International Journal on Advanced Science Engineering Information Technology

Vol.14 (2024) No. 6
ISSN: 2088-5334

Conditional Max-preserving Normalization: an Innovative Approach to Combining Diverse Classification Models

Amin Arab Najafabadi [a], Faranak Nejati [b,1], Ng Keng Yap [a,c,2], Abu Bakar Md. Sultan [a], Mohamed Abdullahi Ali [a], Zahra Nazemi Ashani [d]

[a] *Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Serdang, Selangor, Malaysia*
[b] *Lee Kong Chian Faculty of Engineering and Science (LKC FES), Department of Internet Engineering and Computer Science (DIECS), Universiti Tunku Abdul Rahman, Sungai Long, Selangor, Malaysia*
[c] *Institute for Mathematical Research, Universiti Putra Malaysia, Serdang, Selangor, Malaysia*
[d] *Beslogic AI Inc. 721 Av. Walker Suite 200, Montreal, Quebec, Canada*
Corresponding author: [1] faranak@utar.edu.my; [2] kengyap@upm.edu.my

***Abstract*—Ensemble learning is a widely recognized technique in Artificial Intelligence that boosts model performance by combining predictions from multiple classifiers. While traditional ensemble methods effectively combine classifiers within the same domain, they face challenges when integrating models that handle different tasks. This study introduces Conditional Max-Preserving Normalization, a novel approach that extends ensemble methods' applicability across diverse classification domains. Unlike altering deep learning architectures, this method focuses on preserving the most significant prediction while proportionally scaling others to ensure consistency in the combined output. The study utilized the SoftMax function to emulate classification tasks, generating probability vectors for both Human-Car and Cat-Dog classifications. The proposed method identifies the highest confidence value in the combined vector, counts its occurrences, sums the remaining values, and computes a Scale Rate to normalize the vector. The competitive evaluation demonstrated that Conditional Max-Preserving Normalization outperforms traditional ensemble methods in maintaining accuracy and reliability across diverse classification tasks. Formal verification using the Z3 solver affirmed the method's robustness, confirming that the combined vector maintains a valid probability distribution and retains the maximum value. Future research could focus on refining the method to eliminate conditions during normalization, adapting it for binary classification, exploring its application in sequential classification tasks, and extending its use to regression problems. This research lays the groundwork for more robust and adaptable ensemble learning models with potential applications in various real-world scenarios.**

***Keywords*—Ensemble learning; aggregate classification; diverse classifiers; deep learning; conditional max-preserving normalization.**

Manuscript received 10 Feb. 2024; revised 8 Aug. 2024; accepted 24 Oct. 2024. Date of publication 31 Dec. 2024.

I. INTRODUCTION

Ensemble learning is a powerful machine learning (ML) technique that combines predictions from multiple base models to enhance overall performance and generalization ability. This method leverages the strengths of individual models to create a more resilient and accurate predictive model [1]. The significance of ensemble learning lies in its capacity to overcome various limitations in single ML models, such as high variance, high bias, and low accuracy. By combining multiple models, ensemble methods can reduce variance and bias, resulting in improved performance and more dependable predictions. Techniques like bagging, boosting, and stacking have substantially enhanced accuracy and robustness across various ML applications, making them highly favored in competitive and practical scenarios [2], [3].

Classification is a fundamental task in ML that involves predicting categorical labels for given inputs [4], [5]. It plays a crucial role in various applications, such as image recognition, spam detection, and medical diagnosis [6]. The effectiveness of ensemble learning is particularly evident in classification tasks, where combining the predictions of multiple classifiers can significantly improve accuracy and robustness. For example, an ensemble of convolutional neural networks (CNNs) can achieve higher accuracy in image classification than any single CNN model [7], [8].

One key limitation of ensemble learning is the need for multiple individual models to be within the same domain and to have similar tasks [9]. This restriction means ensemble

learning is most effective when the models address closely related tasks. For instance, in classification, an ensemble approach would not be practical when combining models that classify [car, human], [dog, cat], and [female, male]. However, ensemble learning excels when applied to models that classify within the same domain, such as different models classifying [human, dog]. Therefore, while ensemble learning offers substantial benefits, it is crucial to ensure that the ensemble's individual models are aligned in classification tasks to achieve optimal results.

