IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 10 August 2025, accepted 28 August 2025, date of publication 5 September 2025, date of current version 30 September 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3606472

RESEARCH ARTICLE

Adjusted Exponential Scaling: An Innovative Approach for Combining Diverse Multiclass Classifications

AMIN ARAB NAJAFABADI[1], NG KENG YAP[1,2], (Senior Member, IEEE), ZAHRA NAZEMI ASHANI[3], FARANAK NEJATI[4], (Member, IEEE), AND MOHAMED ABDULLAHI ALI[1,5]

[1]Department of Software Engineering and Information Systems, Universiti Putra Malaysia, Serdang 43400, Malaysia
[2]Institute for Mathematical Research, Universiti Putra Malaysia, Serdang 43400, Malaysia
[3]Beslogic Inc., Montreal, QC H4C 2H5, Canada
[4]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 43000, Malaysia
[5]Faculty of Engineering and Technology, Salaam University, Mogadishu, Somalia

Corresponding author: Ng Keng Yap (kengyap@upm.edu.my)

ABSTRACT Ensemble learning plays a crucial role in deep learning (DL) by improving classification accuracy through the aggregation of multiple classifiers. However, traditional methods, such as bagging, boosting, and stacking, face limitations when combining heterogeneous classifiers trained on unrelated tasks, owing to their reliance on closely aligned objectives. To address these challenges, we propose Adjusted Exponential Scaling (AES), a novel approach that combines diverse classifiers using probabilistic outputs, such as those generated by the Softmax function. AES eliminates the need to modify DL architectures or interfere with their training processes, thereby significantly reducing computational costs while ensuring compatibility with existing systems. We demonstrate AES scalability in simulations, successfully combining the outputs of up to 40 classifiers while maintaining probabilistic integrity. A formal verification using the Z3 Satisfiability Modulo Theories (SMT) solver confirmed its accuracy and reliability. AES is designed for simplicity and adaptability, making it particularly well suited for dynamic applications such as deepfake detection and medical diagnostics, where real-time combination is critical. By providing a scalable and computationally efficient solution, AES expands the applicability of ensemble learning and establishes a foundation for future extensions to sequential and binary tasks, further enhancing its versatility in machine learning challenges.

INDEX TERMS Ensemble learning, multiclass classification, adjusted exponential scaling (AES), classifier integration, formal verification.

I. INTRODUCTION

Ensemble learning is a cornerstone of modern machine learning that enhances prediction accuracy, reduces variance, and mitigates bias by leveraging multiple models [1], [2]. Techniques such as bagging, boosting, and stacking have demonstrated effectiveness across various domains, including deepfake detection and medical diagnostics, by integrating complementary model strengths [3], [4].

Classification is a fundamental machine learning task that assigns inputs to predefined categories with high precision and reliability. Ensemble methods enhance classification performance by addressing challenges such as overfitting and bias through the combined strengths of diverse models [5], [6]. For example, convolutional neural networks (CNNs) in an ensemble can improve image classification by leveraging distinct feature representations. These capabilities make ensemble learning particularly valuable in critical applications where accuracy is essential, such as healthcare diagnostics and fraud detection [7], [8].

The associate editor coordinating the review of this manuscript and approving it for publication was Zhan-Li Sun.

Despite these advantages, conventional ensemble techniques face significant challenges when integrating heterogeneous classifiers trained on diverse, unrelated tasks. These methods generally assume a shared label space and architectural compatibility. Approaches such as stacking, which relies on a meta-learner for output optimization, require additional training and architectural modifications, thereby increasing computational overhead and reducing accessibility in real-world deployments [9], [10]. These limitations become particularly evident when attempting to merge models with distinct classification objectives, such as recognizing fruits, animals, and gender simultaneously [11], [12], [13]. This highlights a clear research gap in enabling architecture-independent, inference-level integration of classifiers across disjoint tasks.

To address this gap, this study introduces AES, a novel architecture-agnostic approach designed to directly combine classifier outputs without requiring additional training or architectural modifications. Unlike traditional ensemble methods, AES operates directly on probabilistic outputs, such as those produced by the Softmax function, ensuring computational efficiency while preserving the probabilistic integrity of predictions. AES offers a scalable and accessible solution for dynamic systems by avoiding modifications to classifier architectures and eliminating the need for meta-learners.

The effectiveness of AES is demonstrated through simulations incorporating up to 40 classifiers while maintaining high precision and scalability. Formal verification using the Z3 Satisfiability Modulo Theories (SMT) solver [14] further validates its accuracy and reliability in preserving probabilistic properties. These attributes establish AES as a robust and generalizable approach for addressing classification challenges in diverse and dynamic environments.

Unlike traditional ensemble methods such as bagging, boosting, and stacking, AES operates entirely at the inference stage without retraining, meta-learners, or modifications to classifier architectures. This enables seamless integration of independently trained models with heterogeneous label spaces, while preserving probabilistic integrity. These properties distinguish AES as a practical and scalable alternative to prior approaches.

This paper is structured as follows: Section II reviews the related literature; Section III details the AES methodology and case study; Section IV presents the formal verification process using the Z3 SMT solver; Section V discusses key findings, limitations, and future directions; Section VI highlights related studies; and Section VII concludes the paper.

II. LITERATURE REVIEW

A. MULTICLASS CLASSIFICATION OUTPUTS

Multiclass classification is a fundamental task in DL that involves assigning inputs to one of several predefined categories. Beyond determining the final prediction, the output of a multiclass classification model reflects the confidence levels, which are critical for high-reliability applications such as medical diagnostics and fraud detection [15], [16], [17]. A thorough understanding of key components, such as logits and the Softmax function, is essential for developing the AES method, as it relies on probabilistic outputs for classifier integration.

1) LOGITS: THE PRECURSOR TO PROBABILITY DISTRIBUTIONS

Logits are raw, unnormalized scores produced by the final layer of a classification model before applying an activation function. These scores indicate the model's confidence in each class, based on the input characteristics. Logits are optimized during training to effectively differentiate between classes, making them essential for decision-making [18], [19], [20]. However, because logits do not form a probability distribution, they require transformation, which is typically achieved using the Softmax function.

