Physics Letters A 555 (2025) 130804

Contents lists available at ScienceDirect

Physics Letters A

journal homepage: www.elsevier.com/locate/pla

Letter

Exploring the Bayes-oriented noise injection approach in neural networks

Caimin An [a], Fabing Duan [a,*], François Chapeau-Blondeau [b], Derek Abbott [c]

[a] Institute of Complexity Science, Qingdao University, Qingdao 266071, PR China
[b] Laboratoire Angevin de Recherche en Ingénierie des Systèmes (LARIS), Université d'Angers, 62 avenue Notre Dame du Lac, 49000 Angers, France
[c] Centre for Biomedical Engineering (CBME) and School of Electrical & Electronic Engineering, The University of Adelaide, Adelaide, SA 5005, Australia

ARTICLE INFO

Communicated by P. Zeppini

Keywords:
Noise injection
Bayesian optimization
Neural network
Noise-boosted activation
Stochastic resonance

ABSTRACT

Noise injection, which involves the addition or multiplication of random variables to the input data or parameters of neural networks, has proven to be an effective strategy for enhancing neural network performance. However, the relationship between the generalization of a neural network and the scale parameter of injected noise is not well understood, and the optimization of noise injection for performance enhancement is a complex non-convex high-dimensional problem. This study investigates various noise injection methodologies across different neural network architectures utilizing the Bayesian optimization approach. The results indicate that, among the diverse noise injection strategies, the intrinsic hyperparameter governing the noise scale, specifically within the context of noise-boosted activations optimized through a Bayesian surrogate model, yields the most stable enhancement in network performance for function approximation, image classification, and image reconstruction tasks. These findings demonstrate the feasibility of the Bayes-oriented noise injection approach in improving the performance of neural networks.

1. Introduction

Noise has become an important informative parameter in the monitoring and measurement of physical systems and processes, useful for instance in reliability analysis of integrated circuits [1,2], error control in analog-to-digital conversion [3–5], signal perception in degraded environments [6], and fault detection in equipment [7]. In recent years, noise injection techniques aimed at improving the generalization capability of neural networks, particularly in resource-limited devices [8,9], have gained significant attention. The optimization of noise integration within neural network architectures, combined with a more comprehensive understanding of its effects, represents a significant direction with potential to explore efficient training network of low-power hardware.

Noise injection, the process of adding or multiplying random variables with certain distribution to input data or parameters of a neural network, is emerging as a powerful approach for reducing the complexity of overparameterized networks [12–15], or for enhancing the generalization abilities of neural networks to unseen data [16–18,20–25], or for speeding up the convergence of backpropagation training process of neural networks [26–28,31,32]. In terms of operational methods, multiplicative noise injection primarily involves multiplying Bernoulli or Gaussian random variables with weight coefficients to prune the neural network model size, a technique commonly known as dropout [12–14]. Multiplicative noise injection, achieved by multiplying the neuron inputs by a zero-one random mask following a certain distribution, is found to offer a flexible alternative for the emulation of a whole range of useful neuron activation functions, such as the Gaussian error linear unit (GELU) [33], the sigmoid linear unit (SiLU/Swish) [33–35], and the noise-boosted activation function [30]. By contrast, additive noise injection offers complementary capabilities, by adding random samples to input data [13,16–19], weights [17,25], labels [25], gradients [15,36], layers [32,37] or activation functions [38,39]. The resulting effectiveness of noise injection in neural networks has been continuously demonstrated, with benefits at various levels, for instance to improve function approximation [16–18,23], classification accuracy [20–22,24,25], and robustness of neural networks [40–43], and also to generate high-fidelity images [44–46].

To theoretically understand the role of noise injection, it has been proven [16–18,20,21] that noise injection is equivalent to Tikhonov regularization in the asymptotic regime of small injected noise and for an infinite number of injected noise samples. However, in the practical training of neural networks, this asymptotic regime is hardly accessible due to the associated lengthy time requirements. For instance, a suboptimal or acceptable non-zero injected noise level is often em-

* Corresponding author.
E-mail address: fabing.duan@hotmail.com (F. Duan).

https://doi.org/10.1016/j.physleta.2025.130804
Received 16 March 2025; Received in revised form 17 June 2025; Accepted 5 July 2025
Available online 9 July 2025
0375-9601/© 2025 Elsevier B.V. All rights reserved, including those for text and data mining, AI training, and similar technologies.

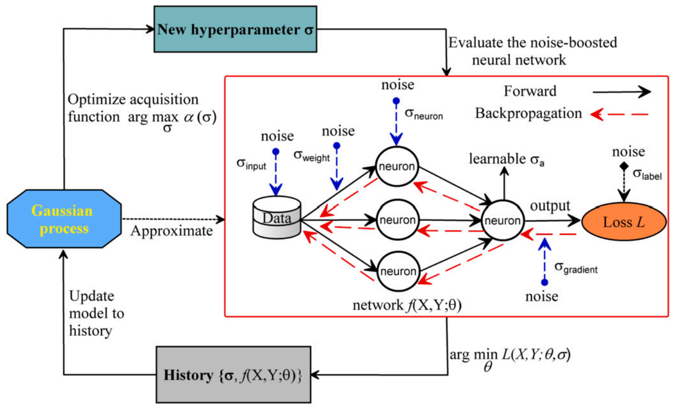

Fig. 1. Flowchart of the Bayes-oriented noise injection approach in a noise-boosted neural network $f(X,Y;\theta)$. The subscripts of the noise level, σ_{input}, σ_{weight}, σ_{neuron}, σ_{gradient}, and σ_{label}, indicate the locations at which noise is injected. A Gaussian process serves as a surrogate model to update the parameters between each network performance estimation and the injected noise level, while recording both the best parameters and the corresponding network performance throughout the process.

pirically determined through a finite number of trials to improve the generalization of networks compared to networks without noise injection [16,17,20]. Moreover, for numerous injected noise components, the practical approach of empirically applying a finite grid search to optimize a set of injected noise levels in a small range, may not lead to a global optimum. This non-convex high-dimensional optimization of the noise parameters represents a significant limit to obtain the best benefit of noise injection in artificial neural networks.

Naturally, a practical and effective approach is to treat noise levels as parameters of the neural network, and use backpropagation to adaptively update and learn them during the training process [22,23,38–40,46–48], to finally achieve a locally optimal feasible solution. Especially, in terms of the biological plausibility of neurons exhibiting inactivation below zero and yielding an unbounded response above zero, certain smooth and non-monotonic activation functions, such as GELU [33], SiLU [33–35] and general noise-boosted neuron models [22,30], have been proposed to mimic the adaptation of neurons. Among these activations [22,30,33–35], the noise level is implicitly integrated into the activation function and can be updated and learned through the stochastic gradient descent (SGD) algorithm.

However, two crucial questions remain unresolved. First, finding the optimal noise level using the SGD algorithm is a highly nonconvex and multimodal optimization challenge. The converged noise level values can vary significantly depending on the initial values, resulting in variations in network performance. Upon the convergence of the noise level, it is unclear whether the obtained noise level value represents a local optimum, a global optimum, or even a saddle point. In addition, the SGD algorithm does not apply constraints on the noise level to ensure its non-negativity, and therefore cannot guarantee convergence to a non-zero non-negative value. This might lead to gradient explosion with respective to the noise level and cause the optimization process of the designed neural network to fail [30]. Second, for the combination of several noise injection strategies applied at different placements in the neural network, it remains uncertain whether noise levels in inputs, weights, activations, hidden layers, labels, and even parameter gradients are all equally beneficial, and how to optimize them collectively as a vector.

In this study, we employ a Bayesian optimization strategy [49,50] to learn injected noise levels in various network placements. This approach contrasts with previous methods of finding optimal noise by an empirical grid search or the SGD algorithm [16–18,20,21]. For each trial in the Bayesian optimization process, network performance is modeled as a Gaussian prior on the injected noise level, which is then updated using the expected improvement derived from the acquisition function. The use of a Bayesian optimization strategy to search for the optimal noise in neural networks [43,51–53] or bistable systems [54] has been extensively studied. A theoretically guaranteed noise injection approach that employs Bayesian optimization to determine the optimal characteristics of noise injected during training was proposed in [43]. It has been shown that, when the perturbations induced by the hardware during actual deployment fall within a guaranteed range, injecting different types of noise can improve the robustness of deep neural networks [43]. Furthermore, it has been demonstrated [51] that injecting noise into neural network weights is equivalent to Bayesian inference on a deep Gaussian process. Based on this equivalence, Yuan et al. proposed a Monte-Carlo noise injection method for uncertainty quantification [51], which involves injecting noise into parameters during training and performing multiple forward propagations during inference. Additionally, it was found that incorporating measurement duration into the Bayesian optimization framework allows the system to learn and adapt optimal noise levels, thereby balancing data quality and experimental efficiency [52]. Inspired by brain asymmetry, an asymmetric stochastic resonance unit-based preprocessing module [53] has been designed for a convolutional neural network-based epilepsy electroencephalogram diagnostic system, wherein Bayesian optimization was also utilized to find optimal parameters for maximizing seizure prediction sensitivity and improving the diagnostic performance of a ResNet-50 classifier. They also investigated an overdamped bistable stochastic resonance model with time-delayed feedback [54], where Bayesian optimization was applied to optimize hyperparameters, including the injected noise levels, thereby improving the computational performance of physical reservoir computing systems.

In this paper, we systematically review and compare various noise injection methods in neural networks, with a particular focus on the impact of noise level parameters embedded in activation functions on network performance. Here, by embedding different types of noise into activation models, we can derive various forms of noise-boosted activation functions, including commonly used ones such as sigmoid and ReLU. Furthermore, Bayesian optimization is employed to evaluate the effectiveness of combining different noise injection strategies across various network placements. Through iterative adjustment of the hyperparameters associated with noise levels, it is observed that the injection of an optimal amount of noise to the input data, network weights, neurons, and gradients (with respect to weights or labels) can enhance neural network performance. Furthermore, this work extends the use of noise-boosted activation functions (e.g., GELU, SiLU) by explicitly optimizing their intrinsic noise level parameters through Bayesian optimization. This builds upon prior work that proposed such activations but did not systematically optimize the associated noise level parameter [22,30,33–35]. It is emphasized that the intrinsic hyperparameter for the noise level in these noise-boosted activations, optimized through a Bayesian surrogate model, leads to the most stable improvements in network performance for tasks such as function approximation, image classification, and image reconstruction. Furthermore, the presence of non-zero optimal noise levels, which are closely related to the stochastic resonance phenomenon in neural networks, is demonstrated in the Bayesian optimization process. This strategy for enhancing neural network performance through noise injection represents a potential application of the benefits of noise in machine learning.

2. Network model and noise injection

2.1. Noise injection approaches and noise-boosted neuron model

The Bayes-oriented noise injection approach for noise-boosted neural networks is illustrated in Fig. 1. In this framework, a neural network without any form of noise injection serves as the reference baseline. A noise-boosted neural network is defined as one in which noise is injected into one or more specific placements: inputs, labels, network weights, neurons or gradients during the backpropagation process. Hyperparameter tuning, in this context, involves finding the optimal amount of noise to be added to the designed neural network. The injected noise ξ is assumed to be scaled, i.e., $\xi = \sigma\xi_n$, where the noise scaling parameter σ represents the intensity of the noise, also referred to as the noise level. The normalized random variable ξ_n has a zero mean and unit variance. This reparameterization method simplifies the process of optimizing the noise-boosted neural network to the task of finding the optimal noise scale σ.

