NashNorm: A Novel Nash Equilibrium-Inspired Batch Normalization Method for Deep Neural Networks

Jincheng Zhang *

Posted Date: 1 July 2025

doi: 10.20944/preprints202506.2531.v1

Keywords: Nash normalization; game-theoretic approach; neural network normalization; strategic feedback; deep learning stability

 doi:10.20944/preprints202506.2531.v1

Article

NashNorm: A Novel Nash Equilibrium-Inspired Batch Normalization Method for Deep Neural Networks

Jincheng Zhang

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

Abstract

Batch Normalization has been widely used in various deep learning architectures to alleviate training instability and accelerate convergence. Despite this, traditional batch normalization mechanisms usually treat neurons as independent individuals and fail to effectively model the potential dependencies between them. This paper introduces the Nash equilibrium idea in game theory for the first time and designs a normalization mechanism based on strategic interaction, called "Nash BatchNorm". By constructing a strategic feedback mechanism between neurons, this method enables neurons to compete with each other as "strategic individuals" during the normalization process, thereby achieving a more stable equilibrium state. Experimental results show that this method achieves improvements in multiple indicators on the MLP architecture, and its versatility also indicates that it can be widely used in various neural network structures such as convolutional networks and Transformers.

Keywords: Nash normalization; game-theoretic approach; neural network normalization; strategic feedback; deep learning stability

1. Introduction

In recent years, in deep learning research, normalization technology has become one of the core mechanisms to improve the learning efficiency and generalization ability of neural networks [1–4]. Specifically, studies have shown that in various architectures such as multilayer perceptrons, convolutional neural networks, and recurrent neural networks, appropriate normalization strategies can significantly reduce the gradient vanishing problem and gradient exploding problem, accelerate the convergence process of the model, and enhance the robustness to changes in input distribution [5–11]. Batch normalization is the most representative technology and has been widely used in various deep neural network structures [12–17]. Its basic idea is to make the input distribution of each layer more stable by normalizing the neuron activation values in each learning batch. This technology not only significantly speeds up the learning process, but also reduces the dependence on initialization and learning rate [18–22].

Although standard batch normalization performs well in practice, it still has certain limitations in theoretical modeling. Its main premise is that the normalization of each neuron can be completed independently without considering the state of other neurons [23–25]. This approach ignores the basic fact that there are complex interactions and dependencies between neurons in multilayer neural networks. In practical systems, neurons should not be viewed as independent processing units, but as participants that jointly influence and evolve in the same ecosystem. This synergistic relationship may lead to information redundancy, local overfitting, and even mode collapse in the learning process, which cannot be effectively modeled by standard batch normalization [26–29].

To this end, this paper rethinks the mechanism of neuron normalization from the perspective of game theory. Specifically, we introduce the Nash equilibrium theory [30–37] as a theoretical basis

 doi:10.20944/preprints202506.2531.v1

and propose a new "Nash normalization" mechanism. Nash equilibrium is one of the core concepts in non-cooperative games. It represents an equilibrium state in which each participant in a system with multiple rational decision makers chooses its own optimal strategy while fully considering the choices of other individuals. In this state, no individual can obtain a better result by unilaterally changing the strategy.

Introducing this concept into the normalization mechanism enables neurons to clearly identify the behavior of other neurons during the normalization process and make "rational responses" instead of treating them as independent individuals. Specifically, by introducing an interactive feedback mechanism based on standard batch normalization, the normalized output of each neuron is affected not only by its own distribution, but also by the collective feedback of the activation states of other neurons during the entire batch normalization process. This mechanism effectively suppresses the abnormal amplification effect of individual neurons caused by random initialization in the early stage of learning, and promotes the convergence of the entire network to a more stable and cooperative trajectory.

The "Nash regularization" mechanism proposed in this paper is not only applicable to the traditional multi-layer perceptron architecture (MLP), but also has good versatility and scalability. It can be seamlessly integrated into various architectures such as convolutional neural networks and graph neural networks, making it a versatile learning optimization module. By modeling the behavior of neurons as participants in non-cooperative games, we provide a new perspective for neural network learning from "individual independent optimization" to "collective rational equilibrium". This transformation is not only innovative in theory, but also shows the potential to improve model performance and stability in practical applications.

