# Introducing Harsanyi-Inspired Adaptive Dropout: A New Paradigm for Dynamic Neural Network Regularization

Jincheng Zhang *

Posted Date: 1 July 2025

doi: 10.20944/preprints202507.0057.v1

Keywords: adaptive dropout; harsanyi's aggregation theorem; neural network regularization; contribution-based masking; generalizable regularization mechanism

 doi:10.20944/preprints202507.0057.v1



*Article*

# Introducing Harsanyi-Inspired Adaptive Dropout: A New Paradigm for Dynamic Neural Network Regularization

**Jincheng Zhang**

Faculty of Science and Technology, Rajabhat Maha Sarakham University, Maha Sarakham 44000, Thailand; zjc1639834588@gmail.com

**Abstract**

Dropout is a widely used regularization technique in deep learning. It prevents overfitting by randomly blocking some neurons during training. However, the traditional Dropout method uses a uniform probability for random discarding, ignoring the importance differences of each feature under a specific input. This paper draws on the core idea of individual contribution distribution in the Harsanyi aggregation theorem and proposes a new adaptive Dropout mechanism, which dynamically adjusts the discard probability according to the importance of neuron activation. Experimental results show that this method shows better accuracy and F1 score than the traditional Dropout model in multiple repeated training. In addition, we emphasize that this mechanism has good versatility and can be extended to various neural network architectures.

**Keywords:** adaptive dropout; harsanyi's aggregation theorem; neural network regularization; contribution-based masking; generalizable regularization mechanism

## 1. Introduction

Against the backdrop of the rapid development of deep learning, neural network models have made great progress in many fields such as image recognition, natural language processing, and speech recognition [1–4]. However, as the model structure continues to grow, the number of parameters also increases dramatically. How to effectively improve the generalization ability of the model and reduce the overfitting problem during training remains an important challenge [5–8]. To solve this problem, researchers have proposed various regularization methods. Among them, dropout is widely used in various neural networks because it is easy to implement and has significant practical effects [9–15].

The core idea of the dropout mechanism is to simulate the uncertainty of the neural structure by randomly setting the output of some neurons to zero during the learning process of the model [16–19]. This "random clipping" method helps the network avoid over-reliance on a few features and enhances the adaptability of the model when facing unknown data [20–22]. However, the standard dropout strategy usually uses a globally fixed dropout probability, ignoring the differences in the role of each neuron in a specific input context [23–26]. In fact, the contribution of a single neuron may vary significantly between different samples and stages. Uniform random dropout may even mistakenly delete important information and hinder the convergence of the model [27–30].

At the same time, the Harsanyi aggregation theorem in the field of economics provides new inspiration [31–36]. This theorem emphasizes that in the process of collective decision-making, fair value distribution should be based on the marginal contribution of each participant to the collective utility. This idea can also be naturally applied to the dropout strategy in neural networks. We regard neurons as "individuals in collective decision-making", and their activation values reflect the "contribution" of each neuron in the current task. By evaluating this contribution and adjusting the

 doi:10.20944/preprints202507.0057.v1



dropout probability accordingly, it may be more reasonable to retain the information flow that is more important to the task result.

Based on this idea, this paper proposes a "contribution-based dropout method". The core of this method is to introduce the statistical information of neuron activation values as a measure of feature importance and construct an adaptive dropout probability mechanism. In specific implementation, the average value of the absolute value of each dimension of the activated feature in the current batch is used as the benchmark, normalized, and formed into an importance index, and then the dropout probability is adjusted inversely, so that "features with high importance are less likely to be discarded, and features with low importance are more likely to be discarded". This mechanism can not only explicitly introduce information selectivity at a low computational cost, but also has high structural compatibility, and can directly replace the standard dropout in various neural networks.

The starting point of this study is not only to improve dropout itself, but also to explore the relationship between the regularization strategy of deep learning and the collective rationality model in economics. We believe that this interdisciplinary thinking shift will not only bring rich ideas to the design of neural networks, but also expand the scope of application of Hasani's aggregation theorem in the field of artificial intelligence.

## 2. Related Work

Since its proposal, the dropout mechanism in neural networks has been widely used in various deep learning models and has become a standard means to solve overfitting and improve the generalization ability of models. The basic idea is to randomly "shield" some neurons in each learning iteration. This makes the model rely on different substructures in different learning processes, thereby reducing the interdependence between parameters. The advantage of this technology is that it is simple and efficient. It only needs to add a random sampling step in the forward propagation process to play a regularization role without significantly increasing the computational overhead.

