DeepAbstraction++: Enhancing Test Prioritization Performance via Combined Parameterized Boxes*

Hamzah Al-Qadasi[1], Yliès Falcone[2], and Saddek Bensalem[1]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, Verimag, 38000 Grenoble, France
[2] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
firstname.lastname@univ-grenoble-alpes.fr

Abstract. In artificial intelligence testing, there is an increased focus on enhancing the efficiency of test prioritization methods within deep learning systems. Subsequently, the DeepAbstraction algorithm has recently become one of the leading techniques in this area. It employs a box-abstraction concept, the efficiency of which depends on the tau parameter, the clustering parameter, that influences the size of these boxes. The conclusion of the previous experiments using tau values of 0.4 or 0.05 has failed to produce optimal results among all experiments. This highlights a significant challenge in the DeepAbstraction framework concerning the appropriate selection of the tau parameter. The selection of the tau value is extremely crucial, given its decisive effect on box size and, subsequently, the stability and efficacy of the framework. Addressing this challenge, we propose a methodology called combined parameterized boxes. This approach leverages the collective verdicts of monitors with various tau values to evaluate network predictions. We assign appropriate weights to these verdicts to ensure that no single verdict influences the decision-making process, thereby ensuring balance. Furthermore, we propose multiple strategies for integrating the weighted verdicts of monitors into a conclusive verdict, such as mean, max, product, and mode. The results of our investigation demonstrate that our approach can notably boost the DeepAbstraction framework's performance. Compared to the leading algorithms, DeepAbstraction++ consistently outperforms its competitors, delivering an increase in performance between 2.38% and 7.71%. Additionally, DeepAbstraction++ brings remarkable stability to the process, addressing a significant shortcoming of the earlier version of DeepAbstraction.

Keywords: Test Prioritization, Deep Learning Systems, Box Abstraction, Runtime Monitoring, Big Data, Effective Labeling, Performance Improvement, Stability.

1 Introduction

Deep Neural Networks (DNNs) have made remarkable strides in a variety of fields, ranging from autonomous driving [21], aviation [15], and healthcare [15]. While these

* This paper is supported by the European Horizon 2020 research and innovation programme under grant agreement No. 956123 and by the French National Research Agency (ANR) in the framework of the Investissements d'Avenir program (ANR-10-AIRT-05, irtnanoelec).

networks have achieved extensive success, they still have drawbacks, particularly concerning their quality and reliability. Notable examples of these weaknesses have manifested in real-world applications, such as incidents involving self-driving cars from Google [22]. These incidents highlight the importance of detecting and correcting malfunctions in software based on Deep Neural Networks (DNNs). Test prioritization is a technique that ranks unlabeled test instances based on their potential to identify errors and select a subset of the entire test dataset, allowing for a selective inspections of a subset from the entire dataset. This not only enhances inspection efficiency but also effectively mitigates the labeling costs that arise from the expanding volumes of data in deep learning. Lastly, test prioritization can help in identifying potential issues earlier before production, thereby preventing catastrophic outcomes.

In the realm of test prioritization algorithms for deep learning systems, which includes approaches, such as Byun et al. [3], DeepGini [7], PRIMA [18], TestRank [12], GraphPrior [5], and CertPri [5], DeepAbstraction [2] has shown a recent and promising performance. Uniquely, DeepAbstraction exploits the capabilities of runtime monitors to prioritize error-revealing test instances. In the literature, runtime monitors [9] are used to supervise the neural network predictions and trigger one of the following verdicts: acceptance, rejection, or uncertainty. Contrarily to traditional runtime verification monitors, which are bound to one or more properties, we consider monitors that check whether a newly observed instance belongs to a set of instances englobing a set of references instances constructed during training. Thus, we construct primarily runtime monitors during training from *box abstractions* [9], which are clusters of instances with similar high-level characteristics. During testing, the specific box abstraction where a test instance is located determines the monitor's verdict. This innovative approach enhances the efficiency of test prioritization, demonstrating the impressive capabilities of DeepAbstraction.

The earlier version of DeepAbstraction, despite showing some level of efficacy, currently reveals several flaws which undermine its overall performance. A notable example is the static nature of the clustering parameter, tau. This parameter, custom-tailored according to training accuracy, has a different optimal value in each experiment. However, this indicates high instability in the performance, posing a considerable challenge to the model's reliability. Moreover, the framework is heavily reliant on the verdict of a single monitor, which is often insufficient to provide a comprehensive understanding of the data distribution.