In summary, the research goal of this paper is to construct a regularization mechanism based on the idea of Nash equilibrium, explore the potential synergy between neurons, and verify its effectiveness in a variety of deep learning tasks. Through this attempt, we hope to provide new ideas for the learning mechanism of deep neural networks and expand the application scope of game theory in the field of machine learning.

2. Conception and Implementation of Nash Normalization Mechanism

In complex neural network systems, the behavior of individual neurons is often affected by the state of other neurons. Although traditional normalization methods, such as batch normalization, have achieved remarkable results in engineering practice, they assume that the normalization of the activation value of each neuron is independent of each other, that is, the normalization process of a neuron does not consider the state changes of other neurons. This processing method may ignore the potential synergistic or competitive relationship between neurons in some cases, especially in deep networks, where multiple neurons may share some functions or information paths, resulting in coupling effects on each other's activation values.

In order to more reasonably model this potential synergistic mechanism, this paper proposes a new normalization strategy - "Nash normalization", which is inspired by the Nash equilibrium theory in game theory. The theory points out that in a system with multiple rational decision makers, when each participant takes the best response to the strategies of others, the entire system will reach a stable equilibrium state. On this basis, we compare each neuron in the neural network to an individual with "strategic awareness". Its activation normalization process is no longer a static, isolated operation, but is based on the perception and feedback adjustment of the states of other neurons, thereby achieving dynamic equilibrium at the system level.

More specifically, we propose the following three core concepts:

Neurons as rational subjects: Each neuron is modeled as a "game-playing" subject in the normalization process, and its goal is to obtain a more coordinated performance in the overall network by adjusting its own state. Different from the operation of "standardizing its own data" in traditional normalization, this subject needs to consider the behavioral feedback of surrounding neurons when adjusting its own activation value.

 doi:10.20944/preprints202506.2531.v1

Strategic feedback mechanism in normalization: We introduce a mutual influence structure between neurons, and use statistical methods (such as synergistic change analysis) to construct an "interaction feedback graph" between neurons, which reflects the extent to which the activation change of a neuron affects the state of other neurons. In the normalization operation, the information of this feedback graph is used as a "reference value" or "regulatory factor" to dynamically adjust the normalization process of each neuron, thereby achieving a more organized and coordinated feature representation.

Dynamic equilibrium evolution of activation distribution: The goal of traditional batch normalization is to make the activation distribution of neurons tend to the standard normal form, while Nash normalization pursues "cooperative stability" - that is, under the interaction of multiple neurons, the activation distribution of the entire neuron set reaches a non-mandatory but inherently coordinated equilibrium state. This distribution may not be completely symmetrical or completely standardized, but it has good stability and generalization capabilities, and is closer to the structural characteristics of complex input distributions in real tasks.

At the implementation level, Nash normalization did not completely subvert the original technical framework of batch normalization, but made structural enhancements to it. We still retain the standardization steps of the mean and variance to ensure basic numerical stability and trainability; at the same time, an additional "feedback regulation module" is added to the process. This module automatically generates a fine-tuning signal based on the interaction relationship between neurons to moderately correct the original normalization result. This mechanism is similar to introducing a readjustment based on collective behavior after each activation normalization, so that the final output can better reflect the dynamic balance relationship between neurons.

It should be pointed out that the proposal of Nash normalization does not depend on a specific network structure or model type. Since its core concept is based only on the normalization operation in the feedforward stage, it can be flexibly integrated into different types of neural network architectures, whether it is a fully connected neural network (MLP), a convolutional neural network (CNN), a graph neural network (GNN) or a transformer model (Transformer). This universality makes Nash normalization not only theoretically valuable for promotion, but also has the potential for practical engineering applications.

In summary, Nash normalization introduces the "strategy feedback" mechanism in game theory on the basis of maintaining the numerical stability of traditional batch normalization, breaking the isolation of normalization calculations between neurons and promoting a more coordinated activation expression. We believe that this normalization method based on collective rationality will bring new inspiration to feature expression and optimization strategies in deep learning.

3. Experimental Design and Evaluation

To verify the effectiveness of the proposed mechanism, we selected a subset of the CIFAR-10 dataset for experiments and compared the performance of Standard Batch Normalization (StandardMLP) and Nash Normalization (NashMLP) under the same network architecture.

The evaluation indicators include accuracy, precision, recall, F1 value, ROC AUC, and training time. All models were run ten times under the same hyperparameter settings, and the average and standard deviation were taken for comparative analysis.

