for future breakthroughs in artificial intelligence.

III. METHODS

A. REPRESENTATION OF BASIS FUNCTIONS

Suppose that we have a data set $\{\mathbf{x}_j\}$ for $1 \leq j \leq n$ and labels $\{\mathbf{y}_j\}$. We would like to find a function $f(x)$ such that

$f(x_j) = y_j$ for all $1 \leq j \leq n$. The Stone-Weierstrass approximation theorem states that any continuous real-valued function on a compact set can be uniformly approximated by a polynomial. Formally:

Theorem 1 (Stone-Weierstrass Approximation Theorem): Suppose f is a continuous real-valued function defined on any closed and bounded subset $X \in^m$ for any $m \in \mathbb{N}$. For every $\epsilon > 0$, there exists a polynomial $p(x_1, x_2, \ldots, x_m)$ such that $|f(x_1, x_2, \ldots, x_m) - p(x_1, x_2, \ldots, x_m)| < \epsilon$ for any $(x_1, x_2, \ldots, x_m) \in X$ The simplicity of polynomial systems makes them very attractive analytically and computationally. They are easy to form and have well-understood properties. The use of polynomials of a given degree as activation functions for all neurons in a single layer seems to be mathematically discouraged in traditional neural network settings because they are not universal approximators. Particularly, Leshno et al. [18] proved the following theorem:

Theorem 2: Let M be the set of functions which are $L^{\infty}_{loc}()$ with the property that the closure of the set of points of discontinuity of any function in M has zero Lebesgue measure. Let $\sigma \in M$. Then for a fixed $x \in^n$,

$$span\{\sigma\{w \cdot x + \Theta\} : w \in^n, \Theta \in\}$$

is dense in $\mathbb{C}(^n)$ if and only if σ is not an algebraic polynomial almost everywhere.

This theorem implies that fully connected feedforward neural networks with a sufficient number of neurons are universal approximators if and only if the activation functions are not polynomials. We note that in this traditional setting, it is assumed that the activation function is the same for every neuron in a given layer. We now give the following extension of the Stone-Weierstrass approximation theorem

Corollary 1: Let $\sigma_p = \frac{x^p}{p!}$ for $0 \leq p < \infty$. Then

$$span\{\sigma_p\{w \cdot x + \Theta\} : w \in^n, \Theta \in\}$$

is dense in $\mathbb{C}(X^n)$ where $X^n \in^n$ is a compact set.

Proof: Notice that $\{\sigma_p\}_{p=0}^{\infty}$ is a basis for the vector space of polynomials over. So since we know that polynomials are dense in $\mathbb{C}(X^n)$ by the Stone-Weierstrass approximation theorem, the result follows.

This corollary implies that if we allow a diverse set of polynomial activation functions in a particular layer, we will still have the result of universal approximation capabilities of feedforward neural networks. Using the same framework as Leshno et al.. [18], in which the output was assumed to be in ^{n}, an extension to higher dimensions can be easily obtained by re-defining $\sigma_p\{w\}$ as a pointwise operation that takes each element of w and raises it to the p^{th} power, e.g. given $w = [2, 3]$, then $\sigma_4\{w\} = [2^4, 3^4]$.

B. ARCHITECTURE OF THE SWAG ALGORITHM

Let $x_j \in^d$ be a data point in our data set $\{x_j\}_{j=1}^n$.

0 Normalize data to be in the interval [0,1].

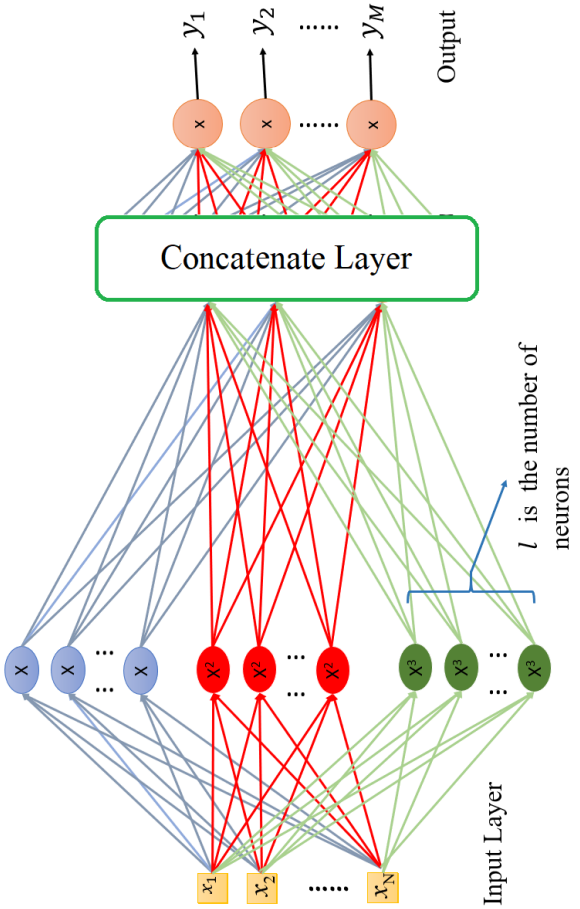


FIGURE 1. Implementation of the SWAG architecture with three groups of monomials of powers 1 through 3, and two layers.

1 Create the first polynomial layer as follows:

1.1 Choose a k for the number of of polynomial terms used (k is a hyperparameter of the model).

1.2 Choose l for the number of neurons that correspond to each monomial of the 1^{st} layer (l is a hyperparameter of the model).

1.3 Create k fully-connected layers with l neurons in each layer, all with x_j as their inputs.

1.3.1 The p^{th} fully-connected layer for $1 \leq p \leq k$ is defined by $\sigma_p\{Wx + b\}$ for $W \in l \times d$, $b \in^l$, and σ_p as defined above.

1.3.2 Initialization of weights are random and drawn from $\mathcal{N}(0, 1)$, a Gaussian distribution with mean 0 and standard deviation 1.

1.4 Vertically concatenate the k layers to form a vector of length $l \cdot k$

2 Create a layer with a linear activation function. This is considered the second layer of SWAG.

3 To add a third and fourth layer, repeat the structure of the previous two layers with the input of the third layer as the output of the second layer. If a third and fourth layer

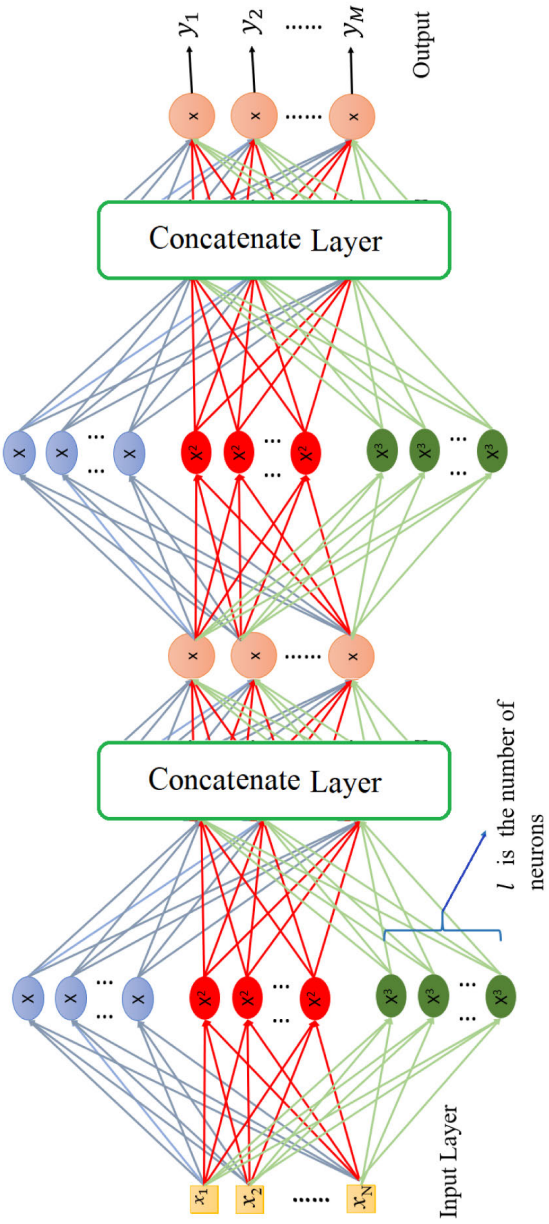


FIGURE 2. Implementation of the SWAG architecture with three groups of monomials of powers 1 through 3, and four layers.

is added, then the first dimension of the matrix used in the second layer is a hyperparameter of the model.

