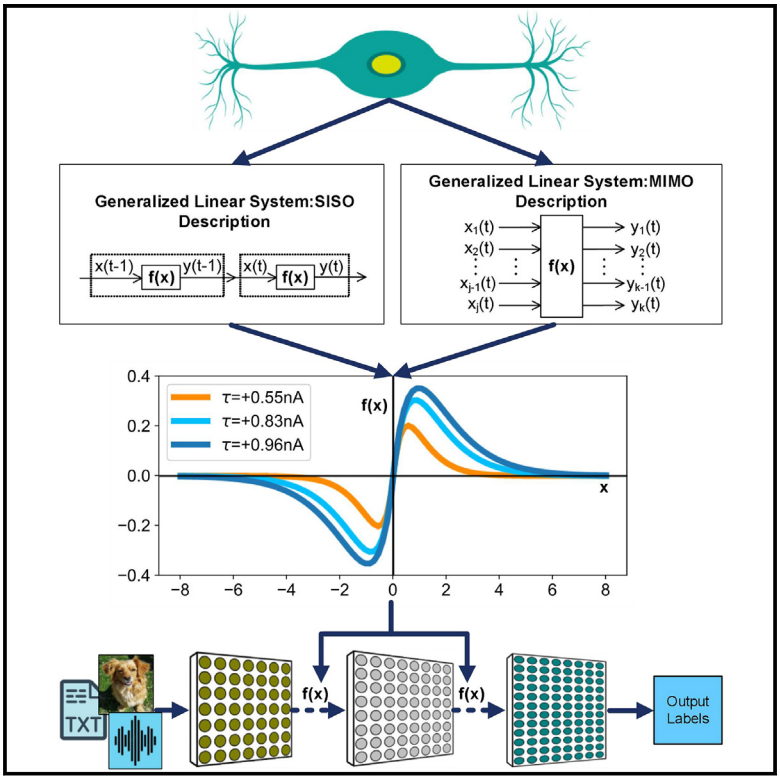
Article

Patterns

Neuron signal attenuation activation mechanism for deep learning

Graphical abstract



Highlights

- Introduce generalized linear system theory to construct neuron input-output model
- A signal attenuation-activation mechanism is proposed for generalized neurons
- The Attenuation (Ant) model is highly consistent with the neuron intracellular recordings
- Ant performs well on various neural network architectures with challenging tasks

Authors

Wentao Jiang, Heng Yuan, Wanjun Liu

Correspondence

lntuyuanheng@163.com

In brief

Although previous studies have identified several activation functions for neural networks, applications often struggle with controlling gradient flow and representing detailed information. This paper presents a model, termed Ant, to formulate an efficient activation function of neurons. Ant is a zero-centered, non-monotonic, smooth, continuous, bounded above, and bounded below activation function. The model is evaluated across several tasks.

Jiang et al., 2025, Patterns 6, 101117
January 10, 2025
https://doi.org/10.1016/j.patter.2024.101117

CellPress

Patterns

CellPress
OPEN ACCESS

Article

Neuron signal attenuation activation mechanism for deep learning

Wentao Jiang,[1,2] Heng Yuan,[1,2,3,*] and Wanjun Liu[1]
[1]Department of Artificial Intelligence, Liaoning Technical University, Huludao 125105, China
[2]These authors contributed equally
[3]Lead contact
*Correspondence: lntuyuanheng@163.com
https://doi.org/10.1016/j.patter.2024.101117

THE BIGGER PICTURE Signal activation in neurons, whereby intricate biochemical pathways and signaling mechanisms can produce different outputs from subtle changes, constitute complex systems to study. The output state varies based on activity patterns, reflecting plasticity. From neuroplasticity, neuron signal activation resembles a "black-box" core with dynamic input-output behavior, integrating and transforming signals from other neurons with varying transmission patterns. Although the neural mechanisms underlying this dynamic behavior are not fully understood, they reflect a wide range of neuronal properties. We test this hypothesis from a generalized linear system perspective by constructing computational models of neuronal activation, which enhances our understanding of the signal-processing properties of generalized neurons. Our results demonstrate strong performance across various neural network architectures.

SUMMARY

Neuron signal activation is at the core of deep learning and broadly impacts science and engineering. Despite growing interest in neuron cell stimulation via amplitude current, the activation mechanism of biological neurons has limited application in deep learning due to the lack of a universal mathematical principle suitable for artificial neural networks. Here, we show how deep learning can go beyond the current learning effects through a newly proposed neuron signal activation mechanism. To achieve this, we report a new cross-disciplinary method for neuron signal attenuation, using the inference of differential equations within generalized linear systems to enhance the efficiency of deep learning. We formulate the mathematical model of the efficient activation function, which we refer to as Attenuation (Ant). Ant can represent higher-order derivatives and stabilize data distributions in deep-learning tasks. We demonstrate the effectiveness, stability, and generalization of Ant on many challenging tasks across various neural network architectures.

INTRODUCTION

Linear systems are the foundation of control theory and are widely used in engineering and scientific fields. The research background of linear systems can be traced back to the early 20th century, when the development of control theory began to attract attention. Early research on linear systems primarily focused on traditional electrical and mechanical systems, such as circuits, generators, and mechanical vibration systems.[1–3] Researchers have established mathematical models to describe the dynamic behavior of these systems and used linear differential equations to characterize their responses. Fundamental concepts such as system stability, controllability, observability, and performance indicators were proposed, leading to the development of many classic system control methods.[4,5]

The research on linear systems is continuously advancing, leading to an expanding scope of inquiry in this field. Linear systems are increasingly being integrated with areas such as signal processing[6] and system identification.[7] As modern control theory progresses, the study of linear systems incorporates more advanced methods, such as adaptive control[8] and optimal control.[9] This evolution has introduced the concept of generalized linear systems, which describe dynamic behaviors using linear differential equations. The origins of linear system studies can be traced back to the early developments in control theory. Research in this area is significant for control and optimization problems in engineering and scientific fields and provides a theoretical basis and practical methods for solving challenges in practical applications.

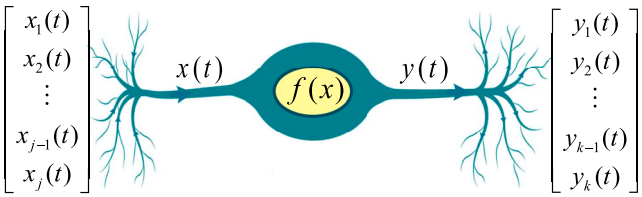


Figure 1. Generalized constrained biological neuron model with SISO (single-input single-output), SIMO (single-input multiple-output), MISO (multiple-input single-output), and MIMO (multiple-input multiple-output)

There is a close relationship between generalized linear systems and linear systems. Generalized linear systems encompass a broader range of linear systems, which serve as a natural representation of objective systems, and offer the ability to describe a wider range of associated performance characteristics. Linear systems have superposition and uniformity, which means that when the input of the system changes, the output of the system can be obtained through the linear combination of various inputs. Generalized linear systems have a more general structure while maintaining linearity and have a broader range of applications in large-scale systems and singular perturbation theory.[10] Generalized linear systems are a natural extension of linear systems, which have more general performance characteristics while maintaining linearity. Generalized linear systems have more prosperous application and research methods than linear systems and can better describe and control actual systems.

Generalized linear systems constitute a cross-disciplinary method that plays a vital role in artificial intelligence. They can describe and process various data types and provide flexible and efficient solutions for different signal-processing tasks. Diverse data types, including frequency domain data, can be modeled and analyzed using generalized linear systems.[11] Whether the data are numerical, a classification, or count data, appropriate models can be established by selecting appropriate probability distributions and connection functions, making generalized linear systems widely applicable in data mining, machine learning, and statistical analysis tasks.

Drawing on the ideas of search-optimization algorithms in machine learning, linear system reduction has achieved high efficiency and accuracy[12]; it verifies that generalized linear systems can extract and represent data features. Selecting appropriate connection functions and regularization methods can extract useful features from the original data for subsequent model training and prediction. This method is beneficial in fields such as computer vision, natural language processing, and speech recognition, as it can improve the model's expressive and generalization abilities.

Generalized linear systems demonstrate high performance and broad application value for online robust feedback and identification tasks.[13,14] By establishing appropriate models, input data can be classified and identified. This method has essential important applications in tasks such as image classification, text classification, speech recognition, and sentiment analysis. The flexibility of generalized linear systems enables them to adapt to different data types and task requirements.

The system modeling methods using generalized linear systems have shown the best performance in dealing with unknown noise and signal jump problems.[15,16] This method can carry out learning and decision making in dynamic environments by establishing appropriate models and defining appropriate reward functions. As a powerful signal analysis and system modeling tool, generalized linear systems could also play an important role in artificial intelligence. The flexibility and efficiency of generalized linear systems enable them to adapt to different types of data and task requirements to provide effective solutions.

Table 1. Similarities and differences of the main activation functions (symbols represent the presence or absence of corresponding attributes)

Activation function	Smooth	Zero centered	Mean near zero	Normalization	Gradient preserved	Saturation	Dead neurons
ReLU	×	×	×	×	✓	×	✓
ELU	×	×	×	×	✓	×	×
Leaky-ReLU	×	×	×	×	✓	×	×
PReLU	×	×	×	×	✓	×	×
DRelu	×	×	×	×	✓	×	×
Selu	×	×	×	×	✓	×	×
GELU	✓	×	×	×	✓	×	×
ACON-C	✓	×	×	×	✓	×	×
Swish-beta	✓	×	×	×	✓	×	×
Mish	✓	×	×	×	✓	×	×
Softmax	✓	×	×	×	×	×	×
SoftPlus	✓	×	×	×	×	×	×
Sigmoid	✓	×	×	×	×	✓	×
Sinc	✓	×	×	×	×	✓	×
Tanh	✓	✓	×	×	×	✓	×
Siren	✓	✓	✓	✓	×	✓	×
Ant	✓	✓	✓	✓	✓	×	×

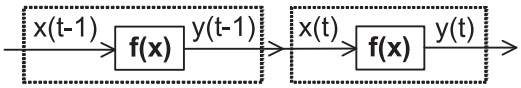


Figure 2. Neuron activation in a time-series neural network with SISO description

Conducting in-depth research on the theory and applications of generalized linear systems can provide strong support for the development and innovation of artificial intelligence.

The successful application of generalized linear systems in various engineering fields has led us to introduce this theory into the realm of computational modeling in neuroscience (Figure 1). Here, we present a model that describes the attenuation-activation mechanism of biological neurons with generalized constraints. This model serves as the activation function for neural networks, enhancing their learning performance across various data science applications.

Background

Non-linear transformation is the core of every neural network, as it enhances the network's performance by introducing non-linearity through activation function $f(\,)$. The neuron-activation function plays a major role in the performance of every neural network. Although neuroscientists have conducted extensive tests on the electrical signals of neurons,[17–19] the signal activation process of neurons still lacks a concise mathematical principle. Currently, in the deep-learning community, the most successful and widely used activation function is the hand-designed Rectified Linear Unit (ReLU),[20–22] which is defined as $f(x) = \max(x,0)$. ReLU has become the default and standard activation function in almost all learning communities thanks to its simplicity and consistent performance.[23,24]

For many years, various neuron-activation functions have been created to replace ReLU, including Swish,[23] Mish,[25] Leaky Rectified Linear Unit (Leaky ReLU),[26] Exponential Linear Unit (ELU),[27] Dynamic Rectified Linear Unit (Dynamic ReLU),[28] and Gaussian Error Linear Unit (GELU),[29] along with many others. However, most activation functions proposed to replace ReLU, either parametric or non-parametric, static or dynamic, are hand-designed to fit properties deemed to be important and have functional profiles similar to those of ReLU or share similar functional properties. Thus, we can consider them as ReLU's generalizations or part of the "the ReLU family."

Although the ReLU family is widely used, these strategies have several drawbacks. The first is that they are unbounded above

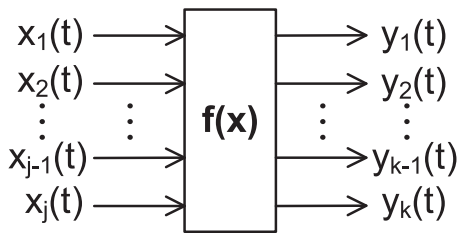


Figure 3. Generalized constrained single neuron with MIMO description

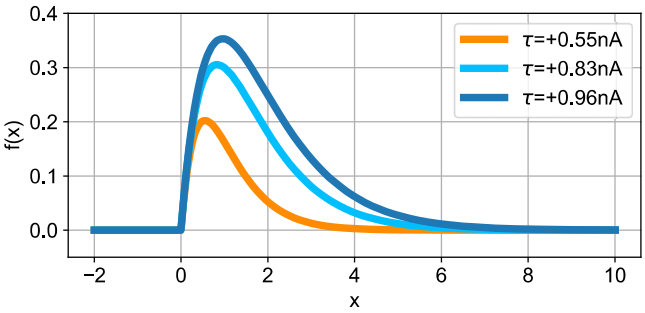


Figure 4. Neuron-activation model of Equation 40 with three current signal range scales

and not saturated on the positive axis, which leads to an internal covariate shift in the training process.[30] The second is that the positive part is linear or approximately linear, which leads to not having the ability to accurately capture intricate details in the underlying signals.[31] The third is that ReLU's generalizations, especially parametric functions and dynamic functions, lead to the extra cost of learning tasks.

The ReLU family has similar functional profiles and properties. They were never beyond the framework set by ReLU. Therefore, neural networks with the ReLU family have an upper bound of performance. In this paper, inspired by theories of linear systems, we break through the old framework of ReLU and propose a new activation function, which we term Attenuation (Ant). Unlike the ReLU family, Ant is a zero-centered, non-monotonic, smooth, continuous, bounded above, and bounded below activation function. Thanks to these properties, Ant can avoid the gradients from vanishing, represent higher-order derivatives, and change data distributions in the networks. We have summarized the similarities and differences of the current main activation functions in Table 1 and have evaluated the activation properties of Ant.

To evaluate the effectiveness and generalization ability of Ant, we conduct extensive experiments on various network architectures applied to a variety of challenging domains such as image classification, two spirals, image fitting, curve fitting, solving

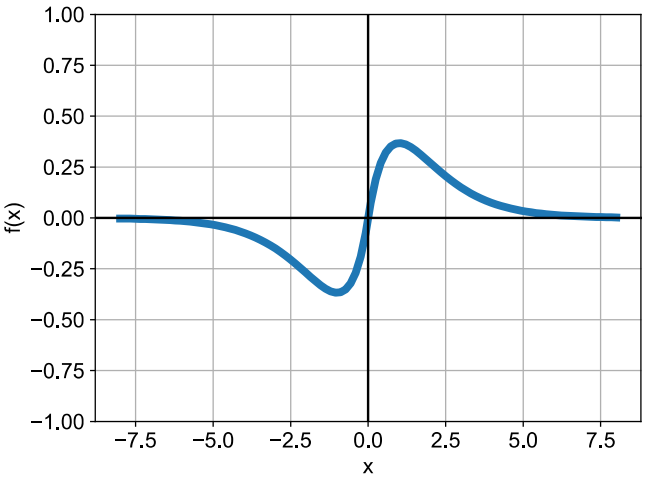


Figure 5. The Ant activation function

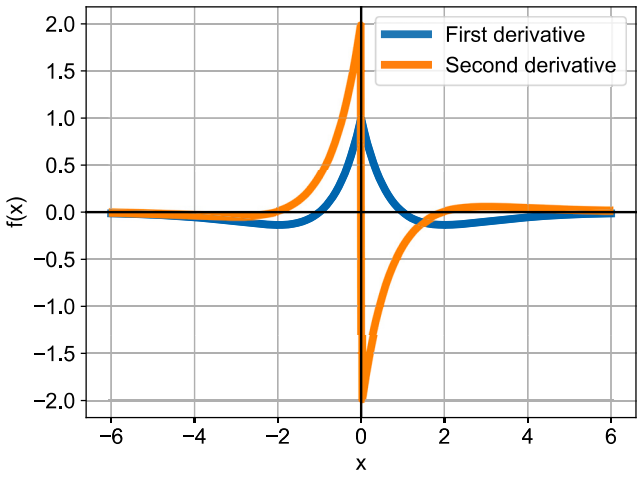


Figure 6. The first and second derivatives of Ant

partial differential equations (PDEs), solving the exclusive-or (XOR) problem, semi-supervised text classification, and natural language processing. Additionally, to evaluate the stability of Ant, we conduct experiments by increasing the number of neural network layers. Experimental results show that Ant consistently outperforms other activation functions on all models and datasets, which demonstrates the effectiveness, stability, and good generalization ability of Ant to match these varying architectures and domains.

