How important are activation functions in regression and classification? A survey, performance comparison, and future directions

Ameya D. Jagtap∗ and George Em Karniadakis

The abstract of the paper introduces the critical role of activation functions in artificial neural networks, particularly in the context of real-world problems. It outlines the paper's main objective, which is to survey the evolution of activation functions used in classification and regression tasks, discussing their advantages, limitations, and developments over the years. The authors promise a comprehensive exploration that includes both classical (fixed) and adaptive activation functions, with a focus on their applications and characterizations. The paper's approach involves a systematic comparison of these functions using popular datasets like MNIST, CIFAR-10, and CIFAR-100, and across different machine learning libraries such as TensorFlow, Pytorch, and JAX. A notable aspect of the paper is its focus on the emerging field of physics-informed machine learning, discussing the specific requirements for activation functions in this area.

The abstract suggests originality through its comprehensive survey and comparative analysis, highlighting the relevance of activation functions as a fundamental component of neural networks. While the abstract does not delve into the technicalities, it indicates a sound methodological approach in exploring activation functions. The authors mention specific findings, such as the inadequacy of ReLU and the effectiveness of alternatives like hyperbolic tangent, swish, and sine, particularly in physics-informed frameworks. These findings imply significant implications for the selection and application of activation functions in various machine learning frameworks. The quality of presentation in the abstract is commendable; it is well-structured, clear, and sets the stage effectively for the paper. Based on the abstract, the paper appears to be a comprehensive and insightful contribution to the field of machine learning, warranting attention from readers interested in neural network design and activation function selection, especially in the context of physics-informed machine learning frameworks.

The introduction of the paper begins by acknowledging the widespread success of Artificial Neural Networks (ANNs) in various industries, highlighting their applications in fields like cybersecurity, manufacturing, healthcare, financial services, food industry, and energy. This establishes the relevance and impact of ANNs in both academia and industry, demonstrating their versatility in handling complex tasks such as logistics and inventory management. The authors emphasize the role of ANNs in processing complex sensory inputs like images, videos, and audio, which underscores the technological advancements and challenges in the field.
The historical context provided about the development of neural networks, starting from the computational model by McCulloch and Pitts to the invention of the perceptron by Rosenblatt, is insightful. It traces the evolution of neural networks, noting the limitations of early networks in processing simple tasks, as highlighted in the seminal book by Minsky and Papert. This historical overview is crucial for understanding the progression and the challenges encountered in the field.
The introduction also touches upon the concept of Deep Neural Networks (DNNs), explaining their theoretical versatility and the initial challenges in training them. The mention of Rumelhart et al.'s groundbreaking study on the backpropagation training algorithm is significant, as it marks

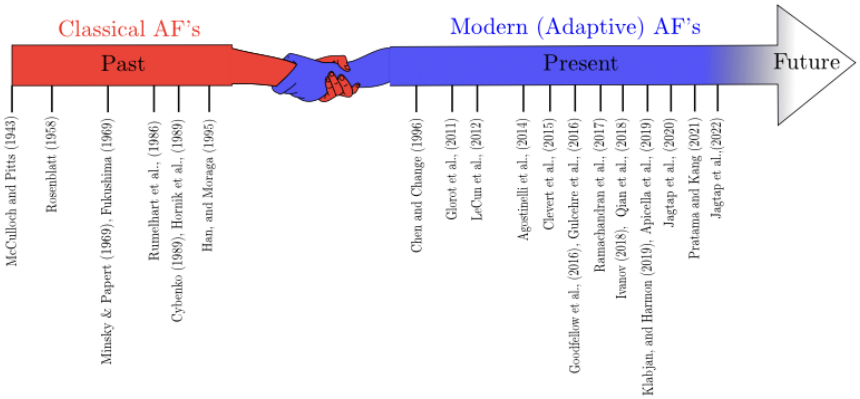
a pivotal moment in the field, leading to transformative developments in computer vision, speech recognition, natural language processing, and Physics-Informed Machine Learning.
A key element of the introduction is the discussion on artificial neurons and the role of the activation function. The paper correctly identifies the activation function as a critical component in determining the performance of an ANN, paralleling the action potential firing in biological brains. This analogy is essential for readers to understand the biological inspiration behind ANNs and the importance of activation functions in enhancing the network's learning capability.
The introduction concludes with the aim of the survey, which is to discuss the various advantages and limitations of both classical (fixed) and modern (adaptive) activation functions. This sets a clear objective for the paper, indicating a thorough examination of the topic.
However, the introduction could benefit from addressing certain questions, such as "Why was it necessary to develop different types of activation functions?" and "Why did early neural networks fail at processing simple tasks?" These questions could provide a deeper understanding of the challenges in the field and the motivation behind the continuous development and refinement of activation functions.

The key contributions are:

1. **Comprehensive Survey of Classical Activation Functions**: The paper thoroughly examines classical (fixed) activation functions, including real-valued functions like rectifier units, and extends to oscillatory and non-standard functions. The discussion includes their properties and suitability for specific tasks.
2. **Applications-Based Taxonomy**: The authors introduce a novel taxonomy of activation functions based on applications, particularly relevant for scientific computation. This includes a detailed look at complex-valued activation functions and their applications in various fields such as robotics and bioinformatics. They also explore quantized activations, focusing on their efficiency in network performance.
3. **Discussion on Adaptive Activation Functions**: The paper discusses state-of-the-art adaptive activation functions, highlighting their advantages over classical functions in terms of training acceleration and prediction accuracy. Various types of adaptive functions are examined in detail.
4. **Focus on Physics-Informed Machine Learning**: The authors delve into the unique requirements of activation functions in physics-informed machine learning, a relatively new and intriguing area. They compare the effectiveness of different activation functions in this context using various datasets and machine learning libraries.
5. **Organizational Structure**: The paper is methodically structured, covering a historical perspective of activation functions, a comparison between biological and artificial neurons, a detailed discussion on different types of activation functions, and a comparative analysis using popular datasets and neural network models.