4 Continue to add layers in this pattern as desired.

5 The matrix used for the final linear activation layer will have its first dimension be the dimension of the output vector.

Figure 1 is a diagram of an example of SWAG using two layers, and Figure 2 is a diagram of an example of SWAG with four layers.

C. OPTIMIZED SWAG

The previous architecture faced a challenge that required computing higher degrees of X. To address this issue.

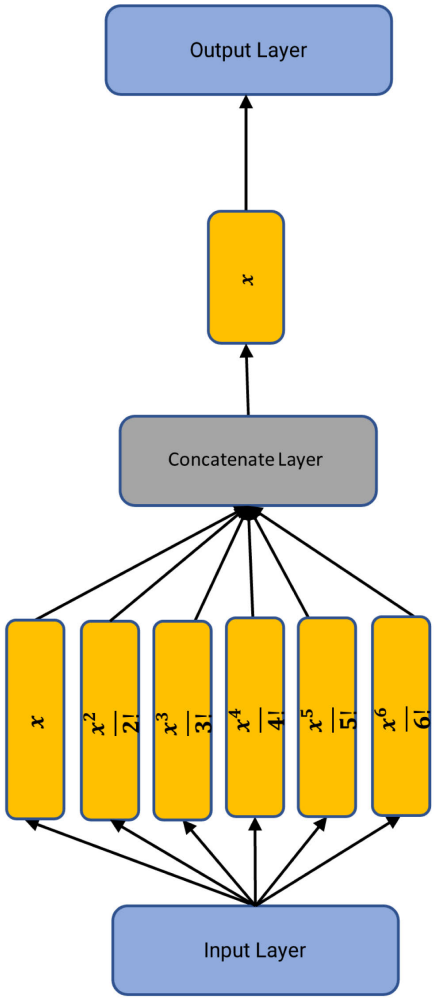


FIGURE 3. Original SWAG in different format.

We propose a new architecture that optimizes the previous flow.

When we have a Taylor series with terms of different degrees, one useful technique is to use the lower-degree terms to compute the higher-degree terms. Specifically, if we have already computed X and X^2, we can use these terms to calculate all the higher-degree terms in the series. For example, to calculate X^3, we can multiply X by X^2. Similarly, to calculate X^4, we can multiply X^2 by itself. This method allows us to compute all the terms in the series without having to compute each term separately, which can be very time-consuming for high-degree polynomials. Therefore, by utilizing X and X^2, we can simplify the process of computing the Taylor series and obtain accurate approximations for the original function.

Another benefit of this architecture is that the derivative of X is constant, and the derivative of X^2 is $2X$, making them easy to compute during backpropagation.

Figure 4 and Figure 3 illustrate the architectural differences between the original SWAG and its optimized version, presented in distinct formats.

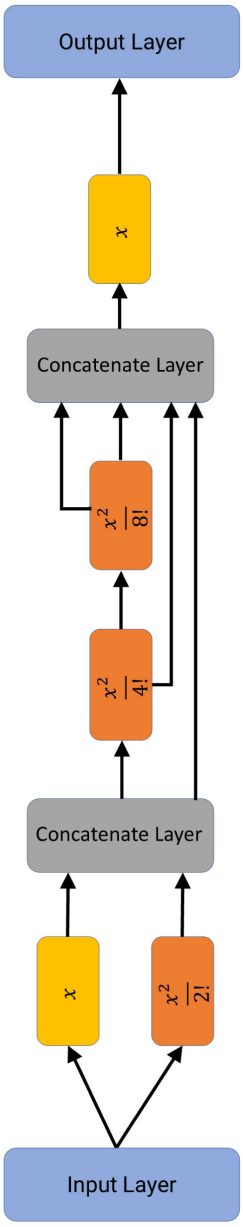


FIGURE 4. Optimized version of SWAG.

D. DATASET OVERVIEW

This study utilizes four fundamental datasets, widely recognized in the field of machine learning research for classification tasks: the Pima Indians Diabetes, Sonar Signal, Ionosphere Radar, and the MNIST Handwritten Digit datasets. Each dataset has been extensively documented in the literature. Notably, [39] discussed the Sonar dataset, [38] explored the Ionosphere dataset, [37] examined the Pima Indians Diabetes dataset, and [40] described the MNIST dataset.

1) PIMA INDIANS DIABETES DATASET

Comprising 768 entries, the Pima Indians Diabetes Dataset is sourced from the corresponding Diabetes Database, presenting instances with eight attributes plus a class label. The attributes, representing essential health metrics such as glucose concentration, blood pressure, skin thickness, insulin levels, BMI, diabetes pedigree function, age, and number of pregnancies, are utilized to predict the likelihood of diabetes onset within five years among Pima Indian women [37].

TABLE 1. Hyperparameters for random search.

Hyperparameter	Values
Batch size	10
Epochs	10
Learning rate	0.01
Dropout rate	0.1
Activation function for layer 1	softmax, relu, tanh, linear
Activation function for layer 2	softmax, relu, tanh, linear
Activation function for layer 3	softmax, relu, tanh, linear
Weight initialization	uniform
Number of neurons in layer 1	50, 100, 150
Number of neurons in layer 2	20, 40, 80

2) SONAR SIGNAL DATASET

The Sonar Signal Dataset consists of 208 observations, each described by 60 attributes, derived from sonar signals rebounding off various materials. These attributes, documenting energy levels across several frequency bands and categorizing observations as 'R' (rock) or 'M' (mine), aim to enable the accurate classification of sonar echoes from the seabed, thus showcasing the model's proficiency in pattern recognition from signal data [39].

3) IONOSPHERE RADAR DATASET

This dataset features 351 instances, each with 34 attributes from radar observations of the ionosphere, distinguishing between ''good'' and ''bad'' structural detections. A ''good'' rating indicates that the radar successfully identified a structure, whereas ''bad'' suggests the opposite. This differentiation aids in evaluating the model's capacity to discern various atmospheric phenomena [38].

4) MNIST HANDWRITTEN DIGIT DATASET

With 70,000 handwritten digits, the MNIST dataset splits into a training set of 60,000 images and a testing set of 10,000 images. Each 28 $\times$ 28 pixel image depicts a digit from 0 to 9, challenging the model to accurately classify handwritten digits, thereby testing its image recognition abilities [40].

5) DATA PREPROCESSING AND PREPARATION

Each dataset was subjected to preprocessing protocols to align with the SWAG model requirements. The Pima Indians Diabetes, Sonar, and Ionosphere datasets underwent normalization to a 0-1 scale to remedy scale variations, with missing data points imputed based on the median of relevant attributes. The MNIST dataset's pixel values were similarly normalized to the [0,1] range, and its labels were

adapted to a one-hot encoding scheme, optimizing the data for classification tasks.

Collectively, these datasets afford a thorough examination of the SWAG model's classification accuracy in various contexts, from medical diagnostics and sonar signal parsing to atmospheric condition discernment and handwritten digit identification. These datasets' diverse nature and application areas underpin a solid basis for evaluating the model's versatility, efficiency, and general effectiveness in classification tasks.