However, traditional dropout techniques usually adopt a uniform dropout probability, that is, the same retention/dropout strategy is applied to all neurons, regardless of their importance in the current task. Although this "equal probability" strategy is unified and easy to implement, it has structural blind spots because it ignores the differences between input samples and the dynamic performance of neurons themselves. In practical applications, this indiscriminate dropout operation may block the activation channels that play a key role in model prediction, resulting in information loss and hindering the further improvement of model performance.

In recent years, researchers have been aware of this problem and are committed to proposing more "intelligent" dropout techniques. One direction is to introduce an attention mechanism to selectively discard neurons after weighting their importance. Another direction is to analyze the impact of each neuron on the loss function through the back-propagation stage based on gradient information and decide whether to retain them in the next round of forward propagation. Although these methods have achieved certain improvements in accuracy, they generally have problems such as complex implementation, high computational cost, and poor scalability, especially in large models and resource-constrained environments, which limits their application scenarios.

Unlike these improvement methods that rely on backward gradients or external attention mechanisms, the Dropout improvement strategy proposed in this paper is only based on the statistical information of the activation values in the forward propagation stage. In each training batch, we only need to calculate the average absolute value of the neuron activation value to evaluate its "importance" or "contribution" in the current sample set, and adjust the Dropout probability accordingly. This process does not require gradient information, has good interpretability, and does not incur additional training overhead. In other words, neurons with larger activation values are considered more important in the current task and are more likely to be retained. On the contrary, blocking these neurons more boldly can improve the robustness of the model.

More importantly, the design concept of this method incorporates the core concept of the Hasani aggregation theorem in economics. The theorem emphasizes that in collective decision-making, the

 doi:10.20944/preprints202507.0057.v1



value of each participant should be evaluated and allocated based on its marginal contribution. Introducing this idea into neural network regularization means that we no longer treat all neurons equally, but dynamically adjust them according to the marginal performance of neurons in the "collective task", thereby achieving fairer and more efficient resource allocation. This research method can be said to be a regularization strategy with a philosophical basis, which introduces the idea of micro-control of social decision-making mechanisms into deep learning models.

In summary, the contribution-based adaptive dropout mechanism proposed in this paper has made many improvements on the basis of existing research. First, we simplify the implementation process by introducing importance evaluation in the forward propagation stage. Second, we improve the explanatory power and rationality of the dropout mechanism by introducing the core principles of social choice theory. Third, we provide a more targeted neural network regularization method without sacrificing efficiency, and hope to further expand its application scope and theoretical depth in future research.

## 3. Method Design

The "Haisani Dropout" mechanism proposed in this study is inspired by the Harsanyi aggregation theorem, which has important theoretical value in economics. The theorem emphasizes that the marginal role played by individuals in collective decision-making should be evaluated accordingly. On this basis, we construct a new dropout strategy for neural network regularization, which aims to dynamically adjust the possibility of neurons being retained according to their actual performance in the current input, thereby achieving more selective structural sparsification.

Specifically, during the training process, when the model is in the state of enabling dropout, we first perform absolute value processing on the activation value of the current neuron, and then calculate the average activation level of each neuron in the entire batch of samples. This value is regarded as the "contribution" or "importance index" of the neuron in the current task. Next, we normalize the contribution of all neurons and map them to a relatively reasonable weight distribution with a uniform range. Based on this distribution, we further construct a set of adaptive Dropout discarding probabilities: the higher the contribution of neurons, the higher the retention probability; and the probability of Dropout is higher for neurons with overall low activation values and judged to have no significant contribution at present.

This probability control method based on neuron activation response is in sharp contrast to the traditional fixed drop rate strategy. Traditional Dropout usually presets a uniform drop rate for each layer, ignoring the individual differences between input samples and neuron responses, which may lead to "mistaken killing" of key neurons or "retention" of redundant channels. The Harsanyi Dropout achieves more refined regularization control through lightweight statistics of activation values, so that model training can retain the core paths in the information flow more specifically while maintaining the advantages of decorrelation, thereby more effectively promoting model convergence and enhancing generalization ability.