The main contributions of this paper can be summarized as follows.

(1) A new signal activation function called Attenuation (Ant) is proposed, which is similar to biological neuron-firing recording.
(2) Ant has both better accuracy and lower loss in simple and complex networks than ReLU, ELU, GELU, Selu, Tanh, Sigmoid, Siren, Swish, and Mish.
(3) Ant is a much more effective activation function for addressing many challenging problems in different domains on various network architectures.

METHODS

The generalized linear system, which is a natural representation of objective systems, is a powerful tool for describing real systems and is widely used in science and engineering. It is more natural and accurate to use the generalized linear system to portray the black-box problems often encountered in real systems, and it can effectively describe its potential performance and characteristics.

A generalized constrained biological neuron can be treated as a system limited by multiple time-domain functions; as shown in Figure 1, the system can be systematically modeled using a generalized linear system. The system has input-output constraints and unidirectional signaling along with four input-output modes, which are SISO (Single-Input Single-Output), SIMO (Single-Input Multiple-Output), MISO (Multiple-Input Single-Output), and MIMO (Multiple-Input Multiple-Output). The SISO mode can describe the overall input-output of the neuron and can be modeled using SISO generalized constraints. And for SIMO and MISO modes, both can be systematically modeled with MIMO generalized constraints. Artificial neurons are a reduction of biological neurons under generalized constraints; they are abstractions of biological neural behaviors, which can be reduced to a few key characteristics.

(1) Axons exhibit SISO behavior through non-linear transformation after integrating over all inputs.
(2) Dendrites and synapses exhibit MIMO behavior through signal non-linear transformation.

The two key characteristics above may seem unrelated but both are based on the non-linear transformation $f(x)$. Inspired by this, we simulated, inferred, and demonstrated the neuron-activation mathematical model using generalized linear

Table 2. Comparison results of neuron models

				Biological plausibility		
Model name	Dimensions	Parameters	Modeling type	Behavioral diversity	Behavioral universality	Computational simplicity
CTRNN[36]	2	8	non-conductance	high	low	×
H-H[37]	4	10	conductance	high	low	×
MCN[38]	4	9	non-conductance	high	low	×
Connor[39]	5	9	conductance	high	low	×
Chay[40]	3	11	conductance	high	low	×
Izhikevich[41]	2	5	non-conductance	high	medium	×
EIF[42]	2	7	non-conductance	medium	medium	×
Wilson[43]	4	8	conductance	medium	medium	×
IFB[44]	2	11	non-conductance	medium	medium	×
aEIF[45]	2	9	non-conductance	medium	medium	×
Mihalas[46]	4	13	non-conductance	low	medium	×
RF[47]	2	3	non-conductance	low	high	✓
LIF[48]	1	5	non-conductance	low	high	✓
QIF[49]	1	3	non-conductance	low	high	✓
Ant	1	2	non-conductance	low	high	✓

Table 3. Overview of the experimental data types and hyperparameters

Section	Data type	Loss function	Optimizer	Learning rate	Batch size	Epoch/iteration
Generalization performance on CNN	image	Categorical_crossentropy	Adam/SGD	0.0001	128	100
Solving two-spirals task	spatial data	Categorical_crossentropy	SGD	0.05	128	1,000
Solving the image-fitting problem	image	Mseloss	Adam	0.01	256	10,000
Solving curve fitting	spatial data	Mseloss	Adam	0.001	100	10,000
Solving the partial differential equation	sequential data	Mseloss	Adam	0.0001	50	10,000
Solving the XOR problem	coordinate data	Mseloss	SGD	0.1	4	1,000
Increasing the number of network layers	image	Categorical_crossentropy	SGD	0.01	128	10
Generalization performance on GNN	textual data	Sparse_categorical_crossentropy	Adam	0.01	256	300
Language translation on Transformer	sequential data	Crossentropyloss	Adam	0.0005	32	20
Text classification on BERT	textual data	Crossentropyloss	RAdam	0.00002	64	10
Emotion recognition on Vision Transformer	image	Crossentropyloss	RAdam	0.00002	16	10
Named entity recognition on LLaMA	textual data	Crossentropyloss	RAdam	0.001	16	10

systems and differential equations. The system modeling process is built on the generalized constrained biological neural model and the constraints of the two key characteristics mentioned above. In a research approach different from that of neuroscientists, we obtained a mathematical model for neuronal activation. The explanation of the derivation process of the proposed model is as follows.

System modeling: SISO description

The SISO description is a system modeling method after integrating all the inputs, which can describe the potential behaviors and characteristics of the system from a macroscopic point of view and can obtain time-domain mathematical equations that can directly characterize the system behavior. Neuron activation is responsible for introducing non-linearity to biological neural networks. The process of neuron activation in generalized constrained neural networks can be realized as $y = f(x)$, where x is the input and y is the output of the activation function $f(x)$. Each connected neuron in neural networks with a single input and a single output can be viewed as a SISO system. Almost all SISO systems observed in practice become non-linear.

Generally, the analysis of non-linear systems is difficult. However, it is possible to approximate most of the non-linear systems by linear systems for small-signal analysis. Thus, we can consider two connected neurons in a neural network as a linear system. From the perspective of generalized linear system analysis, a neuron can be illustrated by a black-box system with one input and one output. The process of signal activation by connected neurons in a time-series neural network can be simply shown as in Figure 2.

There are two different neurons of the same type in Figure 2. Inside the neuron, there is a function $f(x)$ with the input signal as the independent variable, while outside the neuron there are functions $x(t)$ and $y(t)$ with time t as the independent variable. These functions describe the input-output relationship of neurons of the same type across two dimensions.

To eliminate the differences between different types of signals and consider a more general form, here we consider $x(t)$ and $y(t)$ as generalized signals. Of course, they can both be concretized by potential differences or other types of signals. For example, Abbott et al.[19] measured the firing signals of thousands of interconnected biological neurons, and their inputs and outputs were recorded as potential difference signals. Of course, the types of signals used in such experiments can also be other types, which are determined by different neuroscience experiments and will not be discussed in depth here. As shown in Figure 2, we discuss a simplified time-series neural network with only two neurons of the same type and establish a SISO description. When two neurons are independently connected without the participation of other neurons, the connection between two adjacent neurons can be regarded as the axon of a biological neuron. The axon serves as the output channel of the neuron and can transmit the signal generated by the neuron to other neurons without loss. Therefore, we can equivalently regard the input of the current neuron as the output of the previous neuron.

As shown in Figure 2, we consider a connected neuron system with two neurons as a time-series neural network, $x(t)$ is the input signal of the current neuron system, and $y(t)$ is the output signal of the current neuron. As mentioned above, $x(t)$ is also the output signal of the previous neuron without any output from others. Thus,

Table 4. Testing accuracy of different activation functions on several image classification datasets across various CNN models

Model	Dataset	Ant	ReLU	ELU	Leaky-ReLU	GELU	ACON-C	Swish-beta
VGG7	CIFAR-10	0.8476	0.8352	0.826	0.8327	0.8204	0.8251	0.8146
	CIFAR-100	0.5711	0.5161	0.5310	0.5348	0.5094	0.5440	0.5073
	Fashion-MNIST	0.9435	0.9404	0.9326	0.9352	0.939	0.9275	0.9386
ResNet18	ImageNette	0.8148	0.7552	0.7508	0.7225	0.7679	0.7676	0.7850
ResNet34		0.8189	0.7850	0.6968	0.6846	0.7070	0.7326	0.7366
ResNet50		0.8290	0.7651	0.6902	0.7152	0.7355	0.7712	0.7704

Table 5. Testing loss of different activation functions on several image classification datasets across various CNN models

Model	Dataset	Ant	ReLU	ELU	Leaky-ReLU	GELU	ACON-C	Swish-beta
VGG7	CIFAR-10	0.4800	0.8093	0.9802	0.8767	0.9491	0.9528	0.9558
	CIFAR-100	1.5927	3.5994	3.5364	3.1913	3.5617	2.3949	3.4485
	Fashion-MNIST	0.1898	0.3326	0.412	0.3466	0.3522	0.3631	0.3853
ResNet18	ImageNette	0.6748	1.3999	1.8977	1.7121	1.7156	1.2048	1.7970
ResNet34		0.7370	1.3796	2.3710	1.8986	1.6620	1.6305	1.5718
ResNet50		0.7578	1.3502	3.2350	1.8488	1.3850	1.9859	1.5091

$$x(t) = y(t-1). \quad \text{(Equation 1)}$$

In the SISO system, input is equal to output plus variation. Taking the current neuron as an example, the signal variation is obtained by subtracting its output signal $y(t)$ from its input signal $x(t)$:

$$\Delta y(t) = x(t) - y(t) = y(t-1) - y(t). \quad \text{(Equation 2)}$$

Assuming the ideal case of a neuron with a constant activation time, the activation time of a single neuron can be set to 1, i.e., concerning normalizing all the activation time to it, calculate the differentiation on $y(t)$ with discrete calculus:

$$\frac{dy}{dt} = \lim_{\Delta t \to 1} \frac{\Delta y}{\Delta t} = \frac{y(t-1) - y(t)}{1} = y(t-1) - y(t). \quad \text{(Equation 3)}$$

Thus,

$$\frac{dy}{dt} = y(t-1) - y(t) = x(t) - y(t) \quad \text{(Equation 4)}$$

or

$$\frac{dy}{dt} + y(t) = x(t). \quad \text{(Equation 5)}$$

It proves convenient to use a compact notation D for the differential operator d/dt. Thus,

$$\frac{dy}{dt} = Dy(t). \quad \text{(Equation 6)}$$

With the notation, Equation 5 can be expressed as

$$Dy(t) + y(t) = x(t) \quad \text{(Equation 7)}$$

or

$$(D+1)y(t) = x(t). \quad \text{(Equation 8)}$$

As mentioned above, a neuron system can be specified by Equation 8.

According to the theories of linear systems, determine the unit impulse response $h(t)$ for neuron system specified by Equation 8. This is a first-order system having the characteristic polynomial

$$\lambda + 1. \quad \text{(Equation 9)}$$

The characteristic root of this system is $\lambda = -1$. Therefore,

$$y_n(t) = ce^{-t}. \quad \text{(Equation 10)}$$

The initial condition is

$$y_n(0) = 1. \quad \text{(Equation 11)}$$

Setting $t = 0$ in Equation 10 and substituting the initial condition just given, we obtain

$$c = 1. \quad \text{(Equation 12)}$$

Therefore,

$$y_n(t) = e^{-t}. \quad \text{(Equation 13)}$$

Thus, the unit impulse response $h(t)$ for the neuron system specified by Equation 8 is

$$h(t) = y_n(t)\mu(t) = e^{-t}\mu(t). \quad \text{(Equation 14)}$$

The function $\mu(t)$ in Equation 14 is the unit step function, which is defined by

$$\mu(t) = \begin{cases} 1 & t \geq 0 \\ 0 & t < 0 \end{cases}. \quad \text{(Equation 15)}$$

We can learn from the theories of linear systems, the system response being the sum of its responses to various impulse components. If we know the system response to an impulse input, we can determine the system response to an arbitrary input (x). Therefore, we determine the system response to unit impulse response $h(t)$ by Equation 14:

$$y(t) = h(t) * h(t) = e^{-t}\mu(t) * e^{-t}\mu(t). \quad \text{(Equation 16)}$$

Table 6. Comparison of Ant and ReLU on ImageNet2012 dataset across different depths of ResNets

		Ant				ReLU			
Model	Dataset	FLOPs(G)	Params(M)	Top-1 accuracy	Top-1 loss	FLOPs(G)	Params(M)	Top-1 accuracy	Top-1 loss
ResNet18	ImageNet 2012	2.2347	11.1339	0.6359	1.6342	2.2347	11.1339	0.6252	2.0138
ResNet34		4.2719	21.3521	0.6773	1.7779	4.2719	21.3521	0.6627	2.1354
ResNet50		4.7688	23.5854	0.7046	1.9287	4.7688	23.5854	0.6919	2.3403
ResNet101		8.4501	42.6484	0.7254	1.9589	8.4501	42.6484	0.7055	2.4251
ResNet152		11.3848	57.1676	0.7291	2.1826	11.3848	57.1676	0.7146	2.6136

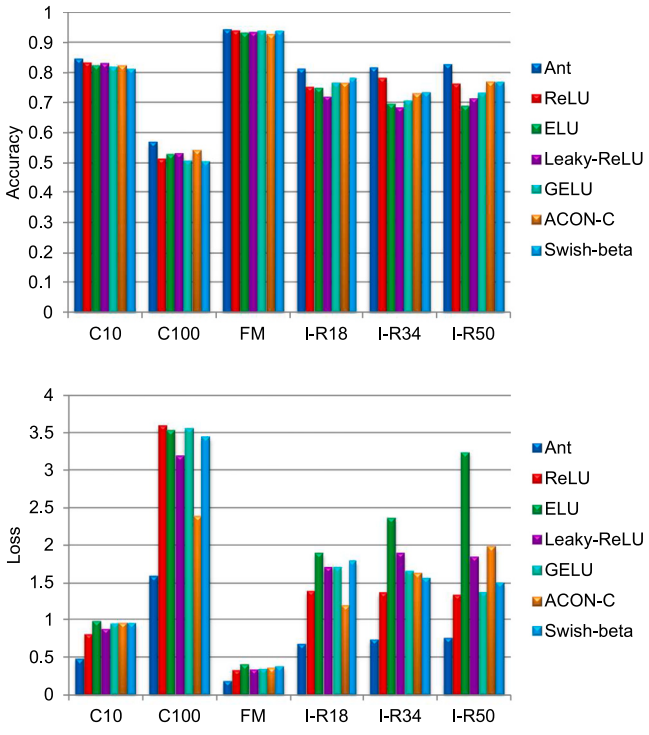


Figure 7. Comparison results of testing accuracy and testing loss for different activation functions (C10, CIFAR-10; C100, CIFAR-100; FM, Fashion-MNIST; I-R18, ImageNette-ResNet18; I-R34, ImageNette-ResNet34; I-R50, ImageNette-ResNet50)

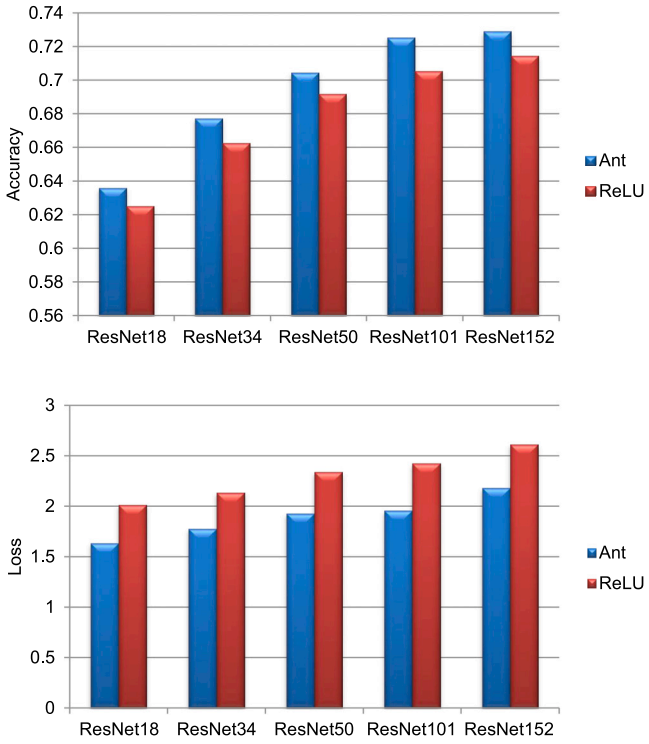


Figure 8. Comparison results of testing accuracy and testing loss for Ant and ReLU on ImageNet

The convolution integral of two functions $h(t)$ and $h(t)$ is denoted symbolically by $h(t) * h(t)$. We can learn from convolution integral that, the system response $y(t)$ specified by Equation 8 is

$$y(t) = te^{-t}\mu(t). \quad \text{(Equation 17)}$$

If we know the system response to an impulse input, we can determine the system response to an arbitrary input (x). That is, as long as we know the system response to an impulse input, we completely know this system. Thus, we consider Equation 17 can represent the SISO neuron system specified by Equation 8.

