A Novel Batch Normalization Approach Inspired by Harsanyi's Aggregation Theorem with Dynamic Feature Attention

Jincheng Zhang *

Posted Date: 14 July 2025

doi: 10.20944/preprints202507.1073.v1

Keywords: Harsanyi aggregation theorem; batch normalization; feature attention; neural network generalization; dynamic weighting mechanism

 doi:10.20944/preprints202507.1073.v1

Article

A Novel Batch Normalization Approach Inspired by Harsanyi's Aggregation Theorem with Dynamic Feature Attention

Jincheng Zhang

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

Abstract

With the continuous development of neural network structures, batch normalization, as a standardization technique, has been widely used in various deep learning models to accelerate convergence and improve model stability. This paper proposes a novel mechanism: introducing the idea of Harsanyi aggregation theorem into batch normalization, constructing a normalization method with attention regulation, so that the model has stronger expressiveness and adaptability when processing feature normalization. This method has not only achieved preliminary verification results in multi-layer perceptrons (MLP), but also has the potential to be promoted in other architectures (such as CNN, Transformer, etc.).

Keywords: Harsanyi aggregation theorem; batch normalization; feature attention; neural network generalization; dynamic weighting mechanism

1. Introduction

Batch normalization, as a key technology of deep neural networks, has been widely used in various models in recent years, especially in accelerating learning, alleviating the gradient vanishing problem, and improving model generalization ability [1–4]. Its basic idea is to standardize the feature distribution of each layer input, thereby reducing the internal covariate shift associated with updating network parameters and promoting network learning [5–10]. However, although this normalization method has achieved remarkable success in multiple tasks, it still has an important potential problem: it treats all channels or feature dimensions equally and does not fully consider the relative importance of each feature in a specific task [11–13].

In practical applications, the roles of various features in model decisions often vary greatly [14–17]. Some features may play an important role in the final classification, prediction, or generation results, while other features may only make a small contribution [18–23]. However, traditional batch normalization methods treat all features as equally important and lack a mechanism to dynamically evaluate and adjust the feature "decision value", which to some extent limits the further improvement of network performance [24–28].

To this end, this paper introduces the main idea of the Hasani Aggregation Theorem [29–35] from the perspective of game theory and proposes an improved strategy. The theorem was originally used to measure the marginal contribution of individuals in collective decision-making and is widely used in collective game modeling in the fields of economics, political science and artificial intelligence. Inspired by this, this paper aims to construct a "marginal feature contribution identification mechanism" by introducing this idea into the normalization stage of deep neural networks to give differentiated importance assessments to different features. Specifically, we design a weight structure based on the attention mechanism on the basis of the traditional batch normalization module, and

 doi:10.20944/preprints202507.1073.v1

dynamically adjust the weight according to the potential marginal contribution of the normalized features. This enables the network to adaptively strengthen or weaken certain key features according to the contextual information of the input samples, thereby achieving more accurate feature representation.

This mechanism not only fills the gap in batch normalization at the feature weight allocation level, but also provides a new idea for the integration of game theory principles and deep learning mechanisms. More importantly, this mechanism has good versatility and portability. It can be applied not only to traditional multi-layer perceptrons (MLPs), but also to convolutional neural networks (CNNs), graph neural networks (GNNs), and Transformer models based on attention mechanisms, showing wide adaptability across structural architectures.

In summary, this study aims to introduce decision-making intelligence elements into the traditional batch normalization method, simulate the differences in the contributions of various features to model predictions in actual tasks, improve the semantic expression ability of the model normalization strategy, and achieve performance optimization on multiple standard evaluation indicators. The experimental results also preliminarily prove the effectiveness and stability of the mechanism in different model architectures.

2. Method Design

This study proposes an improved batch normalization module inspired by the Harsanyi aggregation theorem, referred to as Harsanyi-BN. The core idea is to introduce a feature-level attention regulation mechanism based on traditional Batch Normalization to dynamically reflect the relative importance of different features in model decision making. This mechanism constructs a discriminative normalization method by simulating the concept of "marginal contribution" in game theory, so that the model no longer treats all features equally during the normalization process, but makes weighted adjustments based on the actual influence of each feature.