In addition, the biggest advantage of this method lies in its structural independence and simplicity in implementation. Specifically, Harsanyi Dropout does not rely on a specific network structure, nor does it require special adjustments to the connection method of the previous and next layers. It can be used as a modular Dropout alternative and can be inserted into any existing neural network framework. Therefore, it can be seamlessly embedded and deployed in traditional multi-layer perceptrons (MLPs) or in modern deep learning architectures such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), graph neural networks (GNNs) and even Transformers. Such versatility gives it good expansion prospects and migration potential, and is particularly suitable for widespread application in large-scale, heterogeneous network systems.

Furthermore, from an explanatory perspective, this mechanism provides a theoretical basis for Dropout based on "neuron importance". With the definition of "marginal contribution" in Harsanyi's theorem, we can regard neurons as "participants" in a collective prediction task, and the size of their role directly affects their "survival probability". This view not only improves the transparency and

 doi:10.20944/preprints202507.0057.v1



controllability of the Dropout mechanism, but also provides model designers with more theoretical tools to analyze the internal behavior of the network.

In summary, "Haisani Dropout" is an innovative regularization mechanism that integrates theory and practice. It takes into account efficiency, explanatory power and versatility, is applicable to various mainstream neural network structures, and can effectively improve model performance without introducing significant computational burden. This method not only injects a new ideological core into the Dropout mechanism, but also opens up a possible research direction for future neural network control strategies based on "individual contribution perception".

## 4. Experimental Design

To verify the effectiveness of the proposed method, we designed a comparative experiment on the standard image classification task CIFAR-10 (5000 images are used for the training set and the test set respectively). Two neural network models are constructed: one is a multi-layer perceptron (MLP) with a conventional fixed Dropout, and the other is an improved MLP with Harsanyi Dropout, keeping other network structures and training strategies completely consistent. The experiment evaluates the average performance of the model under multiple indicators by repeating the training ten times.

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

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

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.4914, 0.4822, 0.4465), (0.247, 0.243, 0.261))
])

train_full = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
test_full = torchvision.datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)
train_subset = Subset(train_full, list(range(5000)))
test_subset = Subset(test_full, list(range(5000)))

train_loader = DataLoader(train_subset, batch_size=128, shuffle=True, num_workers=2)
test_loader = DataLoader(test_subset, batch_size=128, shuffle=False, num_workers=2)
```

 doi:10.20944/preprints202507.0057.v1



```
num_classes = 10

class BaselineMLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(3*32*32, 256)
        self.relu1 = nn.ReLU()
        self.dropout1 = nn.Dropout(0.5)
        self.fc2 = nn.Linear(256, 128)
        self.relu2 = nn.ReLU()
        self.dropout2 = nn.Dropout(0.5)
        self.fc3 = nn.Linear(128, num_classes)

    def forward(self, x):
        x = self.flatten(x)
        x = self.relu1(self.fc1(x))
        x = self.dropout1(x)
        x = self.relu2(self.fc2(x))
        x = self.dropout2(x)
        x = self.fc3(x)
        return x

class HarsanyiDropout(nn.Module):
    def __init__(self, base_p=0.5):
        super().__init__()
        self.base_p = base_p

    def forward(self, x):
        if not self.training or self.base_p == 0:
            return x
        with torch.no_grad():
            abs_x = torch.abs(x)
            importance = abs_x.mean(dim=0)
            importance_norm = (importance - importance.min()) / (importance.max() - importance.min() + 1e-8)
            adaptive_p = self.base_p * (1 - importance_norm)
        adaptive_p_expand = adaptive_p.unsqueeze(0).expand_as(x)
        mask = torch.bernoulli(1 - adaptive_p_expand).to(x.device)
        out = x * mask / (1 - adaptive_p_expand + 1e-8)
        return out
```

 doi:10.20944/preprints202507.0057.v1



```
class HarsanyiMLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(3*32*32, 256)
        self.relu1 = nn.ReLU()
        self.harsanyi_dropout1 = HarsanyiDropout(base_p=0.5)
        self.fc2 = nn.Linear(256, 128)
        self.relu2 = nn.ReLU()
        self.harsanyi_dropout2 = HarsanyiDropout(base_p=0.5)
        self.fc3 = nn.Linear(128, num_classes)

    def forward(self, x):
        x = self.flatten(x)
        x = self.fc1(x)
        x = self.relu1(x)
        x = self.harsanyi_dropout1(x)
        x = self.fc2(x)
        x = self.relu2(x)
        x = self.harsanyi_dropout2(x)
        x = self.fc3(x)
        return x

def train_model(model, dataloader, criterion, optimizer, epochs=5):
    model.train()
    for epoch in range(epochs):
        running_loss = 0.0
        for inputs, labels in dataloader:
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

def evaluate_model(model, dataloader, compute_roc_auc=False):
    model.eval()
    all_preds = []
    all_labels = []
    all_probs = []
    with torch.no_grad():
        for inputs, labels in dataloader:
```

 doi:10.20944/preprints202507.0057.v1