For different placements in networks shown in Fig. 1, the noise scale σ is specifically designated as σ_{input}, σ_{weight}, σ_{neuron}, σ_{gradient}, or σ_{label}. Here, the hyperparameter set $\boldsymbol{\sigma} = \{\sigma_{\text{input}}\mathbb{1}, \sigma_{\text{weight}}\mathbb{1}, \sigma_{\text{neuron}}\mathbb{1}, \sigma_{\text{gradient}}\mathbb{1}, \sigma_{\text{label}}\mathbb{1}\}$ and the indicator function $\mathbb{1} = 1$ if noise injection is present, otherwise it equals zero. For instance, consider a fully connected neural network with one hidden layer represented as

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{W}_2\,\varphi(\boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{b}_2, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^{N\times 1}$ is the input vector, $\boldsymbol{W}_1 \in \mathbb{R}^{K\times N}$ and $\boldsymbol{W}_2 \in \mathbb{R}^{M\times K}$ are weight matrices, $\boldsymbol{b}_1 \in \mathbb{R}^{K\times 1}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{M\times 1}$ denote bias vectors, the network parameter set $\boldsymbol{\theta} = \{\boldsymbol{b}_1, \boldsymbol{W}_1, \boldsymbol{b}_2, \boldsymbol{W}_2\}$ and $\varphi(x)$ is the activation function of hidden neurons. Let $\{\boldsymbol{x}(i), \boldsymbol{y}(i)\}_{i=1}^{S}$ denote S examples of the data set for training the network in a supervised learning manner, where the label vector is $\boldsymbol{y} \in \mathbb{R}^{M\times 1}$. The empirical loss function $L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}, \boldsymbol{\sigma})$ can be calculated on the network output $f(\boldsymbol{x}, \boldsymbol{\theta})$ and the label $\boldsymbol{y}$ for current network parameters $\boldsymbol{\theta}$ and the noise level set $\boldsymbol{\sigma}$. Then, $\boldsymbol{\theta}$ can be updated through the mini-batch gradient descent approach [22,55]

$$\theta^t = \theta^{t-1} - \gamma^t \sum_{\boldsymbol{x}\in\mathcal{B}} \frac{\partial L(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}, \boldsymbol{\sigma})}{\partial \boldsymbol{\theta}}\Big|_{\theta=\theta^{t-1}} \tag{2}$$

with a mini-batch $\mathcal{B}$ sampled from the input space and the learning rate $\gamma^t > 0$ at the t-th training epoch.

In the aforementioned training process, the types of noise injection approaches can be expressed as

$$\widetilde{\boldsymbol{x}} = \boldsymbol{x} + \sigma_{\text{input}}\,\boldsymbol{\xi}_{\boldsymbol{x}}, \tag{3}$$

$$\widetilde{\boldsymbol{y}} = \boldsymbol{y} + \sigma_{\text{label}}\,\boldsymbol{\xi}_{\boldsymbol{y}}, \tag{4}$$

$$\widetilde{\boldsymbol{\theta}} = \boldsymbol{\theta} + \sigma_{\text{weight}}\boldsymbol{\xi}_{\theta}, \tag{5}$$

$$\widetilde{\varphi}(\boldsymbol{x}) = \varphi(\boldsymbol{x}) + \sigma_{\text{neuron}}\boldsymbol{\xi}_{\varphi}, \tag{6}$$

$$\widetilde{\frac{\partial L}{\partial \boldsymbol{\theta}}} = \frac{\partial L}{\partial \boldsymbol{\theta}} + \sigma_{\text{gradient}}\boldsymbol{\xi}_{L}, \tag{7}$$

where the normalized random vectors $\boldsymbol{\xi}_{\boldsymbol{x}}$, $\boldsymbol{\xi}_{\boldsymbol{y}}$, $\boldsymbol{\xi}_{\theta}$, $\boldsymbol{\xi}_{\varphi}$ and $\boldsymbol{\xi}_L$ with zero-mean and the identity matrix have the same dimensions as $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}, \varphi$ and $\partial L/\partial\boldsymbol{\theta}$, respectively. It is noted that the noise injection approach indicated in Eq. (6) is equivalent to the method of injecting noise between layers as described in [44,45], because Eq. (6) can be viewed as the input passing through an activation function followed by the addition of noise $\sigma_{\text{neuron}}\xi_{\varphi}$, with the sum of the two acting as the input to the next layer.

As shown in Fig. 1, the sigmoid activation function, defined as $\varphi(u) = (1 + e^{-u})^{-1}$, and the rectified linear unit (ReLU), expressed as $\varphi(u) = \max(u, 0)$ [56], are two representative activations commonly employed in artificial neural network architectures. However, these commonly used activation functions do not contain learnable parameters, thereby attributing the learning capability of the neural network solely to the connection weights between neurons. This viewpoint is somewhat limited, as the learning of hyperparameters associated with these activation functions has also been shown to mimic the adaptive behavior of neurons, potentially leading to enhanced performance.

In order to introduce these learnable activations, we start from the suprathreshold stochastic resonance (SSR) model [10] consisting of an array of McCulloch-Pitts [11] neurons

$$h(u) = \frac{1}{2}[1 + \text{sgn}(u)], \tag{8}$$

yielding the output of the SSR model

$$\bar{h}(u) = \frac{1}{T}\sum_{t=1}^{T} h(u + \xi_t). \tag{9}$$

Here, sgn(u) denotes the signum function, u is the common input and ξ_t $(t = 1, 2, \cdots, T)$ are mutually independent noise components with the common probability density function (PDF) $f_\xi(\xi)$. Then each neuron yields a response of unity with probability

$$\varphi(u) = \mathbb{E}_\xi[h(u + \xi)] = \int_{-\infty}^{\infty} h(u + \xi) f_\xi(\xi) d\xi$$
$$= \int_{-u}^{\infty} f_\xi(\xi) d\xi = 1 - F_\xi(-u), \tag{10}$$

where $F_\xi(u) = \int_{-\infty}^{u} f_\xi(\xi)d\xi$ denotes the cumulative distribution function (CDF). For a sufficiently large number T of neurons [10], the output $\bar{h}(u)$ of the SSR model approaches close to the average firing probability $\varphi(u)$, i.e. $\lim_{T\to\infty}\bar{h}(u) = \varphi(u)$. Therefore, we consider the average firing probability $\varphi(u)$ as the output of the array of McCulloch-Pitts neurons, treating it as a type of activation function that persists even for negative inputs ($u < 0$) under the influence of background noise ξ_t. Then, Eq. (10) forms a unified noise-boosted activation model with learnable noise-related hyperparameters [22,30,73].

For instance, when considering injected noise with a Gaussian PDF $f_\xi(u) = \exp(-u^2/2\sigma_a^2)/\sqrt{2\pi\sigma_a^2}$, Eq. (10) simplifies into the Gaussian error unit (GEU)

$$\varphi(u; \sigma_a) = \frac{1}{2} + \frac{1}{2}\text{erf}(u/\sqrt{2}\sigma_a), \tag{11}$$

where the Gauss error function $\text{erf}(u) = 2/\sqrt{\pi}\int_0^u e^{-t^2}dt$. For logistic noise ξ with its PDF $f_\xi(u) = e^{-u/\sigma_a}/[\sigma_a(1 + e^{-u/\sigma_a})^2]$ and CDF $F_\xi(u) = 1/(1 + e^{-u/\sigma_a})$, Eq. (10) yields the variant sigmoid activation

$$\varphi(u; \sigma_a) = (1 + e^{-u/\sigma_a})^{-1}. \tag{12}$$

Here, the hyperparameter σ_a is just the scale parameter of the injected noise [22,30].

It is noted that the activation indicated in Eq. (10) generates continuous outputs ranging from zero to one, effectively representing the likelihood of the gate being open or closed [30,33,34]. Therefore, by using the probability from Eq. (10) as a gating mechanism, the input x is activated with that probability and deactivated with the complementary probability. This probabilistic gating mechanism results in a unidirectional saturated activation function model

$$\varphi(u) = u[0 \times F_\xi(-u) + 1 \times (1 - F_\xi(-u))] = u[1 - F_\xi(-u)]. \tag{13}$$

Correspondingly, from Eqs. (11), (12) and (13), the activations of the Gaussian error linear unit (GELU) [30,33]

$$\varphi(u; \sigma_a) = \frac{u}{2} + \frac{u}{2}\text{erf}(u/\sqrt{2}\sigma_a) \tag{14}$$

and the sigmoid linear unit (SiLU) [22,30,33–35]

$$\varphi(u; \sigma_a) = u(1 + e^{-u/\sigma_a})^{-1} \tag{15}$$

are unified into a consolidated noise-boosted activation model [30] based on the type of injected noise.

Thus, this model of Eq. (13) further derives various activation functions according to different noise types. For example, the exponential linear unit (ExLU)

$$\varphi(u; \sigma_a) = \begin{cases} u, & u \geq 0, \\ ue^{u/\sigma_a}, & u < 0 \end{cases} \tag{16}$$

can be derived for an exponential noise PDF $f_\xi(u) = (e^{-u/\sigma_a})/\sigma_a$ on the support interval $u \in [0, \infty)$, and the Rayleigh linear unit (RayLU) can be expressed as

$$\varphi(u; \sigma_a) = \begin{cases} u, & u \geq 0, \\ ue^{-\frac{u^2}{2\sigma_a^2}}, & x < 0 \end{cases} \tag{17}$$

based on the Rayleigh-distributed injected noise with PDF $f_\xi(u) = u(e^{-u^2/2\sigma_a^2})/\sigma_a^2$ $(x \geq 0)$ and zero otherwise. Both ExLU and RayLU exhibit similar characteristics to GELU and SiLU [30]. This indicates that these activation functions can benefit from the incorporation of injected noise, thereby enhancing their modeling capabilities across various neural network architectures.

Here, it is also emphasized that the learnable parameter σ_a in Eqs (11)–(17), which actually represents the injected noise level in these activations, can be updated by the gradient descent approach [22,30], and in this context, it is treated as the hyperparameter to be optimized by the Bayesian surrogate model. Therefore, the hyperparameter set is expanded as $\boldsymbol{\sigma} = \{\sigma_{\text{input}}\mathbb{1}, \sigma_{\text{label}}\mathbb{1}, \sigma_{\text{weight}}\mathbb{1}, \sigma_{\text{neuron}}\mathbb{1}, \sigma_{\text{gradient}}\mathbb{1}, \sigma_a\mathbb{1}\}$, as shown in Fig. 1.

2.2. Bayesian optimization of noise injection

Since the generalization ability of a neural network has no explicit functional structure with respect to the hyperparameter set $\boldsymbol{\sigma}$, and its evaluation is time-consuming to perform, finding the optimal set $\boldsymbol{\sigma}$ of injected noise levels to maximize the generalization ability is a nonconvex high-dimensional problem. In such cases, a commonly used surrogate model is Gaussian process [49,50,57], which models the loss function $L(\boldsymbol{\sigma})$ over the search space of $\boldsymbol{\sigma} \in \mathbb{R}_+^d$, where d is the dimensionality, and each element $\sigma \in \boldsymbol{\sigma}$ is nonnegative. Assume the observation set $\mathcal{D} = \{\boldsymbol{\sigma}_i, L(\boldsymbol{\sigma}_i)\}_{i=1}^n$ is with a multivariate Gaussian distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, the posterior is also a Gaussian process. Here, the covariance matrix $\boldsymbol{\Sigma}$ is frequently calculated by the kernel $\boldsymbol{\kappa}$ on the input space, e.g., the squared exponential kernel $\boldsymbol{\kappa}(\boldsymbol{\sigma}_i, \boldsymbol{\sigma}_j) = e^{-\|\boldsymbol{\sigma}_i - \boldsymbol{\sigma}_j\|^2}$, to compute the element $\boldsymbol{\Sigma}_{ij}$ in the i-th row and j-th column of the covariance matrix $\boldsymbol{\Sigma}$ [49,50,57]. This kernel function will be employed in all the Bayesian optimization experiments below, and it will not be mentioned again in the subsequent discussions. Given a possible test point $\boldsymbol{\sigma}_*$, the corresponding variable of the loss $L_*(\boldsymbol{\sigma}_*)$ has the distribution $\mathcal{N}(\hat{\mu}(\boldsymbol{\sigma}_*), \widehat{\text{var}}(\boldsymbol{\sigma}_*))$ with mean

$$\hat{\mu}(\boldsymbol{\sigma}_*) = \boldsymbol{\Sigma}_*^\top \boldsymbol{\Sigma}^{-1} L_*(\boldsymbol{\sigma}_*) \tag{18}$$

and covariance matrix

$$\widehat{\text{var}}(\boldsymbol{\sigma}_*) = \boldsymbol{\Sigma}_{**} - \boldsymbol{\Sigma}_*^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{\Sigma}_*, \tag{19}$$

with $\boldsymbol{\Sigma}_{**} = \boldsymbol{\kappa}(\boldsymbol{\sigma}_*, \boldsymbol{\sigma}_*)$ and $\boldsymbol{\Sigma}_* = \boldsymbol{\kappa}(\boldsymbol{\sigma}, \boldsymbol{\sigma}_*)$ [49,50,57]. Here, the mean $\hat{\mu}(\boldsymbol{\sigma})$ represents the model prediction, and the variance $\widehat{\text{var}}(\boldsymbol{\sigma})$ indicates the posterior uncertainty [49,50,57].