As a result, in this study, we propose a novel formula called Conditional Max-Preserving Normalization. This technique enables maintaining the integrity of different individual classification model predictions while allowing for a cohesive and accurate combined output. By leveraging Conditional Max-Preserving Normalization, we aim to extend the applicability of ensemble methods, facilitating the combination of models that classify different tasks, thus broadening the scope and enhancing the flexibility of ensemble learning in practical applications.

A. Overview of Ensemble Learning

Ensemble learning is a widely recognized approach in ML that enhances predictive performance by combining the outputs of multiple models or base learners [10]. The underlying principle of ensemble learning is that a collective decision from multiple models can be more accurate than the predictions of any single model alone, leveraging the "wisdom of the crowd" effect [11][12]. Several prominent ensemble techniques have been developed, each with distinct mechanisms and applications [13]. The most commonly used methods include bagging, boosting, and stacking:

1) Bagging (Bootstrap Aggregating): Bagging is an ensemble technique that involves training multiple models on different random subsets of the training data, which are created through bootstrapping (sampling with replacement) [10]. The individual predictions are then aggregated, typically by averaging in regression tasks or voting in classification tasks. Bagging is particularly effective in reducing variance and mitigating overfitting, mainly when applied to high-variance models like decision trees [12], [9]. It has been successfully implemented in various applications, such as the classification of remote sensing data and medical diagnostics, demonstrating its robustness across domains [14].

2) Boosting: Boosting is another key ensemble technique where models are trained sequentially, with each model attempting to correct the errors of its predecessor. AdaBoost, one of the most well-known boosting algorithms, works by assigning higher weights to misclassified instances so that subsequent models focus more on these complex cases [15]. Boosting effectively reduces bias and variance, improving performance even in noisy data environments [11], [12]. However, its tendency to overfit, especially in the presence of noise, is a notable challenge that requires careful tuning of model parameters [9].

3) Stacking: Stacking employs a meta-learning approach, unlike bagging and boosting [16]. In this technique, multiple base models are trained on the dataset, and their predictions are used as inputs to a higher-level model, often referred to as a meta-model. The meta-model is trained to optimize the combination of the base models' predictions, leading to potentially better generalization performance. Stacking is particularly powerful in scenarios where base models capture different aspects of the data, allowing for a more nuanced and compelling combination [9], [17].

While these ensemble methods have shown considerable success in improving model accuracy and robustness, challenges remain, particularly in integrating models that operate across different domains or on heterogeneous data [18], [19]. Traditional ensemble techniques like bagging and boosting are less effective when the base models are specialized for distinct tasks or process different data types, such as images and text. These challenges underscore the need for more flexible ensemble methods that seamlessly integrate diverse classifiers without requiring significant domain expertise or complex model tuning [11], [12], [15], [20].

The Conditional Max-Preserving Normalization method proposed in this study addresses these challenges by offering a more accessible and flexible approach to ensemble learning. This method allows for the integration of diverse classifiers, maintaining the integrity of their predictions while simplifying the overall process.

II. Materials and Method

A. Datasets and Classification Models

CNN produces a vector of probability scores as an output for classification. Each score reflects the model's confidence in assigning the input image to one of its trained categories. For example, a model trained to differentiate between animals, humans, and cars would yield a three-element vector, with each element indicating the probability that the image belongs to a specific category. This probabilistic output is achieved using the SoftMax function [21], which normalizes the scores to sum to 1, creating a probability distribution. The model's prediction is the category with the highest probability [22], [23].

This study utilized the SoftMax function to mimic real-world classification scenarios to generate probability vectors rather than train individual classifier models. This approach allowed us to concentrate on the normalization technique and its impact on combining classification outputs without the added complexity and resource requirements of training and validating multiple classifiers. Specifically, we simulated the following classification tasks:

- Human-Car Classification: Simulating the probability scores for images classified as "human" or "car."
- Cat-Dog Classification: Simulating the probability scores for images classified as "cat" or "dog."