To address these issues, we implement substantial improvements in DeepAbstraction framework. We introduce a dynamic approach to selecting tau, which effectively replaces the previous fixed-value system. This new method offers greater adaptability to the unique distribution of each dataset. Secondly, we overcome the limitations of relying on a single monitor by incorporating a new concept, "combined parameterized boxes," which harnesses the collective verdicts of various monitors. This feature increases the verdict's accuracy. Thirdly, when there is a conflict among different verdicts, we introduce a weighting system to balance the decision-making process. This system assigns unique weights to rejection, acceptance, and uncertainty verdicts, ensuring the decisions made are both effective and fair. Finally, we develop multiple combination strategies for the final verdict of monitors to allow the system to decide the most effective conclusion.

If two test instances share the same combined verdict score, the algorithm uses Gini index scores for refined prioritization. These improvements collectively ensure a more effective and stable DeepAbstraction performance.

In this paper, we consolidate our main contributions as follows:

- We comprehensively analyze the earlier version of DeepAbstraction, highlighting the weaknesses compromising its performance and reliability.
- We introduce the concept of "combined parameterized boxes" to leverage the collective verdicts of multiple monitors, enhancing the accuracy of our system's decisions.
- We establish a unique weighting system with a combination strategy that balances the decision-making process when conflicts arise among different verdicts of monitors, optimizing the fairness of the system's decisions.

We organize the paper as follows: Section 2 introduces DeepAbstraction's principles. Section 4 details the role of clustering, the tau selection challenge, and our proposed solution. The experimental setup and summary of main experiments are outlined in Section 5. Section 6 focuses on answering research questions and evaluating our system. Finally, Section 7 reflects on our solution to the tau selection challenge, its impact on DeepAbstraction, and potential future enhancements.

2 Background

This section provides the foundational knowledge needed to grasp the key concepts of DeepAbstraction [2], including the runtime monitors and the statistical scoring function, e.g., the Gini index. Furthermore, we briefly explain the workflow of DeepAbstraction.

2.1 Runtime Monitors

Several studies have been carried out on runtime monitoring, including [4, 9, 11, 19]. In particular, DeepAbstraction adopts the three-verdict monitor system as developed by [19]. These monitors are the key to monitoring the predictions made by the neural network once it's operational. More specifically, the monitors assess the predictions of the neural network and trigger one of three possible verdicts: rejection, uncertainty, or acceptance. Subsequently, we will revisit the process of building these runtime monitors and explain their role briefly. We refer the reader to [9] to understand more how to construct boxe-abstraction monitors in details.

Monitor Construction Upon the completion of the training process, each training instance is thoroughly analyzed. This process includes extracting the high-level features from the penultimate layer (just before the softmax layer) with the corresponding predicted class. Subsequently, these features are grouped into unique subsets depending on whether the neural network correctly classifies these features. The method known as *box abstraction* is then applied. This approach forms n-dimensional boxes in the

feature space, representing clusters of either correctly or incorrectly classified training instances. In essence, each box is a bounding box encompassing a cluster of training instances. Following this, there are two categories of boxes for every class: those encompassing correctly classified instances and those for misclassified ones. The core idea behind this technique is the observation that samples from the same class exhibit similar patterns in the feature space, enabling the system to more effectively assess and categorize new test samples.

Monitor Operation During the neural network evaluation, the decision of the monitors is highly affected by the location of the test instance within the predefined boxes. More specifically, monitors approve the neural network predictions when the test instance is within a box containing correctly classified samples. Conversely, if a test instance is within the box that contains inaccurately classified instances, the monitors reject the prediction. The state of uncertainty arises for the monitor when the test instance lies within an area that overlaps the correctly and incorrectly classified boxes. Lastly, the monitors reject the prediction, if the test instance is outside all the boxes.

2.2 Gini Index

The Gini Index (GI) or Gini Impurity value is customized from the decision tree in machine learning [14] to be used in deep learning as a measure to estimate the uncertainty of the probability distribution in the output layer. Thus, we consider the Gini index, here initiated for DNN. It ranges between 0 and 1, where zero indicates no uncertainty in the prediction, implying that the Deep Neural Network (DNN) is entirely certain of its prediction. On the other hand, as the value approaches 1, it suggests that the neural network is increasingly uncertain. The calculation of Gini Impurity (GI) is as follows:

$$GI = 1 - \sum_{i=1}^{C} p_i^2 \tag{1}$$

where p_i is derived from the output layer of the DNN which is often a softmax activation function. Thus, p_i is the probability of an instance being classified to class i and $\sum_{i=1}^{C} p_i = 1$ and C is the total number of classes.