From the observation set $\mathcal{D}$, we find the current minimization of $L_{\min}$, and seek to maximize the marginal increment

$$\max\{L(\boldsymbol{\sigma}), L_{\min}\} - L_{\min} = \max\{L(\boldsymbol{\sigma}) - L_{\min}, 0\} \tag{20}$$

at a new location of $\boldsymbol{\sigma}$. Then, the acquisition function of the expected improvement [49,50,57]

$$\alpha(\boldsymbol{\sigma}) = \mathbb{E}_L\left[\max\{L(\boldsymbol{\sigma}) - L_{\min}, 0\}\right] = (L_{\min} - \hat{\mu}(\boldsymbol{\sigma}))\Phi\left(\frac{L_{\min} - \hat{\mu}(\boldsymbol{\sigma})}{\sqrt{\widehat{\text{var}}(\boldsymbol{\sigma})}}\right) + \sqrt{\widehat{\text{var}}(\boldsymbol{\sigma})}\,\phi\left(\frac{L_{\min} - \hat{\mu}(\boldsymbol{\sigma})}{\sqrt{\widehat{\text{var}}(\boldsymbol{\sigma})}}\right) \tag{21}$$

determines which point in $\boldsymbol{\sigma}$ should be evaluated next. Here $\mathbb{E}_L(\cdot)$ denotes the expectation operator with respect to the Gaussian distribution of L, and $\Phi(u)$ and $\phi(u)$ are CDF and PDF of the standard Gaussian random variable. In practice, the noise level σ is assumed to have a uniform prior over the interval $[0, a]$, we can utilize the tree-structured Parzen estimator (TPE) approach [57] or the quasi-Newton method [49,58] to optimize $\boldsymbol{\sigma}$ by Bayesian optimization [59,60] in the subsequent experiments.

3. Main results

In the following experiments, we will show that the hyperparameter set $\boldsymbol{\sigma}$ of the noise injection optimized by the acquisition function in Eq. (21) can improve the generalization of neural networks.

3.1. Motivated example of noise injection for function approximation

As a motivating example (see source codes in [61]), we first consider a fully connected $1 \times N \times 1$ sigmoid neural network based on the activation in Eq. (12) ($\sigma_a = 1$) to fit the forest function on noisy observations

$$y = e^{-(x-2)^2} + e^{-\frac{(x-6)^2}{10}} + (x^2 + 1)^{-1} + \xi, \tag{22}$$

where the Gaussian background noise ξ is with zero-mean and variance 0.12. The samples of noisy data $\{x_i, y_i\}_{i=1}^{n=19}$ in the interval $[-2, 10]$ are shown in Fig. 2 (a). The sigmoid neural network is trained on this noisy data for 1.5×10^4 epochs using the Adam optimizer [66] and the learning rate $\gamma = 0.01$ in Eq. (2). The choice of network architecture, particularly the size N of the hidden layer, is critical for balancing model complexity and representational capacity. Here, a hidden layer of size $N = 14$ is employed, and similar results are observed for other $1 \times N \times 1$ networks with N approximately the size of the dataset. During training, the noise level parameter embedded within the GEU activation function is optimized over the range $[0, 10]$, whereas for networks employing explicit noise injection approaches, the corresponding noise levels are optimally determined within the range $[0, 0.3]$. In particular, when the noise is injected into the network weights, the optimization of noise level is constrained to the narrower range $[0, 0.01]$.

It is seen in Fig. 2 (a) that the output (solid line) of the sigmoid network fits the noisy data well with a small mean-squared-error (MSE) of $(1.89 \pm 0.03) \times 10^{-5}$, as indicated in Fig. 2 (b). However, when compared to the true signal illustrated in Fig. 2 (a) (dashed line), the trained sigmoid network exhibits the overfitting effect. For instance, for 10 groups of the testing set $\{x_j, y_j\}_{j=1}^{n=30}$, Fig. 2 (b) shows that the trained sigmoid network has poor generalization to new observation data and presents higher MSEs (dashed line) in the order of $(4.0 \pm 0.45) \times 10^{-2}$.

Subsequently, in accordance with Eqs. (3)–(17), various noise injection approaches are experimentally implemented to enhance the generalization performance of the designed neural network for the function approximation task. The number of Monte Carlo experiments for the direct injection of noise samples is set to 10, while the number of Bayesian optimization trials is fixed at 20. Unless explicitly stated otherwise, all subsequent experiments followed this configuration. Figs. 2 (c)–(h) depict the network outputs with the optimized noise levels obtained through Bayesian optimization. It is observed that the optimal injected noise effectively avoids the overfitting behavior of neural networks, resulting in outputs close to the true signal.

Table 1 presents the training and testing MSEs for the designed network under different noise injection methods, and the generalization gap between training and testing MSEs is minimized for these noise injection approaches at the optimal noise level obtained by the Bayesian optimization. Among them, the fully connected $1 \times 14 \times 1$ neural network with GEU activations achieves a testing MSE in the order of $(1.94 \pm 0.04) \times 10^2$ at the optimal noise level $\sigma_a = 5.65$, which is of the same order as the training MSE of $(1.01 \pm 0.01) \times 10^{-2}$, as illustrated in Fig. 2 (a).

Of course, we can also combine two or more noise injection approaches to enhance the generalization performances of neural net-

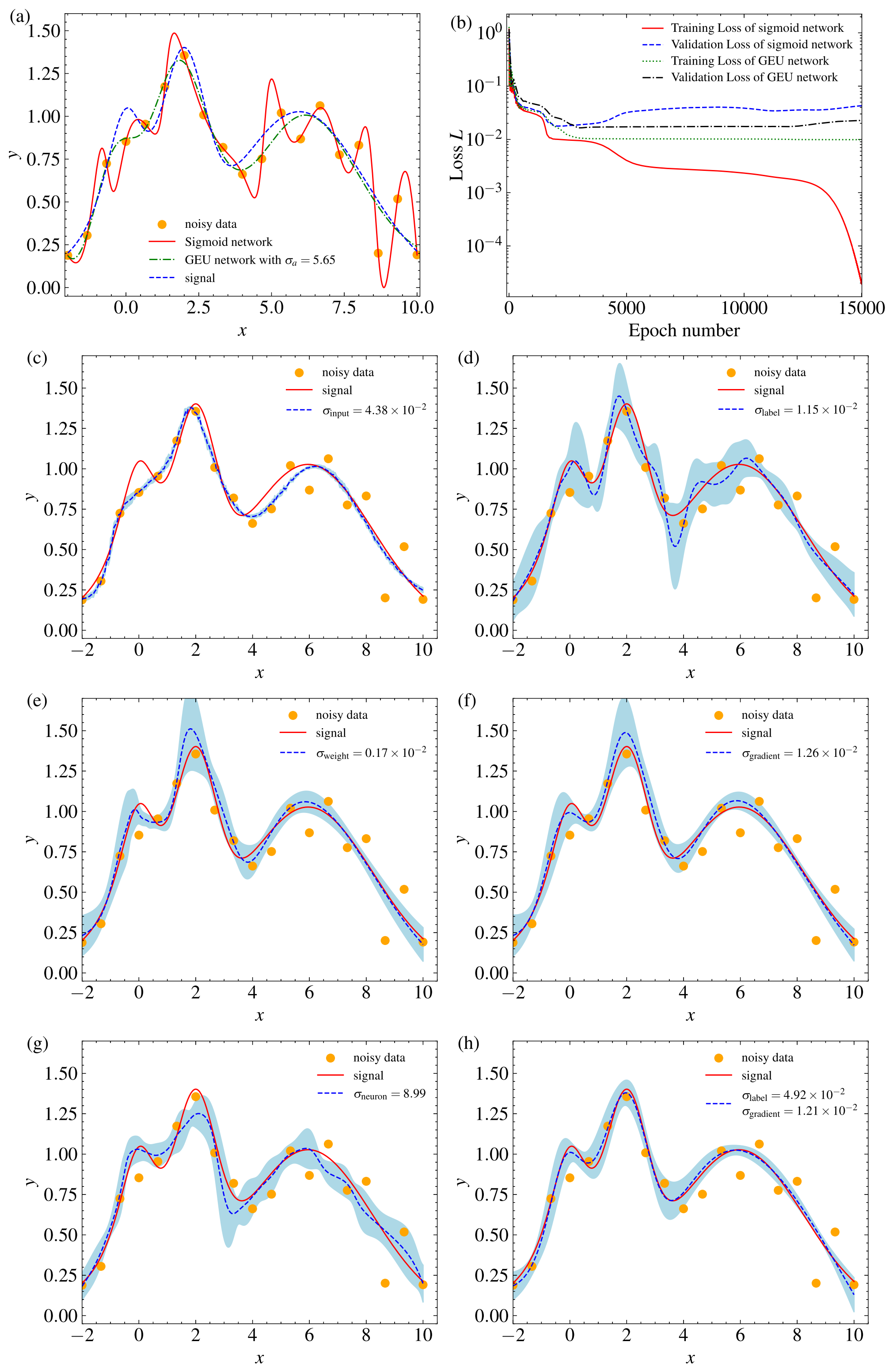

Fig. 2. (a) Outputs of the sigmoid network (solid line) and the GEU network (dash-dot line). For comparison, the target forest function (dashed line) of Eq. (22) and the observations (●) are also plotted. (b) The training and validation losses of the sigmoid and GEU networks. Statistical average outputs of the designed neural network with noise injection into (c) the input, (d) the label, (e) weights, (f) gradients, (g) hidden neurons, and (h) both the labels and the gradients. The optimal noise level parameters obtained via Bayesian optimization are also given in the inserts, and the blue shaded region around the solid line represents an error bar corresponding to ± one standard deviation over 10 experimental trials.

Table 1
MSEs of the neural network with various noise injection approaches for function approximation at the corresponding optimized noise levels.

