Journal of Engineering and Technology for Industrial Applications

ITEGAM-JETIA

Manaus, v.11 n.56, p.20-28. November/December., 2025.
DOI: https://doi.org/10.5935/jetia. v11i56.2156

ISSN ONLINE: 2447-0228

RESEARCH ARTICLE OPEN ACCESS

A NOVEL BELLMAN-INSPIRED GATED ACTIVATION MECHANISM FOR DYNAMIC NEURAL NETWORKS

Jincheng Zhang[1]

[1]Faculty of Science and Technology, Rajabhat Maha Sarakham University, Maha Sarakham 44000, Thailand.

[1]http://orcid.org/0009-0005-1860-0009

Email: zjc1639834588@gmail.com

ARTICLE INFO

Article History
Received: June 2, 2025.
Revised: September 20, 2025.
Accepted: November 1, 2025.
Published: November 30, 2025.

Keywords:
Bellman Equation,
Gated Activation Function,
Dynamic Neural Networks,
Context-Aware Activation,
Modular Activation Mechanism.

ABSTRACT

In deep learning, activation function, as the core mechanism of nonlinear transformation, determines the upper limit of the model's expressive power. However, traditional activation functions are mostly static, point-by-point transformations, lacking the ability to respond to and adjust input structures. This paper proposes a "gated activation mechanism" based on the idea of the Bellman equation in reinforcement learning, which introduces state valuation and future reward modeling into the neural network activation mechanism. Experiments were conducted by replacing the activation method in the standard multilayer perceptron (MLP) structure. The results show that this mechanism brings stable performance improvements without increasing the inference overhead. More importantly, this activation method has good versatility as a module and can be widely embedded in a variety of deep neural networks, providing a new paradigm for activation function design.

1. INTRODUCTION

As the foundation of deep learning, the performance of neural networks depends largely on the design of activation functions [1-6]. The main function of activation functions is to enable neural network models to approximate complex nonlinear functions and have stronger expressive power by introducing nonlinear factors [7-12]. At present, widely used activation functions include ReLU (rectified linear unit), GELU (Gaussian error linear unit), Swish, etc. These functions have shown good stability and convergence in practical tasks [13-18]. However, it should be noted that these mainstream activation functions are essentially static single-point mappings, whose outputs only depend on the current input value and lack the ability to dynamically understand and adjust input features in a broader context. This static characteristic may, to some extent, limit the ability of neural networks to model complex spatiotemporal patterns and multi-level semantics [19-24].

At the same time, the Bellman equation in reinforcement learning provides a completely different perspective [25-34]. By describing the recursive relationship between the current state value and the future expected value, the Bellman equation effectively describes how the system adjusts the current behavior strategy based on future feedback during the decision-making process. The core features of this mechanism are dynamics and recursion, that is, the current output is not only based on the current input state, but also related to the potential "value" in the future. This concept leads to the following thinking: Is it possible to build a similar mechanism in neural networks so that the activation function is no longer an isolated static operation, but a dynamic mapping with prediction, selection and context awareness?

Based on the above ideas, this paper proposes a gated activation mechanism that integrates the idea of Bellman equation. In this mechanism, we introduce the modeling of the input state (i.e., "current state") and its potential contribution in the feature space (i.e., "future value"), and use Bellman-type gating functions to weight and combine the two to achieve a dynamic activation process. In other words, the output of the activation function no longer depends solely on the current value, but is dynamically adjusted according to the combination of the "current performance" and "future potential" of the input. This mechanism not only enhances the expressive power of

the activation process, but also provides new ideas for building neural networks with long-term dependencies and information sorting capabilities. To verify the effectiveness of this mechanism, we conducted experiments using the classic multi-layer perceptron (MLP) structure. The results show that compared with the traditional ReLU activation function, the Bellman gate activation function proposed in this paper has achieved slight but stable performance improvements in multiple evaluation indicators such as accuracy, precision, recall and F1 score. At the same time, this method improves the nonlinear modeling ability of the model while maintaining computational efficiency, and almost does not cause additional overhead to the inference time. More importantly, the gate mechanism is a module with diverse functions, independent structure and controllable parameters. It can be seamlessly extended to complex architectures such as convolutional neural networks (CNNs), graph neural networks (GNNs), and even future Transformers, improving their dynamic adaptability and feature representation capabilities. In summary, the main contribution of this paper is that a general gated activation mechanism is proposed based on Bellman's ideas. By introducing the "state value" dynamic feedback structure, this mechanism effectively improves the nonlinear modeling ability and contextual adaptability of neural networks, providing a new and promising research direction for activation function design.