2) THE SOFTMAX FUNCTION: FROM LOGITS TO PROBABILITIES

The Softmax function converts logits into a probability distribution, ensuring that the sum of all class probabilities is equal to one. This enables the probabilistic interpretation of model predictions, which is critical for AES because it combines the outputs of multiple classifiers. The Softmax function is mathematically expressed as:

$$P(y = i \mid x) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \tag{1}$$

where z_i denotes the logit for class i and K is the total number of classes.

For example, in a three-class classification, a probability vector of $[0.2, 0.7, 0.1]$ indicates that the model is most confident in the second class, assigning it to probability of 70%. This probabilistic representation is fundamental for AES, because it normalizes the classifier outputs before integration [20].

B. COMBINING MULTICLASS CLASSIFICATION

Ensemble learning is widely used to improve classification performance by integrating multiple models. It enhances accuracy and robustness by leveraging the strengths of diverse classifiers. Among the various approaches, ensemble learning aggregates the outputs of multiple models to improve predictive performance [21]

- **Bagging (bootstrap aggregating):** Trains multiple models on different bootstrap samples of the training data. Predictions are combined by averaging (regression) or majority voting (classification) [24], [25], [26].
- **Boosting:** Sequentially trains models, with each new model focusing on correcting errors from the previous one [29]. AdaBoost assigns higher weights to misclassified instances to improve accuracy [30].
- **Stacking:** Uses multiple base models to train a meta-model that optimally combines their outputs [33].

```python
def generate_random_vector(size):
    random_values = np.random.rand(size)
    exp_values = np.exp(random_values - np.max(random_values))
    softmax_vector = exp_values / np.sum(exp_values)
    return softmax_vector

def generate_random_vectors():
    num_vectors = np.random.randint(2, 41)
    vectors = [generate_random_vector(np.random.randint(2, 6)) for _ in range(num_vectors)]
    return vectors
```

FIGURE 1. Generation of random classifier outputs using the softmax function, demonstrating the probability distribution of class scores.

```python
def combine_outputs(vectors):
    return np.concatenate(vectors)
```

FIGURE 2. Illustration of classifier outputs combined into a unified decision vector V_i, improving prediction accuracy.

While highly effective, stacking increases computational complexity due to meta-model training [36], [37].

C. CONTRIBUTION OF THIS STUDY

While ensemble learning enhances the accuracy and robustness of classification, existing methods struggle to integrate outputs from diverse multiclass classifiers. Techniques such as stacking require modifications to DL architectures, including training a meta-learner or adjusting model weights, which increases computational complexity and necessitates substantial DL expertise [4], [9], [39], [40].

To address these challenges, this study introduces AES, a scalable method for combining diverse classifier outputs without modifying model architectures. AES leverages probabilistic outputs, such as those generated by the Softmax function, to seamlessly integrate classifier predictions. Unlike traditional ensemble techniques, AES eliminates architectural modifications and additional training, offering a computationally efficient and accessible solution.

By preserving the probabilistic integrity of the classifier outputs, AES provides a scalable and flexible method for integrating classifiers trained on unrelated tasks, making it highly suitable for real-world applications.

III. METHODOLOGY

A. DATASETS AND CLASSIFICATION MODELS

As detailed in Section II-A, the output of a multiclass classification task is the probability vector produced by the Softmax function. Rather than training DL models, we simulated classifier outputs by generating Softmax-based probability vectors, replicating the probabilistic distributions observed in real-world classification tasks. This approach allows for efficient evaluation of classifier combination techniques without the computational overhead associated with model training.

To ensure diversity in classifier outputs, we generated random probability vectors for classifiers ranging from 2 to 40 classes, with each output containing between 2 and 5 probability scores. This randomness minimizes bias and introduces diversity, mirroring variations found in real-world classification systems. The process is illustrated in Fig. 1, which visualizes the probability distributions generated by the Softmax function for the simulated classifiers.

```python
def adjusted_exponential_scaling(vector):
    max_value = np.max(vector)
    max_count = np.sum(vector == max_value)
```

FIGURE 3. Illustration of the process for identifying max in V_i and determining its occurrences.

```python
if max_count == 1:
    sum_exponentials = np.sum(np.exp(vector[vector != max_value]))
else:
    sum_exponentials = np.sum(np.exp(vector))
```

FIGURE 4. Illustration of the sum of exponentials calculation for scaling the combined output vector.

```python
if max_count == 1:
    scale_rate = (1 - max_value) / sum_exponentials
else:
    scale_rate = 1 / sum_exponentials
```

FIGURE 5. Illustration of the computation process for the scale rate (SR).

B. KEY STEPS IN PROPOSING THE AES METHOD

The AES method follows a structured process to efficiently normalize and combine classifier outputs. This process ensures that probabilistic integrity is maintained while addressing cases in which classifiers assign equal confidence scores to multiple classes. The following steps systematically detail the AES process.

1) **Combine classifier outputs** (V_i)**:** The outputs of all classifiers are concatenated into a single vector, consolidating the predictions into a unified structure (Fig. 2).
2) **Identify the most confident prediction** (max)**:** The next step involves identifying max from vector V_i, where max denotes the highest confidence score predicted by any classifier. Preserving this value ensures that the most confident prediction remains dominant in the combined output (Fig. 3).
3) **Compute the Sum of the Exponentials** (S)**:** After identifying max, the sum of exponentials is calculated to scale the combined outputs based on confidence levels. The computation differs depending on whether a single or multiple max values exist:

$$S = \sum_{i \neq \max} e^{v_i} \tag{2}$$

If multiple max values exist, the sum is computed across all values, including max:

$$S = \sum_{i} e^{v_i} \tag{3}$$

4) **Derive the Scale Rate** (SR)**:** Once S is determined, the scale rate is calculated to normalize the outputs. If there is only one max, SR is computed as:

$$SR = \frac{1 - \max}{S} \tag{4}$$

If multiple max values exist, the scale rate is adjusted as:

$$SR = \frac{1}{S} \tag{5}$$

```
if max_count == 1:
    scaled_vector = np.where(vector != max_value, scale_rate * np.exp(vector), max_value)
else:
    scaled_vector = scale_rate * np.exp(vector)

return scaled_vector
```