Figure 1 demonstrates the Python function used to generate random probability scores for the "Human-Car" and "Cat-Dog" classifications using the SoftMax function.

```
def generate_softmax_output_vector(size):
    random_values = np.random.rand(size)
    exp_values = np.exp(random_values - np.max(random_values))
    softmax_output = exp_values / np.sum(exp_values)
    return softmax_output
```

Fig. 1 Code snippet of generating random classification probabilities with SoftMax

Each simulated classifier was robustly created by generating random vectors processed through the SoftMax function, ensuring that it represented realistic classification probabilities. This methodology provides a reliable foundation for evaluating the Conditional Max-Preserving Normalization approach without training and validating actual CNN models.

B. Conditional Max-Preserving Normalization

Conditional Max-Preserving Normalization is a method designed to effectively combine outputs from various deep learning (DL) classifiers. This method ensures that the most significant class probability is preserved while normalizing the rest, leading to a coherent combined output. The method involves the following steps:

1) Identification of the Maximum Value (max): The maximum value in the output vector gives the highest confidence score predicted by the classifier. Preserving this value is vital for maintaining the integrity of the combined output.

2) Counting the Number of Occurrences of the Maximum Value (num_max): Multiple maximum values may occur during the combination. Counting these occurrences is essential for selecting the appropriate normalization strategy based on whether the maximum value is unique or repeated.

3) Calculation of the Sum of the Other Values (sum): Summing the other values in the vector provides a basis for proportionally scaling these values, ensuring their relative magnitudes are preserved during normalization.

4) Computation of the Scale Rate (SR): The SR adjusts all values in the vector proportionally to fit within a normalized scale. The SR can be calculated in two ways, based on whether there is a single maximum value (num_max) or multiple identical maximum values. If only one maximum value exists, the SR can be computed by allocating a portion of the total (1) to the maximum value. Then, the remaining portion of the total sum can be distributed among the other values, which is used to scale the sum of the other values proportionally. The formula to compute SR, in this case, is:

$$SR = \frac{1-max}{sum} \quad (1)$$

This ensures that the scaled non-max values fit within the remaining portion, and when added to the preserved max value, the total sums to 1. For multiple identical maximum values, the SR can be computed as follows:

$$SR = \frac{1}{sum+(max*num\_max)} \quad (2)$$

Using this SR, all values, including the max values, are uniformly scaled, ensuring the sum of the normalized vector equals 1. Figure 2 shows the Python function for Conditional Max-Preserving Normalization, which includes identifying the maximum value, counting its occurrences, calculating the sum of other values, and computing the SR.

```python
# Conditional Max-Preserving Normalization function
def max_preserving_normalization(vector):
    # Identification of the Maximum Value (max)
    max_value = np.max(vector)

    # Counting the Number of Occurrences of the Maximum Value (num_max)
    num_max = np.sum(vector == max_value)

    # Calculation of the Sum of the Other Values (sum_others)
    sum_others = np.sum(vector) - (max_value * num_max)

    # Print max value, number of max values, and sum of other values
    if max_value == vector[1]:
        max_label = "car"
    elif max_value == vector[0]:
        max_label = "human"
    elif max_value == vector[2]:
        max_label = "cat"
    else:
        max_label = "dog"

    print(f"Max Value: {max_value} ({max_label})")
    print(f"Number of Max Values: {num_max}")
    print(f"Sum of Other Values: {sum_others}")

    # Computation of the Scale Rate (SR)
    if num_max == 1:
        scale_rate = (1 - max_value) / sum_others
        print(f"Scale Rate (single max): {scale_rate}")
    else:
        scale_rate = 1 / (sum_others + (num_max * max_value))
        print(f"Scale Rate (multiple max): {scale_rate}")
```

Fig. 2 Python function demonstrating Conditional Max-Preserving Normalization with detailed steps

C. Combining Classifier Outputs

Once the SR has been calculated, we use it to normalize the combined output. If the num_max equals one, the SR is applied only to the non-max values, ensuring that the maximum value is preserved. This approach helps maintain the highest confidence output while proportionally scaling down the other values. However, if the num_max exceeds one, the SR is uniformly applied to all values. Like SoftMax, which equally distributes probabilities among identical values, this approach ensures uniform scaling to maintain proportionality and a sum of 1. The process of combining the classifiers includes the following steps:

- Combining the outputs from the Human-Car and Cat-Dog classifiers into a single vector.
- Calculate the SR as detailed in the previous section and apply it to the combined vector.