IV. RESULTS

In the field of machine learning, there are three primary methods for hyperparameter tuning: human-designed machine learning models [19], AutoML [20], and the greedy search algorithm [29].

Human-designed machine learning models rely on expert knowledge and experience to select hyperparameters and construct the model architecture [30]. This approach necessitates extensive domain expertise, and the resulting models may not always be optimal for a specific use case [30].

AutoKeras, an automated machine learning (AutoML) tool, employs a neural architecture search algorithm to generate deep learning models automatically [24]. This method requires less human intervention and is generally faster than manual design. However, it may only consistently produce the optimal architecture for some use cases [24].

The greedy search algorithm, another method, seeks optimal hyperparameters within a predefined deep learning architecture. It involves testing various hyperparameter combinations and selecting the best-performing one, although it can be computationally intensive and time-consuming [29].

This study compared our proposed model's performance with random search, AutoKeras, and human-designed machine learning models. Our objective was to demonstrate the effectiveness of our model and its ability to outperform both traditional and automated hyperparameter optimization techniques and expert-designed models. This comparison provides insights into each method's strengths and limitations, highlighting the importance of choosing the most suitable hyperparameter tuning approach for a specific task [26].

We generated a dataset containing random numbers between 0 and 1 to conduct this analysis, which we divided into training and test sets. We also created three random mathematical functions to evaluate algorithm performance across different data types. In addition to manually designing two deep learning architectures, we investigated random search and AutoKeras for comparison purposes [25].

For the AutoKeras implementation, the max trials parameter was set to 10 [28]. We also trained SWAG with $l = 50$, $k = 8$, and three layers, where the first dimension of the second layer was set to 50. The standard mean squared loss function with the Adam optimizer was employed to test model accuracy [23].

For the random search, the following hyperparameters were applied:

For the AutoKeras implementation, the max trials parameter was set to 10. We also trained SWAG with $l = 50$, $k = 8$, and three layers, where the first dimension of the second layer was set to 50. The standard mean squared loss function with the Adam optimizer was employed to test model accuracy [23].

We generated three random functions by selecting coefficients for four distinct terms: a power function, a sigmoid function, an exponential function, and a logarithmic function. Each function was defined as a summation of these four terms using the randomly selected coefficients. The generated function is given as follows:

$$F(x) = a_0 x^{a_1} + a_2 \left(\frac{1}{1+e^{-a_3 x}}\right) + a_4 e^{a_5 x \times 0.1} + a_6 \log(a_7 x) \tag{1}$$

In this equation, $a_0, a_2, a_4, a_6 \in (-10, 10)$ and $a_1, a_3, a_5, a_7 \in (1, 50)$ are coefficients.

The randomly generated functions for this experiment are as follows:

$$F_1 = \frac{1}{2}x^2 - 5\left(\frac{1}{1+e^x}\right) \tag{2}$$

$$F_2 = 6x^5 - 3\left(\frac{1}{1+e^x}\right) + e^x - 9\log_{10}(x) \tag{3}$$

$$F_3 = 22x^{20} - \frac{1}{1+e^x} + 2e^x + 5\log_{10}(x) \tag{4}$$

For $1 \leq i \leq 3$,

$$Y_{i_{\text{train}}} = F_i(X_{\text{train}}) \tag{5}$$

$$Y_{i_{\text{test}}} = F_i(X_{\text{test}}) \tag{6}$$

A. DETAILED ANALYSIS OF MODEL PERFORMANCE FOR FUNCTION APPROXIMATION TASKS

To provide a more in-depth analysis of the performance of the five different models in approximating the three functions (F_1, F_2, and F_3), we will consider the following aspects:

- Loss value trends and convergence
- Model complexity and computational efficiency
- Robustness and generalization capabilities

1) LOSS VALUE TRENDS AND CONVERGENCE

- **Architecture 1** demonstrates a steady decline in loss values for F_3 and F_2. However, the loss values remain relatively constant for F_1, showing no significant improvement. Overall, Architecture 1 exhibits suboptimal performance across all three functions.
- **Architecture 2** shows limited improvement in loss values for all three functions. Loss values plateau or slightly increase over time, indicating poor performance in approximating the functions.
- **AutoKeras** exhibits a significant decrease in loss values across all three functions, demonstrating quick convergence and strong potential for function approximation. This model outperforms Architectures 1 and 2 by a considerable margin.

TABLE 2. Comparison of final loss values between SWAG and AutoKeras for functions 4-13.

Function	SWAG Loss	AutoKeras Loss
Function 4	0.02	0.013
Function 5	0.116	0.2
Function 6	0.037	0.501
Function 7	0.03	0.067
Function 8	0.121	0.462
Function 9	0.047	1.416
Function 10	0.326	0.604
Function 11	1.384	2.041
Function 12	0.012	0.028
Function 13	0.161	5.02

- **Random Search** achieves the lowest loss values among all models for all three functions, indicating excellent performance. However, the convergence is slower than that of AutoKeras and SWAG, taking more epochs to achieve optimal loss values.
- **SWAG** displays a remarkable reduction in loss values over the epochs for all three functions, achieving low loss values and demonstrating excellent convergence properties.

2) MODEL COMPLEXITY AND COMPUTATIONAL EFFICIENCY

- Architecture 1 and Architecture 2, despite their different architectures, both exhibit suboptimal performance in approximating the given functions. This suggests that they may not be well-suited for the task, and other architectures should be explored.
- AutoKeras leverages an automated search for an optimal model architecture, resulting in improved performance across all three functions. This method provides a good balance between model performance and computational efficiency.
- Random Search, although yielding the best performance, is computationally expensive, taking 200 times longer than the other models. This could be a significant drawback when dealing with larger datasets or more complex functions.
- SWAG balances model performance and computational efficiency, achieving low loss values and demonstrating quick convergence.

3) ROBUSTNESS AND GENERALIZATION CAPABILITIES

- AutoKeras and SWAG consistently show promising results across all three functions, suggesting that these models have robust and generalizable architectures. This indicates that they are likely to perform well on a wider range of function approximation tasks.
- Architecture 1 and 2, on the other hand, exhibit poor performance and limited generalization capabilities, making them unsuitable for the given tasks.
- While Random Search achieves the best performance, its computational inefficiency and slower convergence may limit its general applicability, especially for larger or more complex tasks.

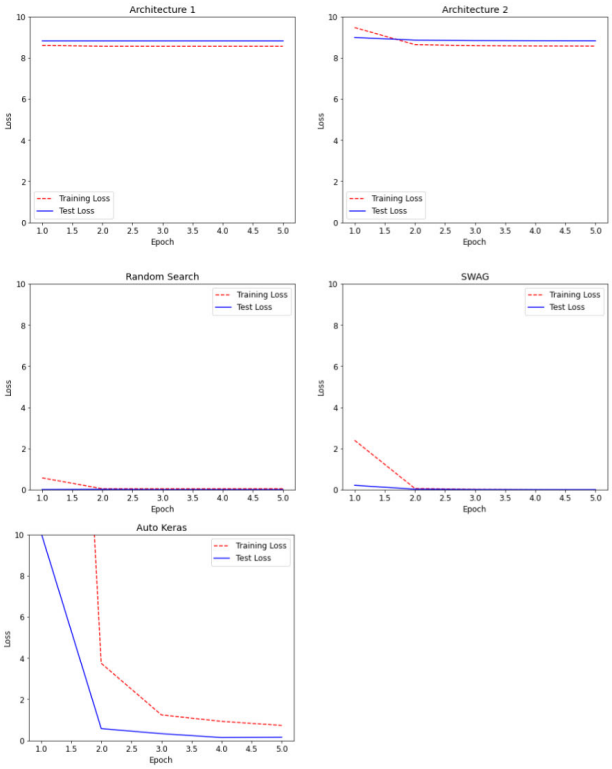


FIGURE 5. $F_1 = \frac{1}{2}x^2 - 5\left(\frac{1}{1+e^x}\right)$: **Five plots show training and validation losses over epochs for models approximating Function 1.**

In conclusion, AutoKeras and SWAG emerge as the most promising candidates for approximating the three functions, providing a good balance between model performance, computational efficiency, and generalization capabilities. Random Search achieves the best performance in terms of loss values but is considerably slower, limiting its practical applicability. Architectures 1 and 2 exhibit suboptimal performance and may not be suitable for the given tasks.