FIGURE 6. Illustration of the normalization process using *SR* to ensure the combined output vector V_i forms a valid probability distribution.

```
Generated Outputs:
Output 1: [0.17222621 0.19292361 0.20509979 0.4297504 ], Sum: 1.0
Output 2: [0.56799866 0.43200134], Sum: 1.0
Output 3: [0.15275767 0.18593914 0.16444555 0.30752869 0.18932895], Sum: 1.0
Output 4: [0.30179763 0.30986149 0.17044085 0.21790004], Sum: 1.0
Output 5: [0.24358076 0.23047202 0.30253207 0.22341515], Sum: 1.0
Output 6: [0.27552598 0.13117472 0.29369374 0.14171382 0.15789173], Sum: 1.0
Output 7: [0.49810643 0.50189357], Sum: 0.9999999999999999
Output 8: [0.26365673 0.25362832 0.23420566 0.24850929], Sum: 1.0000000000000002
Output 9: [0.16722676 0.20968661 0.25362941 0.16673598 0.20272124], Sum: 1.0000000000000002
Output 10: [0.2536986  0.26054697 0.25266641 0.23308802], Sum: 1.0
Output 11: [0.25264035 0.29876744 0.26148754 0.18710467], Sum: 1.0
Output 12: [0.1146131  0.26690752 0.20632623 0.19372202 0.21843113], Sum: 0.9999999999999999
Output 13: [0.31148993 0.68851007], Sum: 1.0
Output 14: [0.3660714 0.4339086 0.20002  ], Sum: 1.0
Output 15: [0.24823523 0.1945392  0.29057654 0.26664903], Sum: 1.0
```

FIGURE 7. Illustration of randomly generated classification outputs using the softmax function. These outputs represent diverse classifier predictions for different class distributions.

```
Combined Outputs (Before Scaling):
[0.17222621 0.19292361 0.20509979 0.4297504  0.56799866 0.43200134
 0.15275767 0.18593914 0.16444555 0.30752869 0.18932895 0.30179763
 0.30986149 0.17044085 0.21790004 0.24358076 0.23047202 0.30253207
 0.22341515 0.27552598 0.13117472 0.29369374 0.14171382 0.15789173
 0.49810643 0.50189357 0.26365673 0.25362832 0.23420566 0.24850929
 0.16722676 0.20968661 0.25362941 0.16673598 0.20272124 0.2536986
 0.26054697 0.25266641 0.23308802 0.25264035 0.29876744 0.26148754
 0.18710467 0.1146131  0.26690752 0.20632623 0.19372202 0.21843113
 0.31148993 0.68851007 0.3660714  0.4339086  0.20002    0.24823523
 0.1945392  0.29057654 0.26664903]
Sum: 15.0
```

FIGURE 8. Illustration of the combined output vector obtained from multiple classifier probability distributions. The sum of probabilities is preserved to ensure a valid representation for further processing.

5) **Normalize the Combined Output:** The final step applies *SR* to normalize the combined output, ensuring a valid probability distribution:

$$v_i' = SR \cdot e^{v_i} \tag{6}$$

- **Single** max**:** *SR* is applied only to non-max values, preserving max as is.
- **Multiple** max**:** *SR* is applied uniformly to all values, maintaining proportional normalization.

C. RESULT

This section presents the implementation of the AES method and demonstrates its application, from the generation of classification outputs to their combination and normalization. AES operates directly on classification model outputs. To simulate real-world scenarios and reduce bias, random classification outputs were generated using the Softmax function, as described in Section III-A. This simulation replicates diverse classifier outputs, enabling a realistic AES evaluation. As illustrated in Fig. 7, 15 outputs were generated, each representing a different classifier handling between two and five classes. These probability distributions sum to one, reflecting the probabilistic nature of the classification tasks.

The first step of the AES method consolidates the probability distributions of multiple classifiers into a single vector, thereby creating a unified representation for further processing. As shown in Fig. 8, the combined vector

```
Maximum Value (max): 0.6885100661720936 (Single Max)
Sum of the Exponentials (S): 72.63897415517118
Scale Rate (SR): 0.004288192908155647
```

FIGURE 9. Illustration of the computation of max, *S*, and *SR*. These key computations ensure the proper normalization of the combined output vector.

```
Normalized Output: (after applying SR)
[0.00509414 0.00520068 0.00526439 0.00659041 0.00756751 0.00660526
 0.00499593 0.00516448 0.00505466 0.0058322  0.00518202 0.00579887
 0.00584582 0.00508506 0.00533221 0.00547092 0.00539967 0.00580313
 0.0053617  0.00564851 0.00488926 0.00575207 0.00494106 0.00502164
 0.00705666 0.00708344 0.00558186 0.00552616 0.00541987 0.00549795
 0.00506874 0.00528859 0.00552617 0.00506625 0.00525188 0.00552655
 0.00556453 0.00552085 0.00541381 0.00552071 0.00578132 0.00556977
 0.0051705  0.00480895 0.00560004 0.00527085 0.00520483 0.00533504
 0.00585535 0.68851007 0.00618382 0.00661787 0.00523772 0.00549644
 0.00520909 0.00573416 0.00559859]
Sum of Normalized Output: 1.0000000000000002
Maximum Value of Normalized Output: 0.6885100661720936
```

FIGURE 10. Illustration of the normalized output vector after applying *SR*. The total sum of the vector is preserved as one, ensuring the validity of the probability distribution.

incorporates all the generated outputs while preserving their contributions. The total sum of the combined outputs was confirmed to be 15.0, corresponding to the sum of the probabilities across all classifiers.

In the next step, key computations prepare the combined vector for normalization. The maximum value (max) is identified within the vector, as illustrated in Fig. 9. In this example, the maximum value is 0.6885, which is classified as a single maximum because no other values are equal. Subsequently, S is calculated by summing the exponential values of all non-maximum elements in the vector. This calculation ensures that *SR* reflects the contributions of non-maximum values, maintaining their relative significance. In this case, S is calculated as 72.639, and *SR* is determined to be 0.00429, ensuring that the combined vector retains its probabilistic nature upon normalization.

Finally, *SR* was applied to normalize the combined vector. As shown in Fig. 10, the normalized vector sums to one, demonstrating the successful completion of the normalization process with minimal floating-point precision error. In particular, the maximum value of the normalized output remains unchanged at 0.6885, preserving the integrity of the highest classification probability.

These results confirm the effectiveness of the AES method in normalizing the output of diverse classifiers while preserving the most confident predictions. Following these steps, the next phase involved formally evaluating the method using rigorous verification.