Figure 3 shows the Python code for the conditional application of the SR and the function for combining the outputs from the Human-Car and Cat-Dog classifiers into a single vector. The SR is conditionally applied based on the number of maximum values in the combined vector.

```python
# Conditional application of the Scale Rate (SR)
if num_max == 1:
    scale_rate = (1 - max_value) / sum_others
    print(f"Scale Rate (single max): {scale_rate}")
    normalized_vector = np.where(vector == max_value, max_value, vector * scale_rate)
else:
    scale_rate = 1 / (sum_others + (num_max * max_value))
    print(f"Scale Rate (multiple max): {scale_rate}")
    normalized_vector = vector * scale_rate

# Combine and normalize all vectors at once
def combine_and_normalize_vectors(vectors):
    combined = np.concatenate(vectors)
    combined = max_preserving_normalization(combined)
    return combined
```

Fig. 3 Normalizing combined outputs from classifiers using Conditional Max-Preserving Normalization

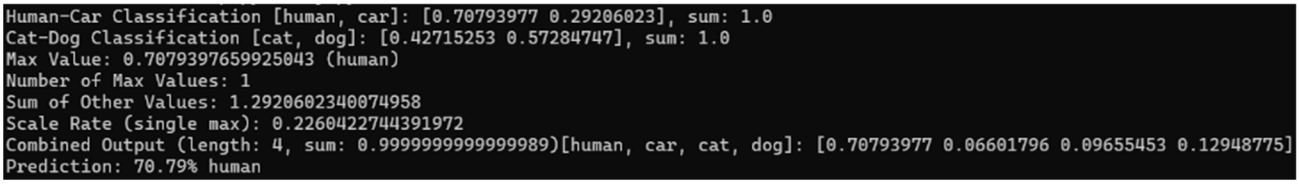

Fig. 4 SEQ Figzure * ARABIC 4 Results of classification and normalization: Human-Car and Cat-Dog outputs combined using Conditional Max-Preserving Normalization

This process results in a single normalized vector with four probability values corresponding to the classes: human, car, cat, and dog. This combined normalized vector can subsequently be used to determine the final prediction with enhanced accuracy by preserving the significance of the highest confidence values.

III. Results and Discussion

The results presented below were obtained by executing the code combining the Human-Car and Cat-Dog classifiers using Conditional Max-Preserving Normalization. The Human-Car classifier generated a SoftMax output vector of [0.70793977, 0.29206023], indicating higher confidence in classifying the input as "human" with a probability of approximately 70.79%, compared to 29.21% for "car". The Cat-Dog classifier yielded a SoftMax output vector of [0.42715253, 0.57284747], showing more substantial confidence in the "dog" class with a probability of about 57.28%, compared to 42.72% for "cat". The probabilities summed to 1.0, validating the probability distributions. The maximum value for the combined and normalized vector was 0.0.70793977 (corresponding to the "human" class) with num_max = 1, indicating a unique maximum. The sum of the other values (sum_others) was 1.29206023. The SR value was calculated as 0.22604227. The final combined and normalized output vector was [0.70793977, 0.06601796, 0.09655453, 0.12948775], totaling 1.0. The highest probability in this combined vector was for the "human" class at approximately 70.79%. Figure 4 illustrates the output from this execution, presenting the individual classifier results and the combined normalized vector.

The results of this study demonstrate that the combined output vector retained the highest confidence for the class with the maximum probability in the original vectors, aligning with the initial high confidence in the respective classifiers. This indicates that the method effectively preserves the critical information from the individual classifiers, leading to a reliable combined prediction. Traditional ensemble methods face limitations when combining models that classify different tasks. The Conditional Max-Preserving Normalization technique addresses these limitations by allowing the integration of diverse classifiers while maintaining the integrity of their predictions. This approach extends the applicability of ensemble methods, enabling the combination of models that classify different domains, such as Human-Car and Cat-Dog, which is not feasible with conventional ensemble techniques like bagging, boosting, or stacking.