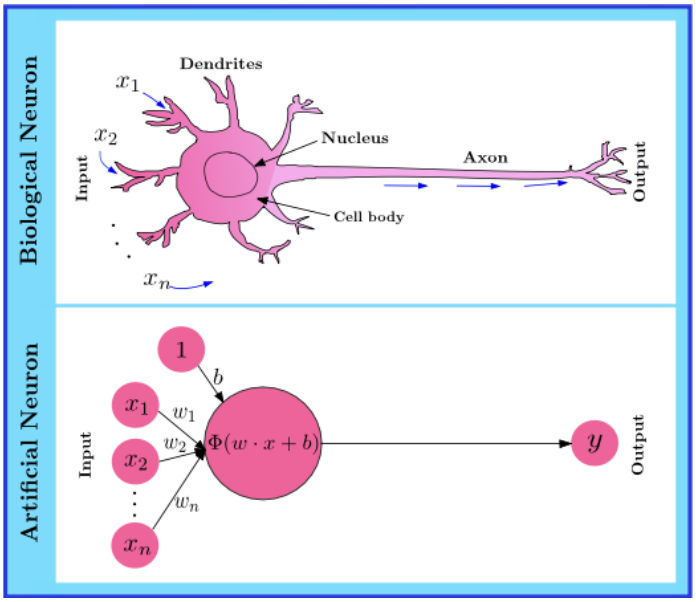
The authors start by explaining the fundamental role of activation functions in neural networks, highlighting their necessity for introducing nonlinearity, which is crucial for the network's ability to learn complex data features. This introduction underscores the importance of activation functions in differentiating neural networks from linear regression models.
Key historical developments and various terminologies associated with activation functions are discussed, reflecting the evolution of thought and application in this field. The transition from early activation functions like the Sigmoid to more recent innovations such as ReLU and its variants is

particularly insightful. It illustrates the ongoing efforts to address challenges like the vanishing gradient problem and the need for functions that are computationally efficient and conducive to gradient-based training.



The section also delves into the rise of adaptive activation functions. These functions represent a significant advancement in the field, offering enhanced flexibility and performance in neural network training. The exploration of various types of adaptive functions, from stochastic to ensemble activations, indicates the breadth and depth of current research in this area.

A comparison of biological and artificial neurons is also presented, providing a clear analogy of how activation functions in neural networks mimic the firing behavior of biological neurons. This comparison not only serves to root the concept in a biological context but also highlights the differences and unique challenges in designing effective artificial neurons.



The following section presents a detailed taxonomy of activation functions, categorizing them based on their characteristics and applications. This classification provides a structured understanding of the diverse range of activation functions used in artificial neural networks.

A. Characterization-Based Taxonomy:

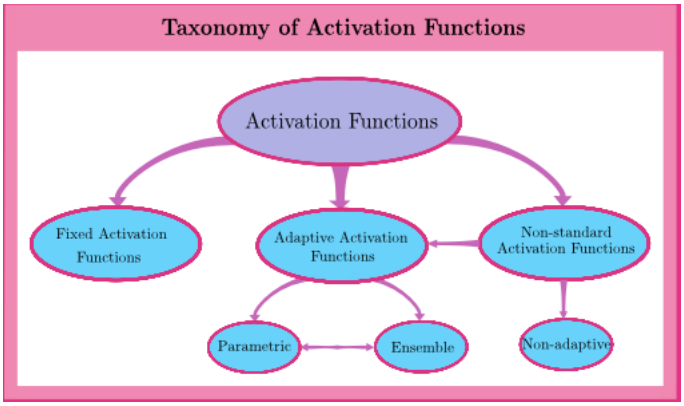
- The section starts with a characterization-based taxonomy, dividing activation functions into linear, nonlinear, fixed, adaptive, and non-standard categories.
- Fixed activation functions include classical activations like rectifiers.
- Adaptive functions, which can modify their behavior during training, are subdivided into parametric and ensemble types.
- Non-standard activations are more specialized and can be either fixed or adaptive.

B. Application-Based Taxonomy:

- The authors propose a novel application-based taxonomy, especially relevant for scientific applications where outputs are real or complex-valued.
- Real-valued activations are common in literature, but the section highlights the growing importance of complex-valued functions for applications like robotics and bioinformatics.
- Additionally, the concept of quantization in activation functions is introduced, which is essential for reducing memory requirements and increasing computational efficiency.

C. Classical Activation Functions:

- The section then delves into various classical activation functions, discussing their properties, advantages, and limitations.
- Linear and piece-wise linear functions, step functions, sigmoid, hyperbolic tangent (tanh), and rectified linear units (ReLU) are examined.
- Each function's historical context, application, and the specific issues they address or suffer from, such as the vanishing gradient problem, are discussed.
- The section also covers improvements and variants of these functions, showcasing the ongoing efforts to enhance neural network performance.



CLASSICAL ACTIVATION FUNCTIONS:

1. **Linear and Piece-wise Linear Functions**

- **Importance**: These are the simplest forms of activation functions. Their main appeal lies in their simplicity and interpretability. It has a constant gradient, and the descent is based on this constant value of gradient.
- **Core Properties**: The range of linear function is $(-\infty,\infty)$, and has a C^1 order of continuity. Piece-wise linear functions offer more flexibility, introducing non-linearity in specific segments of the input.
- **Typical Uses**: They are often used in regression problems or in the output layer for regression-based neural networks.
- **Advantages/Drawbacks**: While easy to compute and understand, linear functions severely limit the network's ability to model complex, non-linear relationships in data. They are generally not used in hidden layers of deep neural networks due to this limitation.