4) COMPARATIVE ANALYSIS OF SWAG AND AUTOKERAS MODELS BASED ON FINAL LOSS VALUES

This section presents a comparative analysis of the SWAG and AutoKeras models, focusing on each function's final test loss values. The results are outlined below:

In Equations (5) to (14), we can observe the various random functions that were generated for our experiment.

In summary, AutoKeras exhibits superior performance for Function 4, while SWAG outperforms AutoKeras for Functions 5-13. Notably, the average runtime for AutoKeras is approximately eight times slower than that of SWAG, which is an important factor to consider when selecting a model for a specific application.

$$f_4(x) = -5x^7 + \frac{1}{1+e^{-6x}} + e^{-4x\times 0.01} + \log(4x), \tag{7}$$

$$f_5(x) = -4x^{10} + \frac{1}{1+e^{-4x}} + e^{4x\times 0.01} + 7\log(3x), \tag{8}$$

$$f_6(x) = -x^6 + \frac{2}{1+e^{-7x}} - e^{-x\times 0.01} + 5\log(7x), \tag{9}$$

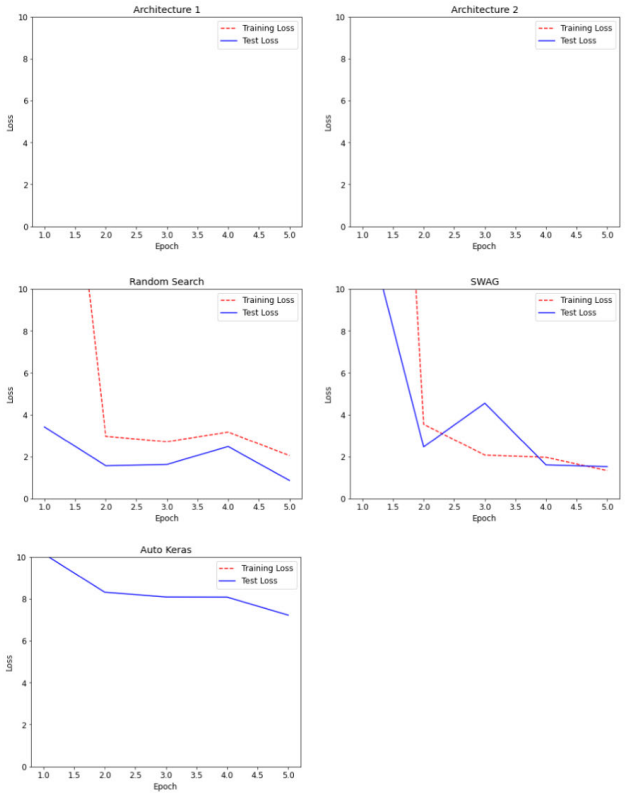


FIGURE 6. $F_2 = 6x^5 - 3\frac{1}{1+e^x} + e^x - 9\log_{10}(x)$: Five plots illustrate training and validation losses for models on Function 2. Architecture 1 and 2 plots excluded due to losses beyond the displayed range.

$$f_7(x) = -3x^{10} + 5\frac{e^{2x}}{1+e^{8x}} - e^{-x\times 0.01} + 3\log(6x), \quad (10)$$

$$f_8(x) = 10x^6 - 2\frac{e^{4x}}{1+e^{8x}} - e^{-x\times 0.01} + 6\log(7x), \quad (11)$$

$$f_9(x) = 8x^9 + 5\frac{e^{8x}}{1+e^{-x}} + e^{-x\times 0.01} + 3\log(7x), \quad (12)$$

$$f_{10}(x) = -8x^8 - 9\frac{e^{x}}{1+e^{-9x}} + 5e^{10x\times 0.01} + 2\log(8x), \quad (13)$$

$$f_{11}(x) = 6x^{10} + 8\frac{e^{5x}}{1+e^{3x}} - e^{-10x\times 0.01} + 2\log(2x), \quad (14)$$

$$f_{12}(x) = 5x^5 + 10\frac{e^{6x}}{1+e^{-2x}} - e^{-x\times 0.01} + 8\log(5x), \quad (15)$$

$$f_{13}(x) = -7x^5 + 10\frac{e^{6x}}{1+e^{-2x}} - 6e^{-x\times 0.01} + 4\log(10x). \quad (16)$$

B. CLASSIFICATION EXPERIMENT: APPLICATION OF SWAG ON DIVERSE DATASETS

Our experiment assessed the efficacy of two machine learning approaches, SWAG and AutoKeras, through a rigorous 10-fold cross-validation process across three diverse datasets: Sonar, Ionosphere, and Pima Indians Diabetes. Unlike previous methods that relied on time-consuming random searches or human-designed architectures, our comparison focused on the innovative SWAG algorithm and AutoKeras

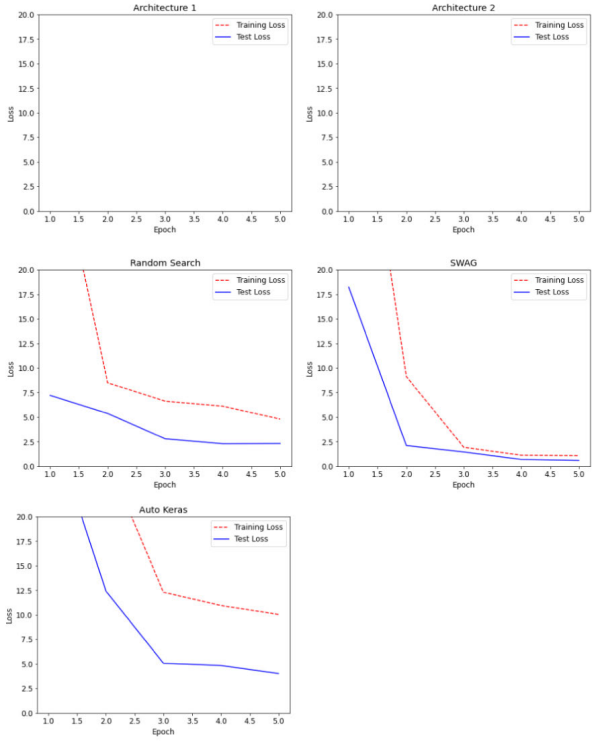


FIGURE 7. $F_3 = 22x^{20} - \frac{1}{1+e^x} + 2e^x + 5\log_{10}(x)$: Figure with five plots showing training and validation losses over epochs for each model on Function 2. Plots for Architecture 1 and 2 are omitted due to losses exceeding the displayed range.