In the test prioritization context, DeepGini [7] and DeepAbstraction utilize GI as a scoring function to estimate the error-revealing capability of each test instance. As a result, instances with high GI are prioritized over other test instances with low GI since they are more likely to show weakness in the model.

Example 3 Assume we have a binary classification with two classes {●, ■}. We also have six test instances with the network predictions. As demonstrated in Table 1, the Gini Impurity (GI) measures the uncertainty of the neural network towards the test instances. We can observe that instance D has the highest impurity score (GI = 0.5), which is the most likely to be misclassified by the network. The prioritization list contains other instances (C, F, E, A, and B) prioritized in descending order, as shown in the last column.

Table 1: Prioritizing Test Data Using the Gini Index (GI)

Instance	Ground truth	DNN output	Predicted class	Gini index	Order
A	●	0.90 , 0.10	●	0.18	5
B	■	0.00 , 1.00	■	0.00	6
C	●	0.40 , 0.60	■	0.48	2
D	■	0.50 , 0.50	●	0.50	1
E	■	0.25 , 0.75	■	0.38	4
F	●	0.35 , 0.65	■	0.46	3

2.3 DeepAbstraction

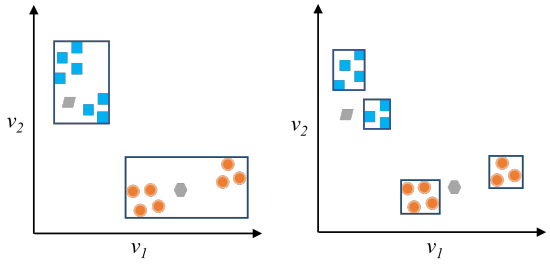
DeepAbstraction framework essentially involves a two-level prioritization process for unlabeled test inputs. Specifically, the initial stage involves high-level prioritization, where we categorize test instances into three groups based on the verdicts of monitors. The subsequent stage encompasses ordering the test instances within each category in decreasing order, guided by the value determined by the scoring function, such as the Gini Index (GI). At the last phase of this process, instances with a GI value of zero are removed from the first two categories since they are redundant, and the network is highly confident. Lastly, a certain number of instances are chosen within the predefined labeling budget. For more details, we refer the reader to the first version of DeepAbstraction [2].

3 Problem Analysis

In this section, we explain the crucial role that clustering plays within the framework. We then discuss the challenge of tau selection and its consequential effects on the overall performance of the framework.

3.1 Clustering

Box-abstraction monitors are built based on the presumption that instances of the same class show similar patterns due to their greater contiguity within the feature space than instances of other classes. However, monitors may incorrectly validate the network's prediction for a new test input that closely mimics instances within a box, even though this instance originates from another class. Therefore, to alleviate this problem of false negatives, the k-means clustering algorithm is applied before box construction. After clustering, each cluster forms a small box rather than a large box for all clusters. Figure 1 illustrates how monitors in Fig. (a) falsely accept the predictions as a square and a circle for parallelogram and hexagon instances, respectively, i.e., novel classes not in the training dataset. After clustering in Fig. (b), the monitors correctly reject the predictions as they are outside all boxes.



(a) Without clustering (b) With clustering

Fig. 1: Novel test instances before and after clustering [2].

3.2 Tau Selection Issue

DeepAbstraction controls the size of each box by a pre-specified parameter, namely the clustering parameter (τ), which has one of the following values: 0.4, 0.3, 0.2, 0.1, 0.05, and 0.01. The dynamic relationship between the value of τ and the box size is such that a decrease in τ value shrinks the box, whereas an increase expands it. The challenge is to select the best τ that optimally reduces the frequency of false negatives while enhancing the number of true positives. The choice of the ideal τ is deeply influenced by the inherent distribution of the dataset, which may vary across different classes. Therefore, DeepAbstraction lacks a definitive guideline for selecting the best tau across several benchmarks. For instance, DeepAbstraction suggests setting τ to 0.05 as a default value for models with a training accuracy of less than 98%, while a τ of 0.4 is for exceptionally accurate models. These values of τ are experimentally validated. However, these default values of τ are empirical consensus rather than optimal values over all benchmarks, as shown in Fig. 2. For instance, when τ is 0.01 in the following experiments achieves better results than τ of 0.05: Exp.3 and Exp.6. Similarly, the performance of DeepAbstraction with τ of 0.4 in Exp.5 is less effective than with τ of 0.3.