2. **Step Function**

- **Importance:** Step functions are foundational and conceptually simple, serving as a basic form of an activation function. Also known as Heaviside or unit step functions.
- **Core Properties:** It outputs a binary result, essentially enabling a 'fire' or 'not fire' decision.
- **Typical Uses:** Historically used in early neural network models, particularly in perceptron.
- **Advantages/Drawbacks:** Its binary nature makes it unsuitable for tasks requiring nuanced outputs. It doesn't allow for backpropagation since its derivative is zero almost everywhere.

3. **Sigmoid Function**

- **Importance:** Once a popular choice for activation due to its smooth, differentiable nature and bounded output.
- **Core Properties:** Sigmoid squashes input values into a small range between 0 and 1, making it useful for binary classification problems.
- **Typical Uses:** Common in binary classification tasks and in the output layer of binary classifiers.
- **Advantages/Drawbacks:** Suffers from vanishing gradient problem, making it less favorable for deep networks. Also, its outputs are not zero-centered, which can lead to inefficient gradient descent.

4. **Hyperbolic Tangent (tanh) Function**

- **Importance:** Preferred over sigmoid for being zero-centered, improving the efficiency of gradient descent.
- **Core Properties:** Outputs values in a range between -1 and 1, which makes it more balanced compared to sigmoid.
- **Typical Uses:** Used in hidden layers of neural networks, particularly beneficial for regression tasks.

- **Advantages/Drawbacks:** Like sigmoid, it suffers from vanishing gradients for extreme input values, although its zero-centered nature partially mitigates this issue.

5. **Rectified Linear Unit (ReLU)**

- **Importance:** Extremely popular in modern neural networks due to its computational efficiency and effectiveness in addressing the vanishing gradient problem.
- **Core Properties:** ReLU activates a neuron only if the input is above a certain threshold (usually zero).
- **Typical Uses:** Widely used in hidden layers of deep neural networks, particularly in convolutional neural networks.
- **Advantages/Drawbacks:** Though it mitigates the vanishing gradient problem and speeds up training, it can lead to "dying ReLU" where neurons stop activating entirely. It's also non-differentiable at zero, which can be a computational concern.

The section on ReLU variants addresses the critical issue of the dying ReLU problem in neural networks. Let's explore these variants in detail, understanding their design, purpose, and impact on network learning:

i. **Leaky ReLU**

- Purpose: It introduces a small, non-zero gradient (defined by the hyperparameter α) when the unit is not active.
- **Importance:** This modification helps mitigate the dying ReLU problem by allowing a small gradient when the neuron would otherwise be inactive.
- **Typical Use:** Used in scenarios where dying ReLU problem is a concern. Often, α is set to 0.01, but it can be tuned for specific applications.

ii. **Randomized Leaky ReLU**

- **Definition:** Like Leaky ReLU, but α is randomly picked from a specified range during training and fixed to its average value during testing.
- **Purpose:** Adds an element of randomness to prevent overfitting and improve generalization.
- **Importance:** It can lead to better performance in some cases due to its randomized nature during training.

iii. **Mirror ReLU**

- **Purpose:** Designed to provide a mirrored activation pattern around the origin, aiming to handle negative inputs more effectively.
- **Importance:** Offers an alternative approach to handle negative inputs compared to traditional ReLU.

iv. **Concatenated ReLU (CReLU)**

- **Purpose:** Concatenates the outputs of a ReLU for both the positive and negative parts of the input.
- **Importance:** It can capture both positive and negative features, but increases model complexity.

v. **Elastic ReLU**
- **Purpose:** Introduces randomness in the activation function to potentially enhance feature learning.
- **Importance:** The randomness can help in exploring different feature spaces but might introduce instability.

vi. **Bounded ReLU**
- **Purpose:** It bounds the output of the ReLU, addressing the issue of unbounded outputs for non-negative inputs.
- **Importance:** Useful in scenarios where controlling the range of activation function output is necessary.

vii. **V-shaped ReLU**
- **Purpose:** Designed to handle negative inputs differently by providing a constant output for very negative inputs.
- **Importance:** It's an alternative approach to handling negative inputs, potentially useful in specific contexts.

viii. **Gaussian Error Linear Units (GELUs)**
- **Purpose:** It's designed to model neuron activation probabilistically, considering the entire distribution of inputs.
- **Importance:** GELU finds utility in models where a probabilistic understanding of neuron activation is beneficial, like in complex language models.

Each of these ReLU variants offers unique advantages and addresses specific limitations of the standard ReLU function. By tweaking the behavior for negative inputs or by introducing bounds and randomness, these variants make it possible to adapt the activation function more closely to the needs of specific neural network architectures and tasks.

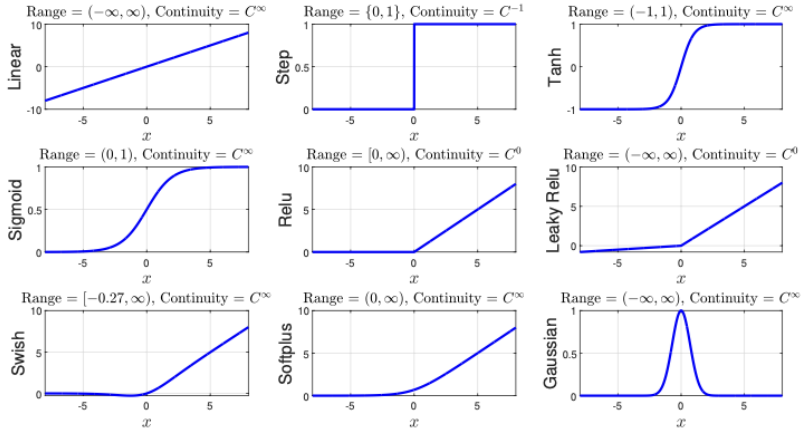


Fig. 5. Classical (fixed) activation functions with their range and continuity.