In terms of implementation structure, the Harsanyi-BN module mainly consists of two parts: one is the traditional batch normalization submodule, which is used to standardize the input features so that they have zero mean and unit variance, thereby stabilizing the training process; the other is a lightweight feature attention submodule, which is composed of a shallow feedforward network and can calculate a set of attention weights based on the original input features. This attention mechanism outputs the dynamic coefficients of each dimensional feature with a Sigmoid activation function to measure its "marginal contribution" and thus weight the normalized output.

The introduction of this weighted method makes the normalization operation no longer a simple linear transformation process, but a feature reorganization process closely related to the task goal. In other words, Harsanyi-BN is not only a numerical normalization process, but also an information filtering mechanism guided by game theory. It can adaptively identify and enhance those feature dimensions that have a greater impact on the model's prediction results, while suppressing invalid or redundant parts, thereby achieving more accurate and efficient feature representation learning.

In addition, the module has strong structural versatility and engineering replaceability. Because its internal logic is highly similar to the traditional BatchNorm module structure, it can be seamlessly integrated into any existing neural network architecture with only a small change in the code level. Whether in multi-layer perceptrons (MLP), convolutional neural networks (CNN), graph neural networks (GNN), or more complex Transformer structures, Harsanyi-BN can be used as a direct replacement for batch normalization layers. Its design does not depend on the specific form or number of features of the input data, nor does it require major changes to the main network structure, so it has good portability and deployment convenience in practical applications.

Furthermore, the Harsanyi-BN module is highly modular in construction and can be integrated with other attention mechanisms (such as channel attention, spatial attention, multi-head attention, etc.) to form a more complex and flexible feature enhancement network component. This extensibility provides the possibility for exploring the integration of normalization and intelligent feature selection

 doi:10.20944/preprints202507.1073.v1

in deep models in the future, and also lays a methodological foundation for building model structures with higher expressiveness and decision-making adaptability.

In summary, the Harsanyi-BN module not only improves the expressiveness of traditional batch normalization, but also successfully introduces the theoretical ideas of game theory into the normalization mechanism of neural networks, promotes the development direction of intelligent feature selection and dynamic normalization mechanism, and brings new possibilities for the structural design of deep learning models.

3. Experimental Design

In order to verify the effectiveness of this mechanism, we constructed two groups of comparative experiments: one group is a traditional MLP model using standard batch normalization; the other group uses the Harsanyi-BN module proposed by us under the same network structure. The experiment uses a randomly selected subset of the CIFAR-10 dataset (500 training samples and 500 test samples). Each model is repeated for ten rounds, and the accuracy, precision, recall, F1 score and AUROC indicators are recorded and averaged.

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
import torch.nn.functional as F
import time
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

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

# ============================
# Data Preparation (500 Train / 500 Test)
# ============================
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

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

train_loader = DataLoader(Subset(train_set, range(500)), batch_size=64, shuffle=True)
test_loader = DataLoader(Subset(test_set, range(500)), batch_size=64, shuffle=False)
```

 doi:10.20944/preprints202507.1073.v1


```
# ============================
# Baseline MLP with BatchNorm
# ============================
class BaselineMLP(nn.Module):
    def __init__(self):
        super(BaselineMLP, self).__init__()
        self.fc1 = nn.Linear(3 * 32 * 32, 512)
        self.bn1 = nn.BatchNorm1d(512)
        self.fc2 = nn.Linear(512, 256)
        self.bn2 = nn.BatchNorm1d(256)
        self.fc3 = nn.Linear(256, 10)

    def forward(self, x):
        x = x.view(x.size(0), -1)
        x = F.relu(self.bn1(self.fc1(x)))
        x = F.relu(self.bn2(self.fc2(x)))
        return self.fc3(x)

# ======================================
# Harsanyi-inspired BatchNorm-enhanced MLP
# ======================================
class HarsanyiBatchNorm1d(nn.Module):
    def __init__(self, num_features):
        super(HarsanyiBatchNorm1d, self).__init__()
        self.bn = nn.BatchNorm1d(num_features)
        self.attn = nn.Sequential(
            nn.Linear(num_features, num_features),
            nn.Sigmoid()
        )

    def forward(self, x):
        bn_out = self.bn(x)
        attn_weights = self.attn(x)
        return bn_out * attn_weights

class HarsanyiBNMLP(nn.Module):
    def __init__(self):
        super(HarsanyiBNMLP, self).__init__()
        self.fc1 = nn.Linear(3 * 32 * 32, 512)
        self.hbn1 = HarsanyiBatchNorm1d(512)
        self.fc2 = nn.Linear(512, 256)
```

 doi:10.20944/preprints202507.1073.v1