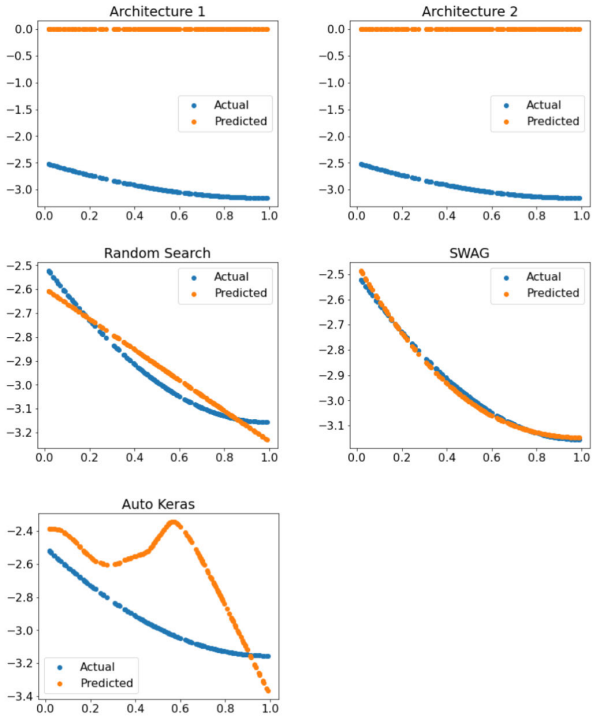


FIGURE 8. $F_1 = \frac{1}{2}x^2 - 5\left(\frac{1}{1+e^x}\right)$: Chart of models' approximations vs. original Function 1, with input on x-axis and output on y-axis.

TABLE 3. SWAG test loss for random functions.

Fn	SWAG Test Loss
Fn 4	[5.04, 1.35, 1.00, 0.73, 0.12, 0.03, 0.03, 0.02, 0.02, 0.02]
Fn 5	[20.57, 3.05, 0.99, 0.43, 0.24, 0.18, 0.14, 0.13, 0.12, 0.12]
Fn 6	[0.34, 0.04, 0.04, 0.04, 0.04, 0.04, 0.04, 0.03, 0.03, 0.04]
Fn 7	[4.83, 0.29, 0.10, 0.04, 0.04, 0.03, 0.03, 0.03, 0.03, 0.03]
Fn 8	[3.49, 0.20, 0.17, 0.16, 0.19, 0.13, 0.13, 0.14, 0.13, 0.12]
Fn 9	[20.79, 1.11, 0.67, 0.35, 0.27, 0.23, 0.16, 0.11, 0.07, 0.05]
Fn 10	[11.71, 1.01, 0.64, 0.58, 0.56, 0.45, 0.38, 0.38, 0.32, 0.33]
Fn 11	[58.57, 3.51, 1.48, 2.21, 2.35, 1.44, 0.87, 0.85, 0.69, 1.38]
Fn 12	[9.35, 0.31, 0.12, 0.07, 0.05, 0.03, 0.02, 0.02, 0.01, 0.01]
Fn 13	[2.53, 0.98, 0.61, 0.42, 0.47, 0.30, 0.26, 0.21, 0.19, 0.16]

TABLE 4. AutoKeras test loss for random functions.

Fn	AutoKeras Test Loss
Fn 4	[0.07, 0.04, 0.05, 0.04, 0.04, 0.03, 0.02, 0.02, 0.02, 0.01]
Fn 5	[0.82, 0.69, 0.48, 0.57, 0.35, 0.43, 0.27, 0.34, 0.25, 0.20]
Fn 6	[3.79, 3.69, 3.62, 3.57, 3.50, 3.44, 3.38, 3.34, 1.61, 0.50]
Fn 7	[0.25, 0.12, 0.13, 0.06, 0.06, 0.10, 0.04, 0.04, 0.06, 0.07]
Fn 8	[1.93, 1.55, 1.23, 1.00, 0.87, 0.72, 0.64, 0.60, 0.50, 0.46]
Fn 9	[1.77, 1.74, 1.74, 1.70, 1.70, 1.65, 1.63, 1.52, 1.54, 1.42]
Fn 10	[2.02, 1.22, 1.42, 1.48, 1.60, 1.30, 1.43, 0.52, 0.46, 0.60]
Fn 11	[5.23, 4.61, 4.00, 3.77, 3.47, 2.89, 2.59, 2.22, 2.21, 2.04]
Fn 12	[0.21, 0.02, 0.14, 0.10, 0.08, 0.04, 0.01, 0.02, 0.03, 0.03]
Fn 13	[6.87, 6.43, 6.23, 5.62, 5.98, 5.28, 5.02, 5.37, 4.82, 5.02]

due to their automation and efficiency in design selection. This analysis presents detailed insights into each dataset's outcomes.

Experimental Setup: We conducted our evaluation using a consistent experimental setup across all datasets. Specifically, the number of training epochs was fixed at 5, and AutoKeras was allowed a maximum of 5 trials to optimize its architecture. This setup ensures a fair comparison between SWAG and AutoKeras regarding performance and efficiency.

1) ANALYSIS ON THE SONAR DATASET

The Sonar dataset, introduced by [39], served as the first test case for our comparison. Our findings are summarized as follows:

- Accuracy and Performance: SWAG showcased superior accuracy, with a mean of 75.43% (±8.10%), outshining AutoKeras's 61.98% (±10.89%). This demonstrates SWAG's robustness in handling the Sonar dataset's complex signal patterns.
- Computational Efficiency: SWAG's computational time was significantly less (20.21 seconds) compared to AutoKeras (333.03 seconds), highlighting SWAG's efficiency and its potential for real-time applications.

2) EVALUATION ON THE IONOSPHERE DATASET

Upon applying the algorithms to the Ionosphere dataset [38], the following observations were made:

- Accuracy Excellence: Continuing its streak, SWAG achieved 87.45% (±4.48%) accuracy on the Ionosphere dataset versus AutoKeras's 81.78% (±8.46%). This underscores SWAG's ability to model the atmospheric disturbances captured in this dataset effectively.
- Speed Advantage: SWAG maintained an execution time (21.63 seconds), far outpacing AutoKeras (305.51 seconds), which speaks to its computational frugality and suitability for high-frequency data processing tasks.

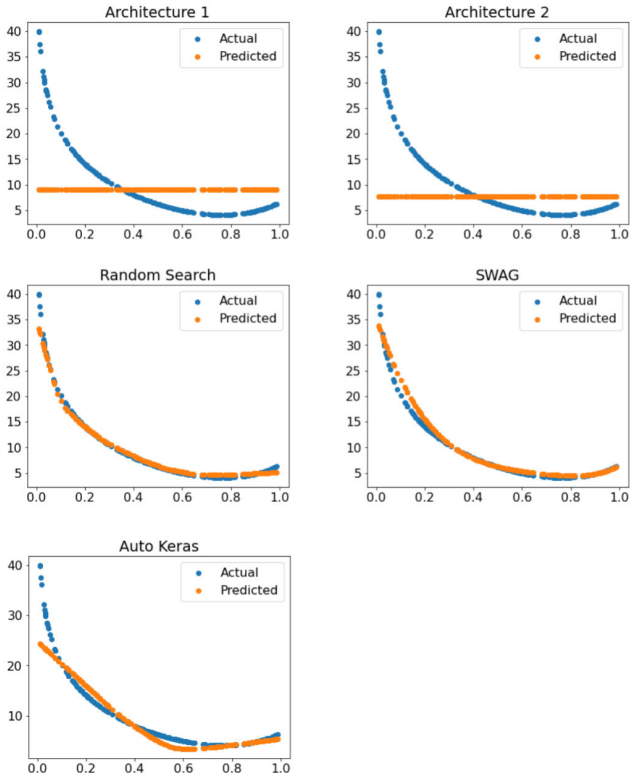


FIGURE 9. Graph comparing model approximations to original F_2, with input on the x-axis and F_2 output on the y-axis.

3) PERFORMANCE ON THE PIMA INDIANS DIABETES DATASET

The evaluation of the Pima Indians Diabetes dataset [37] yielded the following results:

- Comparable Accuracy: Both algorithms performed similarly, with SWAG slightly edging out with 76.31% (±3.41%) accuracy against AutoKeras's 76.17% (±2.86%). This parity suggests that both methods are equally adept at managing the clinical data variability inherent in the diabetes dataset.
- Efficiency Gains: SWAG completed its analysis in 39.49 seconds, compared to AutoKeras's 244.52 seconds, reinforcing the efficiency pattern observed in previous datasets.