As mentioned above, we apply Equation 17 as a system model in a SISO description, so Equation 17 can be expressed as

$$f(t) = te^{-t}\mu(t). \quad \text{(Equation 18)}$$

The SISO description can only characterize the behavior and characteristics of the system at a macro level. If a more accurate system model is desired, the system needs to be modeled from a micro level using the MIMO description method.

System modeling: MIMO description

The SISO mostly deals with the external description of the system. For neuronal systems with multiple input-output mode constraints, the SISO description is inadequate, and we need to use the MIMO description method to deal with an internal description of the system for micro-level system modeling.

The most suitable method for modeling MIMO systems at present is state-space analysis, which can determine all possible system outputs based on system inputs and system initial states as well as all possible system initial states based on system inputs and system outputs. It is an internal description method of the system and can provide mathematical models of great generality for MIMO system modeling.

For the SIMO, MISO modes included in Figure 1 can all be modeled using the MIMO description method. A single-neuron system with multiple input and output constraints is represented as shown in Figure 3, where the system can be treated as a black box with a set of observable input variables $x_1(t), x_2(t), \ldots, x_j(t)$ corresponding to another set of observable output variables $y_1(t), y_2(t), \ldots, y_k(t)$, and we can establish the state-space equation of this system to solve for the system state function $f(x)$. Thus, we can establish the state-space equation

$$\dot{\mathbf{f}} = \mathbf{A}\mathbf{f} + \mathbf{B}x, \quad \text{(Equation 19)}$$

$$\sum_{n=1}^{k} \mathrm{y}_n(t) = \mathbf{f}\left(\sum_{m=1}^{j} \mathrm{x}_m(t)\right), \quad \text{(Equation 20)}$$

where $\mathbf{f}$ is the state variable, $\mathbf{A}$ and $\mathbf{B}$ are the parameter variables, which vary with time, x is the input variable, and y is the output variable. Equation 20 is the generalized constrained prerequisite of the state-space equation from Figure 3. Now multiplying both sides of Equation 19 by Laplace transform $e^{-\mathbf{A}t}$, we obtain

$$e^{-\mathbf{A}t}\dot{\mathbf{f}} = e^{-\mathbf{A}t}\mathbf{A}\mathbf{f} + e^{-\mathbf{A}t}\mathbf{B}x \quad \text{(Equation 21)}$$

or

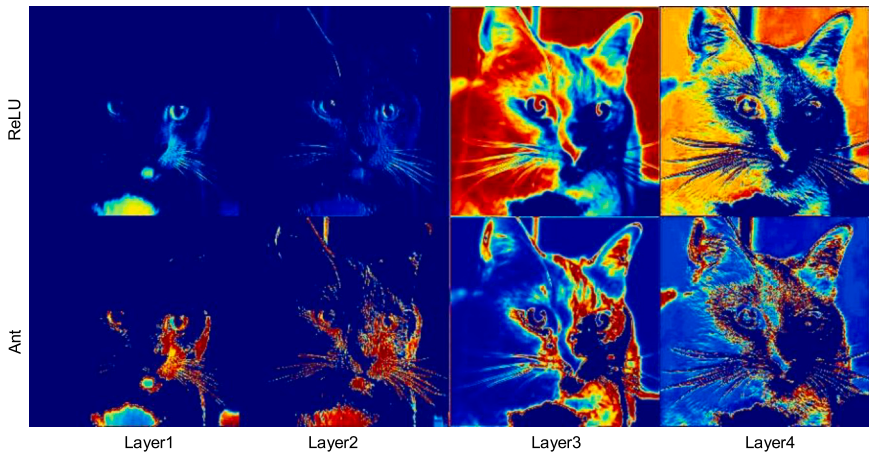


Figure 9. Feature maps of different layers activated by ant and ReLU in ResNet (red pixels represent positive features with high response, while blue pixels represent negative features with low response)

$$e^{-\mathbf{A}t}\dot{\mathbf{f}} - e^{-\mathbf{A}t}\mathbf{A}\mathbf{f} = e^{-\mathbf{A}t}\mathbf{B}x. \quad \text{(Equation 22)}$$

Calculate the differentiation on the left side of Equation 21:

$$\frac{d}{dt}\left[e^{-\mathbf{A}t}\mathbf{f}\right] = \left(\frac{d}{dt}e^{-\mathbf{A}t}\right)\mathbf{f} + e^{-\mathbf{A}t}\dot{\mathbf{f}} = -e^{-\mathbf{A}t}\mathbf{A}\mathbf{f} + e^{-\mathbf{A}t}\dot{\mathbf{f}}. \quad \text{(Equation 23)}$$

Compared with Equation 23, it can be seen that the left side of Equation 22 is

$$\frac{d}{dt}\left[e^{-\mathbf{A}t}\mathbf{f}\right]. \quad \text{(Equation 24)}$$

Hence,

$$\frac{d}{dt}\left[e^{-\mathbf{A}t}\mathbf{f}\right] = e^{-\mathbf{A}t}\mathbf{B}x. \quad \text{(Equation 25)}$$

Integrating both sides of Equation 25 from 0 to t yields

$$e^{-\mathbf{A}t}\mathbf{f}\Big|_0^t = \int_0^t e^{-\mathbf{A}\tau}\mathbf{B}x(\tau)d\tau \quad \text{(Equation 26)}$$

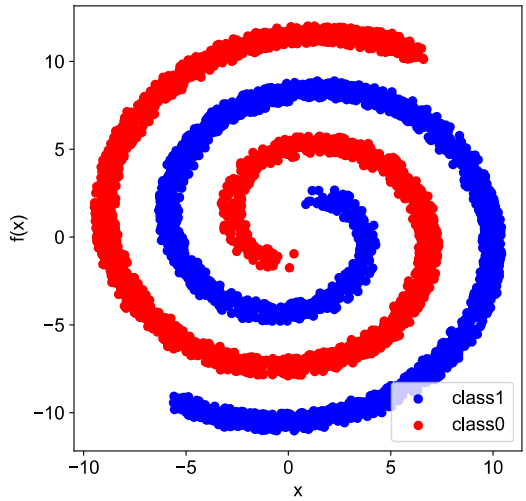


Figure 10. Training points of two-spirals task

or

$$e^{-\mathbf{A}t}\mathbf{f}(t) - \mathbf{f}(0) = \int_0^t e^{-\mathbf{A}\tau}\mathbf{B}x(\tau)d\tau. \quad \text{(Equation 27)}$$

Hence,

$$e^{-\mathbf{A}t}\mathbf{f}(t) = \mathbf{f}(0) + \int_0^t e^{-\mathbf{A}\tau}\mathbf{B}x(\tau)d\tau. \quad \text{(Equation 28)}$$

Multiplying both sides of Equation 28 forward by another Laplace transform $e^{\mathbf{A}t}$, we have

$$\mathbf{f}(t) = e^{\mathbf{A}t}\mathbf{f}(0) + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{B}x(\tau)d\tau. \quad \text{(Equation 29)}$$

The first term on the right side of Equation 29 represents the zero-input component, and the second term represents the zero-state component. The result of Equation 29 can be approximated using the definition of matrix convolution in a generalized linear system, so Equation 29 can be approximated as

$$\mathbf{f}(t) = e^{\mathbf{A}t}\mathbf{f}(0) + e^{\mathbf{A}t}\mathbf{B}x(t). \quad \text{(Equation 30)}$$

Since the limit of convolution integral on the right side of Equation 30 is from 0 to t, all elements on the right side implicitly have the multiplicative term $\mu(t)$:

$$\mathbf{f}(t) = e^{\mathbf{A}t}\mathbf{f}(0)\mu(t) + e^{\mathbf{A}t}\mathbf{B}x(t)\mu(t) = [\mathbf{f}(0) + \mathbf{B}x(t)]e^{\mathbf{A}t}\mu(t). \quad \text{(Equation 31)}$$

This is the desired solution to the state-space equation. In the next step, we will combine the SISO description and the MIMO description to solve the final system state function $f(x)$.

Combined modeling: SISO and MIMO

We take the system state function $f(x)$ described by SISO and MIMO as a neuron-activation function with generalized constraints, and thus we need to solve the generalized solutions of the parameter matrices **A** and **B** jointly by the two description methods.

Two of the key mathematical models are Equation 18 and Equation 31, and comparing the two equations above, we obtain a joint system:

Table 7. Testing accuracy, loss, and average convergence time of different activation functions on two-spirals task

	Task	Ant	ReLU	ELU	Leaky-ReLU	GELU	ACON-C	Swish-beta
Accuracy	two spirals	0.9820	0.7518	0.9690	0.9740	0.9730	0.5095	0.9620
Loss		0.0613	0.3772	0.0787	0.0799	0.1161	–	0.0956
Time (s)		0.0031	0.0283	0.0125	0.0076	0.0143	–	0.0145

$$te^{-t}\mu(t) = [\mathbf{f}(0) + \mathbf{B}x(t)]e^{\mathbf{A}t}\mu(t). \quad \text{(Equation 32)}$$

From Equation 20, the joint system satisfies homogeneity; thus,

$$te^{-t} = [\mathbf{f}(0) + \mathbf{B}x(t)]e^{\mathbf{A}t}. \quad \text{(Equation 33)}$$

Therefore,

$$\begin{matrix} t = \mathbf{f}(0) + \mathbf{B}x(t) \\ e^{-t} = e^{\mathbf{A}t} \end{matrix}. \quad \text{(Equation 34)}$$

From Equation 34 we find that

$$\begin{matrix} \mathbf{A} = -1 \\ \mathbf{B} = [t - \mathbf{f}(0)]/x(t) \end{matrix}. \quad \text{(Equation 35)}$$

The zero-state response of the system is 0, and $x(t)$ is the unit time input when t converges to the minimum discrete time 1; thus, $x(1) = \mu(1) = 1$. Therefore,

$$\begin{matrix} \mathbf{B} = 1 \\ \mathbf{f}(0) = 0 \end{matrix}. \quad \text{(Equation 36)}$$

From Equation 35 we obtain

$$t = \mathbf{B}x(t). \quad \text{(Equation 37)}$$

Thus, the system state function $f(x)$ can be described as

$$f[x(t)] = x(t)e^{-x(t)}\mu[x(t)]. \quad \text{(Equation 38)}$$

When t is a time constant, Equation 38 is approximately reduced as

$$f(x) = xe^{-x}\mu(x). \quad \text{(Equation 39)}$$

We take Equation 39 as the generalized constrained neuron-activation mechanism with a signal range scale parameter τ. As an activation mechanism, Equation 39 can be expressed as

$$f(x) = xe^{-x/\tau}\mu(x). \quad \text{(Equation 40)}$$

The neuron-activation mechanism, which has a high degree of similarity between the mathematical waveform of Figure 4 and the biological neuron-firing recording shown in Figure S9D of Abbott et al.,[19] is highly intuitive and interpretable from the perspective of the generalized linear system. It can be seen that this activation mechanism induces a gradual decay of the input signal over time, which brings the system close to a zero-input steady state as a resting (non-signaling) neuron[32] for the next input.

The waveform shown in Figure S9D of Abbott et al.[19] is a neuron-firing recording with potential differences as input and output signals, and it contains some obvious pulse signals. The pulse neuron model is a simplified and abstract representation of biological neurons. It studies the functions and mechanisms of the nervous system by simulating the pulse-firing behavior of neurons. It is a neural model that takes potential difference as input and pulse or pulse frequency as output. The proposed Ant model and its applications do not involve pulse-related content. Ant establishes a neuron model with a generalized abstract signal as the input-output relationship, and this generalized abstract signal can be a potential difference; for example, Figure 4 is the result of concretizing the generalized abstract signal as a potential difference. As can be seen from the comparison, the output waveform of Ant has a highly similar potential difference waveform to that of the biological neuron-firing recording, with the main difference being that the potential difference waveform of the neuron-firing recording has multiple pulse spikes, whereas the output waveform of Ant is smoother and has no pulse spikes. This suggests that Ant has some biological plausibility in terms of mainstream signaling with the main characteristics of biological neuron-firing recording.