II. RELATED WORKS

Activation functions are an important part of neural networks and have gone through multiple stages of development and innovation. The earliest activation functions are represented by S-type functions and Tanh functions. These functions give the neuron output nonlinear characteristics by mapping the input to a finite interval, thereby improving the network's ability to learn complex functions. However, the S-type function has a gradient saturation problem, which causes the gradient to disappear during the back-propagation process, reducing the learning efficiency of the deep network. To solve these problems, researchers proposed the ReLU (Rectified Linear Unit) activation function, which replaces the traditional nonlinear mapping with a simple linear truncation form, greatly reducing the gradient vanishing problem and improving computational efficiency. Due to its excellent performance, the ReLU activation function has become the mainstream in the field of deep learning.

Later, with the deepening of research, more improved activation functions emerged, such as Leaky ReLU and PReLU, which solve the "dead neuron" problem of ReLU by introducing negative semi-axis linear responses. In recent years, adaptive activation functions such as Swish and Mish have further improved the network's representation ability and generalization performance by utilizing nonlinear transformations and learnable parameters. These activation functions often take into account the characteristics of smoothness, nonlinearity, and gradient stability, thereby greatly improving the learning effect and final performance of deep neural networks. Generally speaking, the design of activation functions mainly focuses on improving the gradient propagation path, enhancing the nonlinear representation ability and improving the generalization performance of the model, but many activation functions are still static functions of single-point mapping and lack the ability to dynamically adjust the input information.

At the same time, the gating mechanism, as an effective tool for dynamically adjusting the information flow, has received widespread attention in the design of neural network structures. The classic long short-term memory network (LSTM) introduces input gates, forget gates and output gates, and selectively controls the retention and discarding of information through gating units, which greatly improves the gradient disappearance and information forgetting problems in traditional long-sequence recurrent neural networks. Similarly, the attention mechanism of the Transformer model models the importance of each element in the sequence through a dynamic weighting scheme of query key values, realizes adaptive aggregation of information, and greatly improves its performance in fields such as natural language processing and computer vision. The success of these gating mechanisms and attention mechanisms highlights the importance of dynamic adjustment and context dependency in deep learning models and inspires the dynamic design of activation functions.

In addition, as the theoretical basis of the field of reinforcement learning, the Bellman equation systematically describes the recursive relationship between the current state and the future expected value in the decision-making process, forming a mathematical framework for dynamic programming. The core idea is to adjust the current decision by predicting future rewards, which reflects a deep understanding of the dynamic dependence of time and state. Although the Bellman equation is widely used in reinforcement learning, it has not been systematically studied and applied in the field of neural network activation function design. The existing activation functions have not yet incorporated concepts such as dynamic weight adjustment and state value fusion, and most of them are in the form of static functions, which limits the ability to deeply control the information flow of the activation function.

In response to this situation, this paper first proposed to introduce the core concept of the Bellman equation into the design of the activation mechanism, and constructed a new type of gated activation unit-Bellman Gate Unit (BGU). This unit realizes dynamic adjustment of the activation output by fusing the weight information of the current input state and the future prediction features. BGU not only breaks the limitations of the single-point mapping of traditional activation functions, but also integrates the idea of recursive value evaluation in reinforcement learning, bringing a new dynamic activation paradigm to neural networks. This innovative design is expected to inject more contextual awareness and dynamic adaptability into the activation function, and further improve the performance of neural network models in complex tasks.

III. METHOD DESIGN: BELLMAN ACTIVATION MECHANISM

In this study, we designed an innovative activation mechanism called "Bellman Activation Mechanism", the core concept of which is derived from the Bellman equation in reinforcement learning. This mechanism aims to break the limitation of traditional activation functions as single-point static mappings, and by introducing dynamic gating strategies, the activation function can dynamically adjust the input information and perceive the context, thereby improving the expressiveness and adaptability of the neural network. Specifically, this mechanism includes the following key components: First, the state estimation layer. This layer is responsible for building a rich and multi-dimensional intermediate representation based on the input data, which we call the current "state". The state here is not just a simple mapping of a single input value, but a comprehensive characterization of the overall input characteristics through multi-layer feature

transformation and fusion. Through the state estimation layer, the network can capture the local and global information of the input and form a comprehensive cognitive basis for the current environment.