4) OVERALL OBSERVATIONS

The comparative analysis between SWAG and AutoKeras, as detailed in Tables 5 and 6, reveals insightful distinctions in their performance across various datasets. In terms of accuracy, SWAG consistently demonstrates comparable or superior results when measured against AutoKeras. Specifically, for the Pima Indians Diabetes dataset, SWAG and AutoKeras exhibit nearly identical average accuracies, with

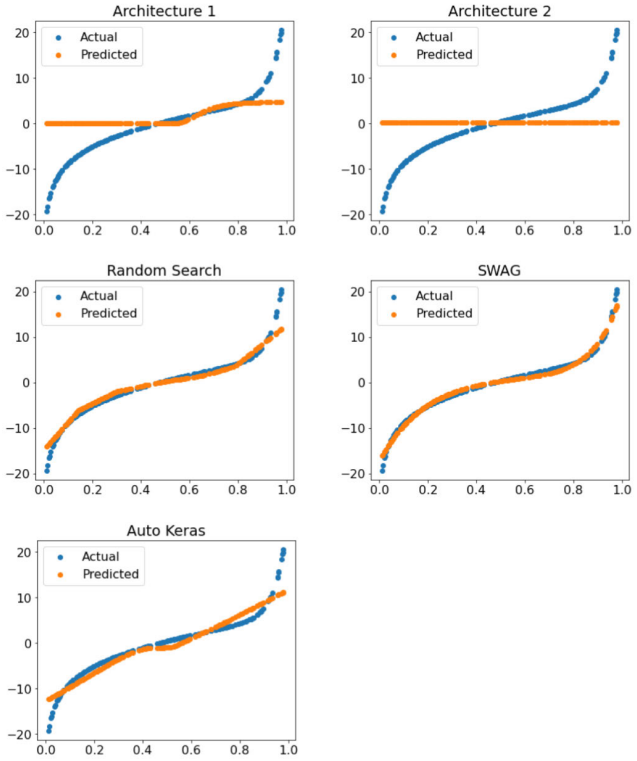


FIGURE 10. Graph of model approximations vs. original F_3, with input on the x-axis and F_3 output on the y-axis.

SWAG slightly leading at 76.31% compared to AutoKeras's 76.17%. This trend is more pronounced in the Ionosphere and Sonar datasets, where SWAG outperforms AutoKeras with average accuracies of 87.45% and 75.43% against 81.78% and 61.98%, respectively. The variance in performance is notably lower for SWAG across these datasets, indicating a more stable and reliable prediction model.

Moreover, the efficiency of SWAG is markedly highlighted in the analysis of runtime performance. SWAG completes its processing significantly faster than AutoKeras across all evaluated datasets, with runtime measurements for SWAG (ranging from 20.21 to 39.49 seconds) substantially lower than those for AutoKeras (ranging from 244.52 to 333.03 seconds). This substantial difference underscores SWAG's potential for applications requiring not only high accuracy and stability but also efficiency in computational resource usage.

This comparative evaluation underscores the efficacy of SWAG in achieving high accuracy with reduced variability and enhanced computational efficiency, making it a compelling choice for diverse machine learning applications.

C. FINAL EXPERIMENT: EVALUATING SWAG ON THE MNIST HANDWRITING DATASET

In the final phase of our investigation, the SWAG framework was applied to the MNIST handwriting dataset [40], a benchmark collection of 70,000 images of handwritten digits ranging from 0 to 9. This dataset served as the basis

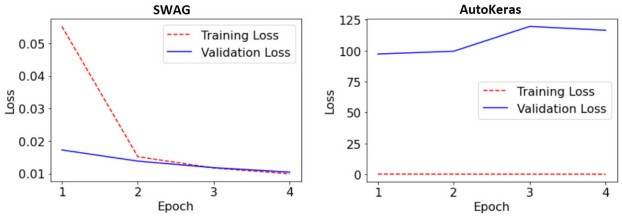


FIGURE 11. This figure depicts training and validation loss trends over epochs for SWAG and AutoKeras on MNIST.

for evaluating the efficacy of SWAG relative to AutoKeras. During preprocessing, images were transformed into vectors of dimension $784 \times 1$to facilitate their use as input. Parameters for the SWAG algorithm were configured with $l = 500$, $k = 7$, and the model architecture was streamlined to incorporate only two layers. The dataset allocation consisted of a training subset encompassing 60,000 images, alongside a test subset comprising 10,000 images.

1) COMPREHENSIVE EVALUATION OF SWAG AND AUTOKERAS ON THE MNIST DATASET

Our in-depth analysis focuses on contrasting the SWAG and AutoKeras models across various metrics on the MNIST dataset:

- Trends in loss values and model convergence
- Accuracy metrics, including both training and validation accuracy
- Training duration and computational efficiency

Trends in Loss Values and Model Convergence:

- **SWAG Model:** Exhibited a consistent decrease in training and validation loss over the epochs, indicating effective learning and convergence towards an optimal solution. The training loss reduced from 0.0552 to 0.0099, alongside a validation loss decrease from 0.0172 to 0.0104, showcasing the model's ability to generalize well as illustrated in Figure 11.
- **AutoKeras Model:** Demonstrated reduced training loss but increased validation loss, suggesting potential overfitting. Specifically, training loss dropped from 0.3001 to 0.2251, while validation loss escalated from 97.3062 to 116.4989, indicating a divergence in model generalization capability.

Accuracy and Validation Accuracy:

- **SWAG Model:** Achieved a remarkable training accuracy of 98.36% and validation accuracy of 97.70%, underlining its superior performance on the MNIST dataset.
- **AutoKeras Model:** Attained lower accuracy levels compared to SWAG, with a training accuracy of 92.97% and validation accuracy of 88.38%, highlighting its limitations in effectively classifying the MNIST digits.

Training Time and Computational Efficiency:

- **SWAG Model:** Notably efficient, requiring only 7.32 seconds to complete the training over four epochs.

TABLE 5. Performance comparison of SWAG and AutoKeras on different datasets.

Fold	Dataset/Method	SWAG	AutoKeras
1	Pima Indians Diabetes	74.03%	79.22%
2	Pima Indians Diabetes	75.32%	76.62%
3	Pima Indians Diabetes	80.52%	76.62%
4	Pima Indians Diabetes	79.22%	77.92%
5	Pima Indians Diabetes	74.03%	72.73%
6	Pima Indians Diabetes	70.13%	71.43%
7	Pima Indians Diabetes	80.52%	76.62%
8	Pima Indians Diabetes	74.03%	77.92%
9	Pima Indians Diabetes	75.00%	72.37%
10	Pima Indians Diabetes	80.26%	80.26%
1	Ionosphere	91.67%	77.78%
2	Ionosphere	80.00%	74.29%
3	Ionosphere	94.29%	91.43%
4	Ionosphere	91.43%	91.43%
5	Ionosphere	85.71%	85.71%
6	Ionosphere	91.43%	82.86%
7	Ionosphere	82.86%	77.14%
8	Ionosphere	85.71%	85.71%
9	Ionosphere	82.86%	88.57%
10	Ionosphere	88.57%	62.86%
1	Sonar	71.43%	61.90%
2	Sonar	80.95%	52.38%
3	Sonar	71.43%	61.90%
4	Sonar	76.19%	61.90%
5	Sonar	90.48%	42.86%
6	Sonar	76.19%	85.71%
7	Sonar	80.95%	71.43%
8	Sonar	66.67%	66.67%
9	Sonar	80.00%	55.00%
10	Sonar	60.00%	60.00%

TABLE 6. Runtime and average accuracy for SWAG and AutoKeras on different datasets.