According to the activation function's writing habit, changing the styles of expression, Equation 40 can be expressed as

$$f(x) = \begin{cases} xe^{-x/\tau} & x \geq 0 \\ 0 & x < 0 \end{cases}. \quad \text{(Equation 41)}$$

In reality, the input x is positive but in the artificial neural network, input can both be positive and negative. So, taking the activation function zero-centered, we obtain

$$f(x) = \begin{cases} xe^{-x/\tau} & x \geq 0 \\ xe^{x/\tau} & x < 0 \end{cases}. \quad \text{(Equation 42)}$$

To recap briefly, Ant is defined as

$$f(x) = xe^{-|x/\tau|}. \quad \text{(Equation 43)}$$

Table 8. Image-fitting precision of different activation functions (intersection-over-union thresholds based on pixel values)

Activation function	mAP	AP_{50}	AP_{75}
Ant	65.019	79.407	67.174
ReLU	36.821	59.195	39.398
Siren	48.616	67.759	52.927

Table 9. PSNR of different activation functions on image fitting

Image	Ant	ReLU	Siren
Cat	70.485	63.975	66.589
Dog	66.924	64.107	65.819
Helmholtz_imag	76.881	60.978	69.053
Helmholtz_real	74.915	59.572	64.808

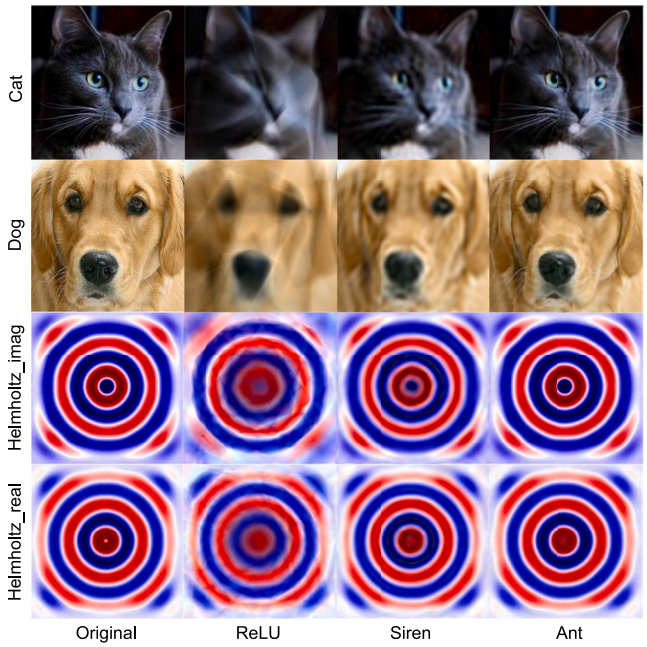


Figure 11. Examples of different activation functions on image fitting

For diversified data-standardization methods, τ can be set as the maximum value within the signal range. We set it to 1 in the experimental section of this paper when the input range has been appropriately normalized to [−1,1] by default; when τ = 1, the first derivative of Ant is

$$f'(x) = \begin{cases} (1-x)e^{-x} & x \geq 0 \\ (1+x)e^{x} & x < 0 \end{cases}, \quad \text{(Equation 44)}$$

and the second derivative of Ant is

$$f''(x) = \begin{cases} (x-2)e^{-x} & x \geq 0 \\ (x+2)e^{x} & x < 0 \end{cases}. \quad \text{(Equation 45)}$$

The graph of Ant is shown in Figure 5. The first and second derivatives of Ant are shown in Figure 6. As shown in Figure 5, Ant is very unlike the ReLU family. Ant is a zero-centered, non-monotonic, smooth, continuous, bounded above, and bounded below activation function. From Ant's mathematical model and its function curve, it can be seen that Ant performs exponential attenuation operations on the strength of all input signals, and the attenuation amplitude has a direct relationship with the strength of the input signals, i.e., the larger the signal strength, the larger the attenuation amplitude, and the smaller the signal strength, the smaller the attenuation amplitude. The attenuation is the

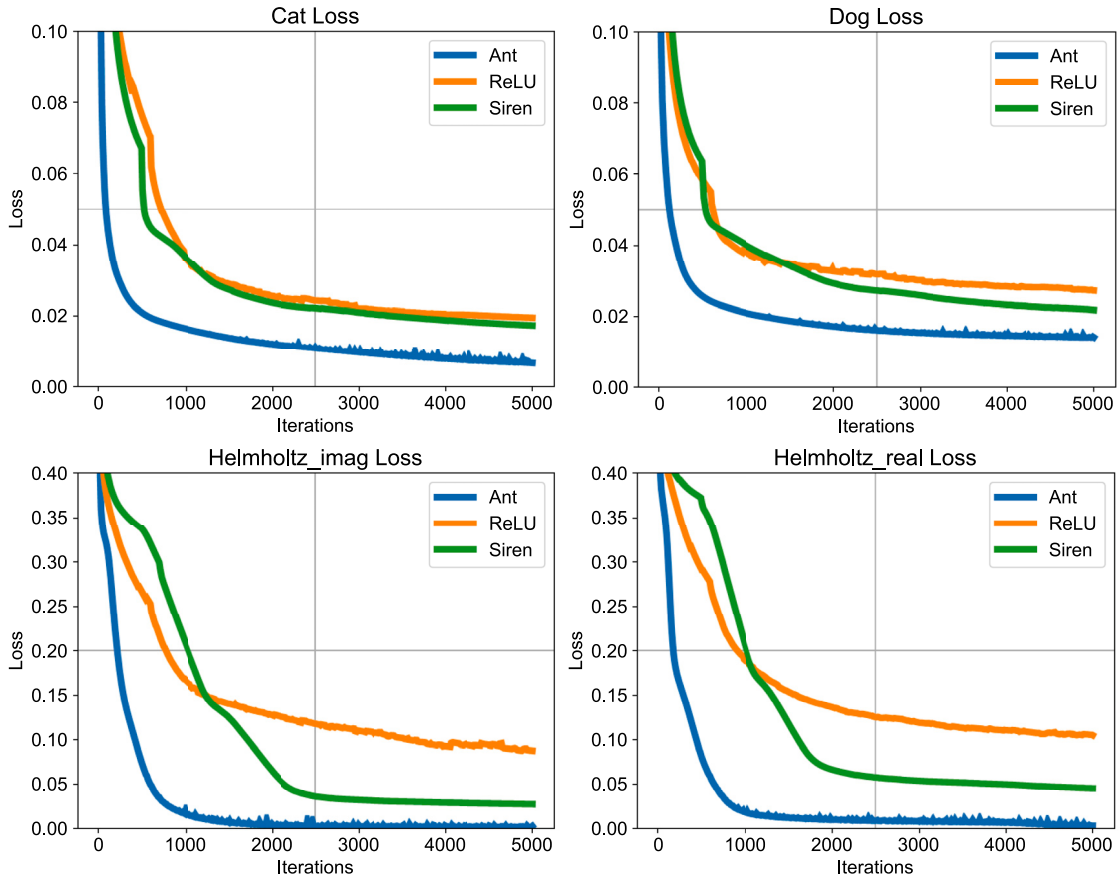


Figure 12. Loss curves of different activation functions on image fitting

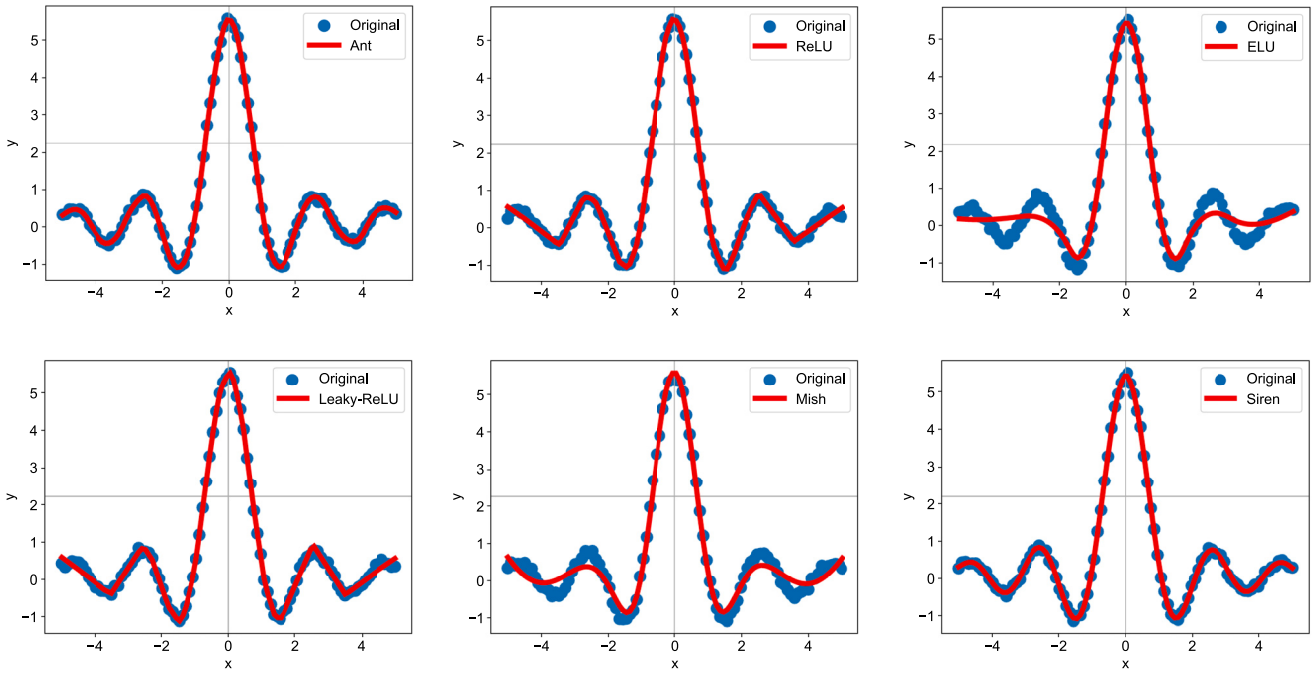


Figure 13. The curves fitted by different activation functions (blue line represents the original training data, and red line represents the fitting result)

feature most distinctive from all other activation functions. Signal attenuation promotes the data distribution of the artificial neural network to be stabilized and helps the neural network to converge to the optimal solution, which is also the essential reason why Ant brings performance enhancement to deep-learning tasks.

As we know, in order to preserve gradient, the ReLU family is unbounded above on the positive half. Because of this property, the ReLU family members do not saturate in the positive half-axis, so they have two strengths: effectively avoiding gradient vanishing and speeding up network convergence. However, this property also leads to two weaknesses. The first is internal covariate shift in the training process,[14] a phenomenon especially obvious in deep architecture. Thus, most networks with the ReLU family need batch normalization to reduce internal covariate shift, which makes network architectures more complicated. The second is that the positive part is linear or approximately linear, which means their second derivative is zero everywhere; these architectures do not have the ability to accurately capture intricate details in the underlying signals.[15]

Different from the ReLU family, Ant is not only bounded below but also bounded above. At first sight, Ant would saturate and vanish of gradient when $x > 6$ and $x < -6$. Actually, however, when the input becomes large ($x > 1$) in the training process, Ant will decrease the large input to obtain a small output. The larger the input, the more likely it is to be decreased. Then, during the next iteration, the input will tend to 0 and the gradient will tend to 1. Similarly, when the input becomes small ($x < -1$), Ant will increase the small input to obtain a large output, and the smaller the input, the more likely it is to be increased. During the next iteration, the input and the gradient will then also tend to 0 and 1, respectively. This property we can call attenuation. Because of this property, Ant function has several advantages.

(1) The first advantage is the mean near zero. The deeper the network, the more near to zero is the mean. The zero-mean activation function enables the gradient to be closer to the natural gradient, thereby enhancing the learning efficiency.[11,33] Figure 19 shows this property in the XOR experiment.

(2) The second advantage is gradient preservation. In the training process, by decreasing the large input and increasing the small input, the vast majority of inputs would fall inside the domain of $-6 \leq x \leq 6$. As shown in Figure 6, the domain of gradient is $[-0.135,1]$ when $-6 \leq x \leq 6$. As $|x|$ increases, the gradients are able to flow and tend to 1 during the iterative process of training. Ant function can prevent the gradients from vanishing and

Table 10. Testing loss and average convergence time of different activation functions on curve fitting

	Ant	ReLU	ELU	Leaky-ReLU	Mish	Siren
Loss	0.001684	0.007156	0.082822	0.006850	0.055355	0.003517
Time (s)	11.4886	15.1071	19.9651	16.8931	42.9088	17.4966

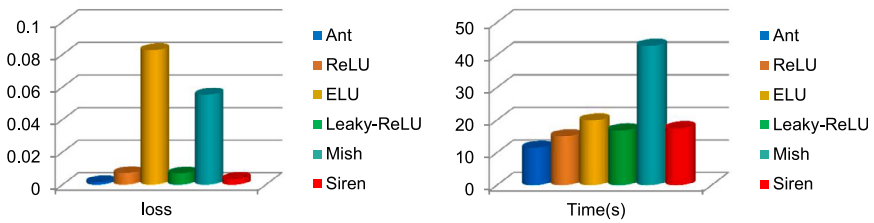


Figure 14. Comparison results of testing loss, and average convergence time for different activation functions on curve fitting

has good gradient-preserving properties. The experiments on increasing the depth of network on convolutional neural network (CNN) and deep neural network (DNN) architectures show this property.

(3) The third advantage is attenuation. As mentioned above, because of the attenuation property, the vast majority of inputs of Ant function would fall inside the domain of $-6 \leq x \leq 6$ and meanwhile, the vast majority of outputs would fall inside the domain of $-1/e \leq y \leq 1/e$. The attenuation property determines the data distribution in the networks, reduces the internal covariate shift, and accelerates network training, which is the normalization effect caused by attenuation. Because of this property, networks with Ant function can perform well with batch normalization, which can be seen in all experiments.

(4) The fourth advantage is zero centering. We can learn from Figure 5 that Ant is a zero-centered activation function. It has been long known that neural networks can learn faster if activation functions in hidden layers are centered around zero.[34,35] Cun et al.[34] presented strict proof of the zero-centered property of effective activation functions. Neural networks with Ant have faster convergence speed than neural networks with other activation functions, which can be seen in the results of the XOR experiment.

(5) The fifth advantage is smoothness and continuity. Ant is a smooth and continuous function. Having a smooth and continuous profile helps make gradients flow and makes it easier to optimize. Meanwhile, Ant is capable of representing higher-order derivatives and fine details. The experiments on image fitting, curve fitting, solving PDEs, and natural language translation show this property.

(6) The sixth advantage is non-saturation and absence of dead neurons. At first sight, it appears that saturation occurs at both ends of the Ant function, just as in Sigmoid, but in reality this saturation rapidly disappears as the network is trained and iterated. The Sigmoid function is monotonically increasing at both ends, and since there is no attenuation operation, the Sigmoid will continue to grow at both ends and will saturate so badly that the gradient tends to zero. However, Ant is different, as discussed in the second property about gradient preservation; when $|x|$ grows, due to the attenuation operation, in the next iteration the output of Ant will tend to be near zero, and thus the saturation will disappear. After attenuation, the output range is $-1/e \leq y \leq 1/e$; within this range there are few zero values and the gradient is close to 1, and almost no dead neurons exist.

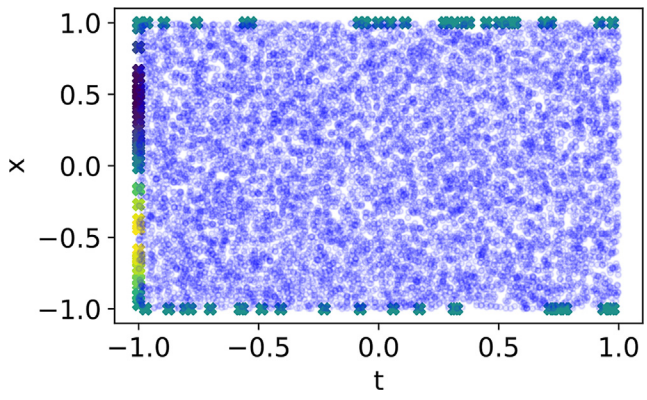


Figure 15. Positions of collocation points and boundary data of Burgers' equation

Comparison of different neuron models

The research on neuron models has a long history, traced back to the end of the 19th century. With the cross-fertilization of computer science, mathematics, physics, and other multi-disciplinary disciplines in the later period, the research on neuron models has continued to develop. It has not only made significant breakthroughs at the theoretical level but has also had a far-reaching impact on practical applications. Neuron models are mainly classified into two types. The first is the non-conductance model, which centers on using abstract mathematical language to depict the input and output mechanisms of neurons without considering the detailed physiological structure of neurons. The second type is biophysical-level models, or conductance models, which aim to simulate neuronal behavior by reproducing the precise electrophysiological properties of neuronal cell membranes and constructing their conductance channels.