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

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import torchvision.transforms as transforms
from torch.utils.data import Subset, DataLoader
```

 doi:10.20944/preprints202506.2531.v1


```
import numpy as np
import time
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score

# Set random seed
torch.manual_seed(0)
np.random.seed(0)

# Device
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Load CIFAR-10 (800 train, 800 test)
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

train_dataset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
test_dataset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)

train_subset = Subset(train_dataset, range(800))
test_subset = Subset(test_dataset, range(800))

train_loader = DataLoader(train_subset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_subset, batch_size=64, shuffle=False)

# MLP config
input_size = 3 * 32 * 32
hidden_size = 256
num_classes = 10

# Standard MLP with BatchNorm
class StandardMLP(nn.Module):
    def __init__(self):
        super(StandardMLP, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.bn1 = nn.BatchNorm1d(hidden_size)
        self.fc2 = nn.Linear(hidden_size, hidden_size)
        self.bn2 = nn.BatchNorm1d(hidden_size)
        self.fc3 = nn.Linear(hidden_size, num_classes)
```

 doi:10.20944/preprints202506.2531.v1


```
    def forward(self, x):
        x = x.view(-1, input_size)
        x = F.relu(self.bn1(self.fc1(x)))
        x = F.relu(self.bn2(self.fc2(x)))
        x = self.fc3(x)
        return x

# Nash BatchNorm (innovative)
class NashBatchNorm1d(nn.Module):
    def __init__(self, num_features, eps=1e-5, momentum=0.1):
        super(NashBatchNorm1d, self).__init__()
        self.gamma = nn.Parameter(torch.ones(num_features))
        self.beta = nn.Parameter(torch.zeros(num_features))
        self.eps = eps
        self.momentum = momentum
        self.register_buffer('running_mean', torch.zeros(num_features))
        self.register_buffer('running_var', torch.ones(num_features))

    def forward(self, x):
        if self.training:
            batch_mean = x.mean(0)
            batch_var = x.var(0, unbiased=False)

            self.running_mean = self.momentum * batch_mean + (1 - self.momentum) *
self.running_mean
            self.running_var = self.momentum * batch_var + (1 - self.momentum) * self.running_var

            x_hat = (x - batch_mean) / torch.sqrt(batch_var + self.eps)
        else:
            x_hat = (x - self.running_mean) / torch.sqrt(self.running_var + self.eps)

        interaction = torch.matmul(x_hat, x_hat.T) / x_hat.size(1)
        strategic_feedback = torch.mean(interaction, dim=1, keepdim=True)
        x_nash = x_hat - strategic_feedback

        return self.gamma * x_nash + self.beta

# MLP with NashBN
class NashMLP(nn.Module):
    def __init__(self):
        super(NashMLP, self).__init__()
```

 doi:10.20944/preprints202506.2531.v1


```
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.bn1 = NashBatchNorm1d(hidden_size)
        self.fc2 = nn.Linear(hidden_size, hidden_size)
        self.bn2 = NashBatchNorm1d(hidden_size)
        self.fc3 = nn.Linear(hidden_size, num_classes)

    def forward(self, x):
        x = x.view(-1, input_size)
        x = F.relu(self.bn1(self.fc1(x)))
        x = F.relu(self.bn2(self.fc2(x)))
        x = self.fc3(x)
        return x

# Train function
def train_model(model, optimizer, criterion, train_loader, num_epochs=5):
    model.train()
    for _ in range(num_epochs):
        for images, labels in train_loader:
            images, labels = images.to(device), labels.to(device)

            outputs = model(images)
            loss = criterion(outputs, labels)

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

# Evaluation
def evaluate_model(model, test_loader):
    model.eval()
    y_true, y_pred, y_prob = [], [], []

    with torch.no_grad():
        for images, labels in test_loader:
            images = images.to(device)
            outputs = model(images)
            probs = F.softmax(outputs, dim=1)
            _, predicted = torch.max(outputs.data, 1)

            y_true.extend(labels.numpy())
            y_pred.extend(predicted.cpu().numpy())
            y_prob.extend(probs.cpu().numpy())
```

 doi:10.20944/preprints202506.2531.v1


```
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)
    y_prob = np.array(y_prob)

    acc = accuracy_score(y_true, y_pred)
    prec = precision_score(y_true, y_pred, average='macro', zero_division=0)
    rec = recall_score(y_true, y_pred, average='macro', zero_division=0)
    f1 = f1_score(y_true, y_pred, average='macro', zero_division=0)
    try:
        auc = roc_auc_score(y_true, y_prob, multi_class='ovr')
    except:
        auc = float('nan')
    return acc, prec, rec, f1, auc