```
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            probs = nn.Softmax(dim=1)(outputs)
            preds = outputs.argmax(dim=1)
            all_preds.append(preds.cpu())
            all_labels.append(labels.cpu())
            all_probs.append(probs.cpu())
    all_preds = torch.cat(all_preds).numpy()
    all_labels = torch.cat(all_labels).numpy()
    all_probs = torch.cat(all_probs).numpy()

    accuracy = accuracy_score(all_labels, all_preds)
    precision = precision_score(all_labels, all_preds, average='macro', zero_division=0)
    recall = recall_score(all_labels, all_preds, average='macro', zero_division=0)
    f1 = f1_score(all_labels, all_preds, average='macro', zero_division=0)

    if compute_roc_auc:
        try:
            from sklearn.preprocessing import label_binarize
            all_labels_bin = label_binarize(all_labels, classes=list(range(num_classes)))
            roc_auc = roc_auc_score(all_labels_bin, all_probs, average='macro', multi_class='ovr')
        except Exception:
            roc_auc = float('nan')
    else:
        roc_auc = float('nan')

    return [accuracy, precision, recall, f1, roc_auc]

def summarize_results(name, all_metrics):
    all_metrics = np.array(all_metrics)
    means = np.mean(all_metrics, axis=0)
    stds = np.std(all_metrics, axis=0)
    metric_names = ['Accuracy', 'Precision', 'Recall', 'F1-Score', 'ROC-AUC']
    print(f"\n===== {name} - Final Summary over 10 Runs =====")
    for i, metric in enumerate(metric_names):
        if np.isnan(means[i]):
            print(f"{metric}: Skipped (ROC-AUC)")
        else:
            print(f"{metric}: Mean = {means[i]:.4f}, Std = {stds[i]:.4f}")
    print("================================================\n")

def run_experiment():
```

 doi:10.20944/preprints202507.0057.v1



```
    epochs = 5
    criterion = nn.CrossEntropyLoss()
    runs = 10

    baseline_all_metrics = []
    harsanyi_all_metrics = []

    for i in range(runs):
        print(f"======= Run {i+1} / {runs} =======")
        # Baseline MLP
        baseline_model = BaselineMLP().to(device)
        baseline_optimizer = optim.Adam(baseline_model.parameters(), lr=0.001)
        train_model(baseline_model, train_loader, criterion, baseline_optimizer, epochs=epochs)
        metrics = evaluate_model(baseline_model, test_loader, compute_roc_auc=False)
        baseline_all_metrics.append(metrics)
        print(f"[Baseline MLP] Acc: {metrics[0]:.4f}, Prec: {metrics[1]:.4f}, Recall: {metrics[2]:.4f}, F1:
{metrics[3]:.4f}")

        # Harsanyi MLP
        harsanyi_model = HarsanyiMLP().to(device)
        harsanyi_optimizer = optim.Adam(harsanyi_model.parameters(), lr=0.001)
        train_model(harsanyi_model, train_loader, criterion, harsanyi_optimizer, epochs=epochs)
        metrics = evaluate_model(harsanyi_model, test_loader, compute_roc_auc=False)
        harsanyi_all_metrics.append(metrics)
        print(f"[Harsanyi MLP] Acc: {metrics[0]:.4f}, Prec: {metrics[1]:.4f}, Recall: {metrics[2]:.4f}, F1:
{metrics[3]:.4f}")

    # Summary
    summarize_results("Baseline MLP", baseline_all_metrics)
    summarize_results("Harsanyi MLP", harsanyi_all_metrics)