We selected 14 representative neuron models, including CTRNN,[36] to compare with the Ant model, and the comparison mainly includes five aspects: model dimension, model parameters, modeling type, biological plausibility, and computational simplicity. The comparison results are shown in Table 2, from which it can be seen that Ant belongs to the non-conductance model, which mainly adopts the generalized linear system to describe the neuron's input and output mechanisms and does not take into account the specific physiological structure of neurons. Ant can simulate fewer neuronal behaviors, so the behavioral diversity of Ant is low, but since Ant is computationally modeled from a variety of input-output mechanisms, it has a

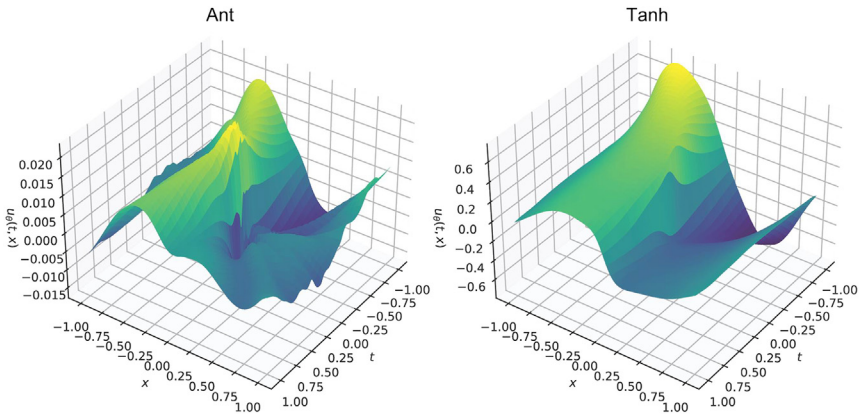


Figure 16. Solutions of Burgers' equation of Ant and Tanh activation functions

high input-output behavioral universality. Compared with other neuron models, Ant has the following two characteristics.

(1) Ant has low behavioral diversity and high behavioral universality compared with other neural models, which can simulate various input-output mechanisms of biological neurons.
(2) Ant has only one dimension and two parameters, the most simplified mathematical description among the currently known neuron models, and this simple computational model has good potential for application.

RESULTS AND DISCUSSION

Experimentally, to test the effectiveness and generalization ability of Ant, we evaluate it on different network architectures and deep-learning tasks. Network architectures include CNN, DNN, graph neural network (GNN), and Transformer. Deep-learning tasks include image classification, two-spirals task, image fitting, curve fitting, solving PDEs, solving the XOR problem, textual classification, and natural language processing. We also conduct experiments by increasing the number of neural network layers to evaluate the stability of Ant. All input data in these experiments are approximately normalized to the range of [−1,1] through transformations of the signal range scale parameters. All the experimental datasets and their detailed information can be found in Table S1.

We compare Ant against several baseline activation functions in these experiments by replacing the activation functions. We select the basic Adam or stochastic gradient descent (SGD) optimizer and avoid using a range of complex optimization strategies, aiming to minimize the interference of a "bag of tricks" and fairly demonstrate the different learning performances of each activation function. For training, we follow the common practice and train all models on the Tesla P100 GPU and report the standard top-1 accuracy, top-1 loss, and other evaluation metrics. To ensure a fair comparison, we use the average result from multiple experiments as the final outcome, and all results are calculated based on multiple independent runs. The mean validation accuracy, mean validation loss, and other mean metrics are given in the corresponding experimental results. All hyperparameters are set according to the actual situation of different application tasks, either general parameters or default parameters, because the parameters set in this way are more general and universal. All parameters are shown in Table 3. These default parameters are also the best hyperparameters for traditional activation functions, which can prove the effectiveness and universality of Ant under ordinary parameter settings. Our results show that Ant exceeds other functions on all tasks, which demonstrates the effectiveness, stability, and good generalization ability of Ant to match these varying architectures and challenging domains.

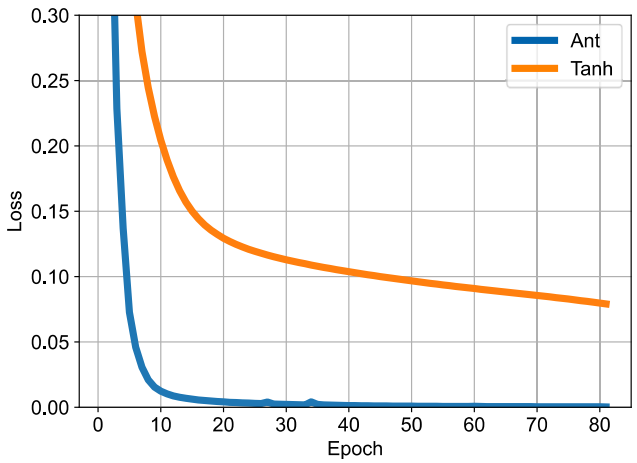


Figure 17. Loss curves of Ant and Tanh activation function on solving Burgers' equation

Generalization performance on CNN

We first conducted several comparison experiments to evaluate the proposed Ant on CNN architectures, including models VGG7[50] and ResNets[51] with different depths (e.g., ResNet18,

Table 11. Testing loss and average convergence time of Ant and Tanh activation functions on solving Burgers' equation

	Ant	Tanh
Loss	1.5248e−04	4.1813e−02
Time (s)	117.7511	412.3585

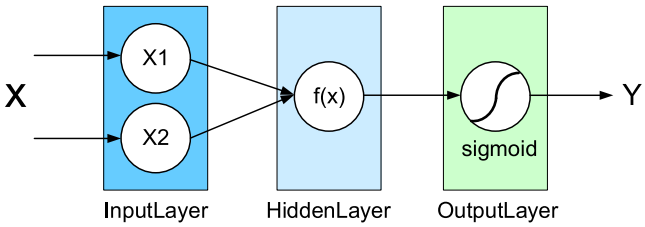


Figure 18. A simple neural network structure designed for solving the XOR problem with the input x1 and x2

ResNet34, ResNet50, ResNet101, ResNet152). We choose standard CIFAR-10,[52] CIFAR-100,[51] Fashion-MNIST,[53] ImageNette,[54] and the challenging ImageNet2012[55] datasets for all CNN experiments. There are numerous factors that could affect CNN's performance. To highlight the role of the activation function, we eliminated the bag of tricks, effectively reducing the interference of tricks on network performance without chasing state-of-the-art results. To minimize the impact of image augmentation on the results, no image augmentation was applied to the small datasets, while only basic image augmentation was used for ImageNet. As the layers increase in the ImageNet experiment, the degree of overfitting often intensifies due to the growing number of parameters. To mitigate the effects of overfitting, employing basic image augmentation is actually beneficial. This includes adjustments to the aspect ratio, scaling, cropping, flipping, color jittering, and the addition of noise. In training, we use a uniformly distributed random method to select channel locations and establish appropriate positive and negative feature ratios for activation, ensuring both comprehensiveness and sparsity of the features. Meanwhile, we choose ReLU, ELU, Leaky-ReLU, GELU, ACON-C,[56] Mish, and Swish-beta as baseline activation functions to compare against.

The experimental results are shown in Tables 4, 5, and 6, and the comparison results in Figures 7 and 8 are especially more obvious. We can see from the results that Ant consistently outperforms other activation functions by unique performance with the highest validation accuracy and the lowest validation loss simultaneously achieved on all models and datasets with the same FLOPs and params. The results also show the stability of Ant to match increasing the number of layers on CNN architectures. As seen in Figure 9, the number of positive features activated by Ant is significantly higher than that of ReLU, and the positive features activated by Ant are all almost in the cat's foreground. In contrast, the positive features activated by ReLU are concentrated in the background and local foreground. Additionally, on visual perception, each layer of the feature map activated by Ant can see the outline of the cat while, on the contrary, ReLU finds it difficult to see the outline of the cat in the third- and fifth-layer networks. The experiments prove that Ant has better stable performance in CNN architectures for image classification due to its strong stability in the flow transmission of positive features in deep networks.

Generalization performance on DNN

Additionally, we evaluate the generalization performance of the proposed Ant on DNN architectures. Ant activation function can easily be extended to other challenging deep-learning tasks, and we show its generalization performance by experiments on two spirals, image fitting, curve fitting, PDEs, and the XOR problem. Meanwhile, we test the stability of Ant to match increasing the number of neural network layers on DNN architectures.

Solving the two-spirals task

The two-spirals task is a benchmark task for non-linear classification.[57,58] The dataset consists of two spirals, each with 2,000 sample data points in a 2D Cartesian space (Figure 10). The objective is to classify sample points close to each of the spirals by using only the (x,y)-Cartesian coordinates. In this section, we evaluate Ant and several baseline activation functions in a two-spirals task on a simple feedforward neural network consisting of an input layer (2 neurons), two fully connected hidden layers (4 neurons in the first hidden layer, 3 neurons in the second hidden layer), and an output layer (2 neurons). The results are shown in Table 7. We can learn from the results that Ant consistently surpasses other activation functions with the highest mean validation accuracy and the lowest mean validation loss.

Solving the image-fitting problem

Ant is capable of represent fine details. In this section, we demonstrate that Ant can represent fine details on image fitting. We choose ReLU and Siren as baseline functions to compare against four natural images including cat, dog, Helmholtz_imag, and Helmholtz_real. We choose a feedforward neural network consisting of an input layer (2 neurons), two fully connected hidden layers (256 neurons in each hidden layer), and an output layer (3 neurons) as experimental network architecture.

Standard evaluation metrics including mean average precision (mAP), AP_{50}, AP_{75}, and mean of the peak signal-to-noise ratio

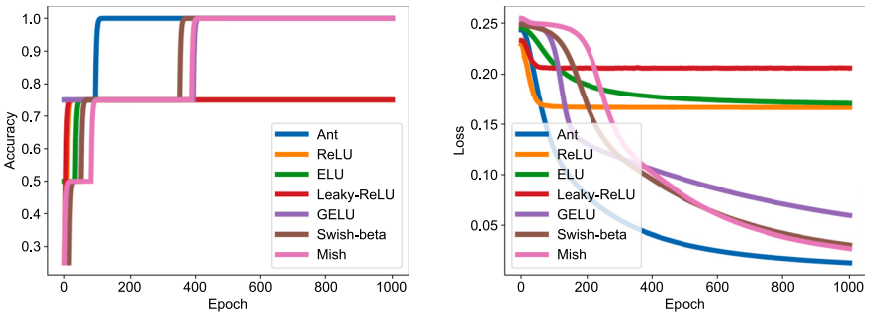


Figure 19. Accuracy and loss curves of different activation functions on the XOR problem

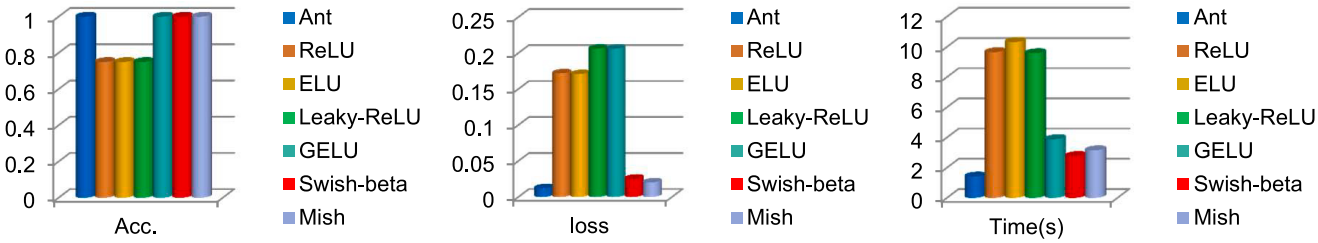


Figure 20. Comparison results of testing accuracy, loss, and average convergence time for different activation functions

(PSNR) are reported in Tables 8 and 9, respectively. We also plot loss curves of fitting images in Figure 12. As seen in Figure 12, the training loss of Ant converges in very stable manner, while the training loss of ReLU fluctuates greatly. Ant representation enables accurate representations of natural image signals due to its smooth, continuous, and non-monotonic properties. The results demonstrate that Ant performance is better than that of ReLU and Siren with both the highest evaluation metrics and mean PSNR and the lowest mean loss. Meanwhile, we offer some examples of fitting images in Figure 11. As seen in Figure 11, Ant accurately reproduces intricate details such as whiskers and hairs better than ReLU and Siren.

Solving curve fitting

Curve fitting is a process of generating a curve that best represents the characteristics of a system using the input dataset. It is the basis of any analytical, comparative, or growth-related statistics.[59] In this section, we benchmark Ant against ReLU, ELU, Leaky-ReLU, Mish, and Siren activation functions on curve fitting. The neural network used in the experiment consists of an input layer (1 neuron), a hidden layer (1,000 neurons), and an output layer (1 neuron).

To illustrate the capability of curve fitting, we use a complex function curve with varying amplitude, which is defined as $y = 1.8 * \sin(3 * x)/x$. Curves fitted by different activation function networks are shown in Figure 13. We can see from Figure 13 that the average error for all the curves is considerably low, while the curve plotted by Ant is smoother and closer to the original curve than other functions. The loss of function curve fitting is reported in Table 10 and Figure 14, in which we can see that Ant consistently has the lowest mean test loss compared with other activation functions.

Solving the partial differential equation

As mentioned above, Ant can prevent the gradients from vanishing. The derivative of Ant is well behaved and can represent fine details. In this section, with a well-behaved derivative, we demonstrate how Ant can be employed to tackle complex boundary value issues for time-series data, such as PDEs.

To demonstrate the performance of Ant for problems pertaining to the solution of PDEs, we will use the Burgers' equation as a canonical example, adopted from Raissi et al.[60] In a single spatial dimension, the Burger's equation coupled with Dirichlet boundary conditions takes the form of

$$\begin{aligned} &\mu_t + \mu\mu_x - (0.01/\pi)\mu_{xx} = 0, \\ &x \in [-1,1], t \in [-1,1], \\ &\mu(-1,x) = -\sin(\pi x), \\ &\mu(t,-1) = \mu(t,1) = 0. \end{aligned} \qquad \text{(Equation 46)}$$

This PDE arises in various disciplines such as traffic flow, fluid mechanics, and gas dynamics, and can be derived from the Navier-Stokes equations.[61]

We construct a neural network approximation of the solution of non-linear PDEs and proceed by approximating $\mu(t,x)$ with a DNN $\mu_\theta(t,x)$, where μ_θ denotes a function realized by a neural network with parameters θ.

$$\mu_\theta(t,x) \approx \mu(t,x). \qquad \text{(Equation 47)}$$

We assume that the training data *Nr* as well as the initial time and boundary data *Nt* and *Nb* are generated by random sampling from a uniform distribution of x and t ($x \in [-1, 1]$, $t \in [-1, 1]$). Here, we choose training data of size $Nt = Nb = 50$ and $Nr = 10{,}000$, respectively. As shown in Figure 15, we illustrate the training data (blue circles) and the positions where the boundary and initial conditions will be enforced (cross marks; color indicates value).

In this experiment, we assume a feedforward neural network followed by eight fully connected hidden layers each containing 20 neurons and each followed by an activation function. We train the model for 5,000 epochs (which takes approximately 2 min) with Adam optimizer, and the learning rate is fixed to 1e−4.

The solutions to Burgers' PDEs plotted by Ant and Tanh are shown in Figure 16. We can see from the results that Ant and Tanh are capable of representing higher-order derivatives, while ANT outperforms Tanh in capturing fine details. The solution surface plotted by Ant is smoother and makes the least error compared to the solution surface plotted by Tanh. The mean validation loss of Ant and Tanh is reported in Figure 17 and Table 11, where we can see that Ant has a lower mean validation loss than Tanh.