Results \ Position	$\boldsymbol{x}$	$\boldsymbol{y}$	θ	$\partial L/\partial\theta$	Sigmoid(σ_{neuron})	Sigmoid	GEU(σ_a)
Training MSE	$(1.03 \pm 0.03) \times 10^{-3}$	$(1.25 \pm 1.36) \times 10^{-3}$	$(5.90 \pm 2.18) \times 10^{-3}$	$(5.39 \pm 1.73) \times 10^{-3}$	$(1.00 \pm 0.45) \times 10^{-3}$	$(1.89 \pm 0.03) \times 10^{-5}$	$(1.01 \pm 0.01) \times 10^{-2}$
Optimal noise level	4.38×10^{-2}	1.15×10^{-2}	1.77×10^{-3}	1.26×10^{-2}	8.99	/	5.65
Testing MSE	$(1.56 \pm 0.38) \times 10^{-2}$	$(3.31 \pm 0.61) \times 10^{-2}$	$(2.74 \pm 0.28) \times 10^{-2}$	$(2.46 \pm 0.30) \times 10^{-2}$	$(2.61 \pm 0.50) \times 10^{-2}$	$(4.0 \pm 0.45) \times 10^{-2}$	$(1.94 \pm 0.04) \times 10^{-2}$

Table 2
MSEs of the neural network with combined noise injection approaches for function approximation at the corresponding optimized noise levels.

Results \ Position	GEU(σ_a), $\boldsymbol{x}$	GEU(σ_a), $\boldsymbol{y}$	θ, $\boldsymbol{x}$	GEU(σ_a), $\partial L/\partial\theta$	$\boldsymbol{y}$, $\partial L/\partial\theta$	GEU(σ_a), $\boldsymbol{x}$, $\partial L/\partial\theta$
Training MSE	$(9.90 \pm 0.29) \times 10^{-3}$	$(9.89 \pm 0.54) \times 10^{-3}$	$(1.63 \pm 0.15) \times 10^{-2}$	$(1.67 \pm 0.09) \times 10^{-2}$	$(6.98 \pm 1.95) \times 10^{-3}$	$(1.38 \pm 0.07) \times 10^{-2}$
Optimal noise level	5.00, 7.15×10^{-2}	4.17, 2.67×10^{-2}	3.90×10^{-3}, 4.20×10^{-4}	2.43×10^{-1}, 1.82×10^{-1}	4.92×10^{-2}, 1.21×10^{-2}	7.53×10^{-1}, 1.23×10^{-1}, 1.16×10^{-1}
Testing MSE	$(1.93 \pm 0.27) \times 10^{-2}$	$(2.12 \pm 0.22) \times 10^{-2}$	$(2.14 \pm 0.41) \times 10^{-2}$	$(2.60 \pm 0.45) \times 10^{-2}$	$(2.29 \pm 0.27) \times 10^{-2}$	$(2.24 \pm 0.17) \times 10^{-2}$

works, and the corresponding results are shown in Table 2. For example, in Fig. 2 (h), we simultaneously injected noise optimized through the Bayesian optimization method into both the labels and the gradients during network training, resulting in outputs that closely match the true signal. This combination approach also achieves a testing MSE of $(2.29 \pm 0.27) \times 10^{-2}$. However, as shown in Tables 1 and 2, methods that directly inject noise samples into the network, as described in Eqs. (3)–(7), result in higher testing MSEs and poorer generalization, compared to the network with GEU activations. Moreover, as shown in Table 1, the statistical error of the loss of the GEU network is very small. This indicates that constructing neural networks with noise-boosted activations is more practical, as it does not require extensive experiments with random noise samples to statistically confirm its generalization performance.

Specifically, for the convex loss function of MSE and injecting noise samples into the weight coefficients of the neural network defined in Eq. (1), applying Jensen inequality to the expectation of the loss function yields

$$\begin{aligned}\mathbb{E}_{\xi_\theta}\left[L(\boldsymbol{x},\boldsymbol{y};\widetilde{\boldsymbol{\theta}})\right] &= \mathbb{E}_{\xi_\theta}\left[\sum_{i=1}^{S}\left|\boldsymbol{y}(i) - \widetilde{\boldsymbol{W}}_2\,\varphi(\widetilde{\boldsymbol{W}}_1\boldsymbol{x}(i)+\widetilde{\boldsymbol{b}}_1)+\widetilde{\boldsymbol{b}}_2\right|^2\right]\\ &\geq \sum_{i=1}^{S}\left|\boldsymbol{y}(\ell) - \boldsymbol{W}_2\,\mathbb{E}_{\xi_\theta}[\varphi(\widetilde{\boldsymbol{W}}_1\boldsymbol{x}(i)+\widetilde{\boldsymbol{b}}_1)]+\boldsymbol{b}_2\right|^2\\ &= \sum_{i=1}^{S}\left|\boldsymbol{y}(\ell) - \boldsymbol{W}_2\Phi(\boldsymbol{W}_1\boldsymbol{x}(i)+\boldsymbol{b}_1)+\boldsymbol{b}_2\right|^2, \qquad (23)\end{aligned}$$

where the noise-boosted activation $\Phi(u) = \mathbb{E}_\xi[\varphi(u+\rho\xi)]$ and $\rho = x(i) + b_1$. This inequality also holds for direct noise injection into the data, labels, and hidden-layer activation functions. It is evident that a noise-boosted network with activation $\Phi(u)$ has a lower testing loss compared to networks with direct noise injection, given identical weight coefficients. If the activation $\varphi(u)$ is the McCulloch-Pitts [11] neuron as defined in Eq. (8), and the injected noise ξ_θ is Gaussian distributed, the activation $\Phi(u)$ in Eq. (23) corresponds to the noise-boosted model in Eq. (11). These results demonstrate the practicality and effectiveness of neural networks employing the activation function defined in Eq. (10), compared to direct noise injection methods. The noise level σ_a in Eqs. (10)–(17) is implicitly embedded in the noise-boosted activation model, making parameter optimization during training more efficient. However, we generally do not utilize the sigmoid function for $\varphi(u)$ when deriving the noise-boosted activation $\Phi(u)$. This is because the expression $\Phi(u) = \mathbb{E}_\xi[\varphi(u+\rho\xi)]$ involves an integral operator, rather than having an explicit solution as in the derivation of the noise-boosted neuron model in Eq. (10). Consequently, the gradient $\partial\Phi(u)/\partial\sigma_a$ of $\Phi(u)$ with respect to σ_a also contains an integral, leading to increased computational complexity during noise level updates.

From the motivated example above and the theoretical analysis, the benefits of noise-enhanced neural networks utilizing activation functions defined in Eq. (10) are evident. Consequently, the subsequent experiments will mainly focus on the designed noise-boosted network architectures. Additionally, for comparison, direct noise injection methods will also be included.

3.2. Noise injection as the stochastic resonance effect

Noise injection in neural networks is closely linked to the phenomenon of stochastic resonance [62]. Here, we argue that stochastic resonance, broadly defined, denotes improved performance in a nonlinear system under an optimal non-zero noise level, compared to its performance without noise. Given the inherent nonlinearity of neurons, a neural network functions as a high-dimensional nonlinear mapping system. The underlying mechanism by which noise injection enhances the generalization performance of neural networks is in line with the principles of stochastic resonance [22,26–30], which has been extensively studied in fields such as physics [71,72,75–78] and metrology [2,7,23,26–29,63–65,70].

To illustrate the consistency between noise injection and the theory of stochastic resonance, let us consider the example of the method of injecting noise $\boldsymbol{\xi}_x$ into the input data $\boldsymbol{x}$ indicated in Eq. (3) with a small noise scale $0 < \sigma_{\text{input}} \ll 1$. Under this condition, a second-order Taylor expansion of $L(\widetilde{\boldsymbol{x}},\boldsymbol{y})$ around $\boldsymbol{x}$ can be expressed as

$$\begin{aligned}L(\widetilde{\boldsymbol{x}},\boldsymbol{y}) \approx L(\boldsymbol{x},\boldsymbol{y}) + \sigma_{\text{input}}\nabla_{\boldsymbol{x}}L(\boldsymbol{x},\boldsymbol{y})^\top\boldsymbol{\xi}_x + \frac{\sigma_{\text{input}}^2}{2}\boldsymbol{\xi}_x^\top\nabla_{\boldsymbol{xx}}^2 L(\boldsymbol{x},\boldsymbol{y})\boldsymbol{\xi}_x\\ + \mathcal{O}(\sigma_{\text{input}}^3), \qquad (24)\end{aligned}$$

where $\nabla_{\boldsymbol{x}}L(\boldsymbol{x},\boldsymbol{y})$ is the gradient of $L(\boldsymbol{x},\boldsymbol{y})$ with respect to $\boldsymbol{x}$, and $\nabla_{\boldsymbol{xx}}^2 L(\boldsymbol{x},\boldsymbol{y})$ represents the Hessian matrix of $L(\boldsymbol{x},\boldsymbol{y})$ with respect to $\boldsymbol{x}$. For a normalized random variable $\boldsymbol{\xi}_x$, $\mathbb{E}_{\xi_x}[\boldsymbol{\xi}_x] = \boldsymbol{0}$, $\mathbb{E}_{\xi_x}[\boldsymbol{\xi}_x\boldsymbol{\xi}_x^\top] = \boldsymbol{I}$ and $\mathbb{E}_{\xi_x}\left[\boldsymbol{\xi}_x^\top\boldsymbol{A}\boldsymbol{\xi}_x\right] = \text{Tr}[\boldsymbol{A}]$ for a symmetric matrix $\boldsymbol{A}$. Based on Eq. (25), the expectation of the empirical loss function $L(\widetilde{\boldsymbol{x}},\boldsymbol{y})$ can be simplified as

$$\mathbb{E}_{\xi_x}\left[L(\widetilde{\boldsymbol{x}},\boldsymbol{y})\right] \approx L(\boldsymbol{x},\boldsymbol{y}) + \frac{\sigma_{\text{input}}^2}{2}\text{Tr}\left[\nabla_{\boldsymbol{xx}}^2 L(\boldsymbol{x},\boldsymbol{y})\right]. \qquad (25)$$

It is seen from Eq. (25) that, from a statistical perspective, injecting noise into the input is equivalent to adding a regularization term $\frac{\sigma_{\text{input}}^2}{2}\text{Tr}\left[\nabla_{\boldsymbol{xx}}^2 L(\boldsymbol{x},\boldsymbol{y})\right]$ to the original loss function $L(\boldsymbol{x},\boldsymbol{y})$. The noise level σ_{input} acts as a learnable regularization parameter. Consequently, this formulation enhances the generalization ability of the network. The theoretical proof of the regularization equivalence in Eq. (25) was first established by Bishop [16], which can be extended to the approaches of injecting noise into the weight coefficients [17], labels [17,18], activation functions [13,38], and gradients [15,36,37]. For the noise-boosted

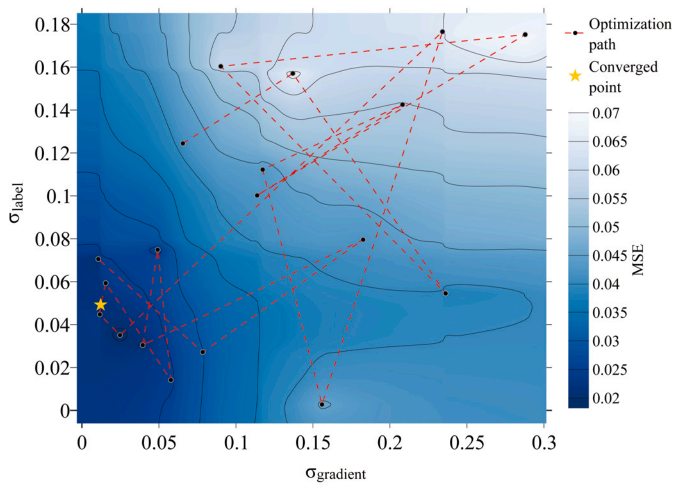

Fig. 3. Level curves for evaluating MSE of the sigmoid network as a function of hyperparameters σ_{label} and σ_{gradient}, with the dashed line indicating the convergence process of the hyperparameters towards the optimal non-zero levels of injected noise. The yellow colored star marks the final converged coordinate of $(\sigma_{\text{gradient}}, \sigma_{\text{label}}) = (0.012, 0.049)$ in the Bayesian optimization process.

activation functions indicated in Eqs. (10) and (13), we also theoretically proved the regularization equivalence for a small injected noise level [22].