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

The output results of the experimental code are as follows:

```
===== Baseline MLP - Final Summary over 10 Runs =====
Accuracy: Mean = 0.3882, Std = 0.0076
Precision: Mean = 0.3872, Std = 0.0063
Recall: Mean = 0.3882, Std = 0.0076
F1-Score: Mean = 0.3772, Std = 0.0101
ROC-AUC: Skipped (ROC-AUC)
==========================================
```

 doi:10.20944/preprints202507.0057.v1



===== Harsanyi MLP - Final Summary over 10 Runs =====

Accuracy: Mean = 0.4075, Std = 0.0052

Precision: Mean = 0.4109, Std = 0.0050

Recall: Mean = 0.4077, Std = 0.0055

F1-Score: Mean = 0.3991, Std = 0.0041

ROC-AUC: Skipped (ROC-AUC)

==========

The results show that the Harsanyi Dropout model is superior to the traditional Dropout model in accuracy, precision, recall and F1 score, and the standard deviation of each indicator is lower, showing a more stable training effect. This shows that the method of introducing feature importance as the basis for Dropout adjustment is more expressive and robust.

## 5. Generality and Scalability Analysis

One of the biggest advantages of the Harsanyi Dropout mechanism proposed in this study is its high generality and flexibility. The mechanism is designed based on the statistical characteristics of neuron activation values in the forward propagation phase and is completely independent of the specific network structure. Therefore, it can be easily transplanted and applied to various neural network architectures without complex modifications or reconstruction of the underlying network.

Specifically, in convolutional neural networks (CNNs), the concept of neurons is usually extended to feature maps (channels) in the convolution layer or activation units at spatial positions. The Harsanyi Dropout mechanism can calculate the overall activation contribution of each feature map and adaptively adjust the drop probability of the corresponding channel. This drop strategy that is dynamically adjusted for each channel is more targeted than the traditional fixed probability Dropout, and can more effectively avoid the loss of important features, thereby improving the performance and generalization ability of the model.

In the more complex Transformer architecture, the core part of the network is the multi-head attention mechanism and the feedforward neural network. Hasani Dropout can be extended to adaptive sparse processing of attention weights, dynamically adjusting the drop probability according to the "contribution" of each attention head or attention weight vector. This technology not only effectively suppresses the redundant information that may exist in the attention mechanism, but also improves the model's information selection ability in sequence modeling and improves the model's ability to capture long-distance dependencies and contextual information.

In addition, the idea of Hasani Dropout is not limited to the traditional Dropout module. As an adaptive regularization strategy based on neuron activation statistics, it is highly scalable in nature. For example, the mechanism can be combined with regularization techniques such as batch normalization and layer normalization to dynamically adjust the parameters and weight decay rates of these regularization modules, thereby further improving training stability and model robustness. In some scenarios, designing a contribution-based weight pruning mechanism can help optimize the neural network structure using neural architecture search (NAS) to make the model lighter and more efficient.

Overall, the design concept of the Harsanyi Dropout mechanism is to dynamically evaluate and select model components based on their "individual contributions". This concept is universal regardless of the architecture or task. With the increasing diversification of the structure and application scenarios of deep learning models, this versatility will provide a broad space for future research and application of Harsanyi Dropout.

## 6. Conclusion and Future Work

This paper proposes an innovative adaptive dropout method. Based on the concept of individual contribution in Harsanyi aggregation theorem, we transform the statistics of neuron activation into

 doi:10.20944/preprints202507.0057.v1



dynamically adjusted dropout probability, thus realizing a more flexible and effective regularization mechanism than the traditional fixed dropout rate. Through experimental verification using the standard multi-layer perceptron (MLP) architecture, the proposed method has achieved significant improvements in multiple indicators such as accuracy, precision, recall and F1 score, demonstrating its excellent performance in improving the generalization ability of the model.

More importantly, this paper not only verifies the effectiveness of the proposed method in MLP, but also highlights its excellent versatility and scalability. Due to its lightweight and structure-independent design, our method can be directly applied to various mainstream deep learning architectures such as convolutional neural networks, recurrent neural networks and Transformer, and can adapt to a wide range of application scenarios. The concept of "contribution evaluation" proposed in our method provides a theoretical basis and technical path for designing smarter and more adaptive neural network regularization strategies in the future.

Future research will be conducted in depth around the following core directions. First, we will further explore the application effect of this mechanism in large-scale Transformer models, especially in long sequence modeling tasks in natural language processing and computer vision, and verify its ability to improve the efficiency and accuracy of the attention mechanism. Secondly, we will combine Hasani's contribution concept and gradient information to design a more fine-grained Dropout control method, dynamically adjust the Dropout strategy during the learning process, and further explore the multi-layer structure of network internal information and its dependencies. Finally, we plan to combine the neural architecture search (NAS) technology with contribution evaluation to automatically optimize the Dropout strategy and its parameter settings to improve its generalization ability and adaptability in multi-task learning and transfer learning scenarios.