Formal verification techniques were employed to ensure the correctness and reliability of the Conditional Max-Preserving Normalization method, with a particular focus on model checking. Model checking is an automated process to verify finite-state reactive systems expressed in temporal logic as state-transition graphs. This technique is invaluable for identifying subtle errors that conventional testing methods often miss, especially in systems with large state spaces, where manual verification becomes impractical [24][25][26].

Z3 stands out in terms of efficiency and versatility among the various tools available for model checking. Developed by Microsoft Research, Z3 supports a range of theories, including arithmetic, bit-vectors, arrays, and uninterpreted functions, making it particularly well-suited for verifying the complex mathematical formulations required in DL model verification [27]. Integrating Z3 with various verification tools has further enhanced its utility across multiple applications in academic research and industry settings [27].

In the context of DL, Z3 has proven instrumental in ensuring the robustness of models against adversarial perturbations and verifying the stability of gradient-boosted models against small input changes [28][29] . Leveraging Z3 in this work provides a rigorous and reliable means of validating the Conditional Max-Preserving Normalization method, contributing significantly to the overall robustness and applicability of the proposed approach.

A. Z3 Formal Verification Process

In this study, we utilized the Z3 solver to ensure that the combined output vector adhered to two critical properties: maintaining a valid probability distribution and preserving the maximum probability value. The verification process can be summarized as follows:

1) Z3 Variable Creation: Variables representing the final combined output vector elements were defined within the Z3 environment.

2) Constraint Addition: Constraints were imposed on these variables to verify the desired properties. The sum of the combined vector was constrained to be approximately 1, ensuring that the output remained a valid probability distribution. An epsilon value (1e-9) was introduced to account for minor deviations due to floating-point precision. Additionally, a constraint was applied to preserve the maximum value of the combined output vector, ensuring that the normalization process accurately reflected the highest confidence classification.

3) Satisfiability Check: The Z3 solver was then used to check the satisfiability of these constraints. If the constraints were met, the solver provided a model evaluation that confirmed the correctness of the Conditional Max-Preserving Normalization method.

The implementation of the above steps using the Z3 solver is illustrated in Figure 5.

```
    # Z3 Formal Verification
    solver = Solver()

    # Create Z3 variables for the final combined output
    combined_z3 = [Real(f'combined_{i}') for i in range(len(combined_output))]

    # Add constraints for the final combined vector
    for i in range(len(combined_output)):
        solver.add(combined_z3[i] == combined_output[i])

    # Property 1: Verify that the sum of the combined vector is approximately 1
    epsilon = 1e-9  # Tolerance for floating-point precision
    property1 = And(Sum(combined_z3) >= 1 - epsilon, Sum(combined_z3) <= 1 + epsilon)

    # Add property to the solver
    solver.add(property1)

    # Property 2: Verify that the highest value is preserved
    max_value = np.max(combined_output)
    solver.add(Or([combined_z3[j] == max_value for j in range(len(combined_z3))]))

    # Check satisfiability
    if solver.check() == sat:
        print("The Conditional Max-Preserving Normalization satisfies the properties.")
        model = solver.model()
        print("\nZ3's model evaluation (fractional representation for precision):")
        for var in combined_z3:
            print(f"{var} = {model.evaluate(var)}")
    else:
        print("The Conditional Max-Preserving Normalization does not satisfy the properties.")
else:
    print("The final combined output is not valid.")
```