The second is the reward estimation layer. The task of this layer is to calculate an immediate "reward" estimate for each input state. The concept of reward comes from the feedback mechanism in reinforcement learning, reflecting the beneficialness or importance of the current state. Here, reward estimation is not just a simple numerical calculation, but also a combination of deep semantic information extracted from input features to provide an immediate reference signal for subsequent decision-making. Through the reward estimation layer, the network obtains the ability to dynamically evaluate the "good or bad" of the input state, which can guide the adjustment of the strength of activation. The third part is the value function approximation layer. This layer undertakes the task of estimating the potential "value" of the input state, that is, it comprehensively considers the current state and its possible future performance from a longer-term perspective to give an overall value assessment. The value function approximation layer simulates the expected effect of potential future results by deeply encoding the state, so that the activation mechanism not only focuses on the present, but also takes into account the possibility and impact of the future. This mechanism helps the network have a certain degree of foresight and is conducive to a more reasonable allocation of activation resources.

Finally, the core Bellman gating mechanism combines the immediate reward with the future value to generate a gating signal between 0 and 1. This gating signal acts as a regulating switch for the activation function, dynamically controlling the activation strength of the input state. The gating mechanism ensures that the activation behavior is no longer a mechanical uniform transformation of each input, but is dynamically adjusted according to the immediate feedback of the current state and future expectations, making the network activation more flexible and accurate. This design greatly improves the sensitivity and adaptability of the activation function in information processing. The biggest difference between this Bellman activation mechanism and the traditional activation function is that it introduces the core idea of sequential decision-making in reinforcement learning. Through the weighted combination of the current state and future value, the behavior of the activation function becomes dependent on the contextual information of the input, and has a certain forward-looking nature, which can effectively simulate the decision-making process in a dynamic environment. It is worth emphasizing that, unlike recurrent neural network structures such as LSTM, we do not introduce the time dimension or sequence state transfer, but apply the Bellman idea to the information flow regulation in the spatial dimension.

This innovative application greatly expands the application boundary of the Bellman equation in deep learning. In addition, the Bellman activation mechanism is designed as a highly modular structure that can be easily embedded in various neural network architectures, including but not limited to complex network structures such as multi-layer perceptron (MLP), convolutional neural network (CNN), and Transformer. It is not only compatible with existing models, but also can significantly improve the model's expressiveness and performance stability through dynamic gating, and has broad application prospects and promotion value. In summary, the Bellman activation mechanism is based on the theoretical basis of reinforcement learning, combined with the activation requirements in deep learning, and proposes a new dynamic activation design idea, which injects unprecedented dynamic adjustment and context perception capabilities into the activation function of the neural network, heralding a new development direction in the field of activation function design.

IV. EXPERIMENTAL SETUP

In order to verify the effectiveness of the Bellman gating mechanism, we built two models on the CIFAR-10 dataset:

Standard MLP model: using ReLU as the activation function;

BellmanGatedMLP model: replacing the activation function with the gating structure proposed in this paper.

Each model was trained and evaluated 10 times, and indicators such as accuracy, precision, recall, F1 score, and inference time were collected to evaluate performance and efficiency.