Given different loss functions $L(\boldsymbol{x}, \boldsymbol{y})$ and the network output $f(\boldsymbol{x}; \boldsymbol{\theta})$, the trace of the Hessian $\nabla^2_{\boldsymbol{xx}} L(\boldsymbol{x}, \boldsymbol{y})$ in Eq. (25) consists of a positive term, $[\nabla_{\boldsymbol{x}} f(\boldsymbol{x}; \boldsymbol{\theta})]^2$, and a term involving $\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}; \boldsymbol{\theta})$, whose definiteness is not guaranteed [16–18]. However, in the presence of a large amount of training data, for example, for the loss functions such as MSE, the regularization term in Eq. (25) can be calculated as

$$\frac{\sigma^2_{\text{input}}}{2} \operatorname{Tr}\left[\nabla^2_{\boldsymbol{xx}} L(\boldsymbol{x}, \boldsymbol{y})\right] = \underbrace{\sigma^2_{\text{input}} \sum_{i=1}^{S} \nabla^2_{\boldsymbol{x}} f(\boldsymbol{x}(i); \boldsymbol{\theta})}_{P_1} + \underbrace{\sigma^2_{\text{input}} \sum_{i=1}^{S} [\boldsymbol{y}(i) - f(\boldsymbol{x}(i); \boldsymbol{\theta})] \nabla_{\boldsymbol{xx}} f(\boldsymbol{x}(i); \boldsymbol{\theta})}_{P_2}, \quad (26)$$

where the term P_1 is positive, and the second term P_2 vanishes as the network output $f(\boldsymbol{x}(i); \boldsymbol{\theta})$ converges to the conditional expectation of the target $\boldsymbol{y}(i)$ in the order of σ^2_{input} [16–18]. Or equivalently, the second term is a higher-order infinitesimal compared to the first term. Consequently, the first positive term P_1 primarily contributes to the regularization effect. It is evident that when the noise level $\sigma_{\text{input}} = 0$, the regularization term vanishes, resulting in the absence of any regularization effect on the generalization performance of network. However, when σ_{input} becomes very large, the penalty imposed by the regularization term on the loss function $L(\tilde{\boldsymbol{x}}, \boldsymbol{y})$ increases significantly, which, naturally, degrades the generalization performance of the network. It is only when σ_{input} is at an optimal value or in a range that the regularization term positively contributes to the generalization ability of the network, as demonstrated in Tables 1 and 2 for the function approximation fitting. Therefore, we refer to the enhancement of the generalization performance of networks through the approach of noise injection as a stochastic resonance effect in a broader sense.

The manifestation of the stochastic resonance effect, induced by the optimal noise intensity explained above, can also be clearly observed during the optimization process of a network for function fitting. Fig. 3 illustrates the level curves for evaluating MSE of the sigmoid network as a function of the hyperparameters σ_{label} defined in Eq. (4) and σ_{gradient} in Eq. (7). From Fig. 3, it is observed that, after 20 iterations of expectation improvement using the acquisition function defined in Eq. (21) via the TPE Bayesian optimization method, the convergence trajectory of the

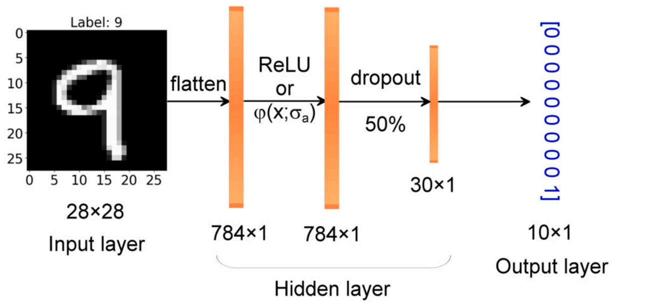

Fig. 4. Block diagram representations of the neural network architecture for MNIST classification.

hyperparameter values $(\sigma_{\text{gradient}}, \sigma_{\text{label}})$ (dashed line) ultimately reaches the optimal values of $(\sigma_{\text{gradient}}, \sigma_{\text{label}}) = (0.012, 0.049)$ marked by the star. The non-zero optimal coordinate $(\sigma_{\text{gradient}}, \sigma_{\text{label}})$ implies that the testing MSE increases for both high and low values of the noise levels σ_{label} and σ_{gradient}. This observation experimentally confirms the manifestation of the stochastic resonance effect [22,26,30,62] within the designed neural network optimized through the TPE Bayesian approach. Moreover, the non-zero level values in Figs. 2 and 3 demonstrate a strategic integration of noise injection into the neural network to enhance its generalization performance. The stochastic resonance effect of other noise injection approaches in the designed neural network is also validated by a non-zero optimal noise level, which accords well to the theoretical exploration of Eq. (25).

However, the assumption of an infinitesimal injected noise level in the proof of regularization equivalence, as presented in Eq. (25) [16,22], does not always hold. This is because the converged (local optimal) noise levels in the training of networks with noise-boosted activation functions, as shown in Tables 1 and 2, are often significantly larger than unity. Consequently, the aforementioned regularization equivalence indicated by Eq. (25), as well as its theoretical interpretation in terms of stochastic resonance, needs further elucidation. Therefore, a more extensive theoretical investigation is necessary to establish the link to stochastic resonance in designed neural networks with noise injection. The obtained results, along with the proof of regularization equivalence in Eq. (25), will provide a theoretical framework for the application of noise injection strategies in neural network training.

3.3. Classifications of gray images

Based on the analysis above, it is evident that noise injection methods are effective for approximating functions in neural networks. Next, we will employ these approaches to enhance the image classification performance of neural networks. The architecture of the neural network is illustrated in Fig. 4, where the activation function $\varphi(x)$ in the hidden layer can be selected from either ReLU or noise-boosted activations $\varphi(x; \sigma_a)$ defined in Eqs (14)-(17). The MNIST dataset comprises grayscale 28×28 images across 10 classes, with 60,000 training examples and 10,000 test images. For this simple image classification task, the fully connected network designed in Fig. 4 can achieve good classification accuracy. Therefore, we chose the architecture shown in Fig. 4 to compare the feasibility of noise injection methods. The designed network of Fig. 4 is trained for 50 epochs, with a batch size of 100, using the Adam optimizer [66] to optimize the weights with a learning rate $\gamma = 0.01$. The optimization ranges for the noise level parameters used here are the same as those chosen in the function approximation example above.

Without any noise injection, the testing accuracy of the neural network depicted in Fig. 4 is 93.53% when the activation function $\varphi(x)$ is set to ReLU. Table 3 presents the corresponding testing accuracies of the ReLU neural network with various noise injection methods applied to the MNIST dataset or in other placements of network. The values following the ± symbol denote one standard deviation computed across 10

Table 3
Testing accuracies of the neural network with various direct noise injection methods on the MNIST dataset at the corresponding optimal noise levels.

Position	Optimal noise level σ	Testing accuracy (%)
$\boldsymbol{x}$	1.02×10^{-2}	94.25 ± 0.40
$\boldsymbol{y}$	1.03	94.52 ± 0.34
$\boldsymbol{\theta}$	3.3×10^{-3}	94.86 ± 0.04
$\partial L/\partial \boldsymbol{\theta}$	1.21×10^{-2}	92.92 ± 0.09
$\partial L/\partial \boldsymbol{\theta}$, $\boldsymbol{x}$	1.42×10^{-3}, 5.11×10^{-2}	94.69 ± 0.03
$\boldsymbol{\theta}$, $\boldsymbol{y}$	6.44×10^{-3}, 1.77×10^{-1}	94.59 ± 0.09

Table 4
Testing accuracies of the neural network with noise injection in activations on the MNIST dataset at the corresponding optimal noise levels.

Activation function	Optimal noise level σ	Testing accuracy (%)
ReLU(σ_{neuron})	2.10	94.83 ± 0.11
GELU(σ_{neuron})	0.28	95.26 ± 0.39
GELU(σ_a)	1.11	95.83 ± 0.08
ExLU(σ_{neuron})	2.32	95.58 ± 0.08
ExLU(σ_a)	2.41	95.84 ± 0.12
RayLU(σ_{neuron})	3.49	95.47 ± 0.04
RayLU(σ_a)	6.24	95.62 ± 0.09
GELU(σ_a), $\boldsymbol{x}$	1.70×10^{-1}, 1.27×10^{-3}	95.72 ± 0.11
ExLU(σ_a), $\boldsymbol{x}$	5.11, 2.10×10^{-3}	95.82 ± 0.11
RayLU(σ_a), $\boldsymbol{x}$	6.99, 1.09×10^{-2}	95.63 ± 0.13

trials. This format is also used in the following sections unless otherwise stated. The results indicate that, besides the method that injects noise into the gradient, other noise injection approaches can enhance the testing accuracy of the designed neural network. Specifically, injecting noise directly into the weight $\boldsymbol{\theta}$ of a ReLU network yields an average classification accuracy 94.86% that exceeds the noise-free ReLU network by 1.33%.

Furthermore, we experimentally explore two major categories of noise injection methods to enhance the testing accuracy of the designed network: directly adding noise to the activation functions in the hidden layers and employing noise-boosted activation functions. As shown in Table 4, the rows listing the names of the activation functions with σ_{neuron} represent the results of directly injecting noise into the activation functions, as described in Eq. (6). In contrast, the rows that include the activation function names along with σ_a indicate the results obtained by optimizing the intrinsic hyperparameter σ_a of the noise-boosted activation functions. It is seen that GELU, ExLU, and RayLU networks, optimized for their noise level hyperparameters σ_a or with noise injected directly via Bayesian optimization, can achieve average testing accuracies larger than 95.0%, exceeding the noise-free ReLU network by approximately 2.0%. As indicated in Table 4, these experimental results further validate the effectiveness of noise injection methods in improving the performance of neural networks for image classification tasks.

Furthermore, we also explore the optimization of noise-boosted networks with injecting optimal noise into data $\boldsymbol{x}$, as shown in Table 4. However, the corresponding results indicate no further improvement in testing accuracy. Consequently, the adoption of neural networks integrated with a noise-boosted activation defined in Eqs (14)-(17) is demonstrated to be a more effective approach.

3.4. Classifications of color images

We further employed a lightweight ResNet-9 network [67], as illustrated in Fig. 5, to classify color images within the CIFAR-10 dataset. ResNet-9 is selected over ResNet-18 and more complex architectures due to constraints in computational resources and the objective of minimizing the size of the noise scale parameters. For the CIFAR-10 classification task, the floating point operations of ResNet-9 are approximately 30%

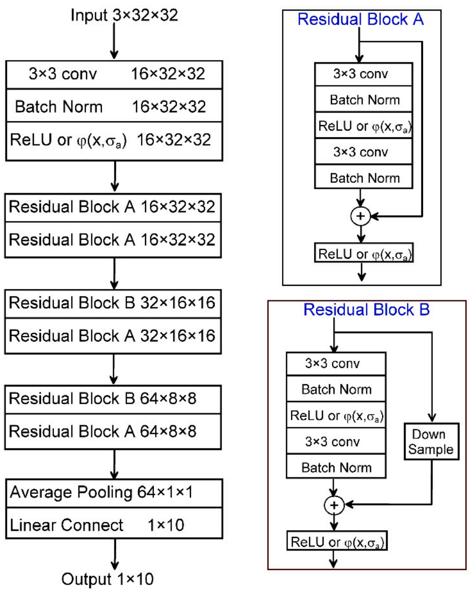

Fig. 5. Diagram representation of the architecture of the ResNet-9 network on the CIFAR-10 dataset.

of those of ResNet-18. The total memory of parameters of ResNet-9 is approximately 0.2 MB, which is significantly smaller than the 11.4 MB required for the parameters of ResNet-18 [67]. Furthermore, ResNet-9 with noise-boosted activation functions contains 13 noise scale parameters, while the noise-boosted ResNet-18 has 17.