# Run experiment 10 times
def run_experiments():
    for model_name, ModelClass in [('StandardMLP', StandardMLP), ('NashMLP', NashMLP)]:
        print(f"\n===== {model_name} =====")
        metrics_all = []

        for run in range(10):
            model = ModelClass().to(device)
            optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
            criterion = nn.CrossEntropyLoss()

            start_time = time.time()
            train_model(model, optimizer, criterion, train_loader, num_epochs=5)
            duration = time.time() - start_time

            acc, prec, rec, f1, auc = evaluate_model(model, test_loader)
            metrics_all.append([acc, prec, rec, f1, auc, duration])

            print(f"Run {run+1:2d} | Acc: {acc:.4f} | Prec: {prec:.4f} | Rec: {rec:.4f} | F1: {f1:.4f} | AUC:
{auc:.4f} | Time: {duration:.2f}s")

        metrics_np = np.array(metrics_all)
        means = metrics_np.mean(axis=0)
        stds = metrics_np.std(axis=0)

        print("\n----- Final Results -----")
        print(f"Average Accuracy     : {means[0]:.4f} ± {stds[0]:.4f}")
```

 doi:10.20944/preprints202506.2531.v1


```
    print(f"Average Precision   : {means[1]:.4f} ± {stds[1]:.4f}")
    print(f"Average Recall      : {means[2]:.4f} ± {stds[2]:.4f}")
    print(f"Average F1 Score    : {means[3]:.4f} ± {stds[3]:.4f}")
    print(f"Average ROC AUC     : {means[4]:.4f} ± {stds[4]:.4f}")
    print(f"Average Time (s)    : {means[5]:.2f} ± {stds[5]:.2f}")

run_experiments()
```

The output of the experimental code is as follows:

```
===== StandardMLP =====
----- Final Results -----
Average Accuracy    : 0.3090 ± 0.0088
Average Precision   : 0.3080 ± 0.0089
Average Recall      : 0.3055 ± 0.0082
Average F1 Score    : 0.2997 ± 0.0086
Average ROC AUC     : 0.7716 ± 0.0070
Average Time (s)    : 1.62 ± 0.11
===== NashMLP =====
----- Final Results -----
Average Accuracy    : 0.3191 ± 0.0143
Average Precision   : 0.3155 ± 0.0139
Average Recall      : 0.3143 ± 0.0136
Average F1 Score    : 0.3089 ± 0.0152
Average ROC AUC     : 0.7724 ± 0.0068
Average Time (s)    : 1.93 ± 0.17
```

The experimental results show that the Nash normalization version shows better generalization ability in all indicators. Especially in terms of precision, recall and F1 value, the improvement is particularly significant, indicating that the introduction of game modeling mechanism can improve the model's ability to distinguish different categories. More importantly, this improvement is achieved without changing the main structure of the network, demonstrating the lightweight and universality of this method.

4. Consideration of the Universality of the Method

This study selected the multi-layer perceptron (MLP) as the experimental platform to verify the feasibility of the Nash regularization mechanism, but it should be emphasized that the proposed method has a high degree of structural independence and wide adaptability. Its core idea is not limited to a specific neural network architecture, and can be flexibly migrated to various mainstream deep learning models, such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), graph neural networks (GNNs), and even the Transformer architecture that has been widely used in recent years.

In the CNN scenario, the structural characteristics of neurons show a high degree of local correlation in the spatial dimension, which allows the Nash regularization mechanism to be further refined into a "local game" mode. In other words, this cannot be achieved by simply relying on full-

 doi:10.20944/preprints202506.2531.v1

channel or batch feedback modeling. The introduction of local perception windows makes it possible to form more fine-grained strategic interactions between neurons in the same receptive field and more accurately reflect the collaborative characteristics in the spatial structure. We believe that this extended method is particularly effective in tasks that are highly sensitive to local information, such as object detection and image segmentation.

In sequence modeling networks such as recurrent neural networks (RNNs) and long short-term memory networks (LSTMs), information propagation in the time dimension has a natural sequence dependency. By introducing Nash regularization mechanisms in such structures, we can build a "time-step policy feedback system". This allows the neuron state at a specific moment to be affected not only by other units in the current time step, but also by the normalized results of past time steps, thereby achieving a modeling method with "cross-time policy stability". This design is expected to improve the stability of the sequence modeling process and reduce long-term dependency problems such as gradient vanishing and explosion.

In graph neural networks (GNNs), the relationship between nodes is non-Euclidean, and the feature update of each node depends on the information propagation of neighboring nodes. By introducing Nash regularization in GNNs, we can build a natural policy feedback mechanism based on the graph structure itself. That is, the regularized behavior of each node responds to the feature distribution of its neighboring nodes, forming a local Nash game driven by the graph topology. This approach is particularly suitable for scenarios that require fine structure recognition, such as social networks and molecular modeling.

In the Transformer model, the attention mechanism itself has the ability to integrate global information. Nash regularization is highly compatible with this, and further integrates the implicit dependency of attention weights into the regularization process to form a regularized feedback network between layers and positions. This not only improves the modeling accuracy of the model for long-range dependencies, but also serves as a natural buffer mechanism to combat attention noise, improving the stability of information fusion from the regularization dimension.

In addition, Nash regularization also has certain anti-interference and robustness enhancement functions. In regularization, each neuron no longer relies solely on its own data points, but actively identifies and refers to the states of other neurons, so that even if a neuron has a significant deviation due to noise or abnormal input, its influence will be suppressed by collective behavior, and the overall regularization will be less likely to be unbalanced. This "collective correction" function is extremely robust when facing noisy samples, low-quality data, or adversarial attack scenarios. Especially in practical applications, training data often has problems such as label errors and collective bias. Nash regularization provides a solution to adjust individual biases and collective behavior.

Therefore, the versatility of Nash regularization is not only reflected in its adaptability to various network structures, but also in its strategies for dealing with various practical challenges (such as high-noise environments, uneven distributions, and drastic input fluctuations). Nash regularization provides a more strategic and collectively intelligent regularization paradigm, bringing new possibilities for the optimization of complex deep models in the future.

5. Conclusion and Future Outlook