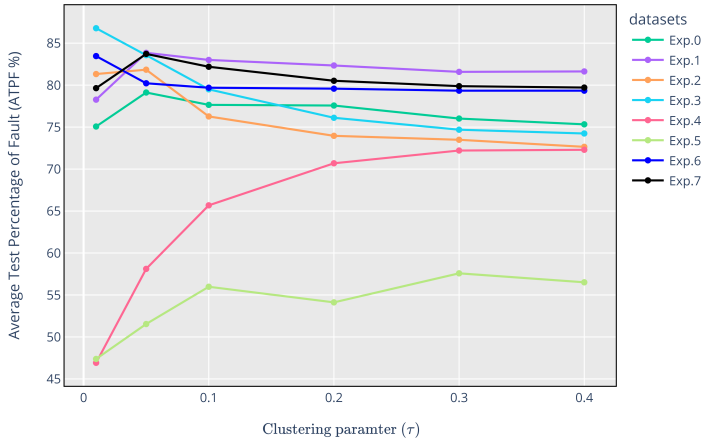


Fig. 2: The effect of clustering parameter τ on performance [2].

4 Approach

In this section, we present the proposed solution, accompanied by its formal definitions. Then, we provide an illustrative example for the updated framework.

4.1 Combined Parameterized Boxes

Figure 2 illustrates that a predefined value of τ cannot effectively improve the performance. There is an observable discrepancy in the performance, i.e., small values of τ work in some cases better than large values, and vice versa. Therefore, we propose an inclusive approach that depends on all monitors' verdicts of different τ to accept or reject the neural network predictions. This integration represents so-called combined parameterized boxes, which collectively involve the predictive potential of every fixed-size box. With this approach, we effectively address the problem of τ selection.

The primary responsibility of monitors is to carefully reject any erroneous predictions. This task gains importance when disagreements arise among monitors of different τ. Amidst such conflict, if even a single verdict signals rejection, the final decision leans towards rejection. Our experimental evaluation further supports the effectiveness of this approach. In response to these findings, we develop a strategic approach to assign weights to the monitors' verdicts. This approach places a higher weight on rejection verdicts than other verdicts, while uncertainty verdicts carry more weight than acceptance ones. It's crucial, however, to maintain a careful balance - the weight differences should not be so significant that the heavily weighted verdicts negate the lesser ones. For instance, overemphasis on rejection verdicts can cancel the contributions of other verdicts, thereby negatively impacting the prioritization performance in subsequent stages. To formalize this approach, we mathematically express the monitor's verdicts in the following order: acceptance [**a**], uncertainty [**u**], and rejection [**r**]:

$$\mathbf{a} = \gamma, \tag{2}$$

$$\mathbf{u} = \mathbf{a} + \beta, \tag{3}$$

$$\mathbf{r} = \mathbf{u} + 2 * \beta \tag{4}$$

where γ and β are arbitrary positive real numbers.

We start to randomly select the values of γ and β. Then we compute the weights of the verdicts according to the above equations. We can also observe that the acceptance weight can be any positive real number except zero since zero denotes no contribution. Furthermore, the uncertainty weight is greater than the acceptance weight with β. Moreover, the rejection weight is larger than the uncertainty weight by $2 * \beta$. Lastly, if we substitute (3) in (4), we infer that the rejection weight is larger than the acceptance weight by $3 * \beta$. In the experimental evaluation section, we will see how different values of β should not affect the performance stability of the combined monitors. In other words, the performance stability is independent of the β selection value.