The complete python code used for the experiment is as follows:

```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torchvision import datasets, transforms
from torch.utils.data import DataLoader, Subset
import time
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import numpy as np

# ----------------------------
# 1. Configuration
# ----------------------------
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
BATCH_SIZE = 64
NUM_CLASSES = 10
NUM_EPOCHS = 10
TRAIN_SIZE = 1000
```

```
TEST_SIZE = 1000
LEARNING_RATE = 0.001
NUM_RUNS = 10

# ----------------------------
# 2. Dataset
# ----------------------------
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

train_dataset_full = datasets.CIFAR10(root='./data', train=True, transform=transform, download=True)
test_dataset_full = datasets.CIFAR10(root='./data', train=False, transform=transform, download=True)

train_dataset = Subset(train_dataset_full, list(range(TRAIN_SIZE)))
test_dataset = Subset(test_dataset_full, list(range(TEST_SIZE)))

train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=BATCH_SIZE, shuffle=False)

# ----------------------------
# 3. Vanilla MLP
# ----------------------------
class MLP(nn.Module):
    def __init__(self, input_dim=3*32*32, hidden_dim=512, output_dim=NUM_CLASSES):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_dim, output_dim)

    def forward(self, x):
        x = x.view(x.size(0), -1)
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# ----------------------------
# 4. Bellman-Gated MLP
# ----------------------------
class BellmanGatedMLP(nn.Module):
    def __init__(self, input_dim=3*32*32, hidden_dim=512, output_dim=NUM_CLASSES, gamma=0.9):
        super(BellmanGatedMLP, self).__init__()
        self.gamma = gamma
        self.state = nn.Linear(input_dim, hidden_dim)
        self.reward = nn.Linear(input_dim, hidden_dim)
        self.value = nn.Linear(hidden_dim, hidden_dim)
        self.fc2 = nn.Linear(hidden_dim, output_dim)
        self.sigmoid = nn.Sigmoid()
        self.relu = nn.ReLU()

    def forward(self, x):
        x = x.view(x.size(0), -1)
        s = self.state(x)
        r = self.reward(x)
        v = self.value(s)
        bellman_gate = self.sigmoid(r + self.gamma * v)
        gated_state = s * bellman_gate
        out = self.fc2(self.relu(gated_state))
        return out

# ----------------------------
# 5. Training Function
```

```
# ----------------------------
def train_model(model, optimizer, criterion, loader):
    model.train()
    for epoch in range(NUM_EPOCHS):
        for images, labels in loader:
            images, labels = images.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(images)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

# ----------------------------
# 6. Evaluation Function
# ----------------------------
def evaluate_model(model, loader):
    model.eval()
    all_preds = []
    all_labels = []
    start_time = time.time()
    with torch.no_grad():
        for images, labels in loader:
            images = images.to(device)
            outputs = model(images)
            _, preds = torch.max(outputs, 1)
            all_preds.extend(preds.cpu().numpy())
            all_labels.extend(labels.numpy())
    end_time = time.time()

    acc = accuracy_score(all_labels, all_preds)
    prec = precision_score(all_labels, all_preds, average='macro', zero_division=0)
    rec = recall_score(all_labels, all_preds, average='macro', zero_division=0)
    f1 = f1_score(all_labels, all_preds, average='macro', zero_division=0)
    infer_time = end_time - start_time

    return {
        'accuracy': acc,
        'precision': prec,
        'recall': rec,
        'f1_score': f1,
        'inference_time_sec': infer_time
    }

# ----------------------------
# 7. Run 10x Experiment for Each Model
# ----------------------------
def run_experiment():
    model_classes = {
        "MLP": MLP,
        "BellmanGatedMLP": BellmanGatedMLP
    }

    results = {}

    for name, model_cls in model_classes.items():
        print(f"\n==== Running 10 Experiments for {name} ====\n")
        all_metrics = {
            'accuracy': [],
            'precision': [],
            'recall': [],
            'f1_score': [],
            'inference_time_sec': []
        }
```

```
        for i in range(NUM_RUNS):
            print(f"Run {i+1} / {NUM_RUNS}")
            model = model_cls().to(device)
            optimizer = torch.optim.Adam(model.parameters(), lr=LEARNING_RATE)
            criterion = nn.CrossEntropyLoss()

            train_model(model, optimizer, criterion, train_loader)
            metrics = evaluate_model(model, test_loader)

            for k in all_metrics:
                all_metrics[k].append(metrics[k])
                print(f"{k}: {metrics[k]:.4f}")

            print("-" * 40)

        # Calculate mean and std
        summary = {}
        for k, v in all_metrics.items():
            mean_val = np.mean(v)
            std_val = np.std(v)
            summary[k] = (mean_val, std_val)

        results[name] = {
            'raw': all_metrics,
            'summary': summary
        }

    # Print summary
    print("\n\n==== Final Summary (Mean ± Std) ====\n")
    for model_name, model_result in results.items():
        print(f"Model: {model_name}")
        for metric, (mean_val, std_val) in model_result['summary'].items():
            print(f"{metric}: {mean_val:.4f} ± {std_val:.4f}")
        print("-" * 50)

if __name__ == "__main__":
    run_experiment()
```