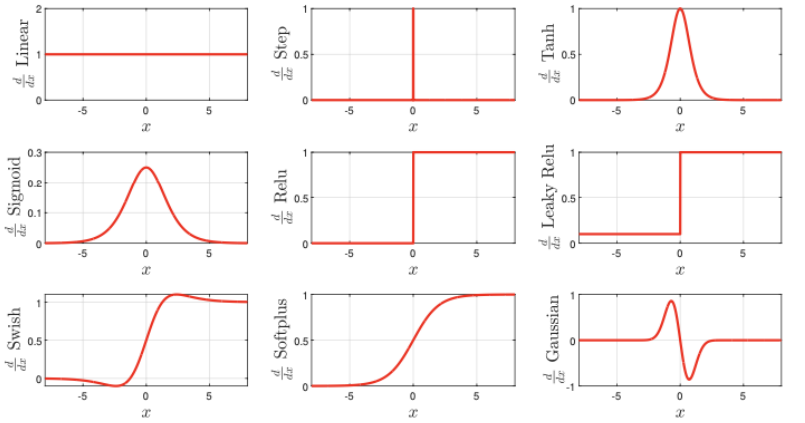


Fig. 6. Derivatives of classical activation functions.

6. Softplus Function

- **Properties:** It is a smooth approximation of the ReLU function and is infinitely differentiable.
- **Importance:** This function addresses the non-differentiability of ReLU at zero, providing a smooth transition.
- **Applications:** Particularly useful in networks where smoothness is a key factor, like in spiking neural networks.

7. Exponential Linear Unit (ELU)

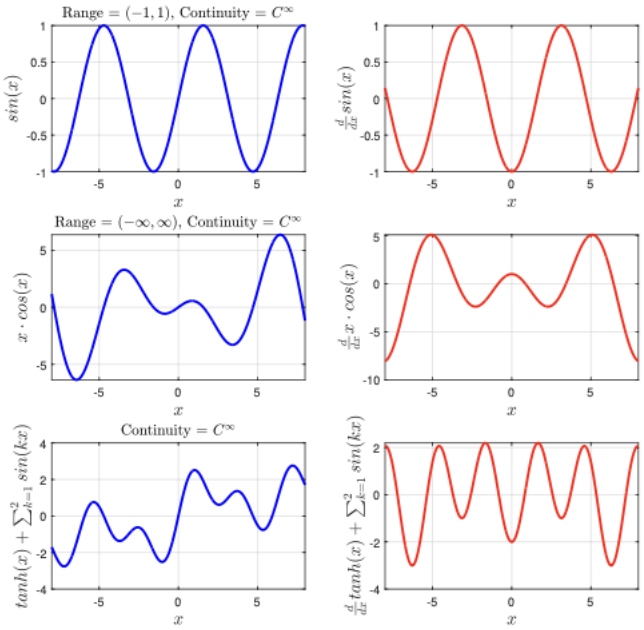
- Definition: ELU is defined as $\Phi(\blacksquare)=\blacksquare(\exp[f_0](\blacksquare)-1)\Phi(x)=\alpha(\exp(x)-1)$ for $\blacksquare\leq 0 x\leq 0$ and $\Phi(\blacksquare)=\blacksquare\Phi(x)=x$ for $\blacksquare>0 x>0$.
- Properties: Offers negative values for $\blacksquare<0 x<0$, making the mean unit activation closer to zero.
- Importance: Helps alleviate the vanishing gradient problem and does not suffer from the dead neuron problem.
- Applications: Beneficial in deep networks where vanishing gradients are a concern.

8. **Mish Function**
- **Properties:** It's unbounded above, bounded below, non-monotonic, and has C^{∞} continuity.
- **Importance:** Mish is a self-gated function, providing a balanced approach to activation.
- **Applications:** Can be used in various architectures, providing a mix of benefits from both ReLU and smooth activation functions.

9. **Radial Activation Functions**
- **Properties:** These functions are typically used in radial basis function (RBF) networks.
- **Importance:** Suitable for problems involving function approximation and classification.
- **Applications:** RBF networks with Gaussian activations have been successful in diverse areas such as pattern recognition and classification tasks.

10. **Wavelet Activation Functions**
- **Properties:** Utilizes wavelets as activation functions, which are effective in identifying important features in high-frequency signals.
- **Applications:** Employed in classification and identification problems, particularly where signal processing is involved.

11. **Oscillatory Activation Functions**
- **Properties:** These functions, like sinusoidal activations, can create multiple decision boundaries, which is essential for complex function approximation like XOR.
- **Importance:** They offer an alternative approach where traditional activation functions might fall short, especially in tasks that require complex decision boundaries.
- Recently a Rowdy activation function was proposed, where the oscillatory noise is injected over the monotonic base activation function with adaptable parameters, thereby making them oscillatory as the optimization process starts. This adaptively creates multiple hypersurfaces to better learn the data set.



12. **Non-standard Activation Functions**

i. **Maxout**

- Definition: Maxout is a piece-wise linear function that generalizes ReLU and leaky ReLU. It computes the maximum over *k* linear feature mappings.
- **Importance:** It enhances optimization by preventing units from remaining idle, a common issue with ReLU. It performs a pooling operation across subspace of *k* linear feature mappings.
- **Applications:** Effective in conjunction with dropout, Maxout is suitable for networks where optimization and prevention of idle units are key.

ii. **Softmax Function**

- **Definition:** Softmax function, also known as Softargmax or normalized exponential function, normalizes an input vector into a probability distribution.
- **Importance:** It is widely used in multi-class classification problems and neural networks dealing with probabilities.
- **Applications:** Softmax is crucial in the output layers of classification networks, especially those targeting multiple classes.