As depicted in Fig. 5, one injected noise level is applied outside the residual blocks, while twelve injected noise levels are integrated within the residual blocks. The batch size is set to 100, and the Adam optimizer [66] is utilized to optimize the weights over 80 epochs. The initial learning rate is set at 0.001, decreasing to one-third of its previous value after 20 epochs. The ranges for the noise level parameters used here are also the same as those chosen in the function fitting example via Bayesian optimization, while for the SGD-based approach, the initial noise levels are randomly initialized as integers within the range $[2, 6]$. The experimental results, including testing accuracies of the ResNet-9 network with various activation functions, are summarized in Table 5 (source codes are provided in [61]). Here, we primarily investigate the optimization of noise injection into the activation functions of hidden layers, or of the intrinsic hyperparameter σ_a of noise-boosted activations. The combinations of noise injection methods are also considered in experiments. Furthermore, during back-propagation, the SGD method [22,30] can be employed to optimize the noise level σ_a. This parameter is initially set as a constant based on empirical selection and subsequently optimized alongside the network weights.

As indicated in Table 5, regardless of the optimization method employed, both the injection of noise into the hidden layer activation functions and the learning of the intrinsic hyperparameter σ_a for the noise-boosted activation achieve a 3% improvement in testing accuracy compared to the 84.48% accuracy achieved by the noise-free ReLU neural network. Specifically, the method of injecting noise to the input data of the GELU neural network results in a testing accuracy of (91.30 ± 4.74)%, a 6% improvement over the noise-free ReLU network. However, it is seen that this result is not stable, as evidenced by the large statistical variance.

As shown in Table 5, the Bayesian optimization approach for the intrinsic hyperparameter σ_a of the noise-boosted activation demonstrates greater applicability and robustness in improving the testing accuracy of the network. Furthermore, although the SGD-optimized network achieves comparable testing accuracies, the initial values of the hyperparameter σ_a must be determined empirically. Learning failures are also observed in networks employing activation functions such as

Table 5
Testing accuracies of the ResNet-9 network with various activations on the CIFAR-10 dataset by the Bayesian or SGD optimization methods.

Activation	Approach	Testing accuracy (%)
ReLU	SGD	84.48 ± 0.17
ReLU(σ_{neuron})	Bayesian	86.91 ± 0.65
GELU(σ_a)	SGD	87.63 ± 0.09
GELU(σ_a)	Bayesian	87.34 ± 0.30
RayLU(σ_a)	SGD	87.22 ± 0.05
RayLU(σ_a)	Bayesian	87.27 ± 0.27
ExLU(σ_a)	SGD	87.66 ± 0.43
ExLU(σ_a)	Bayesian	87.32 ± 0.44
$\boldsymbol{\theta}$	Bayesian	24.31 ± 2.95
$\partial L/\partial\boldsymbol{\theta}$	Bayesian	53.76 ± 9.92
$\boldsymbol{x}$, GELU(σ_a)	Bayesian	91.30 ± 4.74
$\boldsymbol{x}$, $\partial L/\partial\boldsymbol{\theta}$	Bayesian	61.49 ± 7.60
$\boldsymbol{\theta}$, GELU(σ_a)	Bayesian	30.74 ± 9.81

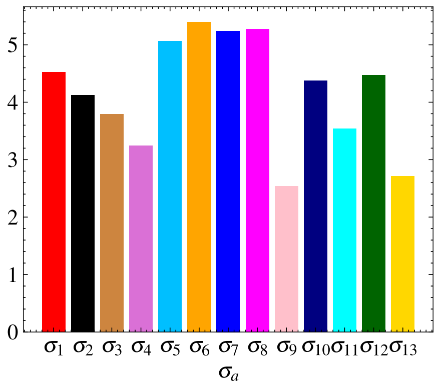

Fig. 6. Injected noise levels σ_a in the trained GELU ResNet-9 network on CIFAR-10 dataset.

ExLU, due to gradient explosion that occurs as σ_a decreases to very small values.

Fig. 6 illustrates the noise levels obtained after Bayesian optimization, corresponding to the 13 noise levels in the GELU neural network. These values represent the optimal results from 20 trials, as determined by the Gaussian surrogate model indicated in Eq. (21). Fig. 7 illustrates the importance scores of 13 injected noise levels (σ_a) in a trained GELU ResNet-9 network. We utilized the Fanova importance evaluator [69] to evaluate the impact of each noise level σ_a on the testing accuracy of the network, which corresponds to the proportion of the performance variance explained by each hyperparameter σ_a. As illustrated in Fig. 7, these variance contributions are represented as the importance scores, allowing for a clear comparison of the influence of each noise level on the network performance. Similar results for other neural networks are not shown here for brevity.

3.5. Image reconstruction

As illustrated in Fig. 8, we design a convolution autoencoder with six convolution layers and five layers that employ ReLU or noise-boosted activation functions for reconstructing images (source codes are provided in [61]). The architecture compresses the spatial dimensions of images from 224×224 to 56×56, enabling effective feature extraction and compression while preserving the essential image characteristics. The fingerprint verification competition (FVC2002) dataset [68] is utilized, where 320 images are divided into training and testing sets, respectively, in a $4:1$ ratio. The learning rate is set to 0.001, the model is trained for 300 epochs using the Adam optimizer [66] with a batch size of 128. For the noise-boosted activation function, the noise level parameter is selected from the range $[0, 6]$. For direct noise injection into gradients and weights, the noise level is chosen from the range $[0, 0.1]$, while for

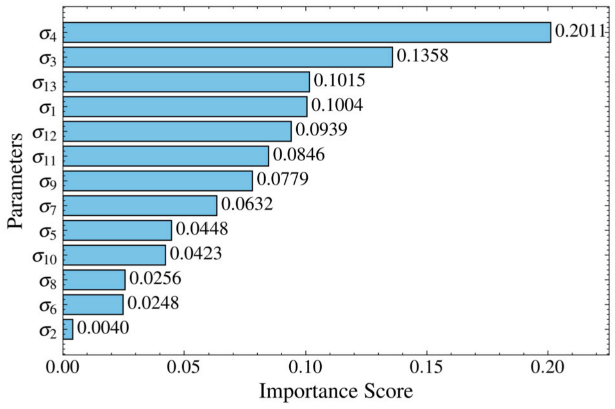

Fig. 7. Importance score of injected noise levels σ_a in the trained GELU ResNet-9 network on CIFAR-10 dataset.

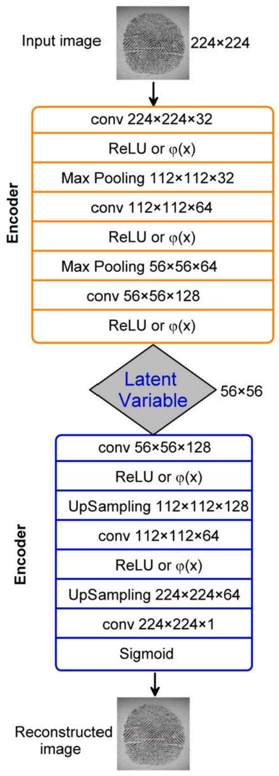

Fig. 8. Diagram representation of the architecture of the designed autoencoder on the FVC2002 dataset.

injecting noise into the input data and labels, the noise level is selected within the range $[0, 1]$. For the SGD-based approach, the initial noise levels are randomly initialized as integers 5 or 6.

The convolutional autoencoder in Fig. 8 was chosen for image reconstruction due to its effective architecture: Six convolutional layers can extract hierarchical spatial features, while ReLU and noise-boosted activations introduce non-linearity and potential robustness. Spatial dimension reduction from 224×224 to 56×56 enables efficient feature compression, retaining essential image characteristics and reducing computational cost.

Fig. 9 (a) describes the testing loss curve over 300 epochs for the noise-free ReLU autoencoder (training loss curves are omitted for clarity, as they closely overlap with the corresponding testing loss curves). Here, the loss function is defined as the MSE between the input and the reconstructed images. As observed in Fig. 9 (a), the loss curves of the noise-free ReLU autoencoder exhibit an oscillatory behavior, which can be

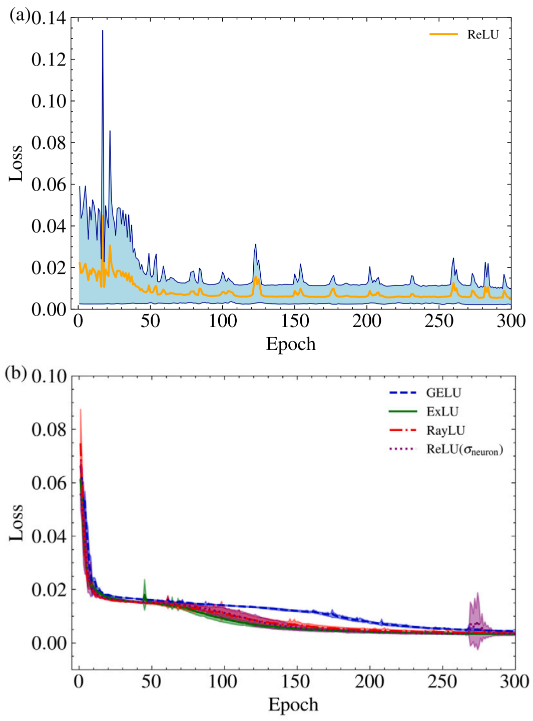

Fig. 9. Testing loss curves of (a) the ReLU autoencoder without Bayesian optimization of noise, and (b) the ReLU autoencoder with optimized noise within the ReLU activation function and autoencoders with GELU, RayLU, or ExLU activation functions. The shaded region represents an error bar corresponding to one standard deviation.

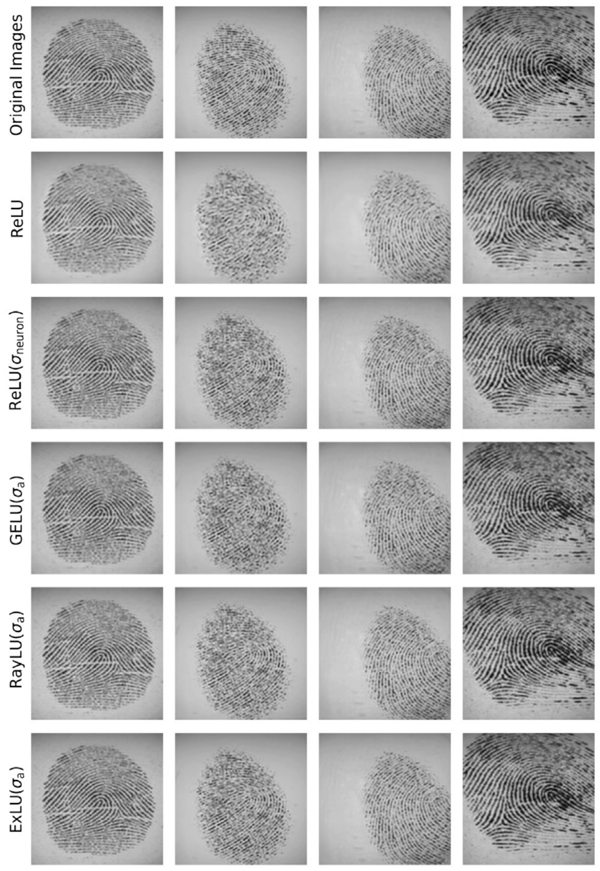

Fig. 10. Reconstructed images by noise-free ReLU and noise-booseted autoencoders for two original image samples of the FVC2002 dataset. Here, noise-booseted autoencoders are trained by the Bayesian optimization method.