V. EXPERIMENTAL RESULTS AND ANALYSIS

The output results of the experimental code are as follows:

==== Final Summary (Mean ± Std) ====

Model: MLP

accuracy: 0.3397 ± 0.0096

precision: 0.3388 ± 0.0065

recall: 0.3359 ± 0.0078

f1_score: 0.3336 ± 0.0091

inference_time_sec: 0.3191 ± 0.1140

Model: BellmanGatedMLP

accuracy: 0.3434 ± 0.0064

precision: 0.3421 ± 0.0071

recall: 0.3392 ± 0.0070

f1_score: 0.3375 ± 0.0071

inference_time_sec: 0.3152 ± 0.0267

The experimental results show that, under the premise of keeping the model structure and training configuration the same, the model using the Bellman gate activation mechanism has achieved slightly better performance than the standard model in all indicators, especially in terms of F1 score and precision. In addition, the inference speed of the model is not significantly affected, indicating that the gating mechanism is practical in terms of computational overhead. This means that we not only improve performance, but also maintain deployment efficiency, providing possibilities for practical applications.

VI. VERSATILITY AND FUTURE APPLICATIONS

Although this study mainly uses multi-layer perceptron (MLP) as an experimental verification platform, the proposed Bellman activation mechanism has a highly modular design and is completely independent of any specific network structure or architecture. In other words, this mechanism can not only be directly applied to the nonlinear transformation layer of the convolutional neural network (CNN), but also to the feedforward activation layer of the Transformer model, seamlessly replacing the traditional activation function, and even has the potential to construct a new attention gating mechanism. With the help of this gating mechanism, the model can adjust the information flow more flexibly and accurately, achieve stronger situational awareness and dynamic adaptability, and improve the overall representation and learning effect.

It is particularly noteworthy that in more complex and emerging fields such as graph neural networks (GNNs), multimodal fusion networks, and modeling tasks in natural language processing, the dependencies between information are not limited to spatial dimensions, but often involve important time, hierarchy, and causal relationships. Traditional static activation functions are difficult to fully capture such complex dependencies and feedback. The advantage of the Bellman gating mechanism itself is that it can integrate current state information and future potential feedback, providing the network with a more reasonable and dynamic information weight strategy. In graph neural networks, the mutual influence and propagation process between nodes have obvious structural dependencies and multi-stage transmission characteristics. The Bellman activation mechanism dynamically adjusts the activation of nodes, enabling the model to better characterize the long-term dependencies and local-global balance between nodes.

In multimodal fusion tasks, the interactive information between different modalities is complex and changes rapidly. The dynamic gating of the Bellman mechanism effectively adjusts the weight distribution of each modality, achieving reasonable fusion and synergistic enhancement of information. In natural language modeling, especially in sequence generation and understanding tasks, Bellman gating can improve the representation depth and prediction accuracy of language models by prospectively adjusting context and future information. In addition, the Bellman activation mechanism has good scalability and adaptability. In the future, it can be combined with methods such as meta-learning and adaptive optimization to further improve the generalization ability in various tasks and data distributions. With the development of hardware and algorithms, this mechanism is expected to become a standard dynamic activation module in deep learning architectures, injecting powerful intelligent decision-making capabilities and flexible response mechanisms into artificial intelligence models.

VII. CONCLUSION

This paper proposes an innovative gating structure that incorporates the Bellman equation, the core theory of reinforcement learning, into the design of neural network activation functions. This paper shows that by constructing a mechanism that combines state estimation, immediate reward and value function, dynamic control of the traditional activation process is achieved, making the activation function an intelligent module that is dynamically adjusted based on the current input state and future potential value, rather than a simple static mapping. The experimental results fully prove that the Bellman activation mechanism achieves significant and stable performance improvements on multiple common evaluation indicators, and maintains the model's efficient reasoning ability without significantly increasing computational overhead. At the same time, the modular design of this mechanism is highly versatile and portable, and can be adapted to various neural network architectures, such as multi-layer perceptron (MLP), convolutional neural network (CNN), and Transformer.