D. CASE STUDY:APPLYING AES ON CLASSIFICATION MODELS

AES is designed to operate at the inference level and can be applied directly to Softmax-normalized outputs from pretrained models without modifying their internal architecture or retraining. To demonstrate the capability of AES to handle diverse classifications in practical scenarios, this section presents a case study involving several lightweight classification models developed using MobileNetV2 [47].

```
Output 1: NoBird: 0.98962, Bird: 0.01038
Output 2: Cat: 0.95556, Dog: 0.04444
Output 3: Car: 0.77805, Human: 0.22195
Output 4: Butterfly: 0.00006, Cow: 0.00007, Flower: 0.00001, Fruit: 0.99985
```

FIGURE 11. Outputs from four trained classifiers in response to a single input image (apple), with *Fruit* receiving the highest probability across all predictions.

1) MODEL OUTPUTS

Each classification model used in this case study was implemented using MobileNetV2, a lightweight convolutional neural network pretrained on ImageNet. Four separate models were fine-tuned for the following tasks: (1) human vs. car, (2) cat vs. dog, (3) bird vs. no-bird, and (4) a four-class model trained on butterfly, cow, flower, and fruit. Each binary classifier was trained on a balanced dataset consisting of 2000 images per class (4000 total), whereas the multiclass model was trained on 8000 images in total. All models were trained using the Adam optimizer with a learning rate of 0.0001 for five epochs. The Softmax activation function was applied to the output layer to produce normalized probability distributions over the target classes.

A single real input image (an apple) was passed through all four trained classifiers to obtain the actual Softmax outputs. Each model produced a normalized probability distribution corresponding to its classification task. Fig. 11 shows the results of each model, confirming the successful generation of distinct task-specific predictions.

As shown in Fig. 11, each classifier produced a valid probability distribution. For the input image of an apple, the four-class model correctly assigned the highest probability (0.9999) to *Fruit*. The other models predicted *no-bird*, *cat*, and *car* as the most likely classes within their respective tasks.

2) APPLYING AES TO MODEL OUTPUTS

Following the generation of individual Softmax outputs, the AES method was applied to combine the predictions from the four classifiers. First, all probability vectors were concatenated into a single vector V_i, resulting in a combined output with a total sum of 4.0, reflecting the sum of the four normalized distributions. AES then identified the highest probability in V_i, which was 0.9999 for the *fruit* class. As this value was the unique maximum, the method proceeded with single-max handling.

The sum of the exponentials for all non-maximum values was calculated as $S = 13.7721$. Based on this value and the maximum probability, the scale rate (SR) was computed as approximately 1.07×10^{-5}. The SR was then applied to the non-maximum elements, while the maximum value (*fruit*) was preserved. The resulting normalized output vector had a total sum of 1.0, confirming its probabilistic validity. Notably, the maximum value remained unchanged, retaining the confidence of the top prediction while integrating outputs from classifiers trained on unrelated tasks.

This example illustrates the practical application of AES with real model outputs, demonstrating its ability to maintain probabilistic integrity and preserve the most confident prediction without requiring architectural modifications. The complete process—including the combined vector, scaling procedure, and final normalized distribution—is presented in Fig. 12.

```
Combined Outputs:
[9.8962e-01 1.0380e-02 9.5556e-01 4.4440e-02 7.7805e-01 2.2195e-01
 6.0000e-05 7.0000e-05 1.0000e-05 9.9985e-01]
Sum: 4.0000

Maximum Value (max): 0.9998526573181152 (Single Max)
Maximum Label: Fruit
Sum of the Exponentials (S): 13.77209758758545
Scale Rate (SR): 1.0698637658332047e-05

Normalized Output: (after applying SR)
NoBird: 2.9999999242136255e-05
Bird: 9.9999997473787752e-06
Cat: 2.9999999242136255e-05
Dog: 9.9999997473787752e-06
Car: 1.9999999494757503e-05
Human: 9.9999997473787752e-06
Butterfly: 9.9999997473787752e-06
Cow: 9.9999997473787752e-06
Flower: 9.9999997473787752e-06
Fruit: 0.9998499751091003

Sum: 1.0
Maximum Value (max): 0.9998526573181152
Maximum Label: Fruit
```

FIGURE 12. Application of AES to real classifier outputs, showing the combined probability vector, the identified maximum value, the computed scale rate (SR), and the final normalized output, confirming probabilistic integrity.

IV. FORMAL VERIFICATION OF THE AES METHOD

Ensuring the correctness of DL systems is critical for high-stakes applications, such as medical diagnostics and autonomous driving. While empirical testing provides useful insights, it cannot offer mathematical guarantees of reliability. Formal verification methods, such as satisfiability modulo theories (SMT) solvers, enable rigorous validation by checking system properties against logical constraints. In this study, the Z3 SMT solver is employed to formally verify that AES outputs satisfy fundamental probabilistic properties.

A. Z3 FORMAL VERIFICATION PROCESS

The verification process was implemented in Z3 through the following steps:

1) **Solver Initialization:** A Z3 solver instance was created to provide the framework for applying constraints and conducting evaluations.
2) **Modeling the Output Vector:** The AES-combined output vector was represented as a set of real-valued variables, each corresponding to a probability in the final distribution.
3) **Constraint Definition:** Two core constraints were applied:
   - *Sum-to-One Normalization:* The total sum of probabilities must equal 1 within a small tolerance (ϵ) to account for floating-point precision.
   - *Maximum Confidence Preservation:* The highest probability in the vector must remain unchanged after normalization, ensuring the top prediction's integrity.

```python
# Formal verification using Z3
solver = Solver()
combined_z3 = [Real(f'combined_{i}') for i in range(len(output))]

# Add constraints
for i in range(len(output)):
    solver.add(combined_z3[i] == output[i])

# Check properties
max_value = np.max(output)
epsilon = 1e-9
solver.add(And(Sum(combined_z3) >= 1 - epsilon, Sum(combined_z3) <= 1 + epsilon))
solver.add(Or([combined_z3[j] == max_value for j in range(len(combined_z3))]))

if solver.check() == sat:
    model = solver.model()
    # Display fractional representation for precision
    print("\nZ3's model evaluation (fractional representation):")
    for var in combined_z3:
        print(f"{var} = {model.evaluate(var)}")
    print("Formal verification passed.")
else:
    print("Formal verification failed.")
```