```
        self.hbn2 = HarsanyiBatchNorm1d(256)
        self.fc3 = nn.Linear(256, 10)

    def forward(self, x):
        x = x.view(x.size(0), -1)
        x = F.relu(self.hbn1(self.fc1(x)))
        x = F.relu(self.hbn2(self.fc2(x)))
        return self.fc3(x)

# ============================
# Training & Evaluation Utils
# ============================
def train_model(model, loader, optimizer, criterion):
    model.train()
    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()

def evaluate_model(model, loader):
    model.eval()
    preds, targets = [], []
    with torch.no_grad():
        for images, labels in loader:
            images = images.to(device)
            outputs = model(images)
            preds.append(outputs.cpu())
            targets.append(labels)
    preds = torch.cat(preds).argmax(dim=1).numpy()
    targets = torch.cat(targets).numpy()
    one_hot_preds = F.one_hot(torch.tensor(preds), num_classes=10).numpy()

    return {
        'Accuracy': accuracy_score(targets, preds),
        'Precision': precision_score(targets, preds, average='macro', zero_division=0),
        'Recall': recall_score(targets, preds, average='macro', zero_division=0),
        'F1': f1_score(targets, preds, average='macro', zero_division=0),
        'AUROC': roc_auc_score(targets, one_hot_preds, average='macro', multi_class='ovr')
    }
```

 doi:10.20944/preprints202507.1073.v1


```
# ============================
# Run 10 Experiments & Aggregate
# ============================
def run_multiple_experiments(model_class, name, runs=10):
    print(f"\n=== Running: {name} for {runs} runs ===")
    all_metrics = {k: [] for k in ['Accuracy', 'Precision', 'Recall', 'F1', 'AUROC']}
    train_times, eval_times = [], []

    for i in range(runs):
        print(f"\n--- Run {i+1} ---")
        model = model_class().to(device)
        optimizer = optim.Adam(model.parameters(), lr=0.001)
        criterion = nn.CrossEntropyLoss()

        start_train = time.time()
        for epoch in range(3):   # short training for speed
            train_model(model, train_loader, optimizer, criterion)
        end_train = time.time()

        start_eval = time.time()
        metrics = evaluate_model(model, test_loader)
        end_eval = time.time()

        train_time = end_train - start_train
        eval_time = end_eval - start_eval
        train_times.append(train_time)
        eval_times.append(eval_time)

        for k, v in metrics.items():
            all_metrics[k].append(v)
            print(f"[{name}][Run {i+1}] {k}: {v:.4f}")

        print(f"[{name}][Run {i+1}] Training Time: {train_time:.2f}s")
        print(f"[{name}][Run {i+1}] Inference Time: {eval_time:.2f}s")

    print(f"\n=== Final Results for {name} ===")
    for k in all_metrics:
        mean_val = np.mean(all_metrics[k])
        std_val = np.std(all_metrics[k])
        print(f"[{name}] {k}: {mean_val:.4f} ± {std_val:.4f}")
```

 doi:10.20944/preprints202507.1073.v1


```
    print(f"[{name}] Avg Training Time: {np.mean(train_times):.2f}s ± {np.std(train_times):.2f}s")
    print(f"[{name}] Avg Inference Time: {np.mean(eval_times):.2f}s ± {np.std(eval_times):.2f}s")

# ============================
# Run All
# ============================
run_multiple_experiments(BaselineMLP, "Baseline MLP")
run_multiple_experiments(HarsanyiBNMLP, "Harsanyi-BN MLP")
```

4. Experimental Results and Analysis

The output results of the experimental code are as follows:

```
=== Final Results for Baseline MLP ===
[Baseline MLP] Accuracy: 0.3016 ± 0.0095
[Baseline MLP] Precision: 0.2811 ± 0.0097
[Baseline MLP] Recall: 0.2926 ± 0.0083
[Baseline MLP] F1: 0.2772 ± 0.0086
[Baseline MLP] AUROC: 0.6074 ± 0.0047
[Baseline MLP] Avg Training Time: 0.71s ± 0.06s
[Baseline MLP] Avg Inference Time: 0.16s ± 0.02s
=== Final Results for Harsanyi-BN MLP ===
[Harsanyi-BN MLP] Accuracy: 0.3088 ± 0.0130
[Harsanyi-BN MLP] Precision: 0.2865 ± 0.0167
[Harsanyi-BN MLP] Recall: 0.2988 ± 0.0125
[Harsanyi-BN MLP] F1: 0.2756 ± 0.0145
[Harsanyi-BN MLP] AUROC: 0.6109 ± 0.0070
[Harsanyi-BN MLP] Avg Training Time: 1.05s ± 0.10s
[Harsanyi-BN MLP] Avg Inference Time: 0.19s ± 0.03s
```

The experimental results show that the batch normalization model with the Harsanyi aggregation mechanism outperforms the traditional batch normalization model in multiple indicators. In particular, the accuracy, precision and AUROC indicators have more stable improvements, indicating that the mechanism can better adjust the importance of features, thereby enhancing the overall judgment ability of the model.

Although the training time has increased slightly, the inference time remains within an acceptable range, indicating that the mechanism does not introduce a significant computational burden and is suitable for deployment in actual scenarios.

5. Generality and Scalability

The core idea of the Harsanyi aggregation theorem, one of the core theories of game theory, is to achieve a reasonable distribution of overall value by evaluating the marginal contribution of individuals in various combinations. The abstractness and generality of this idea make it have a wide range of application potential in machine learning, especially in important subtasks such as feature selection, attention models and normalization strategies, and show excellent theoretical consistency and practicality.

This study introduces the idea of the Harsanyi aggregation theorem into the batch normalization module and constructs a Hassanyi-BN module with a dynamic weight function. This mechanism not only shows stable performance improvement in the multi-layer perceptron (MLP) structure, but more

 doi:10.20944/preprints202507.1073.v1

importantly, it is independent of the structural features and data modality of the specific model. Its design is universal regardless of the model and task, and can be integrated into various deep neural networks as an independent pluggable normalization component.

In convolutional neural networks (CNNs), Harsanyi-BN can directly replace the traditional batch normalization layer. In image data processing, there is obvious spatial redundancy and expression differences between the feature channels of CNN models, so the dynamic weighting mechanism of Harsanyi-BN can effectively improve the model's attention to high-value channels and enhance the effects of image recognition and feature extraction. This is especially effective for visually challenging scenes such as complex backgrounds and low contrast.

In the Transformer architecture, a normalization mechanism is usually used between the multi-head attention module and the feedforward network to stabilize training and suppress gradient fluctuations. Introducing Harsanyi-BN into the Transformer architecture can guide attention to more discriminative information fragments through feature weighting while maintaining training stability. This mechanism is not only expected to improve the understanding ability of natural language processing, but also help improve the efficiency of information flow in long sequence modeling.

In graph neural networks (GNNs), the sharing and propagation of features between nodes is the core mechanism, and the information contribution of different nodes and edges is often uneven. Therefore, Harsanyi-BN can dynamically adjust the normalization strength according to the distribution of node or subgraph features, enhance the adaptability of the model to handle complex topological structures and heterogeneous graph data, and further improve the robustness and generalization ability of tasks such as node classification and edge prediction.

In addition to its wide adaptability, Harsanyi-BN can also be integrated with other attention mechanisms to build a more expressive normalization framework. For example, it can be combined with channel attention mechanisms (such as SE modules) to further adjust the granularity of feature weights; or combined with spatial attention mechanisms, the weights of feature dimensions and spatial dimensions are jointly applied to build a multi-dimensional attention normalization strategy for specific tasks. In addition, this mechanism can also be extended to cross-modal tasks such as visual question answering and image-text matching to guide the weight adjustment and normalization process between multiple information sources.

From the implementation perspective, the Harsanyi-BN module is lightweight and flexible to deploy. It does not rely on specific hardware or complex operations and can run efficiently even in resource-constrained environments. At the same time, the mechanism has good adaptability to changes in data distribution and can maintain stable performance even in distributed scenarios such as federated learning and transfer learning.