Future research directions can focus on the further application and optimization of this mechanism in more complex neural network structures and diversified task scenarios. For example, we can explore how the Bellman activation mechanism can be deeply integrated with graph neural networks, multimodal fusion models, and natural language processing to improve the expressiveness and generalization capabilities of the model. In addition, we can also consider integrating more reinforcement learning strategies and meta-learning techniques to build a smarter and more adaptive activation function design framework, and evolve the neural network activation mechanism from the traditional "static point function" to a more flexible "dynamic gating mechanism", which will open a new chapter in the field of deep learning model activation.

VIII. REFERENCES

[1] A. Chandrasekhar and K. Suresh, "TOuNN: Topology optimization using neural networks," Struct. Multidiscip. Optim., vol. 63, no. 3, pp. 1135–1149, 2021.

[2] H. Yin, P. Seiler, and M. Arcak, "Stability analysis using quadratic constraints for systems with neural network controllers," IEEE Trans. Autom. Control, vol. 67, no. 4, pp. 1980–1987, 2021.

[3] J. W. Lee et al., "Privacy-preserving machine learning with fully homomorphic encryption for deep neural network," IEEE Access, vol. 10, pp. 30039–30054, 2022.

[4] T. De Ryck, S. Lanthaler, and S. Mishra, "On the approximation of functions by tanh neural networks," Neural Netw., vol. 143, pp. 732–750, 2021.

[5] K. C. Onyelowe et al., "Evaluating the compressive strength of recycled aggregate concrete using novel artificial neural network," Civil Eng. J., vol. 8, no. 8, pp. 1679–1693, 2022.

[6] A. D. Jagtap, Y. Shin, K. Kawaguchi, and G. E. Karniadakis, "Deep Kronecker neural networks: A general framework for neural networks with adaptive activation functions," Neurocomputing, vol. 468, pp. 165–180, 2022.

[7] A. M. Atto, S. Galichet, D. Pastor, and N. Méger, "On joint parameterizations of linear and nonlinear functionals in neural networks," Neural Netw., vol. 160, pp. 12–21, 2023.

[8] H. Sivaprasad et al., "Fatigue damage prediction of top tensioned riser subjected to vortex-induced vibrations using artificial neural networks," Ocean Eng., vol. 268, Art. no. 113393, 2023.

[9] Y. Tong et al., "Polynomial fitting algorithm based on neural network," ASP Trans. Pattern Recognit. Intell. Syst., vol. 1, no. 1, pp. 32–39, 2021.

[10] B. Coker, W. P. Bruinsma, D. R. Burt, W. Pan, and F. Doshi-Velez, "Wide mean-field Bayesian neural networks ignore the data," in Proc. Int. Conf. Artif. Intell. Statist., 2022, pp. 5276–5333.

[11] C. Lee, H. Hasegawa, and S. Gao, "Complex-valued neural networks: A comprehensive survey," IEEE/CAA J. Autom. Sinica, vol. 9, no. 8, pp. 1406–1426, 2022.

[12] X. Zhao et al., "A review of convolutional neural networks in computer vision," Artif. Intell. Rev., vol. 57, no. 4, p. 99, 2024.

[13] C. S. Vikranth et al., "Computer assisted diagnosis of breast cancer using histopathology images and convolutional neural networks," in Proc. 2nd Int. Conf. Artif. Intell. Signal Process. (AISP), Feb. 2022, pp. 1–6.

[14] R. Zhang, S. Bilige, and T. Chaolu, "Fractal solitons, arbitrary function solutions, exact periodic wave and breathers for a nonlinear partial differential equation by using bilinear neural network method," J. Syst. Sci. Complex., vol. 34, no. 1, pp. 122–139, 2021.

[15] A. D. Jagtap, Z. Mao, N. Adams, and G. E. Karniadakis, "Physics-informed neural networks for inverse problems in supersonic flows," J. Comput. Phys., vol. 466, Art. no. 111402, 2022.

[16] N. Ma, X. Zhang, M. Liu, and J. Sun, "Activate or not: Learning customized activation," in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2021, pp. 8032–8042.