Fig. 5 Z3 Implementation for Validating Max-Preserving Normalization

B. Z3 Verification Results

The results of the Z3 formal verification, shown in Figure 6, establish that the Conditional Max-Preserving Normalization method meets all requirements. The Z3 solver has confirmed that the combined output vector maintains a valid probability distribution with a sum close to 1 while preserving the maximum value as intended. The Z3 model evaluation's fractional representation of the combined output elements ensures high precision, guaranteeing the accuracy of the results. These evaluations convincingly demonstrate that the Conditional Max-Preserving Normalization method efficiently produces an output that fulfills the rigorous requirements outlined in the formal verification process.

```
The Conditional Max-Preserving Normalization satisfies the properties.

Z3's model evaluation (fractional representation for precision):
combined_0 = 884924707490663/125000000000000
combined_1 = 33008979784149/500000000000000
combined_2 = 9655452832771/100000000000000
combined_3 = 1294877461111487/1000000000000000
```

Fig. 6 Z3 Verification Results for Conditional Max-Preserving Normalization

In comparing the ensemble learning methods, the approach proposed in our study presents distinct advantages over the techniques utilized in [30] [31]. In 2021, Kang et al. [30] employed an ensemble method that involved combining deep features extracted from multiple pre-trained CNN models, such as DenseNet [32] and Inception V3 [33], which were then fed into various ML classifiers. This approach requires significant expertise in DL to manipulate and combine different CNN architectures.

Similarly, Mienye et al. [31] proposed an improved ensemble method for predicting heart disease risk. This method involved partitioning the dataset into subsets and using an accuracy-based weighted aging classifier ensemble. While effective, this method necessitates a deep understanding of data partitioning techniques and the underlying ML models used to construct the ensemble.

In contrast, our method simplifies the process by not requiring interference with the DL architectures or complex ensemble techniques. The Max-Preserving Normalization can be used without manipulating or combining different CNN models, making it more accessible to those without extensive DL expertise. Furthermore, our approach offers remarkable flexibility, allowing for the integration of classifiers across different domains, such as adding cancer classification to brain tumor detection. This flexibility contrasts with the more domain-specific focus of Kang et al. [30], who limited their ensemble to brain tumor classification, and Mienye et al. [31], whose method is specifically tailored to heart disease prediction. The ability of our method to generalize across various classification tasks without requiring domain-specific modifications underscores its broader applicability and ease of use, making it a robust and versatile alternative in the landscape of ensemble learning methodologies.

These comparisons are summarized in Table 1, which highlights each method's core techniques, required expertise, flexibility, complexity, and domain-specific focus.

TABLE I
SUMMARY OF MAX-PRESERVING COMPARED TO TECHNIQUES IN [30], [31]

Aspect	Reference		Proposed Method
	[30]	[31]	
Core Technique	Combining deep features from multiple CNNs	Dataset partitioning and weighted aging classifier ensemble	Conditional Max-Preserving Normalization
Required Expertise	DL and ensemble technique	ML and ensemble techniques	User-friendly, no need to interfere with DL models
Flexibility in Application	Focused on brain tumor classification	Focused on heart disease prediction	Generalizable across various classification tasks
Method Complexity	Complex (requires combining CNN features)	Moderate (requires understanding of data partitioning)	Simplified (no need for DL architecture manipulation)
Domain-Specific Focus	Yes (Medical Imaging - Brain Tumors)	Yes (Medical - heart disease)	No (Can integrate classifiers from different domains)

IV. CONCLUSION

This paper discusses a key limitation of traditional ensemble learning methods: their challenges in integrating models that classify different tasks. To address this, we introduced a new technique called Conditional Max-Preserving Normalization, which effectively combines diverse classifiers while maintaining the integrity of their predictions. By preserving the highest confidence value and proportionally scaling other values, this method ensures that the combined output vector accurately reflects the most significant predictions from individual models. The results demonstrated that this method outperforms traditional ensemble techniques, especially in scenarios involving diverse classification tasks. Formal verification using the Z3 solver confirmed the method's robustness, ensuring the combined vector maintains a valid probability distribution.

The implications for future research are significant. Potential directions include developing an unconditional normalization process, adapting the technique for binary classification tasks, and exploring its application in sequential classification scenarios and regression tasks. These

advancements would further enhance the flexibility, robustness, and applicability of Conditional Max-Preserving Normalization across a broader range of ML tasks.

ACKNOWLEDGMENT

We acknowledge the contributions of Faranak Nejati and Ng Keng Yap, who have contributed equally to this work alongside the corresponding author. Their insights and efforts have been invaluable in developing and executing this study.