Table 6
Training and testing losses of the autoencoder with various activations on the FVC2002 fingerprints dataset by the Bayesian or SGD optimization methods.

Activation	Approach	Training loss (10^{-3})	Testing loss (10^{-3})
ReLU	SGD	4.68 ± 4.04	5.28 ± 4.41
ReLU (σ_{neuron})	Bayesian	2.98 ± 0.49	3.36 ± 0.51
GELU (σ_a)	Bayesian	3.94 ± 0.05	4.43 ± 0.05
RayLU(σ_a)	Bayesian	3.20 ± 0.32	3.56 ± 0.26
ExLU(σ_a)	Bayesian	2.92 ± 0.08	3.28 ± 0.04
GELU(σ_a), $\boldsymbol{x}$	Bayesian	3.94 ± 0.23	4.42 ± 0.26
RayLU(σ_a), $\boldsymbol{x}$	Bayesian	3.04 ± 0.09	3.36 ± 0.11
ExLU(σ_a), $\boldsymbol{x}$	Bayesian	3.22 ± 0.08	3.60 ± 0.12
GELU(σ_a)	SGD	3.86 ± 0.45	4.36 ± 0.48
RayLU(σ_a)	SGD	4.26 ± 0.36	4.78 ± 0.45
ExLU(σ_a)	SGD	4.10 ± 0.52	4.56 ± 0.60

Table 7
Training time and GPU memory usage of noise-boosted autoencoders on the FVC2002 fingerprint dataset.

Activation	Approach	Training time (s)	GPU memory(MB)
ReLU	SGD	187.64	8,419
ReLU (σ_{neuron})	Bayesian	215.72	13,361
GELU (σ_a)	Bayesian	260.53	13,909
RayLU(σ_a)	Bayesian	257.47	14,301
ExLU(σ_a)	Bayesian	239.30	20,381
GELU(σ_a), $\boldsymbol{x}$	Bayesian	243.37	14,305
RayLU(σ_a), $\boldsymbol{x}$	Bayesian	260.49	15,491
ExLU(σ_a), $\boldsymbol{x}$	Bayesian	250.10	12,345
GELU(σ_a)	SGD	265.37	17,049
RayLU(σ_a)	SGD	266.25	14,305
ExLU(σ_a)	SGD	232.58	12,739

attributed to the presence of four distinct types of sensor fingerprint images in the FVC2002 dataset [68]. Conversely, as illustrated in Fig. 9 (b) (dotted line), the testing loss of the ReLU autoencoder enhanced with Bayesian-optimized noise levels $\sigma_{\text{neuron}} = \{0.29, 0.65, 0.42, 0.11, 0.77\}$ across five hidden layers achieves lower loss values, and effectively restrains the oscillations. Fig. 9(b) also illustrates the testing loss curves for the designed autoencoders employing GELU, RayLU, and ExLU activation functions. For clarity, the training loss curves are omitted, as they are close to the corresponding testing loss curves. These noise-boosted autoencoders, incorporating Bayesian-optimized noise levels across five hidden layers, achieve demonstrably lower loss values compared to the noise-free ReLU autoencoder. Representative samples of the reconstructed images obtained from the designed autoencoders are illustrated in Fig. 10. In particular, the ExLU autoencoder, achieving the testing MSE of $(3.28 \pm 0.04) \times 10^{-3}$, outperforms the ReLU autoencoder. This result demonstrates the efficacy of noise-boosted neuron models in unsupervised machine learning applications.

Furthermore, it is seen in Table 6 that combining noise injection at the input and activations of GELU, RayLU, and ExLU networks does not improve reconstruction error. In addition, the SGD method can not decrease the reconstruction errors of the GELU, RayLU, and ExLU networks, as shown in Table 6.

For this complex fingerprint image reconstruction task, Table 7 compares training time and GPU memory usage across autoencoders with different activation functions for both Bayesian optimization and SGD method. As shown in Table 7, autoencoders based on the noise-boosted activation functions have higher training time and GPU memory consumption than the ReLU-based autoencoder, due to the introduction of learnable noise level parameters at each layer. However, when employing noise-boosted activations, the per-trial differences in training time and GPU memory consumption between Bayesian optimization and SGD are minor. Considering the improvements in performance, the autoencoder constructed with noise-boosted activation functions is still of practical significance.

4. Discussion

In this study, we employed the Bayesian optimization to evaluate various noise injection methods in neural networks for applications in function approximation, image classification, and image reconstruction. The obtained results demonstrate that injecting noise into input data or different placements of the neural network can enhance performance compared to noise-free networks. The Bayesian optimization approach effectively finds the optimal noise levels for various noise injection methods within constrained ranges.

Furthermore, we provide a theoretical derivation of the Taylor expansion for the loss function of designed networks under the assumption of small injected noise. Then, the benefits of non-zero optimal noise injection during neural network training were theoretically analyzed in relation to the stochastic resonance mechanism. The statistical results in Tables 1–6 and Figs. 2–9 confirm the effectiveness of noise injection approaches in the designed networks.

While the direct injection of reparameterized noise samples requires a certain amount of statistical computation, it offers less operational simplicity compared to methods that enhance network performance by embedding noise levels within the noise-boosted activation functions of hidden layers, as demonstrated in the design of the ResNet-9 network for image classification and the autoencoder for image reconstruction. Consequently, if the neural network architecture is based on activations without learnable parameters, the direct injection of reparameterized noise samples can serve as a practical strategy. However, when the network design is adaptable, our results indicate that directly constructing neural networks with noise-boosted activation functions is more practical, as such networks can achieve superior and stable performance compared to noise-free ones through Bayesian optimization. This study also explores the physical mechanisms of noise injection in activations and the optimization of noise in machine learning applications, while opening the possibility to further research in areas such as time series forecasting and generative adversarial networks.

Some open questions still need to be addressed in future studies. In this paper, we theoretically investigated various types of noise within the activation functions and derived the corresponding noise-boosted activation models of Eqs. (10) and (13), while only considering Gaussian noise in the direct injection methods for data, weights, gradients, and other components. A valuable direction for future research is to explore the effects of different noise types in these direct injection methods, with the aim of further enhancing network performance through Bayesian optimization. It is worth investigating whether the combination of different noise types in direct injection methods and noise-boosted activation models could lead to further improved and more robust network performance.

In addition, the noise-boosted activations in Eqs. (14)–(17) degenerate to ReLU when the hyperparameter $\sigma_a = 0$. Therefore, we consider the comparison of these noise-boosted networks with the noise-free ReLU network as a form of ablation study [74], as it effectively ablates the learnability introduced by the noise injection into the activation functions. More exploration is possible with ablation studies to further assess the effectiveness of noise injection in such conditions. It is also important to note that the computational cost of Bayesian optimization increases significantly as the dimensionality of the noise scale parameters grows, particularly when applying noise injection to enhance large-scale neural networks. For instance, we experimentally compared the testing accuracy of the ReLU ResNet-34 network with a ResNet-34 network employing noise-boosted activations on the CIFAR-10 dataset. The noise-boosted ResNet-34 networks achieved testing accuracies approximately 90%, which is comparable to that of the noise-free ResNet-34 network (code sources in [61]). Moreover, the designed noise-boosted ResNet-34 model incorporates 33 noise scale parameters, which considerably increases the computational time required for Bayesian optimization, without yielding any further improvement in network performance. Using the SGD optimizer, the GPU and CPU memory usage of a GELU ResNet-34 model were 598.95 MB and $1,474.08$ MB, respectively, with a total training time of $3,750.91$ s. For a single Bayesian optimization trial, the GPU memory usage was 506.73 MB, the CPU memory usage was $1,416.42$ MB, and the training time was $3,549.05$ s. In contrast, the ReLU ResNet-34 required only 232.73 MB of GPU memory, $1,365.77$ MB of CPU memory, and $1,611.52$ s for training. This presents a significant limitation for the application of noise-boosted neural networks in these conditions. Therefore, future research may be useful to develop more scalable and computationally efficient optimization methods for utilizing the noise injection approach in large-scale neural networks. For example, evolutionary algorithms can simulate natural selection to evolve populations of hyperparameter configurations [48]. Meta-learning approaches can utilize experience from hyperparameter optimization across related tasks to efficiently optimize hyperparameters in new tasks [21,49,52]. Moreover, transfer learning offers another promising direction [79], where optimal hyperparameters selected on large-scale datasets can be transferred to new tasks as initial configurations or priors in Bayesian optimization.

CRediT authorship contribution statement

Caimin An: Writing – original draft, Software, Data curation. **Fabing Duan:** Writing – original draft, Formal analysis, Conceptualization. **François Chapeau-Blondeau:** Writing – review & editing, Methodology. **Derek Abbott:** Writing – review & editing, Methodology.

Declaration of competing interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service and/or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled.

Acknowledgements

This work is supported by the Natural Science Foundation of Shandong Province of China (No. ZR2021MF051). DA gratefully acknowledges support from the Australian Research Council (DP240103404 and FL240100217).

Data availability

Data will be made available on request.

References

[1] M.M. Jevtić, Noise as a diagnostic and prediction tool in reliability physics, Microelectron. Reliab. 35 (3) (1995) 455–477.

[2] G.P. Harmer, B.R. Davis, D. Abbott, A review of stochastic resonance: circuits and measurement, IEEE Trans. Instrum. Meas. 51 (2) (2002) 299–309.

[3] K. Wiesenfeld, F. Moss, Stochastic resonance and the benefits of noise: from ice ages to crayfish and SQUIDs, Nature 373 (6509) (1995) 33–36.

[4] R.A. Wannamaker, S.P. Lipshitz, J. Vanderkooy, Stochastic resonance as dithering, Phys. Rev. E 61 (1) (2000) 233–236.

[5] L. Gammaitoni, Stochastic resonance and the dithering effect in threshold physical systems, Phys. Rev. E 52 (5) (1995) 4691–4698.

[6] J. Liu, Z. Li, Y. Gao, Q. Miao, L. Yang, D. Wang, P. Qi, Adaptive stochastic resonance enhanced weak linear frequency modulated signal perception in low signal-to-noise ratio environments, Meas. Sci. Technol. 36 (2025) 016102.

[7] C. Yang, Z. Qiao, R. Zhu, X. Xu, Z. Lai, S. Zhou, An intelligent fault diagnosis method enhanced by noise injection for machinery, IEEE Trans. Instrum. Meas. 72 (2023) 3534011.

[8] L. Chen, K. An, D. Huang, X. Wang, M. Xia, S. Lu, Noise-boosted convolutional neural network for edge-based motor fault diagnosis with limited samples, IEEE Trans. Ind. Inform. 19 (9) (2022) 9491–9502.

[9] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, N. Sebe, Binary neural networks: a survey, Pattern Recognit. 105 (2020) 107281.

[10] N.G. Stocks, Suprathreshold stochastic resonance in multi-level threshold systems, Phys. Rev. Lett. 84 (11) (2000) 2310–2313.

[11] W. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133.

[12] J. Sietsma, R.J.F. Dow, Creating artificial neural networks that generalize, Neural Netw. 4 (1) (1991) 67–79.

[13] Y. Grandvalet, Anisotropic noise injection for input variables relevance determination, IEEE Trans. Neural Netw. 11 (6) (2000) 1201–1212.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.

[15] A. Orvieto, A. Raj, H. Kersting, F. Bach, Explicit regularization in overparametrized models via noise injection, in: International Conference on Artificial Intelligence and Statistics (PMLR, 2023, pp. 7265–7287.

[16] C.M. Bishop, Training with noise is equivalent to Tikhonov regularization, Neural Comput. 7 (1) (1995) 108–116.

[17] G. An, The effects of adding noise during backpropagation training on a generalization performance, Neural Comput. 8 (3) (1996) 643–674.