[17] C. Xie et al., "Optimized functional linked neural network for predicting diaphragm wall deflection induced by braced excavations in clays," Geosci. Front., vol. 13, no. 2, Art. no. 101313, 2022.

[18] A. M. Roy, R. Bose, and J. Bhaduri, "A fast accurate fine-grain object detection model based on YOLOv4 deep neural network," Neural Comput. Appl., vol. 34, no. 5, pp. 3895–3921, 2022.

[19] K. Linka and E. Kuhl, "A new family of constitutive artificial neural networks towards automated model discovery," Comput. Methods Appl. Mech. Eng., vol. 403, Art. no. 115731, 2023.

[20] I. Chillotti, M. Joye, and P. Paillier, "Programmable bootstrapping enables efficient homomorphic inference of deep neural networks," in Cyber Secur. Cryptogr. Mach. Learn., CSCML 2021, pp. 1–19, Springer, 2021.

[21] M. Reyad, A. M. Sarhan, and M. Arafa, "A modified Adam algorithm for deep neural network optimization," Neural Comput. Appl., vol. 35, no. 23, pp. 17095–17112, 2023.

[22] G. Habib and S. Qureshi, "Optimization and acceleration of convolutional neural networks: A survey," J. King Saud Univ. Comput. Inf. Sci., vol. 34, no. 7, pp. 4244–4268, 2022.

[23] A. Solanki and S. Pandey, "Music instrument recognition using deep convolutional neural networks," Int. J. Inf. Technol., vol. 14, no. 3, pp. 1659–1668, 2022.

[24] M. M. Elsagheer and S. M. Ramzy, "A hybrid model for automatic modulation classification based on residual neural networks and long short term memory," Alexandria Eng. J., vol. 67, pp. 117–128, 2023.

[25] S. Daudin and B. Seeger, "A comparison principle for semilinear Hamilton–Jacobi–Bellman equations in the Wasserstein space," Calc. Var. Partial Differ. Equ., vol. 63, no. 4, p. 106, 2024.

[26] S. Meyn, "The projected Bellman equation in reinforcement learning," IEEE Trans. Autom. Control, 2024.

[27] J. Weston, D. Tolić, and I. Palunko, "Application of Hamilton–Jacobi–Bellman equation/Pontryagin's principle for constrained optimal control," J. Optim. Theory Appl., vol. 200, no. 2, pp. 437–462, 2024.

[28] C. Fromonteil, R. Tricarico, F. Cesa, and H. Pichler, "Hamilton–Jacobi–Bellman equations for Rydberg-blockade processes," Phys. Rev. Res., vol. 6, no. 3, Art. no. 033333, 2024.

[29] H. V. Tran, Z. Wang, and Y. P. Zhang, "Policy iteration for exploratory Hamilton–Jacobi–Bellman equations," Appl. Math. Optim., vol. 91, no. 2, p. 50, 2025.

[30] Y. Fei, Z. Yang, Y. Chen, and Z. Wang, "Exponential Bellman equation and improved regret bounds for risk-sensitive reinforcement learning," Adv. Neural Inf. Process. Syst., vol. 34, pp. 20436–20446, 2021.

[31] M. Gonzalez-Atienza, D. Vanoost, M. Verbeke, and D. Pissoort, "Decision algorithm based on the modified Bellman equation to deal with EMI-induced errors in Hamming-based communications," IEEE Trans. Electromagn. Compat., vol. 65, no. 3, pp. 667–678, 2023.

[32] H. E. Wiltzer, D. Meger, and M. G. Bellemare, "Distributional Hamilton–Jacobi–Bellman equations for continuous-time reinforcement learning," in Proc. Int. Conf. Mach. Learn., 2022, pp. 23832–23856.

[33] C. Jimenez, A. Marigonda, and M. Quincampoix, "Dynamical systems and Hamilton–Jacobi–Bellman equations on the Wasserstein space and their L2 representations," SIAM J. Math. Anal., vol. 55, no. 5, pp. 5919–5966, 2023.

[34] M. Eigel, R. Schneider, and D. Sommer, "Dynamical low-rank approximations of solutions to the Hamilton–Jacobi–Bellman equation," Numer. Linear Algebra Appl., vol. 30, no. 3, Art. no. e2463, 2023.