COMPLEX VALUED ACIVATION FUNCTIONS

- **Definition**: CVNNs are designed to handle data in complex number form, making them suitable for specific domains.
- **Properties**: Choosing an appropriate complex activation function for CVNNs is challenging due to Liouville's theorem, which implies a bounded, complex-differentiable function must be constant.
- **Applications**: CVNNs find utility in fields like remote sensing, acoustics, opto-electronics, and robotics.

1. **Multi-Valued Neuron Activation (Aizenberg et al.):**
   - Divides the complex plane into k equal sectors, mapping the entire plane onto the unit circle.

2. **Complex-Sinum Function (Jankowski et al.):**
   - Utilizes multivalued threshold logic.
   - Designed for complex-valued inputs and outputs.

3. **Split Activation Function:**
   - These are bounded but non-analytic functions, preferred in CVNN design.
   - Balances the need for complex data handling with the limitations imposed by complex differentiation.

4. **Phase-Amplitude Functions (Noest, Georgious and Koutsougeras)**:
   - These functions process the amplitude and phase of complex numbers separately.

5. **Analytic and Bounded Activation Functions (Kim and Adali)**:
   - These functions are analytic and bounded but have singular points, requiring careful scaling of inputs and weights.

6. **Möbius Transformation-Based Activation**:
   - A conformal mapping of the complex plane, also known as a bilinear transformation.

7. **Complex-Valued ReLU (Guberman, Arjovsky et al.)**:
   - A complex extension of ReLU, some versions being phase-sensitive.

8. **Cardioid Activation Function (Virtue et al.)**:
   - A complex extension of ReLU that is sensitive to the phase of the input.

Each of these advanced activation functions, whether non-standard like Maxout and Softmax or complex-valued for CVNNs, broadens the spectrum of neural network applications. They allow for tailored solutions to specific data types and problems, particularly in areas where traditional activation functions fall short. The choice of

activation function significantly influences the behavior and performance of the neural network, making it a crucial aspect of neural network design and implementation.

QUANTIZED ACTIVATION FUNCTIONS

Quantized Neural Networks (QNNs) focus on improving the efficiency of neural network models through quantization, which involves converting continuous or high-precision inputs into a lower precision format. The main highlights of this section include:

1. **Efficiency and Performance:** Quantization aims to improve the efficiency of neural networks without degrading performance. It allows for the use of bitwise operations in forward and back-propagation, enhancing computational speed.
2. **Components of Quantization:** Weights, activations, and gradients can be quantized. This helps in reducing the size of the model and, in the case of gradients, reduces communication costs in distributed training.
3. **Spiking Neural Networks (SNNs):** SNNs use quantized activation levels, known as 'spike events', to further simplify computations, converting weight-activity products into straightforward additions.
4. **Implementation Examples:**
    - Vanhoucke et al. quantized activations to 8 bits post-training using a sigmoid function.
    - Binary activations (Sign function) have been used for approximating ReLU units.
    - Cai et al. proposed a half-wave Gaussian quantizer for ReLU approximation.

5. **Gradient Mismatch and Solutions:** A notable issue with quantized activations is the gradient mismatch problem, where the assumed and real activation functions differ. The Straight Through Estimation (STE) derivative approximation has been proposed to address this.

A QUEST TOWARDS AN OPTIMAL ACTIVATION FUNCTION

Which activation function should we use? This is one of the most basic and meaningful questions that could be posed. As discussed earlier, there is no rule of thumb for choosing the optimal activation function, which strongly depends on the problem under consideration. This motivates us to ask another meaningful question: do we need an activation function that adapts itself as per the requirements, thereby avoiding the local minima by changing the loss landscape dynamically?

In this way, the adaptive activation functions can beat any standard (fixed) activation function of the same type. The quest for an optimal activation function revolves around finding a function that adapts to the specific requirements of a given problem, potentially altering the loss landscape dynamically to avoid local minima. Key points include:

Adaptive Activation Functions: These functions can adjust themselves according to the problem's requirements, potentially outperforming fixed activation functions. They can be parameterized or involve ensemble methods.

1. **Parametric Activation Functions:**
    - These include variations of ReLU and ELU with additional parameters for more control over the function shape, which can improve learning dynamics and performance.
    - Generalized hyperbolic tangent function with additional scalar values for adaptability.
    - Swish activation function, is defined as:
      $\Phi(x) = x \cdot \text{Sigmoid}(\beta \cdot x)$, where β is a tuning parameter.
    - E-swish and p-swish functions, providing more flexibility than the original Swish activation.
    - Layer-wise and neuron-wise locally adaptive activation functions, introducing trainable parameters at different levels of the neural network.

These sections emphasize the ongoing innovations in neural network design, particularly in terms of activation functions. Quantized and adaptive activation functions represent significant steps towards creating more efficient and effective neural networks, tailored to specific computational and application requirements.

This section of the paper discusses advanced adaptive activation functions, focusing on the development and application of parameterized and evolutionary designed activation functions for neural networks. Evolutionary Algorithms are new activation functions are being explored using evolutionary algorithms that can automatically discover and optimize activation functions for specific tasks.

i. **Parametric ReLU (PReLU)**: Enhances ReLU by introducing a parameter that adjusts the slope for negative inputs, addressing the dead neuron problem.
ii. **S-Shaped ReLU:** Combination of 3 linear functions which perform a mapping of R -> R.