Table 12. Testing accuracy, loss, and average convergence time of different activation functions on the XOR problem

	Ant	ReLU	ELU	Leaky-ReLU	GELU	Swish-beta	Mish
Accuracy	1.00	0.75	0.75	0.75	1.00	1.00	1.00
Loss	0.0123	0.1720	0.1709	0.2061	0.2061	0.0255	0.02030
Time (s)	1.4124	9.6506	10.3233	9.5815	3.8929	2.7708	3.1512

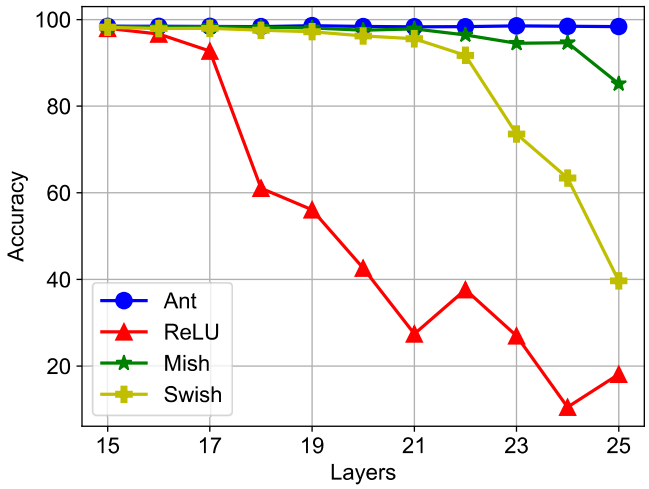


Figure 21. Testing accuracy of different activation functions with increasing depth of the network on the MNIST dataset

Solving the XOR problem

The XOR classification task represents a general learning cognitive challenge. This type of pattern classification is well studied in comparative experimental psychology and is considered as a common categorization benchmark in machine learning or artificial neural networks.[62] The XOR problem cannot be solved by a simple perceptron[63]; we need a neural network with at least one hidden layer to solve the problem. Here, we define a simple neural network, shown in Figure 18, to solve the XOR problem.

As shown in Figure 18, the neural network consists of an input layer (2 neurons), a hidden layer (1 neuron), and an output layer (1 neuron). The structure of the neural network is very simple so that we can test the limiting performances of activation functions.

For this task, we choose ReLU, Leaky-ReLU, GELU, Swish-beta, and Mish activation functions to compare against. The results are shown in Figures 19 and 20, and Table 12. The results show the strong performance of Ant. ReLU and Leaky-ReLU have low accuracy in the experiments. GELU, Swish-beta, and Mish have the same accuracy as Ant, but Ant has a faster convergence speed and lower loss.

Increasing the number of network layers

The depth of the network has significant effects on the performance of different activation functions. In this section, we show the stability of Ant when increasing the depth of the network on DNN architecture. In the experiment, our experimental setup and dataset are the same as for Mish, except that we do not use the batch normalization layers.

As shown in Figure 21, both Swish and ReLU exhibit a significant drop in accuracy beyond the 15th layer. Meanwhile, Swish markedly decreases in accuracy after 21 layers, whereas Ant's performance is stable without a decrease in accuracy during the experiment because of its stable data distribution and stable gradient distribution, which are shown in Figures 22 and 23, respectively. As seen in Figures 22 and 23, the Ant network has a uniform distribution of data in different layers during the training process, and the gradient distribution in different layers is also very stable, with almost no gradient explosion and gradient vanishing. Under the same training conditions, the amount of data in the ReLU network reduced significantly in different layers during the training process, and the gradient distribution is not stable enough, shown by obvious large gradient fluctuation. The experiment results prove the stability of Ant in increasingly large networks with both stable data distribution and gradient distribution.

Generalization performance on GNN

We explore the generalization ability of Ant on GNN architecture such as the graph convolutional network (GCN).[64,65] GCN has

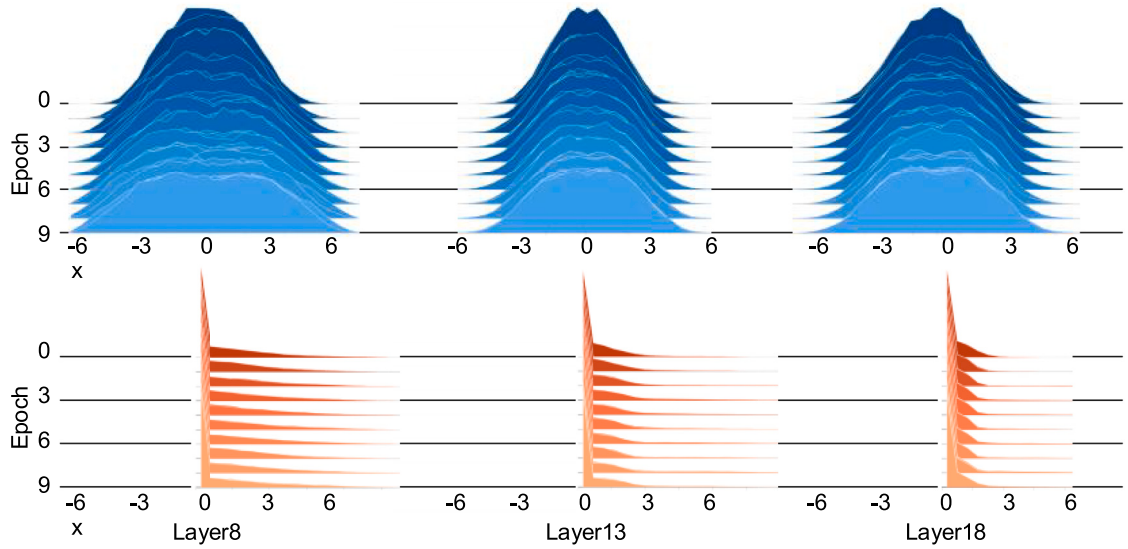


Figure 22. Data distribution of Ant and ReLU in different layers at different training times (blue is Ant, red is ReLU)

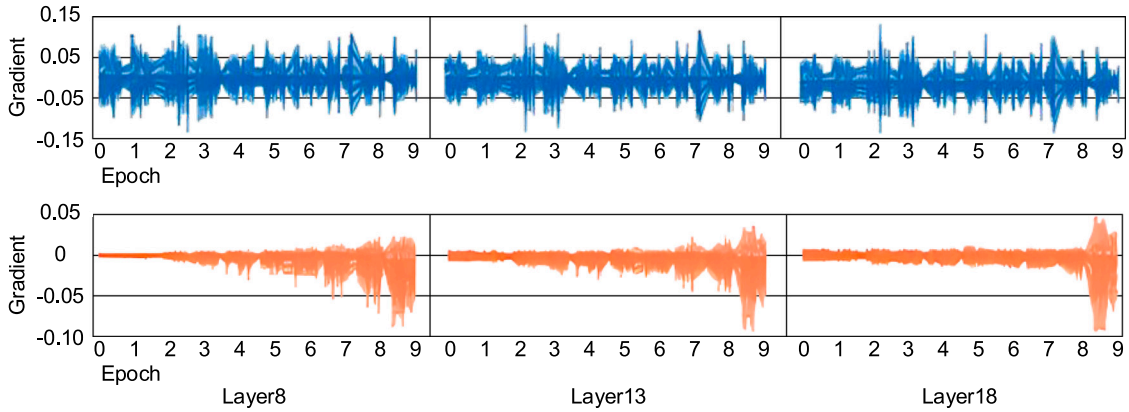


Figure 23. Gradient distribution of Ant and ReLU in different layers at different training times (blue is Ant, red is ReLU)

become a standard and powerful deep-learning approach for text-classification problems on graph-structured data. Recently, GCN has shown strong performance in various application areas on real-world datasets.[66,67]

In the experiment, we use a 2-layer (each with 32 neurons) GCN with a 64-dimensional hidden variable for comparing different activation functions. We choose a standard citation network dataset, Cora, for semi-supervised text classification on GCN architecture. The results are shown in Table 13 and Figure 24. We can see from the results that Ant outperforms Tanh and other functions in improving the average validation accuracy by 4.5% and decreasing the average validation loss by 21%. This shows the strong performance of Ant on GCN architectures.

Generalization performance on Transformer

Language translation on Transformer

Transformer[68] is a self-attention-based neural network architecture, which mainly processes sequence data through encoding and decoding structure and attention modules and has a wide range of applications in multiple fields of deep learning.[69] In this paper, we select a natural language translation task to test the activation effect of Ant in the Transformer model consisting of three encoders, three decoders, eight encoding multi-head attention modules, and eight decoding multi-head attention modules. Since the Transformer architecture consists of multiple block units, each block contains multi-head attention and a feedforward network, and the feedforward network is a non-linear feature space containing a large number of parameters that is responsible for memorizing patterns and relationships. For this reason, we test the performance of Ant by just replacing the activation functions (including ReLU, GELU, Swish, and Mish) in the feedforward network layers. The dataset chosen for the experiment is Multi30K, which is an extension of the Flickr30K dataset with 31,014 English descriptions and 155,070 German descriptions, respectively; the detailed information of the dataset is shown in Table S1. The experimental parameters are set up by choosing the common parameter values of peers, and the specific parameters are shown in Table 3. The evaluation metrics are PPL, Bleu, and Meteor. The experimental results are shown in Table 14 and Figure 25, Ant outperforms the other four activation functions in the Transformer model. In terms of model complexity, Ant has the smallest PPL score compared to other activation functions, and Ant's translation results on the test set are closest to standard results. On German-English translation, Ant's Bleu metric is 0.68 higher compared to the other activation functions, indicating that Ant has the highest translation accuracy. In the English-German bilingual translation task, Ant's Meteor metric ranks first with 74.912, which proves that Ant's translation semantics are more accurate than those of other activation functions. The comparison results, which can be seen clearly in Figure 25, show that Ant is a strong candidate activation function in the Transformer architecture.

Table 13. Testing accuracy, loss, and average convergence time of different activation functions on the Cora dataset

	Dataset	Ant	Tanh	Selu	Softmax	Sigmoid
Accuracy	Cora	0.8679	0.8050	0.8408	0.8366	0.7913
Loss		0.5358	0.8156	0.6293	0.6613	0.8937
Time (s)		4.0900	4.1860	5.6295	5.4192	5.1701

Text classification on BERT

BERT[70] is an improved Transformer-based model with a wide range of applications in natural language processing.[71] To test the performance of Ant on the BERT model, we use the BERT model to perform fine-tuning experiments on the GoEmotions dataset, a large sentiment-labeled dataset for sentiment analysis that contains rich textual data for training and evaluating the performance of sentiment analysis models. The experimental results are shown in Table 15, demonstrating the effect of different activation functions on the model performance.

It is observed that ReLU performs well on this task with an accuracy of 0.7556. The performance of ReLU activation function is in the middle to upper range of all activation functions, which indicates that it provides a better classification performance when dealing with the sentiment analysis task on

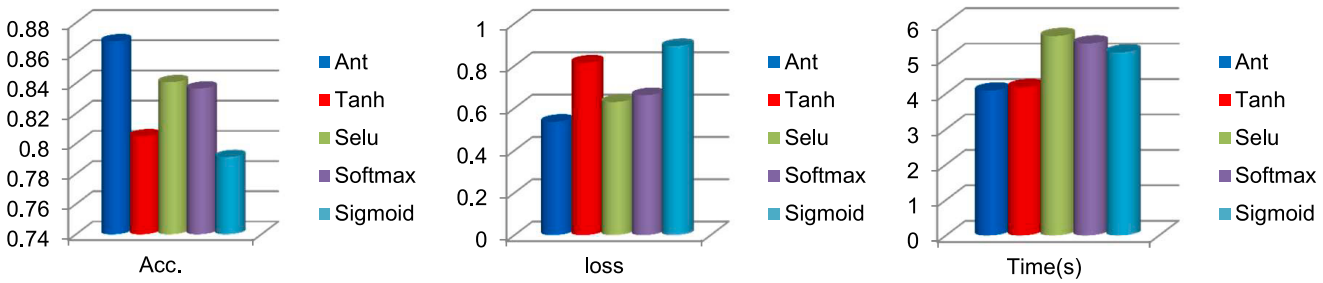


Figure 24. Comparison results of testing accuracy, loss, and average convergence time for different activation functions on Cora

GoEmotions, especially on the micro-F1 score, which usually means that the model has a more balanced overall performance when dealing with samples in the dataset. GELU slightly underperforms ReLU on the GoEmotions dataset, with an accuracy of 0.7418. GELU function attempts to improve the performance of ReLU by introducing the properties of a normal distribution but does not show a significant advantage on the GoEmotions dataset, and, in particular, underperforms ReLU on the macro-F1 score.

The performance of Swish on this task is also noteworthy. Swish attempts to optimize the performance of ReLU by introducing a weighting factor, especially when dealing with non-linear relationships. On GoEmotions, Swish function performs well on the micro-F1 score, which suggests that it provides better classification performance, especially when distinguishing between different emotion categories.

Mish has suboptimal performance on GoEmotions. Mish further optimizes the performance of Swish function by introducing an additional linear correction term. Mish outperforms Swish in test accuracy, and the macro-F1 score is slightly higher than that of Swish, which indicates that the Mish function's overall performance is slightly higher than that of Swish.

Finally, Ant performs best on the GoEmotions dataset with an accuracy of 0.7595, which is 0.925% higher than the average of the other four activation functions, as shown in Figure 26, Ant is optimal in all three evaluation metrics. Ant introduces the attenuation and gradient-preservation properties, which will further optimize the BERT model's learning ability. Among all the activation functions, Ant performs well on the micro-F1 score, which indicates that it can provide optimal classification performance.

Table 14. Natural language translation results based on Transformer architecture

Activation function	Multi30k German-English			Multi30k English-German		
	PPL	Bleu	Meteor	PPL	Bleu	Meteor
ReLU	5.592	37.199	75.108	4.420	34.345	74.503
GELU	6.740	37.069	74.799	4.422	35.179	74.646
Swish	5.531	37.760	74.971	4.384	35.356	74.832
Mish	5.591	37.485	74.931	4.398	35.436	74.699
Ant	5.508	38.003	75.127	4.378	35.531	74.912

Emotion recognition on Vision Transformer

Vision Transformer[72,73] is also an improved Transformer-based model, which has performed excellently in the field of computer vision. To test the performance of Ant on Vision Transformer, we use Vision Transformer to perform fine-tuning experiments on the FER2013 dataset, which is used for emotion recognition and contains rich images of human face expressions for training and evaluating the performance of the emotion recognition model. A 20% slice of the dataset was used to accommodate the GPU memory. The experimental results are shown in Table 16, demonstrating the effect of different activation functions on the model's performance.

We obtained a set of experimental data after trying different activation functions in Vision Transformer and testing it on the FER2013 dataset. These data provide us with information about the impact of different activation functions on model performance, mainly in two key metrics: test accuracy and test loss.

We can see that Ant has the most outstanding performance, with an accuracy of 0.9895, significantly higher than the accuracy of other activation functions. This result indicates that Ant can extract the facial expression features in FER2013 more effectively, which significantly improves the emotion recognition accuracy. This is closely followed by Mish with an accuracy of 0.9782, which also shows a high level of performance. The accuracy of ReLU activation function is 0.9765, which is comparable to that of Mish but slightly lower than the first two. The accuracy of Swish and GELU activation functions is 0.9747 and 0.9712, respectively, which is relatively low, suggesting that the two activation functions perform well in Vision Transformer but do not perform as well as Ant and Mish.

From the perspective of test loss, Ant also performs the best, with a loss rate of only 0.0692, meaning that the model has the least uncertainty in prediction and the best fit. Mish has a loss rate of 0.0710, which is slightly higher than that of Ant but still exhibits lower loss. ReLU has a loss rate of 0.0751, which is not much different from that of Mish. The loss rates of Swish and GELU are 0.0963 and 0.1025 respectively, which are relatively high, indicating that the model does not fit the dataset as well as Ant and Mish under these two activation functions.