Regularization mechanism is an important component of deep learning models and is widely used due to its positive effects such as numerical stability, improved convergence speed and dynamic learning adjustment. The "Nash regularization" mechanism proposed in this paper further pushes this classic operation to the realm of strategic intelligence. It deeply integrates the theoretical framework of "Nash equilibrium" in game theory into the learning process of neural networks, paving the way for regularized modeling based on multi-agent interactive feedback.

Nash regularization realizes a new regularization paradigm by modeling each neuron as an individual with a "rational strategy". The regularization operation is not simply normalizing specific activations, but is integrated into a dynamic game system so that all neurons can participate in a cooperative equilibrium state. Experimental results show that this mechanism not only improves the

 doi:10.20944/preprints202506.2531.v1

stability of model learning and accelerates the convergence process, but also significantly improves the generalization ability and anti-interference performance of the model in the test phase. Especially in complex data distribution and non-stationary input environments, Nash regularization shows robustness advantages that traditional methods do not have.

More importantly, the method has good scalability and theoretical developability. It is redesigned based on the universal behavior of regularization and does not depend on a specific model structure, so it has good cross-structure adaptability. This wide applicability lays a solid foundation for future expansion and practice.

In the future, we believe that the research and application of Nash regularization is also worth exploring in the following directions:

Multi-architecture cross-domain verification and promotion: Extend the mechanism to more types of neural networks, such as deep residual networks (ResNet), visual transformers (ViT), and graph attention networks (GAT), and verify its performance consistency in various task scenarios.

Dynamic feedback modeling mechanism optimization: By introducing more sensitive policy feedback weight mechanisms, such as dynamic adjustment strategies based on attention distribution, adaptive adjacency matrices, and reinforcement learning signals, the behavioral intelligence of the regularization stage is further improved.

Fusion learning framework integration: Integrate the Nash regularization mechanism into a wider range of learning paradigms such as meta-learning, federated learning, and multi-task learning to build a "multi-agent regularized collaborative system" for distributed and heterogeneous environments.

Theoretical modeling and interpretation analysis: From the perspective of information theory or game theory, the behavioral stability, convergence conditions and optimality of Nash regularization are systematically modeled and mathematically explained, laying the foundation for subsequent theoretical research.

In summary, Nash regularization, as a new mechanism that integrates strategic thinking and regularization technology, not only provides a more intelligent and systematic information processing solution, but also brings a new perspective of game intelligence to the development of deep neural networks. With the continuous optimization and expansion of related mechanisms, its position in future intelligent systems will become increasingly important, and it is expected to become a new important milestone in the field of deep learning regularization.

References

1. Zheng, S., Gao, Z., Sun, F. K., Boning, D., Yu, B., & Wong, M. D. (2024). Improving neural ode training with temporal adaptive batch normalization. Advances in Neural Information Processing Systems, 37, 95875-95895.
2. Kim, Y., & Panda, P. (2021). Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. Frontiers in neuroscience, 15, 773954.
3. Lakshmi, T. V., & Krishna Reddy, C. V. (2024). Classification of skin lesions by incorporating drop-block and batch normalization layers in representative CNN models. Arabian Journal for Science and Engineering, 49(3), 3671-3684.
4. Lubana, E. S., Dick, R., & Tanaka, H. (2021). Beyond batchnorm: Towards a unified understanding of normalization in deep learning. Advances in Neural Information Processing Systems, 34, 4778-4791.
5. Talat, N., Alsadoon, A., Prasad, P. W. C., Dawoud, A., Rashid, T. A., & Haddad, S. (2023). A novel enhanced normalization technique for a mandible bones segmentation using deep learning: batch normalization with the dropout. Multimedia Tools and Applications, 82(4), 6147-6166.

 doi:10.20944/preprints202506.2531.v1

6. Wu, Y., Chi, Z., Wang, Y., Plataniotis, K. N., & Feng, S. (2024, March). Test-time domain adaptation by learning domain-aware batch normalization. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 38, No. 14, pp. 15961-15969).
7. Gao, S. H., Han, Q., Li, D., Cheng, M. M., & Peng, P. (2021). Representative batch normalization with feature calibration. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 8669-8679).
8. Yi-Ge, E., Wu, M., & Chen, Z. (2025, April). Reducing Divergence in Batch Normalization for Domain Adaptation. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 39, No. 21, pp. 22155-22163).
9. Brock, A., De, S., Smith, S. L., & Simonyan, K. (2021, July). High-performance large-scale image recognition without normalization. In International conference on machine learning (pp. 1059-1071). PMLR.
10. Hu, F., Chen, A. A., Horng, H., Bashyam, V., Davatzikos, C., Alexander-Bloch, A., ... & Shinohara, R. T. (2023). Image harmonization: A review of statistical and deep learning methods for removing batch effects and evaluation metrics for effective harmonization. NeuroImage, 274, 120125.