4.2 Combination Strategy

Numerous strategies exist to merge the weighted verdicts of different monitors and yield a final, cumulative verdict. We highlight a few of these approaches below, with a more comprehensive evaluation of each to follow in the experimental evaluation section:

- *Mean:* This strategy involves adding all weighted verdicts and dividing the total by the number of verdicts, in this case, six—corresponding to the τ values of 0.4, 0.3, 0.2, 0.1, 0.05, and 0.01.
- *Max:* This strategy selects the largest weighted verdict among the six. If a rejection is present, it automatically becomes the final verdict, followed by uncertainty or acceptance verdicts, as applicable.
- *Product:* As the name suggests, this strategy takes the product of all weighted verdicts as the final verdict.
- *Mode:* This fundamentally operates as a voting strategy, where the final verdict is the weighted verdict appearing most frequently among the others.

By evaluating these strategies, we aim to provide insight into their efficacy and relevance in the context of our monitoring system.

4.3 Illustrative Example

Imagine we task a neural network with a binary classification scenario where it should differentiate between plane and bird. As depicted in Fig. 3, the predictions highlighted in red represent misclassifications and should, therefore, be prioritized. Conversely, the ones highlighted in blue indicate the correct classification. We can observe that the upper part of Fig. 3 shows DeepAbstraction version 1, which consists of 6 verdicts according to the τ value. In this version, the user should select *only one* verdict depending on the τ value determined by the model's training accuracy. However, there are some cases where training accuracy does not sufficiently capture the model's learning capability, e.g., overfitting. Ultimately, it's crucial to consider verdicts of false negatives and false positives, which are marked in red, to assess the predictive performance of the monitors.

Our process begins by assigning weights to the acceptance, uncertainty, and rejection verdicts, according to eq. (2)-(4), yielding respective values of 1, 4, and 10. From there, we incorporate the verdicts of different monitors using the *mean* combination strategy to ascertain the final verdict. Following this, we prioritize the test instances based on the final verdict. However, in cases where two test instances possess an equal combined verdict score, we turn to the GI score for prioritization. For example, the fourth and fifth test instances have a combined verdict of 4.5, prompting the algorithm to prioritize the fourth over the fifth based on their GI scores. Finally, after the prioritization completion, we label the first n test instances. Here, n represents the predetermined labeling budget, setting the threshold for the number of instances to be labeled.

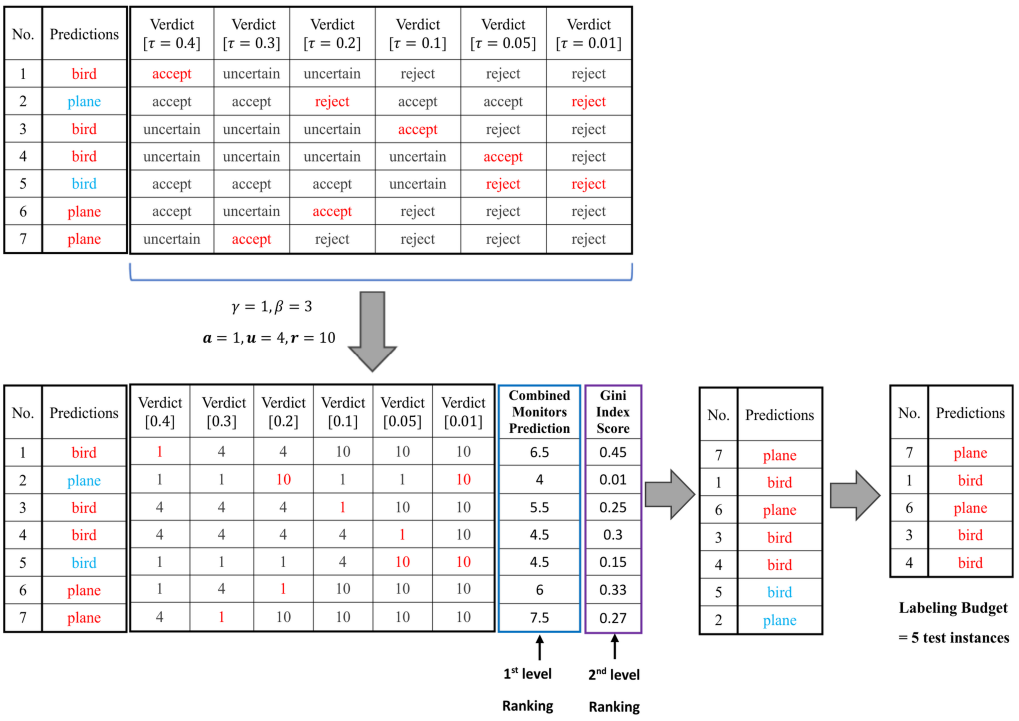


Fig. 3: Transition from DeepAbstraction to DeepAbstraction++.