From the three indicators of test accuracy, test loss, and the confusion matrix shown in Figure 27, we can conclude that Ant outperforms other activation functions in

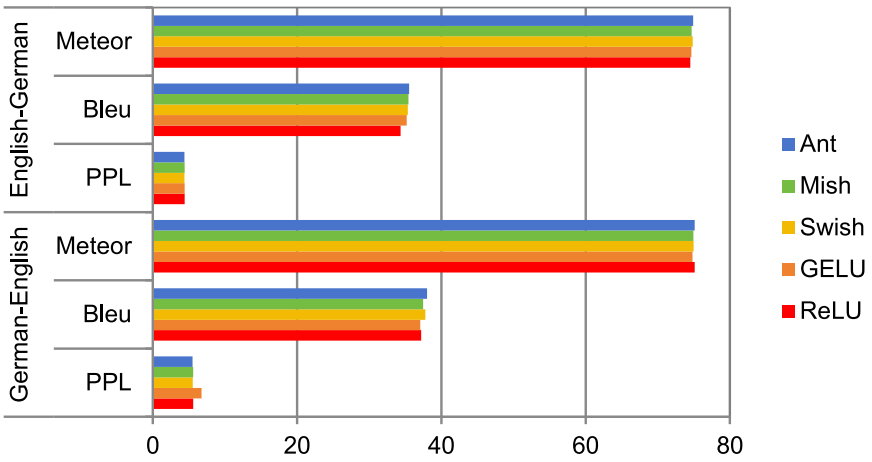


Figure 25. Comparison results of testing PPL, Bleu, and Meteor for different activation functions on Multi30k

distinguishing the two highly similar expressions, disgust and anger, and Ant has the most superior application in Vision Transformer. It not only significantly improves the accuracy but also reduces the loss rate because Ant has zero-centering and gradient-preserving properties, which give it better discriminative ability when dealing with similar expressions.

Named entity recognition on LLaMA

LLaMA[74–76] is a large language model, based on the Transformer architecture, which has shown excellent performance in the field of natural language processing. To test the performance of Ant on LLaMA, we use LLaMA to perform fine-tuning experiments by substituting different activation functions on the CoNLL 2003 dataset, which consists of Reuters news data over 10 years and contains rich textual data of named entities. A 50% slice of the dataset was used to accommodate the GPU memory. The experimental results are shown in Table 17, which demonstrates the effect of different activation functions on the performance of LLaMA.

From the perspective of test accuracy, Ant has the most outstanding performance with an accuracy of 0.9676, significantly higher than that of other activation functions. This indicates that Ant is more effective in improving the performance of the model when dealing with CoNLL2003, especially in terms of accuracy for the named entity recognition tasks. This is closely followed by GELU with an accuracy of 0.9637, which also performs quite well. ReLU has an accuracy of 0.9621, similar but slightly lower than that of GELU. Swish and Mish have relatively low accuracy of 0.9615 and 0.9602, suggesting that both activation functions perform less well than Ant, GELU, and ReLU.

Table 15. Multi-label text-classification results based on BERT

Activation function	Accuracy	F1 score (micro)	F1 score (macro)
ReLU	0.7556	0.8243	0.7553
GELU	0.7418	0.8160	0.7421
Swish	0.7463	0.8201	0.7474
Mish	0.7578	0.8232	0.7502
Ant	0.7595	0.8270	0.7562

From the view of test loss, Ant also performs the best with a loss rate of 0.1061, which indicates that the model has less uncertainty in prediction and fits better. GELU has a loss rate of 0.1155, slightly higher than that of Ant. ReLU has a loss rate of 0.1147, similar to that of GELU. Swish and Mish have relatively high loss rates of 0.1158 and 0.1256, respectively, which may indicate that these two activation functions do not fit the model as well as Ant and GELU when dealing with the CoNLL2003 dataset.

Among other measures of model performance, such as precision, recall, and F1 score, Ant also performs well. The F1 score is the reconciled average of precision and recall, thus Ant has the highest F1 score of 0.7758, which indicates that not only is it accurate but it also handles the entity recognition task, and it also contributes significantly to the recall improvement. The F1 score of GELU activation function is 0.7435, which is slightly lower than that of Ant but still better than other activation function scores. The F1 score of ReLU activation function is 0.7470, and the F1 scores of Swish and Mish are 0.7290 and 0.7410, respectively, which are relatively low.

Combining the test accuracy, test loss, and other evaluation metrics, we can conclude that Ant has a superior effect on LLaMA. As can be seen from the comparison in Figure 28, it not only significantly improves the accuracy, precision, recall, and F1 score of the model but also maintains a low loss rate, due to the zero-centeredness and attenuation properties of Ant, which gives it a better detailed semantic feature extraction ability and semantic recognition ability when dealing with CoNLL2003.

Conclusions and limitations

In this work, we propose an attenuation-activation mechanism, called Ant, by introducing generalized linear system theory for computational modeling of neurons. The proposed mechanism facilitates the precise representation of intricate signals in the deep-learning framework. Unlike the ReLU family, Ant is a zero-centered, non-monotonic, smooth, continuous, bounded above, and bounded below activation function. Because of these features, Ant can prevent gradients from vanishing,

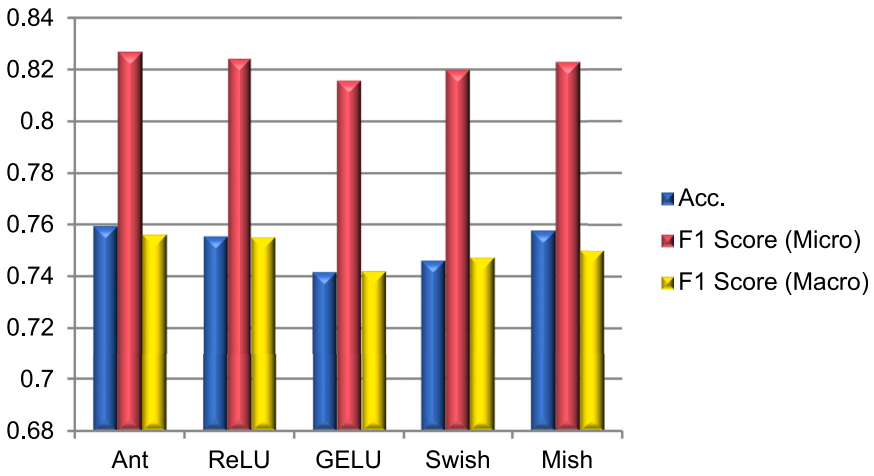


Figure 26. Comparison results of testing accuracy, micro-F1, and macro-F1 for different activation functions on GoEmotions

represent higher-order derivatives, and change data distributions in the networks. Neural networks with Ant have faster convergence speed, stronger representation capability, and simpler structures than neural networks with other activation functions.

We ran several experiments on CNN, DNN, GNN, and Transformer architectures to test Ant's stability and generalization ability for various challenging learning tasks. Experimental results show that Ant has stronger stability and higher generalization ability than other activation functions. Ant performs well in all tasks in general, proving its effectiveness. Furthermore, Ant is very unlike the ReLU family, and we expect that the Ant activation function proposed in this paper brings significant progress to applying deep learning in multiple fields of science and engineering.

Ant has the property of attenuation over the full input range, and this attenuation property motivates artificial neural networks with data-distribution stability and gradient-preserving characteristics. Ant displayed good performance in DNN among the applications of different neural network architectures. Nonetheless, Ant also has limitations. On one hand, it uses generalized linear systems and differential equations to describe the input and output mechanisms of neurons and does not consider the specific physiological structure of neurons; therefore, Ant has low neuron behavioral diversity. On the other hand, due to the application requirements, we fixed the time domain as a constant during the theoretical derivation, resulting in the loss of the cumulative effect of neuron signal attenuation over time. In future research, we plan to study the cumulative effect of signal attenuation in the time domain.

Table 16. Emotion recognition results based on Vision Transformer

Activation function	Accuracy	Loss
ReLU	0.9765	0.0751
GELU	0.9712	0.1025
Swish	0.9747	0.0963
Mish	0.9782	0.0710
Ant	0.9895	0.0692

RESOURCE AVAILABILITY

Lead contact

The lead contact for this study is Heng Yuan (lntuyuanheng@163.com).

Materials availability

This study did not generate new unique materials.

Data and code availability

The source codes and datasets for Ant are publicly available online in a Zenodo repository.[77]

ACKNOWLEDGMENTS

This work was supported by the Research Funds for the Top Youth Talent Program (grants 551901027047, 552001008036, 552101001053, and 552210300086), the Liaoning Natural Science Foundation (grant 20170540426), and the Science and Technology Project Fund of Provincial Department of Education (grant LJYL049).

AUTHOR CONTRIBUTIONS

W.J. and H.Y. initiated and supervised the research. W.J. deduced the attenuation-activation function. H.Y. conceptualized and created the models. W.J. performed the experiments and analysis. H.Y. wrote the original draft with substantial input from the other authors. W.L. contributed to the discussion of the results. All authors participated in reviewing and editing the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.patter.2024.101117.

Received: May 3, 2024
Revised: July 16, 2024
Accepted: November 20, 2024
Published: December 16, 2024

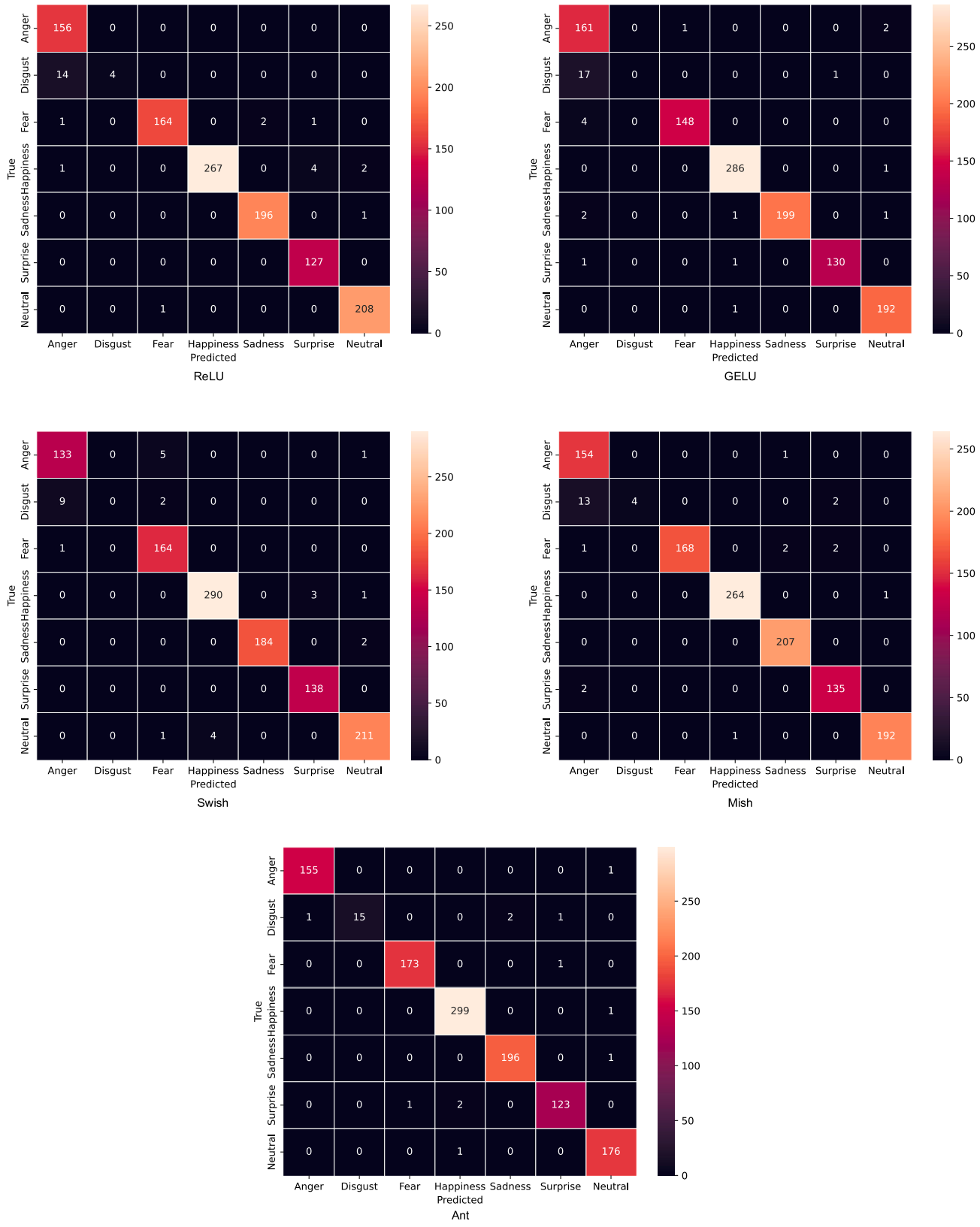


Figure 27. Confusion matrix for different activation functions on FER2013

Table 17. Named entity recognition results on LLaMA

Activation function	Accuracy	Loss	Precision	Recall	F1 score
ReLU	0.9621	0.1147	0.7182	0.7781	0.7470
GELU	0.9637	0.1155	0.7182	0.7707	0.7435
Swish	0.9615	0.1158	0.7113	0.7732	0.7410
Mish	0.9602	0.1256	0.7111	0.7478	0.7290
Ant	0.9676	0.1061	0.7526	0.8006	0.7758