FIGURE 13. Z3 verification workflow: initialization, output vector modeling, constraint definition, and evaluation.

```
Z3's model evaluation (fractional representation):
combined_0 = 1273535969039573/25000000000000000
combined_1 = 5280678074842199/100000000000000000
combined_2 = 1052877915420291/20000000000000000
combined_3 = 1310823229587/2000000000000000
combined_4 = 7567509896573479/10000000000000000
combined_5 = 6605262974536359/10000000000000000
combined_6 = 6244909339623521/12500000000000000
combined_7 = 1032896132200273/2000000000000000
combined_8 = 6318327355531163/12500000000000000
combined_9 = 5832198991791883/10000000000000000
combined_10 = 1036403397316109/2000000000000000
combined_11 = 5798869810998583/10000000000000000
combined_12 = 5845820122202037/10000000000000000
combined_13 = 1271264275726321/2500000000000000
combined_14 = 5322981975535881/10000000000000000
combined_15 = 5479916579045497/10000000000000000
combined_16 = 1079933561963357/2000000000000000
combined_17 = 2981565158375807/5000000000000000
combined_18 = 5361697157458533/10000000000000000
combined_19 = 1129701550059891/2000000000000000
combined_20 = 9778511877569869/20000000000000000
combined_21 = 5752066330791347/10000000000000000
combined_22 = 4941056781954483/10000000000000000
combined_23 = 5021642854202089/10000000000000000
combined_24 = 3528329973012749/5000000000000000
combined_25 = 2766966848583369/3906250000000000
combined_26 = 1395465118503501/2500000000000000
combined_27 = 2763081517765903/5000000000000000
combined_28 = 5419865900668771/10000000000000000
combined_29 = 5497946729238563/10000000000000000
combined_30 = 3167962195133881/6250000000000000
combined_31 = 1057718370691799/2000000000000000
combined_32 = 1185233808405201/2000000000000000
combined_33 = 2533126243478741/5000000000000000
combined_34 = 2625941432968173/5000000000000000
combined_35 = 2763275724023679/5000000000000000
combined_36 = 1391132297684243/2500000000000000
combined_37 = 552084992230201/1000000000000000
combined_38 = 2165524723480771/4000000000000000
combined_39 = 2760353017766991771/5000000000000000000
combined_40 = 2890662383174787/5000000000000000
combined_41 = 1113953101635671/2000000000000000
combined_42 = 2585251761132311/5000000000000000
combined_43 = 4080948780342897/10000000000000000
combined_44 = 1120007087670353911/2000000000000000000
combined_45 = 2108340007683871/4000000000000000
combined_46 = 5204832039818881/10000000000000000
combined_47 = 5335040839993969/10000000000000000
combined_48 = 1171069497919439/2000000000000000
combined_49 = 868637582715117/1250000000000000
combined_50 = 1236764756887411/2000000000000000
combined_51 = 661787291703413711/1000000000000000000
combined_52 = 2095086157769891/4000000000000000
combined_53 = 2748220084725013/5000000000000000
combined_54 = 1627839701072771/3125000000000000
combined_55 = 3583852431136173/6250000000000000
combined_56 = 2799294058100757/5000000000000000
Formal verification passed.
```

FIGURE 14. Z3 verification results showing fractional representation of the normalized output vector, confirming sum-to-one normalization and maximum confidence preservation.

4) **Verification and Model Evaluation:** The solver checked constraint satisfiability and returned a fractional representation of the normalized vector, confirming compliance with the specified properties.

Fig. 13 shows the verification workflow implemented in Z3.

B. Z3 VERIFICATION RESULTS

The Z3 solver confirmed that AES consistently satisfies both defined properties: the normalized output vector sums to one, and the maximum confidence score is preserved. These results validate AES's ability to maintain probabilistic integrity during output integration, ensuring its suitability for deployment in critical classification tasks. Fig. 14 presents the verification results.

V. DISCUSSION

A. KEY FINDINGS AND IMPLICATIONS

This section discusses the implications of the AES method and highlights its impact on ensemble learning and classification tasks. The AES method effectively addresses a major challenge in ensemble learning—integrating diverse classifiers without requiring modifications to DL architectures.

By leveraging only the outputs of the classifiers, AES eliminates the need for architectural modifications or meta-learners, significantly reducing computational overhead. This approach simplifies classifier integration, making AES accessible even to individuals with limited expertise in ensemble learning techniques.

AES has demonstrated capability to handle up to 40 classifiers in simulations, highlighting its scalability. This level of integration is critical for high-dimensional classification problems in which multiple models contribute to decision-making in real-world applications. In addition, AES enables incremental integration of classifiers without interrupting system operations, offering flexibility in dynamic environments. This characteristic is particularly valuable in real-world applications, such as deepfake detection and medical diagnostics, in which system adaptability is crucial.

The formal verification process using the Z3 SMT solver further validated the reliability and accuracy of AES. By ensuring that the normalized outputs retain their probabilistic integrity and that the highest probability remains unchanged, AES can be positioned as a robust solution for diverse classification environments. These findings underscore AES as a practical, efficient, and scalable method to address challenges in ensemble learning.

B. COMPUTATIONAL COST AND APPLICABILITY

A key advantage of AES is its minimal computational overhead. Operating entirely at the inference level, AES combines Softmax-normalized outputs without modifying model architectures, retraining, or employing meta-learners. This design eliminates the need for backpropagation, gradient updates, or joint optimization, making AES inherently lightweight and fast. In practice, AES requires only basic mathematical operations and can be efficiently executed on standard hardware without GPU acceleration, confirming its suitability for real-time and resource-constrained environments.

Traditional ensemble methods, such as stacking, are not directly applicable to AES use cases. Stacking typically assumes that all classifiers operate over a shared label space and produce output vectors of equal dimensionality. In contrast, AES supports the integration of classifiers trained on unrelated tasks with distinct output dimensions, such as combining a binary classifier (e.g., human vs. car) with a multiclass model (e.g., butterfly, cow, flower, and fruit). Consequently, AES's efficiency cannot be directly compared to stacking; instead, it provides a novel capability that complements its computational simplicity and adaptability.

C. FUTURE DIRECTIONS

Although AES has substantial strengths, there are opportunities for further research and development. One important avenue is extending AES to handle sequential and binary classification tasks, and regression problems. These expansions would broaden the applicability of AES to a wider range of machine learning challenges.