iii. **Flexible ReLU (FReLU)**: Adds a bias term to ReLU to capture negative values, allowing a shift in the rectified point.
iv. **Paired ReLU**: Consists of a pair of ReLU functions with scalable parameters and thresholds, which can be trained to fit the data more closely.
v. **Parametric ELU**: Aims to eliminate the need to manually set parameters at every layer by incorporating trainable parameters that adjust the activation function's shape.
vi. **Fast ELU**: Provides an approximation of ELU with a faster computation, aiming to retain ELU's accuracy benefits while improving computational efficiency.
vii. **Multi-bin Trainable Linear Units (MTLU)**: Offers a piece-wise linear activation function with multiple segments, each with its own set of trainable parameters, allowing for a more nuanced and adjustable activation pattern.
viii. **Mexican ReLU**: Combines the traditional ReLU with a Mexican hat wavelet function to add complexity and adaptability.
ix. **Stochastic/Probabilistic Activation Functions**: Incorporate randomness into the activation to prevent saturation and enhance learning, particularly beneficial for activations with a strong saturation for large input values.
x. **Fractional Adaptive Activation Functions**: Use fractional calculus to create activation functions that can encapsulate and generalize many existing activation functions.
xi. **Ensemble Adaptive Activation Functions**: Combine multiple activation functions to optimize the network's performance, allowing the selection of the most suitable activation as a discrete optimization problem.

xii. **Performance Comparison**: The table shows a comparison of different activation functions such as Sigmoid, ReLU, Leaky ReLU, PReLU, ELU, SELU, Swish, and Mish across various datasets like MNIST, CIFAR-10, and CIFAR-100. This comparison provides insights into which activation functions perform better under certain conditions.
xiii. **Neural Network Architectures**: The performance of these activation functions is tested in different neural network architectures, including MobileNetV2 and VGG16, providing a practical perspective on the adaptability and efficiency of these functions in real-world applications.

2. **Stochastic/probabilistic Activation functions:**
The activation functions discussed here incorporate randomness directly into the activation process, with the intention to help the network learn better and escape potential issues like saturation that can occur with standard activation functions like ReLU.

i. **Probabilistic ReLU (ProbAct):** This function adds a noise term s to the standard ReLU function, where s is a combination of a mean μ (which can be static or learnable) and a perturbation term z, which itself is a product of a standard deviation σ and a normally distributed random variable ϵ. This addition aims to preserve some degree of gradient even when the unit output reaches saturation, thus allowing for better backpropagation through the network.

ii. **Rand Softplus:** A function that blends the ReLU and softplus functions with a stochastic component ρ, allowing for adaptability and potentially helping to maintain learning progress in saturated regimes.

- **Advantages:** These functions can potentially allow for gradients to propagate even when the activation function's output is in a saturated state, which could otherwise lead to stalled learning or the vanishing gradient problem. By introducing noise, these functions aim to maintain a flow of gradient information, which can be crucial for the learning process, particularly in deep networks.
- **Drawbacks:** As these functions generalize the ReLU function, they inherit its drawbacks, such as being unbounded and potentially leading to exploding gradients.
- The introduction of noise can also make the behavior of the network less predictable and potentially harder to tune, as the noise needs to be carefully controlled.
  Overall Review: The section effectively presents an innovative approach to activation functions that could potentially address some of the limitations of traditional functions like ReLU. The introduction of noise into the activation function is an interesting strategy that may help networks avoid stagnation during training. The section clearly explains how these functions are structured and the rationale behind their design.

The concept of incorporating randomness directly into the activation process aligns with the broader trend in neural network research seeking to make models more robust and capable of learning complex patterns. However, the stochastic nature of these functions also introduces a level of unpredictability that could affect the stability of the training process.

The discussion could be enhanced by providing empirical evidence or case studies demonstrating the effectiveness of these stochastic functions compared to their traditional counterparts. Additionally, addressing how the introduction of noise affects the convergence and overall training time would provide a more complete picture of the trade-offs involved in using such functions.

3. Fractional Adaptive activation Functions

- **Introduction to Fractional Activation Functions:** The text begins by placing fractional activation functions within the broader landscape of neural network research. It mentions the potential benefits of these functions, such as tunable hyperparameters that could lead to more nuanced models. This is a good way to establish the relevance of the research.
- **Gamma Function and Fractional Calculus:** The authors talk about leveraging the Gamma function in fractional calculus to define new types of activation functions, like the fractional ReLU.
- **Fractional ReLU Definition:** The fractional ReLU is defined with a piecewise function and involves a fractional derivative.

- **Discussion on Specific Functions:** The paper mentions the transformation of tanh function to a quadratic hyperbolic secant function when a specific parameter *a* is set to zero.

4. Ensemble adaptive activation functions

- **Ensemble Activation Functions:** The idea is to combine different activation functions to capitalize on the strengths of each. Earlier works by Xu and Zhang, as well as Ismail et al., are referenced, highlighting approaches that involve adaptive ensembles and selection of activation functions based on optimization criteria.
- **Adaptive Coefficients and Binary Indicators:** These ensembles use a mix of sub-functions with adaptive coefficients and binary indicators, which can turn on or off certain functions in the ensemble, providing a way to dynamically adjust the activation function during the learning process.
- **Stochastic Control and Linear Combination:** References to Chen [72] and Jin et al. [56] indicate approaches that involve multiple activation functions for each neuron and the combination of linear functions, respectively, to tackle stochastic control and optimization problems.
- **Reinforcement Learning:** Ramachandran et al. [35] and Qian et al. [53] use reinforcement learning to combine functions and optimize activation functions in a data-driven way, showcasing an application of reinforcement learning in tuning activation functions.
- **Polynomial and Kernel-based Functions:** There is mention of polynomial activation functions, which can approximate any smooth function given enough terms, and kernel-based non-parametric activation functions, which allow for a more fine-tuned approach to defining activation functions.
- **General Frameworks for Adaptive Activation:** The section also touches on broader frameworks such as Kronecker Neural Networks (KNN) and various forms of network architectures that incorporate adaptive activation functions, like the self-learnable activation functions in SLAF networks.
- **Examples and Variations:** Examples of specific activation function variations are given, including parametric ReLU, exponential linear units (ELU), and their scaled versions. The text mentions the flexibility of the Kronecker network in becoming different types of feed-forward networks depending on the choice of activation functions.
- **Hybrid and Genetic Algorithms:** The text also discusses the use of genetic algorithms for combining activation functions, hinting at a hybrid approach that allows for the crossover of various activation function characteristics.
- **Stochastic Selection and SLAF Networks:** Stochastic selection of activation layers and networks with self-learnable activation functions (SLAF) indicate a trend towards more adaptability and learning-based selection of activation functions within neural networks.
- **Empirical Comparisons:** The section concludes with a reference to empirical work by Nanni et al. [190], who compared various convolutional neural network activation functions using an ensembling method. This suggests that the field is moving towards a more empirical approach where different activation functions are compared in practice to determine their effectiveness in various settings.