In summary, the design of Harsanyi-BN is not only a structural improvement of the traditional batch normalization mechanism, but also an innovative exploration of the fusion of game theory principles and deep learning. Its versatility is reflected in its compatibility with various neural network architectures, and its scalability is reflected in its integration with other attention mechanisms and its adaptability to cross-domain applications. In the future, this mechanism is expected to become a candidate for a new generation of standardized components and be widely used in various artificial intelligence tasks and systems.

6. Conclusion

Based on the Harsanyi aggregation theorem in classical game theory, this paper systematically proposes and implements an improved batch normalization method (Harsanyi-BN module). This method introduces a feature-level attention mechanism and dynamically adjusts the weights of normalized features, so that the neural network can adaptively identify and highlight key features that contribute significantly to the final recognition result. This design not only inherits the advantages of traditional batch normalization learning stability and fast convergence speed, but also effectively improves the performance of the model in multiple performance indicators, showing practical value and innovative significance in the field of deep learning.

 doi:10.20944/preprints202507.1073.v1

Experimental results show that compared with traditional batch normalization, the Harsanyi-BN module has achieved certain performance improvements in multiple indicators, and this is achieved under the premise of a slight increase in computational cost, which reflects the efficiency and practicality of the mechanism. In particular, the improvement of key indicators such as precision, recall rate and area under the curve shows that the dynamic weighting mechanism has played a positive role in promoting feature selection and information utilization. In addition, Harsanyi-BN has good adaptability and is compatible with various network structures. Therefore, it is not limited to a specific model and has a wide range of application potential.

Future research directions can be deepened and expanded in many aspects. First, we will apply this mechanism to larger and more complex neural network architectures, such as deep convolutional networks, complex Transformer variants, and graph neural networks, to explore its performance and advantages in high-dimensional data and complex tasks. Secondly, combined with the currently popular multimodal learning and self-supervised learning frameworks, we will further explore the potential of Harsanyi-BN in multi-channel information fusion and unsupervised feature learning, and improve the generalization ability and robustness of the model.

In addition, the Harsanyi-BN mechanism also has the potential to be integrated with other advanced attention mechanisms and regularization techniques. By combining the spatial attention mechanism, channel attention mechanism and dynamic sparsity mechanism, it is expected that a more intelligent, flexible and efficient regularization module will be constructed in the future, thereby further improving the expressiveness and adaptability of neural networks. In practical applications, this mechanism can also be extended to resource-constrained environments such as edge computing and mobile devices, thereby promoting the popularization and optimization of intelligent systems.

In summary, the batch normalization improvement scheme based on the Hassanyi aggregation theorem proposed in this paper injects new theoretical perspectives and practical ideas into the regularization design of neural networks, which not only brings about the improvement of model performance, but also provides rich inspiration and reference materials for subsequent related research. We hope that in the future, we can carry out more in-depth research based on this framework to promote the innovation of neural network structure and mechanism, and promote the development of artificial intelligence technology in a more efficient and intelligent direction.

References

1. Rahman, T., & Islam, M. S. (2023). MRI brain tumor detection and classification using parallel deep convolutional neural networks. Measurement: Sensors, 26, 100694.
2. 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.
3. Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., & Parmar, M. (2024). A review of convolutional neural networks in computer vision. Artificial Intelligence Review, 57(4), 99.
4. Duan, C., Ding, J., Chen, S., Yu, Z., & Huang, T. (2022). Temporal effective batch normalization in spiking neural networks. Advances in Neural Information Processing Systems, 35, 34377-34390.
5. Dubey, V., & Katarya, R. (2021). BNDNN: Batch Normalization Based Deep Neural Network for Predicting Flood in Urban Areas.