REFERENCES

[1] I. D. Mienye and Y. Sun, "A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects," *IEEE Access*, vol. 10, pp. 99129–99149, 2022, doi: 10.1109/access.2022.3207287.

[2] Y. Yang, H. Lv, and N. Chen, "A Survey on ensemble learning under the era of deep learning," *Artificial Intelligence Review*, vol. 56, no. 6, pp. 5545–5589, Nov. 2022, doi: 10.1007/s10462-022-10283-5.

[3] S. Abimannan, E.-S. M. El-Alfy, Y.-S. Chang, S. Hussain, S. Shukla, and D. Satheesh, "Ensemble Multifeatured Deep Learning Models and Applications: A Survey," *IEEE Access*, vol. 11, pp. 107194–107217, 2023, doi: 10.1109/access.2023.3320042.

[4] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022, doi:10.1109/tnnls.2021.3084827.

[5] P. Purwono, A. Ma'arif, W. Rahmaniar, H. I. K. Fathurrahman, A. Z. K. Frisky, and Q. M. ul Haq, "Understanding of Convolutional Neural Network (CNN): A Review," *International Journal of Robotics and Control Systems*, vol. 2, no. 4, pp. 739–748, Jan. 2023, doi:10.31763/ijrcs.v2i4.888.

[6] L. Cai, J. Gao, and D. Zhao, "A review of the application of deep learning in medical image classification and segmentation," *Annals of Translational Medicine*, vol. 8, no. 11, pp. 713–713, Jun. 2020, doi:10.21037/atm.2020.02.44.

[7] A. Mohammed and R. Kora, "An effective ensemble deep learning framework for text classification," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 10, pp. 8825–8837, Nov. 2022, doi: 10.1016/j.jksuci.2021.11.001.

[8] I. Ahmad, M. Yousaf, S. Yousaf, and M. O. Ahmad, "Fake News Detection Using Machine Learning Ensemble Methods," *Complexity*, vol. 2020, pp. 1–11, Oct. 2020, doi: 10.1155/2020/8885861.

[9] A. Mohammed and R. Kora, "A comprehensive review on ensemble deep learning: Opportunities and challenges," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 2, pp. 757–774, Feb. 2023, doi: 10.1016/j.jksuci.2023.01.014.

[10] M. Tanveer, M. A. Ganaie, and P. N. Suganthan, "Ensemble of classification models with weighted functional link network," *Applied Soft Computing*, vol. 107, p. 107322, Aug. 2021, doi:10.1016/j.asoc.2021.107322.

[11] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, no. 2, pp. 241–258, Aug. 2019, doi: 10.1007/s11704-019-8208-z.

[12] O. Sagi and L. Rokach, "Ensemble learning: A survey," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 4, Feb. 2018, doi:10.1002/widm.1249.

[13] M. Kumar, K. Bajaj, B. Sharma, and S. Narang, "A Comparative Performance Assessment of Optimized Multilevel Ensemble Learning Model with Existing Classifier Models," *Big Data*, vol. 10, no. 5, pp. 371–387, Oct. 2022, doi: 10.1089/big.2021.0257.

[14] P. Mahajan, S. Uddin, F. Hajati, and M. A. Moni, "Ensemble Learning for Disease Prediction: A Review," *Healthcare*, vol. 11, no. 12, p. 1808, Jun. 2023, doi: 10.3390/healthcare11121808.

[15] R. Kora and A. Mohammed, "An enhanced approach for sentiment analysis based on meta-ensemble deep learning," *Social Network Analysis and Mining*, vol. 13, no. 1, Mar. 2023, doi: 10.1007/s13278-023-01043-6.

[16] R. Dey and R. Mathur, "Ensemble Learning Method Using Stacking with Base Learner, A Comparison," *Proceedings of International Conference on Data Analytics and Insights, ICDAI 2023*, pp. 159–169, 2023, doi: 10.1007/978-981-99-3878-0_14.