In other words, the introduction of the Harsanyi Dropout mechanism not only enriches the theory and practice in the field of neural network regularization, but also provides a new research perspective for dynamic model control combined with subsequent contributions. With the continuous development of related technologies, we believe that this mechanism will play an increasingly important role in the learning and application of deep learning models in the future.

## References

1. Pham, H., & Le, Q. (2021, May). Autodropout: Learning dropout patterns to regularize deep networks. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 11, pp. 9351-9359).
2. Basnet, R. B., Johnson, C., & Doleck, T. (2022). Dropout prediction in Moocs using deep learning and machine learning. Education and Information Technologies, 27(8), 11499-11513.
3. Wu, L., Li, J., Wang, Y., Meng, Q., Qin, T., Chen, W., ... & Liu, T. Y. (2021). R-drop: Regularized dropout for neural networks. Advances in neural information processing systems, 34, 10890-10905.
4. Omar, A., & Abd El-Hafeez, T. (2024). Optimizing epileptic seizure recognition performance with feature scaling and dropout layers. Neural Computing and Applications, 36(6), 2835-2852.
5. Kiruthiga, D., & Manikandan, V. (2023). Levy flight-particle swarm optimization-assisted BiLSTM+ dropout deep learning model for short-term load forecasting. Neural Computing and Applications, 35(3), 2679-2700.
6. Nguyen, D., Barkousaraie, A. S., Bohara, G., Balagopal, A., McBeth, R., Lin, M. H., & Jiang, S. (2021). A comparison of Monte Carlo dropout and bootstrap aggregation on the performance and uncertainty estimation in radiation therapy dose prediction with deep learning neural networks. Physics in Medicine & Biology, 66(5), 054002.
7. Hu, J., Weng, B., Huang, T., Gao, J., Ye, F., & You, L. (2021). Deep residual convolutional neural network combining dropout and transfer learning for ENSO forecasting. Geophysical Research Letters, 48(24), e2021GL093531.

 doi:10.20944/preprints202507.0057.v1




8. Abdar, M., Samami, M., Mahmoodabad, S. D., Doan, T., Mazoure, B., Hashemifesharaki, R., ... & Nahavandi, S. (2021). Uncertainty quantification in skin cancer classification using three-way decision-based Bayesian deep learning. Computers in biology and medicine, 135, 104418.
9. Bacanin, N., Zivkovic, M., Al-Turjman, F., Venkatachalam, K., Trojovský, P., Strumberger, I., & Bezdan, T. (2022). Hybridized sine cosine algorithm with convolutional neural networks dropout regularization application. Scientific Reports, 12(1), 6302.
10. Wang, J. (2025). Credit Card Fraud Detection via Hierarchical Multi-Source Data Fusion and Dropout Regularization. Transactions on Computational and Scientific Methods, 5(1).
11. Ali, A. A. A., & Mallaiah, S. (2022). Intelligent handwritten recognition using hybrid CNN architectures based-SVM classifier with dropout. Journal of King Saud University-Computer and Information Sciences, 34(6), 3294-3300.
12. Zunino, A., Bargal, S. A., Morerio, P., Zhang, J., Sclaroff, S., & Murino, V. (2021). Excitation dropout: Encouraging plasticity in deep neural networks. International Journal of Computer Vision, 129(4), 1139-1152.
13. Ait Skourt, B., El Hassani, A., & Majda, A. (2022). Mixed-pooling-dropout for convolutional neural network regularization. Journal of King Saud University-Computer and Information Sciences, 34(8), 4756-4762.
14. Zhang, D., Li, C., Lin, F., Zeng, D., & Ge, S. (2021, August). Detecting Deepfake Videos with Temporal Dropout 3DCNN. In IJCAI (pp. 1288-1294).
15. Horvath, S., Laskaridis, S., Almeida, M., Leontiadis, I., Venieris, S., & Lane, N. (2021). Fjord: Fair and accurate federated learning under heterogeneous targets with ordered dropout. Advances in Neural Information Processing Systems, 34, 12876-12889.
16. Wei, M., Gu, H., Ye, M., Wang, Q., Xu, X., & Wu, C. (2021). Remaining useful life prediction of lithium-ion batteries based on Monte Carlo Dropout and gated recurrent unit. Energy Reports, 7, 2862-2871.
17. Kong, X., Liu, X., Gu, J., Qiao, Y., & Dong, C. (2022). Reflash dropout in image super-resolution. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 6002-6012).
18. Guo, D., Wang, X., Gao, K., Jin, Y., Ding, J., & Chai, T. (2021). Evolutionary optimization of high-dimensional multiobjective and many-objective expensive problems assisted by a dropout neural network. IEEE transactions on systems, man, and cybernetics: systems, 52(4), 2084-2097.
19. Lin, J., He, C., & Cheng, R. (2022). Adaptive dropout for high-dimensional expensive multiobjective optimization. Complex & Intelligent Systems, 8(1), 271-285.
20. Zhang, J., Phoon, K. K., Zhang, D., Huang, H., & Tang, C. (2021). Deep learning-based evaluation of factor of safety with confidence interval for tunnel deformation in spatially variable soil. Journal of Rock Mechanics and Geotechnical Engineering, 13(6), 1358-1367.
21. González, E. G., Villar, J. R., & de la Cal, E. (2021). Time series data augmentation and dropout roles in deep learning applied to fall detection. In 15th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2020) 15 (pp. 563-570). Springer International Publishing.
22. Liu, Y., Hu, R., & Balaprakash, P. (2021, May). Uncertainty quantification of deep neural network-based turbulence model for reactor transient analysis. In Verification and Validation (Vol. 84782, p. V001T11A001). American Society of Mechanical Engineers.
23. Abbaszadeh Shahri, A., Shan, C., & Larsson, S. (2022). A novel approach to uncertainty quantification in groundwater table modeling by automated predictive deep learning. Natural Resources Research, 31(3), 1351-1373.
24. Mubarak, A. A., Cao, H., & Ahmed, S. A. (2021). Predictive learning analytics using deep learning model in MOOCs' courses videos. Education and Information Technologies, 26(1), 371-392.