6. Kim, Y., & Panda, P. (2021). Revisiting batch normalization for training low-latency deep spiking neural networks from scratch. Frontiers in neuroscience, 15, 773954.
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. Benz, P., Zhang, C., Karjauv, A., & Kweon, I. S. (2021). Revisiting batch normalization for improving corruption robustness. In Proceedings of the IEEE/CVF winter conference on applications of computer vision (pp. 494-503).
9. 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.

 doi:10.20944/preprints202507.1073.v1

10. Muhammad, A., Shamshad, F., & Bae, S. H. (2023). Adversarial attacks and batch normalization: A batch statistics perspective. IEEE Access, 11, 96449-96459.
11. Kim, Y., & Panda, P. (2021). Optimizing deeper spiking neural networks for dynamic vision sensing. Neural Networks, 144, 686-698.
12. Zhang, Q., Xiao, J., Tian, C., Chun-Wei Lin, J., & Zhang, S. (2023). A robust deformed convolutional neural network (CNN) for image denoising. CAAI Transactions on Intelligence Technology, 8(2), 331-342.
13. Musallam, A. S., Sherif, A. S., & Hussein, M. K. (2022). A new convolutional neural network architecture for automatic detection of brain tumors in magnetic resonance imaging images. IEEE access, 10, 2775-2782.
14. Dursun, G., Tandale, S. B., Gulakala, R., Eschweiler, J., Tohidnezhad, M., Markert, B., & Stoffel, M. (2021). Development of convolutional neural networks for recognition of tenogenic differentiation based on cellular morphology. Computer Methods and Programs in Biomedicine, 208, 106279.
15. Tang, S., Zhu, Y., & Yuan, S. (2022). Intelligent fault identification of hydraulic pump using deep adaptive normalized CNN and synchrosqueezed wavelet transform. Reliability Engineering & System Safety, 224, 108560.
16. Zheng, H., Wu, Y., Deng, L., Hu, Y., & Li, G. (2021, May). Going deeper with directly-trained larger spiking neural networks. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 12, pp. 11062-11070).
17. Huang, Z., Lu, W. J., Hong, C., & Ding, J. (2022). Cheetah: Lean and fast secure {Two-Party} deep neural network inference. In 31st USENIX Security Symposium (USENIX Security 22) (pp. 809-826).
18. Szepesi, P., & Szilágyi, L. (2022). Detection of pneumonia using convolutional neural networks and deep learning. Biocybernetics and biomedical engineering, 42(3), 1012-1022.
19. Ghaderizadeh, S., Abbasi-Moghadam, D., Sharifi, A., Zhao, N., & Tariq, A. (2021). Hyperspectral image classification using a hybrid 3D-2D convolutional neural networks. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 14, 7570-7588.
20. Li, W., Tang, Y. M., Yu, K. M., & To, S. (2022). SLC-GAN: An automated myocardial infarction detection model based on generative adversarial networks and convolutional neural networks with single-lead electrocardiogram synthesis. Information Sciences, 589, 738-750.
21. Ergen, T., & Pilanci, M. (2021, July). Revealing the structure of deep neural networks via convex duality. In International Conference on Machine Learning (pp. 3004-3014). PMLR.
22. Fdez, J., Guttenberg, N., Witkowski, O., & Pasquali, A. (2021). Cross-subject EEG-based emotion recognition through neural networks with stratified normalization. Frontiers in neuroscience, 15, 626277.
23. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks.
24. Wang, K., Abid, M. A., Rasheed, A., Crossa, J., Hearne, S., & Li, H. (2023). DNNGP, a deep neural network-based method for genomic prediction using multi-omics data in plants. Molecular Plant, 16(1), 279-293.
25. Fang, Z., Wang, Y., Peng, L., & Hong, H. (2021). Predicting flood susceptibility using LSTM neural networks. Journal of Hydrology, 594, 125734.
26. Al-Smadi, M., Hammad, M., Baker, Q. B., & Al-Zboon, S. A. A. (2021). A transfer learning with deep neural network approach for diabetic retinopathy classification. International Journal of Electrical and Computer Engineering, 11(4), 3492.
27. 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.
28. Yuan, H., & Kelley, D. R. (2022). scBasset: sequence-based modeling of single-cell ATAC-seq using convolutional neural networks. Nature Methods, 19(9), 1088-1096.
29. Karni, E., & Weymark, J. A. (2024). Impartiality and relative utilitarianism. Social Choice and Welfare, 63(1), 1-18.
30. Nebel, J. M. (2022). Aggregation without interpersonal comparisons of well-being. Philosophy and Phenomenological Research, 105(1), 18-41.
31. Billot, A., & Qu, X. (2021). Utilitarian aggregation with heterogeneous beliefs. American Economic Journal: Microeconomics, 13(3), 112-123.

 doi:10.20944/preprints202507.1073.v1

32. Pitis, S. (2023). Consistent aggregation of objectives with diverse time preferences requires non-markovian rewards. Advances in Neural Information Processing Systems, 36, 2877-2893.
33. Pivato, M., & Tchouante, É. F. (2024). Bayesian social aggregation with almost-objective uncertainty. Theoretical Economics, 19(3), 1351-1398.
34. Zhi-Xuan, T., Carroll, M., Franklin, M., & Ashton, H. (2024). Beyond preferences in ai alignment. Philosophical Studies, 1-51.
35. Peres, R. Á. (2021). The Harsanyi-Rawls Debate: political philosophy as decision theory under uncertainty. Manuscrito, 44(2), 89-127.