11. Liao, Z., Hezbri, N., Quétu, V., Nguyen, V. T., & Tartaglione, E. (2025, April). Till the Layers Collapse: Compressing a Deep Neural Network through the Lenses of Batch Normalization Layers. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 39, No. 18, pp. 18702-18710).
12. Mirza, M. J., Micorek, J., Possegger, H., & Bischof, H. (2022). The norm must go on: Dynamic unsupervised domain adaptation by normalization. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 14765-14775).
13. Chai, J., Zeng, H., Li, A., & Ngai, E. W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. Machine Learning with Applications, 6, 100134.
14. Choi, S., Kim, T., Jeong, M., Park, H., & Kim, C. (2021). Meta batch-instance normalization for generalizable person re-identification. In Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (pp. 3425-3435).
15. Luo, X., Meng, Q., Chen, W., Wang, Y., & Liu, T. Y. (2021, December). Path-BN: Towards effective batch normalization in the Path Space for ReLU networks. In Uncertainty in Artificial Intelligence (pp. 834-843). PMLR.
16. Saeedi, S., Rezayi, S., Keshavarz, H., & R. Niakan Kalhori, S. (2023). MRI-based brain tumor detection using convolutional deep learning methods and chosen machine learning techniques. BMC Medical Informatics and Decision Making, 23(1), 16.
17. Shah, D., Trivedi, V., Sheth, V., Shah, A., & Chauhan, U. (2022). ResTS: Residual deep interpretable architecture for plant disease detection. Information Processing in Agriculture, 9(2), 212-223.
18. Fekri, M. N., Patel, H., Grolinger, K., & Sharma, V. (2021). Deep learning for load forecasting with smart meter data: Online Adaptive Recurrent Neural Network. Applied Energy, 282, 116177.
19. Nayoga, B. P., Adipradana, R., Suryadi, R., & Suhartono, D. (2021). Hoax analyzer for Indonesian news using deep learning models. Procedia Computer Science, 179, 704-712.
20. Chattopadhyay, A., & Maitra, M. (2022). MRI-based brain tumour image detection using CNN based deep learning method. Neuroscience informatics, 2(4), 100060.
21. Rahman, T., & Islam, M. S. (2023). MRI brain tumor detection and classification using parallel deep convolutional neural networks. Measurement: Sensors, 26, 100694.
22. Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., & Ahmad, R. (2022). CNN-LSTM: hybrid deep neural network for network intrusion detection system. IEEE Access, 10, 99837-99849.

 doi:10.20944/preprints202506.2531.v1

23. Liu, H., Ma, R., Li, D., Yan, L., & Ma, Z. (2021). Machinery fault diagnosis based on deep learning for time series analysis and knowledge graphs. journal of signal processing systems, 93, 1433-1455.
24. Jain, P. K., Saravanan, V., & Pamula, R. (2021). A hybrid CNN-LSTM: A deep learning approach for consumer sentiment analysis using qualitative user-generated contents. Transactions on Asian and Low-Resource Language Information Processing, 20(5), 1-15.
25. Ya, T. U., Yun, L. I. N., Haoran, Z. H. A., Yu, W. A. N. G., Guan, G. U. I., & Shiwen, M. A. O. (2022). Large-scale real-world radio signal recognition with deep learning. Chinese Journal of Aeronautics, 35(9), 35-48.
26. Kang, M., Cho, I., Park, J., Jeong, J., Lee, K., Lee, B., ... & Park, I. (2022). High accuracy real-time multi-gas identification by a batch-uniform gas sensor array and deep learning algorithm. ACS sensors, 7(2), 430-440.
27. Smadi, A. A., Abugabah, A., Al-Smadi, M. K., & Al-Smadi, A. M. (2024). Smart medical application of deep learning (MUNet) for detection of COVID-19 from chest images. Journal of Wireless Mobile Networks Ubiquitous Computing and Dependable Applications, 15(1), 133-153.
28. Majumder, S., & Kehtarnavaz, N. (2021). Multitasking deep learning model for detection of five stages of diabetic retinopathy. IEEE Access, 9, 123220-123230.
29. Zhang, Y. D., Satapathy, S. C., Liu, S., & Li, G. R. (2021). A five-layer deep convolutional neural network with stochastic pooling for chest CT-based COVID-19 diagnosis. Machine vision and applications, 32, 1-13.
30. Hsieh, Y. G., Antonakopoulos, K., & Mertikopoulos, P. (2021, July). Adaptive learning in continuous games: Optimal regret bounds and convergence to Nash equilibrium. In Conference on Learning Theory (pp. 2388-2422). PMLR.
31. Bakhtyar, B., Qi, Z., Azam, M., & Rashid, S. (2023). Global declarations on electric vehicles, carbon life cycle and Nash equilibrium. Clean Technologies and Environmental Policy, 25(1), 21-34.
32. Qian, Y. Y., Liu, M., Wan, Y., Lewis, F. L., & Davoudi, A. (2021). Distributed adaptive Nash equilibrium solution for differential graphical games. IEEE Transactions on Cybernetics, 53(4), 2275-2287.
33. Ye, M., Han, Q. L., Ding, L., & Xu, S. (2023). Distributed Nash equilibrium seeking in games with partial decision information: A survey. Proceedings of the IEEE, 111(2), 140-157.
34. Wang, Z., Liu, F., Ma, Z., Chen, Y., Jia, M., Wei, W., & Wu, Q. (2021). Distributed generalized Nash equilibrium seeking for energy sharing games in prosumers. IEEE Transactions on Power Systems, 36(5), 3973-3986.
35. Saxena, N., Sarkar, B., Wee, H. M., Reong, S., Singh, S. R., & Hsiao, Y. L. (2023). A reverse logistics model with eco-design under the Stackelberg-Nash equilibrium and centralized framework. Journal of Cleaner Production, 387, 135789.
36. Zou, Y., Huang, B., Meng, Z., & Ren, W. (2021). Continuous-time distributed Nash equilibrium seeking algorithms for non-cooperative constrained games. Automatica, 127, 109535.
37. Ye, M., Li, D., Han, Q. L., & Ding, L. (2022). Distributed Nash equilibrium seeking for general networked games with bounded disturbances. IEEE/CAA Journal of Automatica Sinica, 10(2), 376-387.