5 Experimental Setup

We conduct the experiments on a system equipped with an Nvidia K80 GPU and 12 GB of RAM, with PyTorch v1.9.0 as the underlying framework. Table 2 summarizes the principal experiments. The configurations used for the primary setup are as follows:

- **Datasets**: MNIST [6], Fashion-MNIST [20], CIFAR10 [10], SVHN [13].
- **Pretrained Model**: ResNet18 [8], GoogLeNet [16], ResNet34 [8], ResNet50 [8], ResNet101 [8], ResNet152 [8], and EfficientNet-B0 [17].
- **Prioritization Algorithms**: DeepGini [7], DeepAbstraction [2], and DeepAbstraction++.
- **Evaluation Metrics**: We use the Weighted Faults Detection Ratio (WFDR) metric to evaluate prioritization algorithms, according to [1]. This metric outperforms other metric in effectively assessing the quality of prioritization algorithms. Unlike other metrics, the WFDR metric involves the prioritization difficulty which highly depends on the dataset size and the labeling budget.
- **Research Questions**:

  ❶ **(Weights Effectiveness)**: How effective are the verdict weights proposed in eq. (2)-(4)?

  ❷ **(Algorithms Effectiveness)**: How effective is DeepAbstraction++ compared to the state-of-the-art (SOTA) algorithms?

  ❸ **(Performance Stability)**: How does tuning the γ and β parameters influence the performance of DeepAbstraction++?

  ❹ **(Combination Strategy Selection)**: Which combination strategy provides better performance in terms of algorithm effectiveness?

Table 2: Details of the datasets and pretrained models.

Exp ID	Dataset	Training Dataset	Test Dataset	Pretained Model	Training Acc. (%)	Test Acc. (%)
Exp 0	CIFAR-10	50,000	10,000	Efficient-B0	94.95	92.86
Exp 1	CIFAR-10	50,000	10,000	ResNet101	88.83	86.97
Exp 2	F-MNIST	60000	10000	Efficient-B0	94.94	94.17
Exp 3	F-MNIST	60,000	10,000	ResNet50	93.11	91.12
Exp 4	MNIST	60,000	10,000	ResNet18	99.36	99.16
Exp 5	MNIST	60,000	10,000	ResNet34	99.29	98.84
Exp 6	SVHN	73,257	26,032	GoogLeNet	95.51	95.07
Exp 7	SVHN	73,257	26,032	ResNet152	94.63	94.10

6 Experimental Evaluation

This section addresses the research questions outlined in Section 5. First, we assess the effectiveness of the verdict weights proposed in eq. (2)-(4). Then, we evaluate the efficacy of DeepAbstraction++ algorithm by contrasting its performance with the SOTA algorithms. Third, we explore the performance stability of DeepAbstraction++ by examining the impacts of tuning the parameters γ and β. Finally, we will determine the best combination strategy.

6.1 Weights Effectiveness

We perform eight experiments as detailed in Table 2 where the combination strategy is the mean, and γ and β are 1 and 3, respectively. In our initial experiment, we aim to confirm the necessity of assigning greater importance to the weight of a rejection verdict compared to other verdicts. Specifically, we study how a single rejection verdict can influence the final combined verdict compared to the other five verdicts.

Our findings suggest that when only one monitor issues a rejection verdict, this results in a final combined verdict of rejection in 30% of misclassified instances, as indicated in Exp.0 of Fig. 4. The last ratio is significantly greater in Exp.1, 3, and 7, standing at 47%, 48%, and 48%, respectively. However, we found that in scenarios where the neural network exhibits high levels of accuracy, a single *rejection* verdict is insufficient to refuse the network prediction, as evidenced by the results of Exp.4 and 5 in Fig. 4. When the number of rejection verdicts increases to three or six, we can further confirm this finding. For example, in Exp.0, we noticed that of the instances resulting in a final combined verdict of true rejection, 39% had three rejection verdicts, and 73% had six rejection verdicts. We consistently observe this trend across all the conducted experiments. It strongly underlines the pivotal role that rejection verdicts play in determining the final decision over other verdicts.