[18] Y. Grandvalet, S. Canu, S. Boucheron, Noise injection: theoretical prospects, Neural Comput. 9 (5) (1997) 1093–1108.

[19] J. Bohorquez, M.F. Lambert, B. Alexander, A.R. Simpson, D. Abbott, Stochastic resonance enhancement for leak detection in pipelines using fluid transients and convolutional neural networks, J. Water Resour. Plan. Manag. 148 (3) (2022) 04022001.

[20] A.K. Seghouane, Y. Moudden, G. Fleury, Regularizing the effect of input noise injection in feedforward neural networks training, Neural Comput. Appl. 13 (2004) 248–254.

[21] M. Igl, K. Ciosek, Y. Li, S. Tschiatschek, C. Zhang, S. Devlin, K. Hofmann, Generalization in reinforcement learning with selective noise injection and information bottleneck, Adv. Neural Inf. Process. Syst. 32 (2019) 13979–13991.

[22] S. Bai, F. Duan, F. Chapeau-Blondeau, D. Abbott, Generalization of stochastic-resonance-based threshold networks with Tikhonov regularization, Phys. Rev. E 106 (1) (2022) L012101.

[23] L. Duan, F. Duan, F. Chapeau-Blondeau, D. Abbott, Noise-boosted backpropagation learning of feedforward threshold neural networks for function approximation, IEEE Trans. Instrum. Meas. 70 (2021) 1010612.

[24] A. Orvieto, H. Kersting, F. Proske, F. Bach, A. Lucchi, Anticorrelated noise injection for improved generalization, in: International Conference on Machine Learning (PMLR, 2022, pp. 17094–17116.

[25] M. Ferianc, O. Bohdal, T.M. Hospedales, M.R.D. Rodrigues, Navigating noise: a study of how noise influences generalisation and calibration of neural networks, Trans. Mach. Learn. Res. (2024) 1–44.

[26] B. Kosko, K. Audhkhasi, O. Osoba, Noise can speed backpropagation learning and deep bidirectional pretraining, Neural Netw. 129 (2020) 359–384.

[27] K. Audhkhasi, O. Osoba, B. Kosko, Noise-enhanced convolutional neural networks, Neural Netw. 78 (2016) 15–23.

[28] O. Adigun, B. Kosko, Noise-boosted recurrent backpropagation, Neurocomputing 559 (2023) 126438.

[29] M.D. McDonnell, D. Abbott, What is stochastic resonance? Definitions, misconceptions, debates, and its relevance to biology, PLoS Comput. Biol. 5 (5) (2009) e1000348.

[30] Y. Ren, F. Duan, F. Chapeau-Blondeau, D. Abbott, Self-gating stochastic-resonance-based autoencoder for unsupervised learning, Phys. Rev. E 110 (1) (2024) 014107.

[31] V. Thomas, F. Pedregosa, B. Merri"enboer, P.A. Manzagol, Y. Bengio, N.L. Roux, On the interplay between noise and curvature and its effect on optimization and generalization, in: International Conference on Artificial Intelligence and Statistics (PMLR, 2020, pp. 3503–3513.

[32] S.H. Lim, N.B. Erichson, L. Hodgkinson, M.W. Mahoney, Noisy recurrent neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 5124–5137.

[33] D. Hendrycks, K. Gimpel, Gaussian error linear units (GELUs), arXiv preprint, arXiv: 1606.08415, 2016.

[34] S. Elfwing, E. Uchibe, K. Doya, Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, Neural Netw. 107 (2018) 3–11.

[35] A. Apicella, F. Donnarumma, F. Isgrø, R. Prevete, A survey on modern trainable activation functions, Neural Netw. 138 (2021) 14–32.

[36] T. Liu, Y. Li, S. Wei, E. Zhou, T. Zhao, Noisy gradient descent converges to flat minima for nonconvex matrix factorization, in: International Conference on Artificial Intelligence and Statistics (PMLR, 2021, pp. 1891–1899.

[37] A. Camuto, M. Willetts, U. Simsekli, S.J. Roberts, C.C. Holmes, Explicit regularisation in Gaussian noise injections, Adv. Neural Inf. Process. Syst. 33 (2020) 16603–16614.

[38] C. Gulcehre, M. Moczulski, M. Denil, Y. Bingio, Noisy activation functions, arXiv preprint, arXiv:1603.00391, 2016.

[39] K. Shridhar, J. Lee, H. Hayashi, P. Mehta, B.K. Iwana, S. Kang, S. Uchida, S. Ahmed, A. Dengel, Probact: a probabilistic activation function for deep neural networks, arXiv preprint, arXiv:1905.10761, 2019.

[40] Z. He, A.S. Rakin, D. Fan, Parametric noise injection: trainable randomness to improve deep neural network robustness against adversarial attack, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 588–597.

[41] B. Erichson, S.H. Lim, W. Xu, F. Utrera, Z. Cao, M. Mahoney, Noisymix: boosting model robustness to common corruptions, in: International Conference on Artificial Intelligence and Statistics (PMLR, 2024, pp. 4033–4041.

[42] Z. Liu, G. Gagnon, S. Venkataramani, L. Liu, Enhance DNN adversarial robustness and efficiency via injecting noise to non-essential neurons, arXiv preprint, arXiv: 2402.04325, 2024.

[43] N. Ye, L. Cao, L. Yang, Z. Zhang, Z. Fang, Q. Gu, G.Z. Yang, Improving the robustness of analog deep neural networks through a Bayes-optimized noise injection approach, Commun. Eng. 2 (1) (2023) 25.

[44] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4401–4410.

[45] R. Feng, D. Zhao, Z.J. Zha, Understanding noise injection in GANs, in: International Conference on Machine Learning (PMLR, 2021, pp. 3284–3293.

[46] S.H. Hong, J.W. Jeong, Dynamic noise injection for facial expression recognition in-the-wild, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023, pp. 5709–5715.

[47] Z. Zhang, J. Jiang, M. Chen, Z. Wang, Y. Peng, Z. Yu, A novel noise injection-based training scheme for better model robustness, arXiv preprint, arXiv:2302.10802, 2023.

[48] L. Xiao, Z. Zhang, K. Huang, J. Jiang, Y. Peng, Noise optimization in artificial neural networks, IEEE Trans. Autom. Sci. Eng. 22 (2025) 2780–2793.

[49] P.I. Frazier, A tutorial on Bayesian optimization, arXiv preprint, arXiv:1807.02811, 2018.

[50] J. Snoek, H. Larochelle, R.P. Adams, Practical Bayesian optimization of machine learning algorithms, Adv. Neural Inf. Process. Syst. 25 (2012) 2951–2959.

[51] X. Yuan, J. Li, E.E. Kuruoglu, Uncertainty quantification with noise injection in neural networks: a Bayesian perspective, arXiv preprint, arXiv:2501.12314, 2025.

[52] B.N. Slautin, Y. Liu, J. Dec, V.V. Shvartsman, D.C. Lupascu, M.A. Ziatdinov, S.V. Kalinin, Measurements with noise: Bayesian optimization for cooptimizing noise and property discovery in automated experiments, Dig. Discov. 4 (4) (2025) 1066–1074.

[53] Z. Shi, Z. Liao, H. Tabata, Enhancing performance of convolutional neural network-based epileptic electroencephalogram diagnosis by asymmetric stochastic resonance, IEEE J. Biomed. Health Inform. 27 (9) (2023) 4228–4239.

[54] Z. Shi, Z. Liao, H. Tabata, Boosting learning ability of overdamped bistable stochastic resonance system based physical reservoir computing model by time-delayed feedback, Chaos Solitons Fractals 161 (2022) 112314.

[55] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533–536.

[56] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the 27th International Conference on Machine Learning (ICML, 2010, pp. 807–814.

[57] J. Bergstra, R. Bardenet, Y. Bengio, B. K'egl, Algorithms for hyper-parameter optimization, Adv. Neural Inf. Process. Syst. 24 (2011) 2546–2554.

[58] D.C. Liu, J. Nocedal, On the limited memory BFGS method for large scale optimization, Math. Program. 45 (1) (1989) 503–528.

[59] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: a next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2623–2631.

[60] M. Balandat, B. Karrer, D. Jiang, S. Daulton, B. Letham, A.G. Wilson, E. Bakshy, BoTorch: a framework for efficient Monte-Carlo Bayesian optimization, Adv. Neural Inf. Process. Syst. 33 (2020) 21524–21538.

[61] https://github.com/resonance-dfb.

[62] R. Benzi, A. Sutera, A. Vulpiani, The mechanism of stochastic resonance, J. Phys. A, Math. Gen. 14 (11) (1981) L453–L457.

[63] Y. Kang, Y. Fu, Y. Chen, Signal-to-noise gain of an adaptive neuron model with gamma renewal synaptic input, Acta Mech. Sin. 38 (2022) 521347.

[64] Z. Liao, K. Ma, M.S. Sarker, H. Yamahara, M. Seki, H. Tabata, Overdamped Ising machine with stochastic resonance phenomena in large noise condition, Nonlinear Dyn. 112 (11) (2024) 8967–8984.

[65] J. Liu, Z. Qiao, X. Ding, B. Hu, C. Zang, Stochastic resonance induced weak signal enhancement over controllable potential-well asymmetry, Chaos Solitons Fractals 146 (2021) 110845.

[66] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2015.

[67] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on ImageNet classification, in: IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.

[68] D. Maio, D. Maltoni, R. Cappelli, J.L. Wayman, A.K. Jain, FVC2002: second fingerprint verification competition, in: International Conference on Pattern Recognition, 2002, pp. 811–814.

[69] F. Hutter, H. Hoos, K. Leyton-Brown, An efficient approach for assessing hyperparameter importance, in: Proceedings of the 31st International Conference on Machine Learning, vol. 30, 2014, pp. 754–762.

[70] D. Rousseau, F. Chapeau-Blondeau, Noise-improved Bayesian estimation with arrays of one-bit quantizers, IEEE Trans. Instrum. Meas. 56 (6) (2007) 2658–2662.

[71] B. Andøand, S. Graziani, Adding noise to improve measurement, IEEE Instrum. Meas. Mag. 4 (3) (2001) 24–30.
[72] M.D. McDonnell, N.G. Stocks, C.E.M. Pearce, D. Abbott, Stochastic Resonance: From Suprathreshold Stochastic Resonance to Stochastic Signal Quantization, Cambridge University Press, Cambridge, 2008.
[73] X. Liu, L. Duan, F. Duan, F. Chapeau-Blondeau, D. Abbott, Enhancing threshold neural network via suprathreshold stochastic resonance for pattern classification, Phys. Lett. A 403 (2021) 127387.
[74] R. Meyes, M. Lu, C. Waubert de Puiseau, T. Meisen, Ablation studies in artificial neural networks, arXiv preprint, arXiv:1901.08644, 2019.
[75] M. Ozer, M. Perc, M. Uzuntarla, Stochastic resonance on Newman–Watts networks of Hodgkin–Huxley neurons with local periodic driving, Phys. Lett. A 373 (10) (2009) 964–968.
[76] W. Zhang, P. Shi, M. Li, D. Han, A novel stochastic resonance model based on bistable stochastic pooling network and its application, Chaos Solitons Fractals 145 (2021) 110800.
[77] E.P. Devine, Phase stochastic resonance in coherent QKD signal transmission with a crosstalk noise in multicore fiber, Phys. Lett. A 456 (2022) 128531.
[78] M. Perc, Stochastic resonance on weakly paced scale-free networks, Phys. Rev. E 78 (3) (2008) 036105.
[79] Y. Kim, J.W. Soh, G.Y. Park, N.I. Cho, Transfer learning from synthetic to real-noise denoising with adaptive instance normalization, in: Proc. IEEE/CVF Conf. Comput. Vision Pattern Recognit. (CVPR), 2020, pp. 3482–3492.