Another direction is the development of a modular AES framework that facilitates seamless integration of diverse classifiers into existing systems. By treating each classifier

as an independent computational unit, this framework facilitates dynamic updating of classifiers without requiring the retraining of the entire system. Such modularity is especially advantageous for autonomous systems and evolving financial models, where adaptability is essential.

Future work will also focus on validating AES on larger, real-world datasets to empirically assess its robustness across diverse domains, dataset scales, and class cardinalities.

By addressing these research directions, AES has the potential to become a foundational approach to classifier integration in machine learning. Its scalability, adaptability, and efficiency make it a strong candidate for future advancements in ensemble-based decision systems.

VI. RELATED WORK

This section reviews existing methods for integrating classifier outputs, highlighting their strengths and limitations. Combining the outputs of diverse classifiers trained on unrelated tasks remains a unique challenge in ensemble learning, as traditional methods such as stacking, bagging, and boosting primarily focus on homogeneous classifiers. These techniques often rely on meta-learners or architectural modifications, making them unsuitable for integrating classifiers with heterogeneous objectives.

Conditional Max-Preserving Normalization (CMPN) [48] was introduced to address this challenge by combining classifier outputs while preserving the probabilistic integrity of the predictions. CMPN employs a max-preserving scaling mechanism, ensuring that the class with the highest probability remains unchanged while the outputs are normalized. This approach allows classifiers trained on unrelated tasks to be integrated seamlessly.

Although CMPN introduced a novel approach to preserving probabilistic integrity in classifier integration, its applicability remains limited. CMPN has been validated using only four classifiers, and its max-preserving scaling does not account for confidence-level variations across classifiers. As a result, CMPN struggles in large-scale ensembles, where classifier outputs exhibit significant variability.

To address these limitations, AES introduces an exponential scaling mechanism, enabling it to handle up to 40 classifiers while maintaining confidence-level consistency. By accounting for variability in classifier outputs, AES improves the reliability of probabilistic integration and expands the applicability of ensemble learning to diverse real-world scenarios. Its demonstrated scalability makes it particularly valuable in applications such as deepfake detection and medical diagnostics, where classifier adaptability and precision are critical.

By focusing exclusively on classifier outputs, AES represents a significant advancement in methods for integrating diverse classifiers. Its scalability, precision, and computational efficiency make it a practical solution for addressing the challenges of ensemble learning in heterogeneous classification environments.

VII. CONCLUSION

This study introduced AES as a robust solution to the limitations of traditional ensemble learning techniques. By leveraging probabilistic outputs, such as those generated by the Softmax function, AES avoids modifying classifier architectures or requiring additional meta-learning, thereby reducing computational complexity. This makes AES both efficient and accessible, even for users with limited expertise in ensemble methods.

AES enhances flexibility by enabling the incremental addition of classifiers without interrupting ongoing processes, making it particularly valuable in dynamic systems. Its scalability and reliability were demonstrated through both simulations and real-world case studies, with formal verification ensuring probabilistic integrity and preservation of the highest classification confidence score. These properties establish AES as a reliable and adaptable method for diverse classification environments.

Future work will explore extending AES to sequential and binary classification tasks, as well as regression problems. Furthermore, developing a modular framework for seamless integration of classifiers could enhance its adoption in real-world applications, such as autonomous systems and financial analytics.

With these advancements, AES has the potential to serve as a foundational framework for classifier integration in diverse machine learning applications, offering a scalable, efficient, and adaptable approach to ensemble learning.

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," *Artif. Intell. Rev.*, vol. 56, no. 6, pp. 5545–5589, Jun. 2023, doi: 10.1007/s10462-022-10283-5.

[3] A. A. Khan, O. Chaudhari, and R. Chandra, "A review of ensemble learning and data augmentation models for class imbalanced problems: Combination, implementation and evaluation," *Expert Syst. Appl.*, vol. 244, Jun. 2024, Art. no. 122778, doi: 10.1016/j.eswa.2023.122778.

[4] A. Mohammed and R. Kora, "A comprehensive review on ensemble deep learning: Opportunities and challenges," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 35, no. 2, pp. 757–774, Feb. 2023, doi: 10.1016/j.jksuci.2023.01.014.

[5] K. Aswini, U. Reddy, A. Nagpal, A. Rana, P. Praveen, and B. S. Z. Abood, "Ensemble learning approaches for big data classification tasks," in *Proc. 10th IEEE Uttar Pradesh Sect. Int. Conf. Electr., Electron. Comput. Eng. (UPCON)*, Dec. 2023, pp. 1545–1550, doi: 10.1109/UPCON59197.2023.10434616.

[6] M. M. Mitu, S. Arefin, Z. Saurav, M. A. Hasan, and D. M. Farid, "Pruning-based ensemble tree for multi-class classification," in *Proc. 6th Int. Conf. Electr. Eng. Inf. Commun. Technol. (ICEEICT)*, May 2024, pp. 481–486, doi: 10.1109/iceeict62016.2024.10534584.

[7] P. Wu, R. Ma, and T. T. Toe, "Stacking-enhanced bagging ensemble learning for breast cancer classification with CNN," in *Proc. 3rd Int. Conf. Electron. Eng. (ICEEM)*, Oct. 2023, pp. 1–6, doi: 10.1109/iceem58740.2023.10319517.

[8] Y. Xie, A. Li, L. Gao, and Z. Liu, "A heterogeneous ensemble learning model based on data distribution for credit card fraud detection," *Wireless Commun. Mobile Comput.*, vol. 2021, no. 1, Jan. 2021, Art. no. 2531210, doi: 10.1155/2021/2531210.

[9] 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.

[10] L. Rokach, *Ensemble Learning: Pattern Classification Using Ensemble Methods*, 2nd ed., Singapore: World Scientific, 2019.

[11] Y. Lu, H. Xu, C. Wang, G. Yan, Z. Huo, Z. Peng, B. Liu, and C. Xu, "A novel strategy coupling optimised sampling with heterogeneous ensemble machine-learning to predict landslide susceptibility," *Remote Sens.*, vol. 16, no. 19, p. 3663, Oct. 2024, doi: 10.3390/rs16193663.

[12] H. Jamalinia, S. Khalouei, V. Rezaie, S. Nejatian, K. Bagheri-Fard, and H. Parvin, "Diverse classifier ensemble creation based on heuristic dataset modification," *J. Appl. Statist.*, vol. 45, no. 7, pp. 1209–1226, May 2018, doi: 10.1080/02664763.2017.1363163.