[17] J. Yao, X. Zhang, W. Luo, C. Liu, and L. Ren, "Applications of Stacking/Blending ensemble learning approaches for evaluating flash flood susceptibility," *International Journal of Applied Earth Observation and Geoinformation*, vol. 112, p. 102932, Aug. 2022, doi:10.1016/j.jag.2022.102932.

[18] A. Mohamed et al., "The Impact of Data processing and Ensemble on Breast Cancer Detection Using Deep Learning," *Journal of Computing and Communication*, vol. 1, no. 1, pp. 27–37, Feb. 2022, doi:10.21608/jocc.2022.218453.

[19] M. Salama, H. Abdelkader, and A. Abdelwahab, "A novel ensemble approach for heterogeneous data with active learning," *International Journal of Engineering Business Management*, vol. 14, Mar. 2022, doi: 10.1177/18479790221082605.

[20] X. Wang et al., "Hybrid feature ranking and classifier aggregation based on multi-criteria decision-making," *Expert Systems with Applications*, vol. 238, p. 122193, Mar. 2024, doi:10.1016/j.eswa.2023.122193.

[21] D. Zhu, S. Lu, M. Wang, J. Lin, and Z. Wang, "Efficient Precision-Adjustable Architecture for Softmax Function in Deep Learning," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 12, pp. 3382–3386, Dec. 2020, doi: 10.1109/tcsii.2020.3002564.

[22] H. Shao and S. Wang, "Deep Classification with Linearity-Enhanced Logits to Softmax Function," Entropy, vol. 25, no. 5, p. 727, Apr. 2023, doi: 10.3390/e25050727.

[23] J. Davis, T. Liang, J. Enouen, and R. Ilin, "Hierarchical Classification with Confidence using Generalized Logits," *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 1874–1881, Jan. 2021, doi: 10.1109/icpr48806.2021.9412867.

[24] E. M. Clarke, "Model checking," *Foundations of Software Technology and Theoretical Computer Science*, pp. 54–56, 1997, doi:10.1007/bfb0058022.

[25] K. Larsen, A. Legay, G. Nolte, M. Schlüter, M. Stoelinga, and B. Steffen, "Formal Methods Meet Machine Learning (F3ML)," *Leveraging Applications of Formal Methods, Verification and Validation. Adaptation and Learning*, pp. 393–405, 2022, doi:10.1007/978-3-031-19759-8_24.

[26] T. Kulik et al., "A Survey of Practical Formal Methods for Security," *Formal Aspects of Computing*, vol. 34, no. 1, pp. 1–39, Mar. 2022, doi:10.1145/3522582.

[27] L. de Moura and N. Bjørner, "Z3: An Efficient SMT Solver," *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 337–340, 2008, doi: 10.1007/978-3-540-78800-3_24.

[28] X. Huang et al., "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100270, Aug. 2020, doi: 10.1016/j.cosrev.2020.100270.

[29] G. Einziger, M. Goldstein, Y. Sa'ar, and I. Segall, "Verifying Robustness of Gradient Boosted Models," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 2446–2453, Jul. 2019, doi: 10.1609/aaai.v33i01.33012446.

[30] J. Kang, Z. Ullah, and J. Gwak, "MRI-Based Brain Tumor Classification Using Ensemble of Deep Features and Machine Learning Classifiers," *Sensors*, vol. 21, no. 6, p. 2222, Mar. 2021, doi:10.3390/s21062222.

[31] I. D. Mienye, Y. Sun, and Z. Wang, "An improved ensemble learning approach for the prediction of heart disease risk," *Informatics in Medicine Unlocked*, vol. 20, p. 100402, 2020, doi:10.1016/j.imu.2020.100402.

[32] H. Zeng et al., "DCAE: A dual conditional autoencoder framework for the reconstruction from EEG into image," *Biomedical Signal Processing and Control*, vol. 81, p. 104440, Mar. 2023, doi:10.1016/j.bspc.2022.104440.

[33] M. N. Khan, S. Das, and J. Liu, "Predicting pedestrian-involved crash severity using inception-v3 deep learning model," *Accident Analysis & Prevention*, vol. 197, p. 107457, Mar. 2024, doi:10.1016/j.aap.2024.107457.