 doi:10.20944/preprints202507.0057.v1



25. Guo, X., Zhang, X., Tian, X., Lu, W., & Li, X. (2022). Probabilistic prediction of the heave motions of a semi-submersible by a deep learning model. Ocean Engineering, 247, 110578.
26. Loftus, T. J., Shickel, B., Ruppert, M. M., Balch, J. A., Ozrazgat-Baslanti, T., Tighe, P. J., ... & Bihorac, A. (2022). Uncertainty-aware deep learning in healthcare: a scoping review. PLOS digital health, 1(8), e0000085.
27. Zoremsanga, C., & Hussain, J. (2024). Particle swarm optimized deep learning models for rainfall prediction: a case study in Aizawl, Mizoram. IEEE Access.
28. Papp, P. A., Martinkus, K., Faber, L., & Wattenhofer, R. (2021). DropGNN: Random dropouts increase the expressiveness of graph neural networks. Advances in Neural Information Processing Systems, 34, 21997-22009.
29. Dolezal, J. M., Srisuwananukorn, A., Karpeyev, D., Ramesh, S., Kochanny, S., Cody, B., ... & Pearson, A. T. (2022). Uncertainty-informed deep learning models enable high-confidence predictions for digital histopathology. Nature communications, 13(1), 6572.
30. Njoku, J. N., Morocho-Cayamcela, M. E., & Lim, W. (2021). CGDNet: Efficient hybrid deep learning model for robust automatic modulation recognition. IEEE Networking Letters, 3(2), 47-51.
31. Pivato, M., & Tchouante, É. F. (2024). Bayesian social aggregation with almost-objective uncertainty. Theoretical Economics, 19(3), 1351-1398.
32. Kinney, D. (2025). Aggregating Measures of Accuracy and Fairness in Prediction Algorithms.
33. Karni, E., & Weymark, J. A. (2024). Impartiality and relative utilitarianism. Social Choice and Welfare, 63(1), 1-18.
34. Nebel, J. M. (2022). Aggregation without interpersonal comparisons of well-being. Philosophy and Phenomenological Research, 105(1), 18-41.
35. Pitis, S. (2023). Consistent aggregation of objectives with diverse time preferences requires non-markovian rewards. Advances in Neural Information Processing Systems, 36, 2877-2893.
36. Billot, A., & Qu, X. (2021). Utilitarian aggregation with heterogeneous beliefs. American Economic Journal: Microeconomics, 13(3), 112-123.