[13] G. Yao, H. Zeng, F. Chao, C. Su, C.-M. Lin, and C. Zhou, "Integration of classifier diversity measures for feature selection-based classifier ensemble reduction," *Soft Comput.*, vol. 20, no. 8, pp. 2995–3005, Aug. 2016, doi: 10.1007/s00500-015-1927-7.

[14] L. D. Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Proc. Int. Conf. Tools Algorithms Constr. Anal. Syst.*, 2008, pp. 337–340, doi: 10.1007/978-3-540-78800-3_24.

[15] M. Bukowski, J. Kurek, I. Antoniuk, and A. Jegorowa, "Decision confidence assessment in multi-class classification," *Sensors*, vol. 21, no. 11, p. 3834, Jun. 2021, doi: 10.3390/s21113834.

[16] R. Ranawana and V. Palade, "Multi-classifier systems: Review and a roadmap for developers," *Int. J. Hybrid Intell. Syst.*, vol. 3, no. 1, pp. 35–61, Apr. 2006, doi: 10.3233/his-2006-3104.

[17] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview," 2020, *arXiv:2008.05756*.

[18] 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.

[19] H. Yan, S. Li, Y. Wang, Y. Zhang, K. Sharif, H. Hu, and Y. Li, "Membership inference attacks against deep learning models via logits distribution," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 5, pp. 3799–3808, Sep. 2023, doi: 10.1109/TDSC.2022.3222880.

[20] M. Li, F. Su, O. Wu, and J. Zhang, "Class-level logit perturbation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 10, pp. 13926–13940, Oct. 2024, doi: 10.1109/TNNLS.2023.3273355.

[21] R.-T. Liaw and Y.-W. Wen, "Ensemble learning through evolutionary multitasking: A formulation and case study," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 8, no. 4, pp. 3081–3094, Apr. 2024, doi: 10.1109/TETCI.2024.3369949.

[22] L. Nanni, L. Trambaiollo, S. Brahnam, X. Guo, and C. Woolsey, "Ensemble of networks for multilabel classification," *Signals*, vol. 3, no. 4, pp. 911–931, Dec. 2022, doi: 10.3390/signals3040054.

[23] Q. Zhang, E. C. C. Tsang, Q. He, and Y. Guo, "Ensemble of kernel extreme learning machine based elimination optimization for multi-label classification," *Knowl.-Based Syst.*, vol. 278, Oct. 2023, Art. no. 110817, doi: 10.1016/j.knosys.2023.110817.

[24] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, Aug. 1996, doi: 10.1007/bf00058655.

[25] M. A. Ganaie, M. Hu, A. K. Malik, M. Tanveer, and P. N. Suganthan, "Ensemble deep learning: A review," *Eng. Appl. Artif. Intell.*, vol. 115, Oct. 2022, Art. no. 105151, doi: 10.1016/j.engappai.2022.105151.

[26] L. Yuningsih, G. A. Pradipta, D. Hermawan, P. D. W. Ayu, D. P. Hostiadi, and R. R. Huizen, "IRS-BAG-integrated radius-SMOTE algorithm with bagging ensemble learning model for imbalanced data set classification," *Emerg. Sci. J.*, vol. 7, no. 5, pp. 1501–1516, Oct. 2023, doi: 10.28991/esj-2023-07-05-04.

[27] C. T. Tran, "Ensemble learning approaches for classification with high-dimensional data," *J. Sci. Technique*, vol. 12, no. 1, Jun. 2023, doi: 10.56651/lqdtu.jst.v12.n1.659.ict.

[28] Z. Khan, M. U. T. Alvi, M. A. Tahir, and S. Memon, "Medical diagnostic by data bagging for various instances of neural network," in *Proc. ICPR Int. Workshops Challenges*, Jan. 2021, pp. 291–298, doi: 10.1007/978-3-030-68793-9_21.

[29] S. van der Zon, O. Z. Ben Mordehai, T. S. Vrijdag, W. van Ipenburg, J. W. Veldsink, W. Duivesteijn, and M. Pechenizkiy, "BoostEMM: Transparent boosting using exceptional model mining," in *Proc. 2nd Workshop Mining Data Financial Appl. (MIDAS)*, Skopje, Macedonia, Sep. 18, 2017, pp. 5–16.

[30] G. Haixiang, L. Yijing, L. Yanan, L. Xiao, and L. Jinling, "BPSO-AdaBoost-KNN ensemble learning algorithm for multi-class imbalanced data classification," *Eng. Appl. Artif. Intell.*, vol. 49, pp. 176–193, Mar. 2016, doi: 10.1016/j.engappai.2015.09.011.

[31] A. Doganer, "Different approaches to reducing bias in classification of medical data by ensemble learning methods," *Int. J. Big Data Anal. Healthcare*, vol. 6, no. 2, pp. 15–30, Jul. 2021, doi: 10.4018/ijbdah.20210701.oa2.

[32] J. Ugirumurera, E. A. Bensen, J. Severino, and J. Sanyal, "Addressing bias in bagging and boosting regression models," *Sci. Rep.*, vol. 14, no. 1, Aug. 2024, Art. no. 18452, doi: 10.1038/s41598-024-68907-5.

[33] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, Jan. 1992, doi: 10.1016/s0893-6080(05)80023-1.

[34] X. Zhu, J. Hu, T. Xiao, S. Huang, Y. Wen, and D. Shang, "An interpretable stacking ensemble learning framework based on multi-dimensional data for real-time prediction of drug concentration: The example of olanzapine," *Frontiers Pharmacol.*, vol. 13, Sep. 2022, Art. no. 975855, doi: 10.3389/fphar.2022.975855.

[35] H. Tian, H. Kong, and C. Wong, "A novel stacking ensemble learning approach for predicting PM2.5 levels in dense urban environments using meteorological variables: A case study in Macau," *Appl. Sci.*, vol. 14, no. 12, p. 5062, Jun. 2024, doi: 10.3390/app14125062.

[36] R. Dey and R. Mathur, "Ensemble learning method using stacking with base learner, a comparison," in *Proc. Int. Conf. Data Anal. Insights*, 2023, pp. 159–169, doi: 10.1007/978-981-99-3878-0_14.

[37] B. Pavlyshenko, "Using stacking approaches for machine learning models," in *Proc. IEEE 2nd Int. Conf. Data Stream Mining Process. (DSMP)*, Aug. 2018, pp. 255–258, doi: 10.1109/DSMP.2018.8478522.