Dataset/Method	SWAG	AutoKeras
Pima Indians Diabetes	76.31% ± 3.41%	76.17% ± 2.86%
Ionosphere	87.45% ± 4.48%	81.78% ± 8.46%
Sonar	75.43% ± 8.10%	61.98% ± 10.89%
Runtime (seconds)	39.49	244.52
	21.63	305.51
	20.21	333.03

This efficiency underscores SWAG's suitability for rapid model development and deployment.

- **AutoKeras Model:** Incurred a significantly longer training duration of 5681.16 seconds (approx. 94.7 minutes), largely due to its exhaustive search for an optimal model architecture. While this automated process aims to achieve high accuracy, it poses practical challenges regarding time efficiency, especially for larger datasets or complex modeling tasks.

In summary, the SWAG model distinctly outperforms the AutoKeras model across all evaluated metrics on the MNIST dataset, including loss convergence, accuracy, and computational efficiency. SWAG's rapid convergence, high accuracy rates, and computational frugality make it an exemplary choice for handwriting digit classification tasks. Conversely, the AutoKeras model, despite its automated architecture optimization, needs to improve in generalization and efficiency, underpinning the superior utility of SWAG for such applications. The experiments were facilitated by Google Colab, with the code and further details accessible at [41], specifically within Chapter 2 and the SWAG Classification notebooks.

V. DISCUSSION

This study introduces SWAG, a novel neural network architecture aimed at optimizing the selection of nonlinear activation functions for enhanced model performance. Unlike traditional approaches that rely on predetermined activation functions, SWAG employs a set of fixed functions, constructing a polynomial basis within each layer. This strategy is grounded in the principles of the Stone-Weierstrass approximation theorem, which asserts that polynomial functions are capable of uniformly approximating any continuous function over a closed interval. Consequently, SWAG preserves the global approximation capability intrinsic to feedforward neural networks while potentially streamlining the computational process.

Our comprehensive evaluation of SWAG, compared with random search techniques, AutoKeras, and manually developed machine learning models, particularly in hyperparameter optimization, sheds light on the strengths and limitations of these methods. This comparative analysis underscores the importance of adopting a bespoke hyperparameter optimization strategy tailored to the specific demands of each task. The findings reveal that AutoKeras and SWAG balance accuracy, computational efficiency, and model generalization effectively. On the other hand, despite achieving optimal loss values, random search methods are hindered by their time-intensive nature, rendering them less practical for real-world applications. Moreover, architectures designed by human experts fell short of expectations, highlighting their limitations in the tasks examined within this study.

A noteworthy observation from the refinement of the SWAG model is the predominant use of X and X^2 as activation functions. The linear derivative of X^2 significantly simplifies the backpropagation process. Additionally, incorporating the NumPy library in several iterations of SWAG has markedly improved computational speed, offering a competitive advantage over other libraries.

VI. FUTURE WORK

The adaptability and computational efficiency of the SWAG model open up promising avenues for future research, particularly in the realm of on-device machine learning. Given its fixed architecture, SWAG can be pre-installed on mobile devices or other portable hardware, allowing for on-the-fly training and deployment of models directly on the device. This capability introduces a paradigm shift in how machine learning models are traditionally deployed, moving from cloud-based to edge computing. Future studies could explore:

- **Optimization for Mobile Devices:** Tailoring SWAG's architecture to maximize efficiency and performance on mobile processors, including adjustments for battery life optimization and memory usage.

- **Real-time Learning and Adaptation:** Investigating SWAG's potential to learn from new data in real-time, enabling personalized and adaptive applications that evolve with user interaction.
- **Cross-Device Compatibility:** Ensuring SWAG's compatibility across a broad spectrum of devices, from smartphones to IoT devices, to foster a wider adoption of on-device learning.
- **Privacy-preserving Machine Learning:** Leveraging SWAG's on-device training capability to enhance data privacy, as data need not leave the user's device for model training or updates.

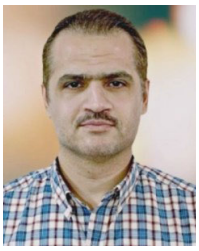
The exploration of these areas will significantly contribute to the advancement of edge computing in machine learning, paving the way for innovative applications that leverage the immediacy and privacy benefits of on-device computation.

ACKNOWLEDGMENT

The authors would like to thank the valuable assistance provided by ChatGPT 4 in enhancing the quality of the articles writing and also would like to thank Google Colab for facilitating the execution of experiments in a collaborative and efficient manner.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[2] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.
[3] H. Chung, S. Joo Lee, and J. Gue Park, "Deep neural network using trainable activation functions," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 348–352.
[4] S. Akbar, M. Hayat, M. Tahir, S. Khan, and F. K. Alarfaj, "CACP-DeepGram: Classification of anticancer peptides via deep neural network and skip-gram-based word embedding model," *Artif. Intell. Med.*, vol. 131, Sep. 2022, Art. no. 102349.
[5] S. Akbar, S. Khan, F. Ali, M. Hayat, M. Qasim, and S. Gul, "IHBP-DeepPSSM: Identifying hormone binding proteins using PsePSSM based evolutionary features and deep learning approach," *Chemometric Intell. Lab. Syst.*, vol. 204, Sep. 2020, Art. no. 104103.
[6] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018, Mar. 2018, Art. no. 068349.
[7] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Jul. 2018.
[8] S. Hayou, A. Doucet, and J. Rousseau, "On the selection of initialization and activation function for deep neural networks," 2018, *arXiv:1805.08266*.
[9] G. Zhang and H. Li, "Effectiveness of scaled exponentially-regularized linear units (SERLUs)," 2018, *arXiv:1807.10117*.
[10] H. W. Lin, M. Tegmark, and D. Rolnick, "Why does deep and cheap learning work so well?" *J. Stat. Phys.*, vol. 168, no. 6, pp. 1223–1247, Sep. 2017.
[11] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, 2013, vol. 30, no. 1, p. 3.
[12] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*.
[13] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 971–980.
[14] R. H. R. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature*, vol. 405, no. 6789, pp. 947–951, Jun. 2000.
[15] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 2146–2153.
[16] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[17] F. Temurtas, A. Gulbag, and N. Yumusak, "A study on neural networks using Taylor series expansion of sigmoid activation function," in *Proc. Int. Conf. Comput. Sci. Appl.*, 2004, pp. 389–397.
[18] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Netw.*, vol. 6, no. 6, pp. 861–867, Jan. 1993.
[19] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019.
[20] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowledge-Based Syst.*, vol. 212, Jan. 2021, Art. no. 106622.
[21] M. A. Zaveri, A. Rios, A. Gupta, and D. P. Wall, "Data labeling for supervised learning," *PLOS Comput. Biol.*, vol. 15, no. 7, 2019, Art. no. e1006855.
[22] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[24] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 1946–1956.
[25] A. Singh, N. Thakur, and A. Sharma, "A review of supervised machine learning algorithms," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, Mar. 2016, pp. 1310–1315.
[26] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020.
[27] DeepLearningSaeid. *SWAG GitHub Repository*. Accessed: Nov. 6, 2023. [Online]. Available: https://github.com/DeepLearningSaeid/Grad
[28] A. Truong, A. Walters, J. Goodsitt, K. Hines, C. B. Bruss, and R. Farivar, "Towards automated machine learning: Evaluation and comparison of AutoML approaches and tools," in *Proc. IEEE 31st Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2019, pp. 1471–1479.
[29] G. Li, G. Qian, I. C. Delgadillo, M. Müller, A. Thabet, and B. Ghanem, "SGAS: Sequential greedy architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1620–1630.
[30] J. Waring, C. Lindvall, and R. Umeton, "Automated machine learning: Review of the state-of-the-art and opportunities for healthcare," *Artif. Intell. Med.*, vol. 104, Apr. 2020, Art. no. 101822.
[31] S. Guarnieri, F. Piazza, and A. Uncini, "Multilayer feedforward networks with adaptive spline activation function," *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 672–683, May 1999.
[32] T.-T. Lee and J.-T. Jeng, "The Chebyshev-polynomials-based unified model neural networks for function approximation," *IEEE Trans. Syst., Man, Cybern., B (Cybern.)*, vol. 28, no. 6, pp. 925–935, Dec. 1998.
[33] Y. Zhiqi, "Gesture learning and recognition based on the Chebyshev polynomial neural network," in *Proc. IEEE Inf. Technol., Netw., Electron. Autom. Control Conf.*, May 2016, pp. 931–934.
[34] L. Ma and K. Khorasani, "Constructive feedforward neural networks using Hermite polynomial activation functions," *IEEE Trans. Neural Netw.*, vol. 16, no. 4, pp. 821–833, Jul. 2005.
[35] S.-S. Yang and C.-S. Tseng, "An orthogonal neural network for function approximation," *IEEE Trans. Syst., Man, Cybern., B (Cybern.)*, vol. 26, no. 5, pp. 779–785, Oct. 1996.
[36] Z. Liao, "Trainable activation function in image classification," 2020, *arXiv:2004.13271*.
[37] M. Kahn, "Diabetes," UCI Mach. Learn. Repository, doi: 10.24432/C5T59G.
[38] V. Sigillito, S. Wing, L. Hutton, and K. Baker, "Ionosphere," UCI Mach. Learn. Repository, 1989, doi: 10.24432/C5W01B.
[39] T. Sejnowski and R. Gorman, "Connectionist bench (sonar, mines vs. rocks)," UCI Mach. Learn. Repository, doi: 10.24432/C5T01Q.
[40] Y. LeCun, C. Cortes, and C. J. C. Burges. (1998). *MNIST Handwritten Digit Database*. [Online]. Available: http://yann.lecun.com/exdb/mnist/
[41] S. Safaei, "SWAG," (Nov. 2023). *GitHub repository*. Version 1.0.0. [Online]. Available: https://github.com/DeepLearningSaeid/Grad