PERFORMANCE OF SOME FIXED AND ADAPTIVE ACTIVATION FUNCTIONS FOR CLASSIFICATION TASKS

This research evaluates various activation functions across common datasets, revealing that while fixed activation functions like Leaky ReLU, ELU, and Sine show good performance on CIFAR-10 and CIFAR-100, adaptive and rowdy activation functions generally outperform them. The study compares the performance of different activation functions, including both fixed types like Sigmoid, Leaky ReLU, and ELU, as well as adaptive ones such as Maxout and ensemble functions. It also benchmarks the performance of these activation functions in neural network models, specifically MobileNet and VGG16, which differ in depth and complexity.

The results, tabulated with accuracy rates, highlight the effectiveness of adaptive activation functions, particularly ensemble and rowdy types, which demonstrate superior performance in most cases. The paper emphasizes that the choice of activation function can significantly influence the learning capabilities and performance of neural networks, suggesting a move towards more flexible and data-driven approaches in activation function design.

Additionally, the research highlights the use of ensemble methods that combine different activation functions, which can be particularly powerful. It also acknowledges the potential of stochastic/probabilistic and complex-valued activation functions, although their performance varies with the dataset and network architecture used.

Crucially, the study provides empirical evidence that supports the hypothesis that adaptive activation functions can offer benefits over traditional fixed types, with the adaptive and rowdy variants showing notable improvements in accuracy across the datasets tested. This suggests that the adaptability of the activation function to the data and task at hand may be a key factor in the design of effective neural network architectures.

The authors also note the advantages of using ensemble activation functions that incorporate multiple sub-functions, allowing the network to dynamically select the most appropriate activation behavior during the training process. This adaptability appears to be a significant factor in the enhanced performance observed in their experiments.

	MNIST	CIFAR-10	CIFAR-100
Sigmoid	97.9 [192]	89.43 ± 0.51 (M), 85.42 ± 0.47 (V)	61.64 ± 0.56 (M), 59.25 ± 0.45 (V)
Tanh	98.21[48]	88.19 ± 1.21 (M), 87.53 ± 0.67 (V)	57.06 ± 2.03 (M), 62.32 ± 0.82 (V)
Swish	99.75 ± 0.11 (M), 99.45 ± 0.26 (V)	95.5 [35]	83.9 [35]
ReLU	99.1 [55], 99.15 [183], 99.53 [56]	95.3 [35], 94.59 [193], 92.27 [56]	83.7 [35], 75.45 [193], 67.25 [56]
Leaky ReLU	98.2 [192],99.58 [56]	95.6[35], 92.32 [53]	83.3[35], 67.3[56]
ELU	98.3 [192]	94.4 [35], 94.01 [193]	80.6 [35], 74.92[193]
PReLU	99.64 ± 0.24 (M), 99.18 ± 0.17 (V)	92.46 ± 0.44 (M), 91.63 ± 0.31 (V)	69.46 ± 0.74 (M), 66.53± 0.69 (V)
SELU	98.42 ± 0.53 (M), 99.02 ± 0.37(V)	93.52 ± 0.63 (M), 90.53 ± 0.36 (V)	70.42 ± 0.75 (M), 68.02± 1.29 (V)
RReLU	99.23 ± 0.53 (M), 99.63 ± 0.6 (V)	90.52 ± 2.14 (M), 90.18 ± 0.91 (V)	68.62 ± 0.42 (M), 65.32 ± 1.74 (V)
GELU	99.72 ± 0.35 (M), 99.26 ± 0.42 (V)	94.76 ± 0.55 (M), 92.67 ± 0.89 (V)	71.73 ± 1.09 (M), 69.61 ± 1.53 (V)
CELU	99.36 ± 0.68 (M), 99.37 ± 0.38 (V)	90.26 ± 0.12 (M), 90.37 ± 0.23 (V)	70.26 ± 1.53 (M), 68.35 ± 0.87 (V)
Softplus	98.69 ± 0.5 (M), 97.36 ± 0.77 (V)	94.9 [35]	83.7 [35]
Mish	97.9 [192]	90.26 ± 0.52 (M), 86.05 ± 0.76 (V)	68.53 ± 0.86 (M), 67.03 ± 1.39 (V)
Maxout	99.55 [127]	90.62 [127]	61.43 [127]
SRS	98.04 ± 0.97 (M), 98.06 ± 0.84 (V)	89.35 ± 0.85 (M), 87.26 ± 1.38 (V)	65.20 ± 1.53 (M), 63.65 ± 2.63 (V)
Ensemble ([180])	99.40 [180]	85.05 ± 0.28 (M), 84.96 ± 0.87 (V)	74.20 [180]
LiSHT	98.74 ± 0.17 (M), 98.32 ± 0.19 (V)	90.78 ± 0.43 (M),87.74 ± 0.36 (V)	56.35 ± 0.58 (M), 58.74 ± 0.99 (V)
Sine	99.10 ± 0.82 (M), 98.63 ± 0.27 (V)	91.64 ± 0.36 (M), 90.59 ± 0.88 (V)	80.98 ± 0.81 (M), 78.42 ± 0.67 (V)
GCU ([126])	97.80 ± 1.02 (M), 90.73 ± 0.87 (V)	90.24 ± 1.96 (M), 89.56 ± 0.18 (V)	77.46 ± 2.91 (M), 76.46 ± 1.61 (V)
Gaussian	98.60 ± 2.22 (M), 96.43 ± 2.57 (V)	90.64 ± 2.76 (M), 89.59 ± 0.28 (V)	76.96 ± 0.41 (M), 74.62 ± 2.69 (V)
Adaptive sine	99.56 ± 0.56 (M), 98.76 ± 0.12 (V)	92.64 ± 0.36 (M), 92.72 ± 0.37 (V)	81.15 ± 0.81 (M), 78.73 ± 0.64 (V)
Adaptive tanh	99.52 ± 0.36 (M), 99.57 ± 0.20 (V)	91.14 ± 0.45 (M), 90.52 ± 0.82 (V)	83.82 ± 0.83 (M), 76.21 ± 1.95 (V)
Adaptive ReLU	99.75 ± 0.25 (M), 98.43 ± 0.32 (V)	96.89 ± 0.93 (M), 94.77 ± 1.16 (V)	**84.58 ± 0.42** (M), 80.30 ± 0.60 (V)
Rowdy Sine	99.74 ± 0.29 (M), 99.24 ± 0.43 (V)	93.87 ± 0.78 (M), 92.82 ± 0.82 (V)	83.73 ± 0.83 (M), 82.14 ± 0.13 (V)
Rowdy tanh	99.45 ± 0.93 (M), 98.65 ± 0.57 (V)	93.78 ± 0.13 (M), 93.26 ± 0.42 (V)	82.34 ± 1.01 (M), 80.97 ± 0.32 (V)
Rowdy ReLU	99.43 ± 0.68 (M), 98.23 ± 0.25 (V)	**95.57 ± 0.84** (M), 94.47 ± 0.92 (V)	82.75 ± 0.51 (M), 79.41 ± 0.75 (V)
Rowdy ELU	**99.79 ± 0.08** (M), 99.06 ± 1.07 (V)	95.45 ± 0.92 (M), 93.27 ± 0.13 (V)	84.29 ± 0.13 (M), 82.40 ± 0.81 (V)