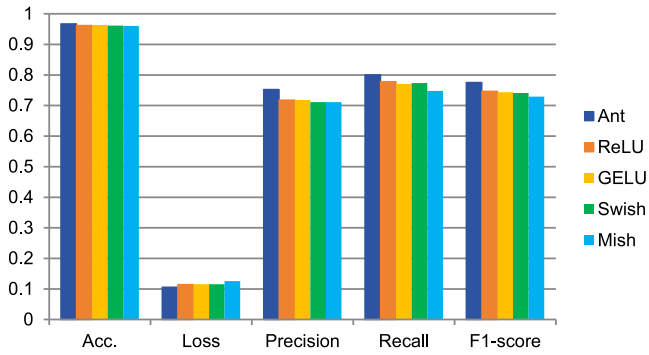


Figure 28. Comparison results for different activation functions on CoNLL2003

REFERENCES

1. Sun, Z. (2010). Stability of piecewise linear systems revisited. Annu. Rev. Control *34*, 221–231. https://doi.org/10.1016/j.arcontrol.2010.08.003.
2. Basin, M., and Maldonado, J.J. (2014). Optimal controller for uncertain stochastic linear systems with poisson noises. IEEE Trans. Industr. Inform. *10*, 267–275. https://doi.org/10.1109/TII.2013.2248160.
3. Tsuburaya, T., Okamoto, Y., Fujiwara, K., and Sato, S. (2014). Performance of preconditioned linear solvers based on minimum residual for complex symmetric linear systems. IEEE Trans. Magn. *50*, 557–560. https://doi.org/10.1109/TMAG.2013.2281410.
4. Zhou, B., and Li, Z.Y. (2015). Truncated Predictor Feedback for Periodic Linear Systems With Input Delays With Applications to the Elliptical Spacecraft Rendezvous. IEEE Trans. Control Syst. Technol. *23*, 2238–2250. https://doi.org/10.1109/TCST.2015.2411228.
5. Xu, F., Li, Z., Nie, Z., Shao, H., and Guo, D. (2019). New Recurrent Neural Network for Online Solution of Time-Dependent Underdetermined Linear System with Bound Constraint. IEEE Trans. Industr. Inform. *15*, 2167–2176. https://doi.org/10.1109/TII.2018.2865515.
6. Rego, F.F., Pascoal, A.M., Aguiar, A.P., and Jones, C.N. (2019). Distributed state estimation for discrete-time linear time invariant systems: A survey. Annu. Rev. Control *48*, 36–56. https://doi.org/10.1016/j.arcontrol.2019.08.003.
7. Vargas, A.N., Agulhari, C.M., Oliveira, R.C.L.F., and Preciado, V.M. (2022). Robust Stability Analysis of Linear Parameter-Varying Systems With Markov Jumps. IEEE Trans. Automat. Contr. *67*, 6234–6239. https://doi.org/10.1109/TAC.2021.3132231.
8. Yang, N., Li, Y., and Shi, L. (2022). Proportional Tracking Control of Positive Linear Systems. IEEE Control Syst. Lett. *6*, 1670–1675. https://doi.org/10.1109/LCSYS.2021.3130638.
9. Batmani, Y. (2023). A Decentralized Event-Triggered State-Feedback Control Technique for Continuous-Time Linear Systems. IEEE Trans. Syst. Man Cybern. Syst. *53*, 2828–2836. https://doi.org/10.1109/TSMC.2022.3219872.
10. Mondié, S., Egorov, A., and Gomez, M.A. (2022). Lyapunov stability tests for linear time-delay systems. Annu. Rev. Control *54*, 68–80. https://doi.org/10.1016/j.arcontrol.2022.09.001.
11. Duan, G.R., and Zhou, B. (2022). Fully Actuated System Approach for Linear Systems Control: A Frequency-Domain Solution. J. Syst. Sci. Complex. *35*, 2046–2061. https://doi.org/10.1007/s11424-022-1361-8.
12. Ahamad, N., Sikander, A., and Singh, G. (2022). A Novel Reduction Approach for Linear System Approximation. Circuits Syst. Signal Process. *41*, 700–724. https://doi.org/10.1007/s00034-021-01816-4.
13. Chen, X., Zhao, S., Liu, F., and Tao, C. (2023). Laplace Distribution Based Online Identification of Linear Systems With Robust Recursive Expectation-Maximization Algorithm. IEEE Trans. Industr. Inform. *19*, 9028–9036. https://doi.org/10.1109/TII.2022.3225026.
14. Batmani, Y., and Najafi, S. (2022). An Improved Design of Event-Triggered Feedback Controllers for Linear Systems Based on Fast and Slow Dynamics. IEEE Trans. Industr. Inform. *18*, 7741–7748. https://doi.org/10.1109/TII.2022.3151808.
15. Wang, C., and Ling, Q. (2023). Stabilization of a Scalar Continuous Time Linear System with Unknown Noise Based on Two-stage Event-triggering. Int. J. Control Autom. Syst. *21*, 1493–1506. https://doi.org/10.1007/s12555-021-1074-0.
16. Wen, S.X., Shi, Y., Pan, Z.R., and Sun, X.M. (2023). Practical Bumpless Transfer Design for Switched Linear Systems: Application to Aeroengines. IEEE Trans. Industr. Inform. *19*, 11910–11919. https://doi.org/10.1109/TII.2023.3254659.
17. Thalhammer, A., Fontanini, M., Shi, J., Scaini, D., Recupero, L., Evtushenko, A., Fu, Y., Pavagada, S., Bistrovic-Popov, A., Fruk, L., et al. (2022). Distributed interfacing by nanoscale photodiodes enables single-neuron light activation and sensory enhancement in 3D spinal explants. Sci. Adv. *8*, eabp9257. https://doi.org/10.1126/sciadv.abp9257.
18. Topalovic, U., Barclay, S., Ling, C., Alzuhair, A., Yu, W., Hokhikyan, V., Chandrakumar, H., Rozgic, D., Jiang, W., Basir-Kazeruni, S., et al. (2023). A wearable platform for closed-loop stimulation and recording of single-neuron and local field potential activity in freely moving humans. Nat. Neurosci. *26*, 517–527. https://doi.org/10.1038/s41593-023-01260-4.
19. Abbott, J., Ye, T., Krenek, K., Gertner, R.S., Ban, S., Kim, Y., Qin, L., Wu, W., Park, H., and Ham, D. (2020). A nanoelectrode array for obtaining intracellular recordings from thousands of connected neurons. Nat. Biomed. Eng. *4*, 232–241. https://doi.org/10.1038/s41551-019-0455-7.
20. Hahnioser, R.H.R., Sarpeshkar, R., Mahowald, M.A., Douglas, R.J., and Seung, H.S. (2000). Digital selection and analogue amplification coexist in a cortex- inspired silicon circuit. Nature *405*, 947–951. https://doi.org/10.1038/35016072.
21. Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009). What is the best multi-stage architecture for object recognition? In Proceedings of the IEEE International Conference on Computer Vision. https://doi.org/10.1109/ICCV.2009.5459469.
22. Nair, V., and Hinton, G.E. (2010). Rectified linear units improve Restricted Boltzmann machines. In ICML 2010 - Proceedings, 27th International Conference on Machine Learning.
23. Ramachandran, P., Zoph, B., and Le, Q.V. (2017). Searching for Activation Functions. Preprint at arXiv. https://arxiv.org/abs/1710.05941.
24. Pesch, I.S., Bestelink, E., Sagazan, O. de, Mehonic, A., and Sporea, R.A. (2022). Multimodal transistors as ReLU activation functions in physical neural network classifiers. Sci. Rep. *12*. https://doi.org/10.1038/s41598-021-04614-9.
25. Misra, D. (2020). Mish: A Self Regularized Non-Monotonic Activation Function. Preprint at arXiv. https://doi.org/10.48550/arXiv.1908.08681.
26. Maas, A.L., Hannun, A.Y., and Ng, A.Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In Proceedings of the 30th International Conference on Machine Learning https://api.semanticscholar.org/CorpusID:16489696.

27. Clevert, D.A., Unterthiner, T., and Hochreiter, S. (2015). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). Preprint at arXiv. https://arxiv.org/abs/1511.07289.

28. Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., and Liu, Z. (2020). Dynamic ReLU. Lect. Notes Comput. Sci. 351–367. https://doi.org/10.1007/978-3-030-58529-7_21.

29. Hendrycks, D., and Gimpel, K. (2016). Gaussian Error Linear Units (GELUs). Preprint at arXiv. https://arxiv.org/abs/1606.08415.

30. Ioffe, S., and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. Preprint at arXiv. https://doi.org/10.48550/arXiv.1502.03167.

31. Sitzmann, V., Martel, J.N.P., Bergman, A.W., Lindell, D.B., and Wetzstein, G. (2020). Implicit neural representations with periodic activation functions. In Advances in Neural Information Processing Systems. https://doi.org/10.5555/3495724.3496350.

32. Roy, K., Jaiswal, A., and Panda, P. (2019). Towards spike-based machine intelligence with neuromorphic computing. Nature *575*, 607–617. https://doi.org/10.1038/s41586-019-1677-2.

33. Liu, X., and Di, X. (2021). TanhExp: A smooth activation function with high convergence speed for lightweight neural networks. IET Comput. Vis. *15*, 136–150. https://doi.org/10.1049/cvi2.12020.

34. Cun, Y.L., Kanter, I., and Solla, S.A. (1991). Eigenvalues of covariance matrices: Application to neural-network learning. Phys. Rev. Lett. *66*, 2396–2399. https://doi.org/10.1103/PhysRevLett.66.2396.

35. Schraudolph, N. (1997). On Centering Neural Network Weight Updates. In Tricks of the Trade (Springer Verlag).

36. Beer, R.D. (1995). On the Dynamics of Small Continuous-Time Recurrent Neural Networks. Adapt. Behav. *3*, 469–509. https://doi.org/10.1177/105971239500300405.

37. Hodgkin, A.L., and Huxley, A.F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. J. Physiol. *117*, 500–544. https://doi.org/10.1113/jphysiol.1952.sp004764.

38. Sun, Y., Zeng, Y., Zhao, F., and Zhao, Z. (2023). Multi-compartment Neuron and Population Encoding improved Spiking Neural Network for Deep Distributional Reinforcement Learning. Preprint at arXiv. https://arxiv.org/abs/2301.07275.

39. Connor, J.A., Walter, D., and McKown, R. (1977). Neural repetitive firing: modifications of the Hodgkin-Huxley axon suggested by experimental results from crustacean axons. Biophys. J. *18*, 81–102. https://doi.org/10.1016/S0006-3495(77)85598-7.

40. Chay, T.R. (1985). Chaos in a three-variable model of an excitable cell. Phys. Nonlinear Phenom. *16*, 233–242. https://doi.org/10.1016/0167-2789(85)90060-0.

41. Izhikevich, E.M. (2003). Simple model of spiking neurons. IEEE Trans. Neural Netw. *14*, 1569–1572. https://doi.org/10.1109/TNN.2003.820440.

42. Fourcaud-Trocmé, N., Hansel, D., Vreeswijk, C.V., and Brunel, N. (2003). How Spike Generation Mechanisms Determine the Neuronal Response to Fluctuating Inputs. J. Neurosci. *23*. https://doi.org/10.1523/jneurosci.23-37-11628.2003.

43. Wilson, H.R. (1999). Simplified dynamics of human and mammalian neocortical neurons. J. Theor. Biol. *200*, 375–388. https://doi.org/10.1006/jtbi.1999.1002.

44. Smith, G.D., Cox, C.L., Sherman, S.M., and Rinzel, J. (2000). Fourier analysis of sinusoidally driven thalamocortical relay neurons and a minimal integrate-and-fire-or-burst model. J. Neurophysiol. *83*, 588–610. https://doi.org/10.1152/jn.2000.83.1.588.

45. Brette, R., and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. J. Neurophysiol. *94*, 3637–3642. https://doi.org/10.1152/jn.00686.2005.

46. Mihalaş, Ş., and Niebur, E. (2009). A generalized linear integrate-and-fire neural model produces diverse spiking behaviors. Neural Comput. *21*. https://doi.org/10.1162/neco.2008.12-07-680.

47. Izhikevich, E.M. (2001). Resonate-and-fire neurons. Neural Netw. *14*, 883–894. https://doi.org/10.1016/S0893-6080(01)00078-8.

48. Fetz, E. (2023). Leaky Integrate and Fire Neurons. In Introducing Computation to Neuroscience: Selected Papers of George Gerstein, A. Aertsen, S. Grün, P.E. Maldonado, and G. Palm, eds. (Springer International Publishing), pp. 1–43. https://doi.org/10.1007/978-3-030-87447-6_1.

49. Ermentrout, G.B., and Kopell, N. (1986). Parabolic bursting in an excitable system coupled with a slow oscillation. SIAM J. Appl. Math. *46*, 233–253. https://doi.org/10.1137/0146017.

50. Simonyan, K., and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. Preprint at arXiv. https://arxiv.org/abs/1409.1556.

51. He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR.2016.90.

52. Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Handbook of Systemic Autoimmune. Diseases *1*, 1–60. https://api.semanticscholar.org/CorpusID:18268744.

53. Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Preprint at arXiv. https://arxiv.org/abs/1708.07747.

54. Howard, J., and Gugger, S. (2020). Fastai: A Layered API for Deep Learning. Information *11*, 108. https://doi.org/10.3390/info11020108.

55. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). ImageNet Large Scale Visual Recognition Challenge. Int. J. Comput. Vis. *115*, 211–252. https://doi.org/10.1007/s11263-015-0816-y.

56. Ma, N., Zhang, X., Liu, M., and Sun, J. (2021). Activate or Not: Learning Customized Activation. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. https://doi.org/10.1109/CVPR46437.2021.00794.

57. Suykens, J.K., and Vandewalle, J. (1999). Training multilayer perceptron classifiers based on a modified support vector method. IEEE Trans. Neural Netw. *10*, 907–911. https://doi.org/10.1109/72.774254.

58. Maliuk, D., and Makris, Y. (2015). An Experimentation Platform for On-Chip Integration of Analog Neural Networks: A Pathway to Trusted and Robust Analog/RF ICs. IEEE Trans. Neural Netw. Learn. Syst. *26*, 1721–1734. https://doi.org/10.1109/TNNLS.2014.2354406.

59. Gonzalez-Diaz, R., Mainar, E., Paluzo-Hidalgo, E., and Rubio, B. (2020). Neural-Network-Based Curve Fitting Using Totally Positive Rational Bases. Mathematics *8*, 2197. https://doi.org/10.3390/math8122197.

60. Raissi, M., Perdikaris, P., and Karniadakis, G.E. (2017). Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. Preprint at arXiv. https://doi.org/10.48550/arXiv.1711.10561.

61. Basdevant, C., Deville, M., Haldenwang, P., Lacroix, J.M., Ouazzani, J., Peyret, R., Orlandi, P., and Patera, A.T. (1986). Spectral and finite difference solutions of the Burgers equation. Comput. Fluids *14*, 23–41. https://doi.org/10.1016/0045-7930(86)90036-8.

62. Grand, C., and Honey, R.C. (2008). Solving XOR. J. Exp. Psychol. Anim. Behav. Process. *34*, 486–493. https://doi.org/10.1037/0097-7403.34.4.486.

63. Nitta, T. (2003). Solving the XOR problem and the detection of symmetry using a single complex-valued neuron. Neural Netw. *16*, 1101–1105. https://doi.org/10.1016/S0893-6080(03)00168-0.

64. Xu, H., Lin, J., Zhang, D., and Mo, F. (2023). Retention time prediction for chromatographic enantioseparation by quantile geometry-enhanced graph neural network. Nat. Commun. *14*, 3095. https://doi.org/10.1038/s41467-023-38853-3.

65. Price, C.C., Singh, A., Frey, N.C., and Shenoy, V.B. (2022). Efficient catalyst screening using graph neural networks to predict strain effects on adsorption energy. Sci. Adv. *8*, eabq5944. https://doi.org/10.1126/sciadv.abq5944.

66. Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. (2020). Simple and deep graph convolutional networks. In 37th International Conference on Machine Learning. https://doi.org/10.5555/3524938.3525099.

67. Chapman, J., Hsu, T., Chen, X., Heo, T.W., and Wood, B.C. (2023). Quantifying disorder one atom at a time using an interpretable graph neural network paradigm. Nat. Commun. *14*, 4030. https://doi.org/10.1038/s41467-023-39755-0.

68. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems. https://doi.org/10.5555/3295222.3295349.

69. Mahowald, K., Ivanova, A.A., Blank, I.A., Kanwisher, N., Tenenbaum, J.B., and Fedorenko, E. (2024). Dissociating language and thought in large language models. Trends Cogn. Sci. *28*, 517–540. https://doi.org/10.1016/j.tics.2024.01.011.

70. Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Preprint at arXiv. https://arxiv.org/abs/1810.04805v2.

71. Riveland, R., and Pouget, A. (2024). Natural language instructions induce compositional generalization in networks of neurons. Nat. Neurosci. *27*, 988–999. https://doi.org/10.1038/s41593-024-01607-5.

72. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. Preprint at arXiv. http://arxiv.org/abs/2010.11929.

73. Wang, Y., Zhang, W., Yip, H., Qu, C., Hu, H., Chen, X., Lee, T., Yang, X., Yang, B., Kumar, P., et al. (2023). SIC50: Determining drug inhibitory concentrations using a vision transformer and an optimized Sobel operator. Patterns *4*, 100686. https://doi.org/10.1016/j.patter.2023.100686.

74. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. (2023). LLaMA: Open and Efficient Foundation Language Models. Preprint at arXiv. https://arxiv.org/abs/2302.13971.

75. Liévin, V., Hother, C.E., Motzfeldt, A.G., and Winther, O. (2024). Can large language models reason about medical questions? Patterns *5*, 100943. https://doi.org/10.1016/j.patter.2024.100943.

76. Sandmann, S., Riepenhausen, S., Plagwitz, L., and Varghese, J. (2024). Systematic analysis of ChatGPT, Google search and Llama 2 for clinical decision support tasks. Nat. Commun. *15*, 2050. https://doi.org/10.1038/s41467-024-46411-8.

77. Jiang, W. T. (2024). Source Code for ANT. Zenodo. https://doi.org/10.5281/zenodo.13984766.