SAEID SAFAEI received the B.Sc. degree in applied mathematics, in 2005, the M.Sc. degree in applied mathematics from Arak University, Iran, in 2008. He is currently pursuing the Ph.D. degree with the University of Georgia, Athens, GA, USA, under the guidance of Prof. Arabnia and with Prof. Gutierrez, Prof. Rasheed, and Prof. Taha serving as a committee members.

With over 15 publications, his work spans machine learning, education, and DNA computing. He has also served as a Reviewer for journals in *BioSystems* and *Supercomputing* and as for several international conferences. In the industry, he has applied his machine-learning expertise to projects for major corporations, such as Walmart, Wells Fargo, Honeywell, and Lowe's. His research interest includes developing faster deep learning topologies using polynomial activation functions, particularly for smaller devices.

ZEROTTI WOODS received the Ph.D. degree in applied mathematics from the University of Georgia, in 2019.

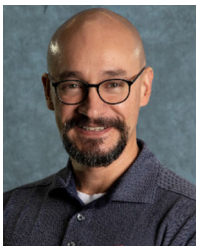
Throughout the Ph.D. degree, his primary area of concentration was deep learning. His thesis, titled "A New Regularization Term for Deep Neural Networks with Applications to Biological Data," showcased his commitment to advancing the field. Following the successful completion of the Ph.D. degree, he embarked on a promising career journey. He joined the Applied Physics Laboratory, The Johns Hopkins University, assuming the role of a Chief Scientist-ISR and T Group within the Force Projection Sector. In this capacity, he continues to contribute his expertise and insights to the field of applied mathematics and deep learning.

KHALED RASHEED received the Ph.D. degree in computer science from Rutgers University, in January 1998. He is currently a Professor with the Department of Computer Science, University of Georgia. He is also the Director of the UGA Institute for Artificial Intelligence, the Director of the Evolutionary Computation and Machine Learning (ECML) Laboratory, and a member of Georgia Informatics Institutes and the UGA Faculty of Robotics. He has authored more than 100 research articles. His research interests include artificial intelligence methods, including genetic algorithms, evolutionary computation, machine learning and artificial intelligence applications, including engineering design optimization, computational biology, and bioinformatics.

THIAB R. TAHA received the Ph.D. degree from Clarkson University, in 1982. He joined UGA, in 1982. He has been a Professor and the Head of the Computer Science Department, UGA, since July 2013. He is also the Director of the UGA CUDA Teaching and Research Centers and the Big Data Consulting Services and Training Center. He is also an Adjunct Faculty Member of the Institute of Bioinformatics, UGA. He has published more than 80 research articles and has given more than 100 invited talks or keynotes at international conferences. His research interests include scientific and distributed computing and software development for solving problems in nonlinear waves, optical fiber communication systems, and biochemical reaction networks. He received the M. G. Michael Award for Research in the Sciences from UGA, in 1985. He was the Fulbright Scholar, from 1995 to 1996. He received several grants from NSF and DOE and the ARO in support of his research. He is a Senior Editor of the *Mathematics and Computers in Simulation* journal, the Co-Editor-in-Chief of the APNUM journal, and has been the Chair and a Conference Coordinator of the IMACS International Conferences on Nonlinear Waves: Computation and Theory, since 1999.

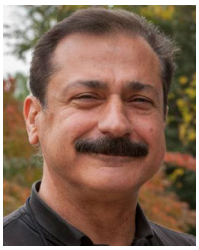
VAHID SAFAEI received the B.Sc. degree in mechanical engineering, in 2006, and the M.Sc. degree in mechanical engineering from Kashan University, Iran, in 2009. He is currently pursuing the Ph.D. degree in mechanical engineering with the University of Isfahan.

His academic contributions include over ten publications, focusing primarily on solid mechanics and DNA computing.



JUAN B. GUTIÉRREZ received the M.Sc. and Ph.D. degrees in mathematics from Florida State University, in May 2005 and December 2009, respectively.

Currently, he holds the position of a Professor in mathematics with The University of Texas at San Antonio (UTSA), in August 2019. In addition to his academic role with UTSA, he serves as the Chair of the Department of Mathematics, where he plays a key administrative role. Throughout his academic career, he has made notable contributions to mathematical research, with a focus on data analysis methods. His research interests include infectious disease modeling, population dynamics, and the dispersal of genetically modified organisms. He is recognized for his expertise in developing mathematical, computational, and statistical models that address practical challenges. Furthermore, he has secured ten grants and authored over 60 research articles, underscoring his significant contributions to the fields of academia and scientific research.



HAMID R. ARABNIA received the Ph.D. degree in computer science from the University of Kent, Canterbury, in 1987. He has been with the University of Georgia, Athens, GA, USA, since 1987, where he is currently a Professor Emeritus in computer science. He is a fellow and an Advisor of the Center of Excellence in Terrorism, Resilience, Intelligence, and Organized Crime Research. He has authored extensively in journals and refereed conference proceedings. He has authored or coauthored about 200 peer-reviewed research publications and also 250 edited research books in his areas of expertise. His research interests include parallel and distributed processing techniques and algorithms, supercomputing, big data analytics (in the context of scalable HPC), imaging science (image processing, computer vision, and computer graphics), other compute-intensive problems, methodologies that promote cross-disciplinary education, health informatics, medical imaging, and security. His most recent activities include studying ways to promote legislation that would prevent cyberstalking, cyber harassment, and cyberbullying. He is also the Editor-in-Chief of *The Journal of Supercomputing* (Springer).