TABLE III

ACCURACY COMPARISON OF DIFFERENT ACTIVATION FUNCTIONS FOR MNIST, CIFAR-10, AND CIFAR-100 DATA SETS. HERE, WE REPORT THE BEST ACCURACY FROM THE LITERATURE WITHOUT CONSIDERING THE ARCHITECTURE USED TO PERFORM THE EXPERIMENTS. ALSO, WE USED MOBILENET [194] (DENOTED BY M) AND *VGG16* [195] (DENOTED BY V) ARCHITECTURES WITH DIFFERENT ACTIVATION FUNCTIONS, WHERE WE REPORT THE MEAN AND STANDARD DEVIATIONS OF 10 DIFFERENT REALIZATIONS. THE *Adaptive* AND *Rowdy* (WITH $K = 3$) REFERS [64] AND [73], RESPECTIVELY. THE BEST RESULTS ARE HIGHLIGHTED IN BOLD.

PHYSICS INFORMED MACHINE LEARNING

PINNs are part of a broader category of Physics-Informed Machine Learning (PIML) algorithms that have gained traction for their ability to solve both forward and inverse problems governed by differential equations, a task notoriously difficult for traditional computational methods. This is achieved by incorporating physical laws directly into the learning process of the neural network.

PINNs are particularly efficient at solving both forward and ill-posed inverse problems, integrating sparse and multi-fidelity data along with the governing physical laws into the loss function of the network. This allows them to leverage information such as the governing equations, boundary conditions, and experimental data, effectively converting a partial differential equation (PDE) problem into an optimization problem. The section also references a range of literature that has contributed to the field, showcasing the rapid development and wide application of these methods in various scientific and engineering domains.

The activation function plays a crucial role in PINNs, affecting the network's ability to satisfy both the data and physical laws. The discussion highlights that traditional activation functions like ReLU and its variants are not suitable for solving problems that require the computation of higher-order derivatives due to their non-differentiable points. Instead, smoother functions such as sine, tanh, and swish are preferred because they allow for the computation of derivatives that are essential in enforcing the physical laws in PINNs.

The text mentions specific cases where PINNs are applied to solve differential equations, such as the Linear Convection Equation, Viscous Burgers Equation, and the Boussinesq Equation, each with increasing complexity and order of derivatives required. It is emphasized that the choice of activation function must take into account the necessity for differentiability, especially when higher-order derivatives are involved, as they are in these cases.

Additionally, the section discusses the importance of the range of the activation function, particularly when dealing with field variables that have known physical bounds. The correct selection of an activation function can ensure that predictions remain within physically plausible ranges, which is crucial for the accurate simulation of physical phenomena.

The authors also compare the performance of various activation functions, including fixed, adaptive, and rowdy types, using different machine learning libraries like TensorFlow, PyTorch, and JAX. The results indicate that adaptive and rowdy activation functions generally provide better predictive accuracy compared to their fixed counterparts for the solution of PDEs.

SUMMARY

The provided summary discusses the critical role of activation functions in neural networks, particularly emphasizing their impact on learning and the accuracy of the network's predictions. Classical activation functions like sigmoid, tanh, ReLU, and their variants are commonly used, with ReLU and its variants being particularly notable for their performance in classification problems. The text also proposes a taxonomy of activation functions more suitable for scientific computation, such as complex-valued activations, and highlights the use of quantized activations for efficiency in large data sets.

Moreover, the summary notes that adaptive activation functions often outperform fixed types due to their ability to better accommodate the intricacies of different problems. Adaptive and rowdy activations show promising results in function approximation and solving differential equations, as evidenced by empirical evaluations using TensorFlow, PyTorch, and JAX machine learning libraries. The discussion concludes by suggesting that the ideal activation function is task-dependent and by highlighting the potential for future research directions in the field of adaptive activations.