[38] M. Zounemat-Kermani, O. Batelaan, M. Fadaee, and R. Hinkelmann, "Ensemble machine learning paradigms in hydrology: A review," *J. Hydrol.*, vol. 598, Jul. 2021, Art. no. 126266, doi: 10.1016/j.jhydrol.2021.126266.

[39] N. Rane, S. P. Choudhary, and J. Rane, "Ensemble deep learning and machine learning: Applications, opportunities, challenges, and future directions," *Stud. Med. Health Sci.*, vol. 1, no. 2, pp. 18–41, Jul. 2024, doi: 10.48185/smhs.v1i2.1225.

[40] M. P. Sesmero, J. M. Alonso-Weber, G. Gutierrez, A. Ledezma, and A. Sanchis, "An ensemble approach of dual base learners for multi-class classification problems," *Inf. Fusion*, vol. 24, pp. 122–136, Jul. 2015, doi: 10.1016/j.inffus.2014.09.002.

[41] K. G. Larsen, A. Legay, G. Nolte, M. Schlüter, M. Stoelinga, and B. Steffen, "Formal methods meet machine learning (F3ML)," in *Proc. Int. Symp. Leveraging Appl. Formal Methods*, 2022, pp. 393–405, doi: 10.1007/978-3-031-19759-8_24.

[42] S. Bensalem, X. Huang, W. Ruan, Q. Tang, C. Wu, and X. Zhao, "Bridging formal methods and machine learning with model checking and global optimisation," *J. Log. Algebr. Methods Program.*, vol. 137, pp. 1–19, Feb. 2024, doi: 10.1016/j.jlamp.2023.100941.

[43] M. Krichen, A. Mihoub, M. Y. Alzahrani, W. Y. H. Adoni, and T. Nahhal, "Are formal methods applicable to machine learning and artificial intelligence?" in *Proc. 2nd Int. Conf. Smart Syst. Emerg. Technol. (SMARTTECH)*, May 2022, pp. 48–53, doi: 10.1109/SMARTTECH54121.2022.00025.

[44] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Proc. 29th Int. Conf. Comput.-Aided Verification (CAV)*, Berlin, Germany, 2017, pp. 3–29, doi: 10.1007/978-3-319-63387-9_1.

[45] Y. Zhang, Z. Wei, X. Zhang, and M. Sun, "Using Z3 for formal modeling and verification of FNN global robustness," 2023, *arXiv:2304.10558*.

[46] G. Einziger, M. Goldstein, Y. Sa'ar, and I. Segall, "Verifying robustness of gradient boosted models," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 2446–2453, doi: 10.1609/aaai.v33i01.33012446.

[47] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.

[48] A. A. Najafabadi, F. Nejati, N. K. Yap, A. B. Md. Sultan, M. A. Ali, and Z. N. Ashani, "Conditional max-preserving normalization: An innovative approach to combining diverse classification models," *Int. J. Adv. Sci., Eng. Inf. Technol.*, vol. 14, no. 6, pp. 1976–1981, Dec. 2024, doi: 10.18517/ijaseit.14.6.17344.

AMIN ARAB NAJAFABADI received the B.S. degree in irrigation engineering from Islamic Azad University, Kermanshah Branch, Kermanshah, Iran, in 2012, and the M.S. degree in information technology management from Asia–Pacific University of Technology and Innovation (APU), Kuala Lumpur, Malaysia, in 2016. He is currently pursuing the Ph.D. degree in software engineering with the Faculty of Computer Science and Information Technology, Kuala Lumpur.

He was an External Co-Researcher at Universiti Tunku Abdul Rahman (UTAR), in 2024, for a UTARRF-funded project on deepfake detection. His research interests include software engineering for deep learning (SE4DL), with a focus on classification, component-based development, and artificial general intelligence (AGI).

NG KENG YAP (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from Universiti Putra Malaysia, in 2001 and 2005, respectively, and the Ph.D. degree in computer science from The University of Manchester, U.K., in 2015.

He is currently a Senior Lecturer with the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia. He has authored articles in IEEE Access and other indexed journals. He has been involved in multiple research projects, including studies on palm oil production analytics, traffic flow analysis, and disruptive technology in construction project management. His research interests include software components, business analytics, and software engineering for artificial intelligence (SE4AI) systems.

Dr. Keng Yap is a Professional Technologist Member of Malaysian Board of Technology (MBOT).

ZAHRA NAZEMI ASHANI received the B.S. degree in statistics from Sheikh Bahaei University, Isfahan, Iran, in 2006, the M.S. degree in statistics from Payame Noor University, Shiraz, Iran, in 2010, and the Ph.D. degree in mathematical statistics from Universiti Putra Malaysia, Selangor, Malaysia, in 2017.

She was affiliated with Universiti Putra Malaysia, in 2022. She is currently a Data Scientist with Beslogic IA Inc., Canada. She has authored research articles in *Journal of Advanced Research in Applied Sciences and Engineering Technology*, *Arabian Journal for Science and Engineering*, and *Modern Applied Science*. Her research interests include machine learning, image processing, and natural language processing.

FARANAK NEJATI (Member, IEEE) received the M.S. degree in computer science (software engineering) from Tabriz University, Tabriz, Iran, in 2012, and the Ph.D. degree in computer science (software engineering) from Universiti Putra Malaysia, Selangor, Malaysia, in 2019.

She was a Senior Software Engineer at Willowglen MSC Berhad, from 2018 to 2023. She is currently an Assistant Professor with Universiti Tunku Abdul Rahman, Malaysia. She has authored articles in IEEE Access and *International Journal on Advanced Science, Engineering and Information Technology*. She has received research grants, including the UTAR Research Fund (UTARRF), for projects on deepfake detection and generative adversarial networks for traffic signal control. Her research interests include deep learning, augmented intelligence, generative artificial intelligence, large language models, component-based software engineering, formal methods and theorem proving, and model checking.

Dr. Nejati is a Registered Graduate Technologist (GT) with Malaysia Board of Technologists (MBOT).

MOHAMED ABDULLAHI ALI is currently pursuing the Ph.D. degree in software engineering with Universiti Putra Malaysia (UPM). His research interests include software quality, component-based software, data-driven software components, machine learning software, and artificial intelligence.