In the second experiment, we contrast the impact of the rejection verdict compared to the other two types: uncertainty and acceptance. The aim is to investigate the influence each verdict type has on the true rejection of the final decision. More specifically, our comparison involves only those instances where the verdicts are consistent across all six monitors. Then, we compute the number of instances in which the six rejection verdicts led to the true rejection of the final verdict. After that, we compare this with

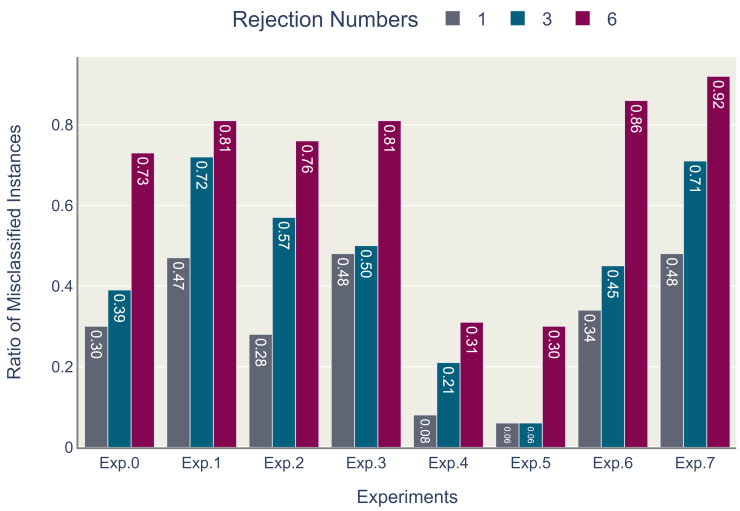


Fig. 4: The impact of the number of rejection verdicts on the final combined verdict.

the total number of six rejection-verdict instances to find the ratio. This procedure is repeated with instances of six uncertainty verdicts and six acceptance verdicts.

Our findings, as depicted in Fig. 5, highlight the considerable influence of the rejection verdicts on the final combined verdict. In six out of eight experiments, this verdict type significantly outperformed the others, with the highest contribution ratio reaching 91% and a median ratio of 78%. On the contrary, instances of uncertainty verdicts exhibit a moderate influence, with a maximum contribution ratio of 42% towards the true rejection of the final verdict. Instances of acceptance verdicts, however, demonstrate minimal impact on rejecting the final verdict.

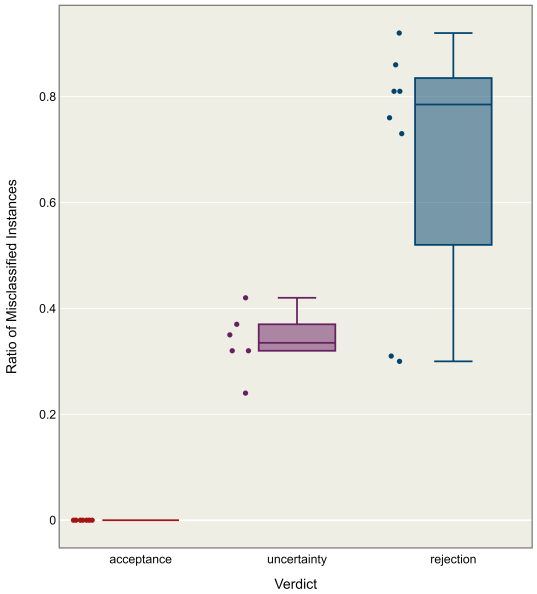


Fig. 5: The Impact of various verdict types on the final combined verdict.

RQ 1 Answer :

Our study reveals that the observed impact of the verdicts on the final verdict aligns with the proposed weights of the verdicts in the eq.(2)-(4).

6.2 Algorithms Effectiveness

Table 3 presents a comparative study on the effectiveness of the DeepGini, DeepAbstraction, and DeepAbstraction++ algorithms, evaluated using the WFDR metric across eight experiments. In each experiment, the combination strategy is the mean, and γ and β are 1 and 3. Table 3 reveals that DeepAbstraction++ consistently outperforms both DeepGini and DeepAbstraction in all experiments, as shown by the positive deltas. The improvements offered by DeepAbstraction++ over DeepAbstraction range from a minimum of +2.38% (in Exp.4) to a maximum of +7.71% (in Exp.6). This demonstrates that the additional optimizations in DeepAbstraction algorithm are effective.

Table 3: Effectiveness of DeepAbstraction++ and other algorithms (WFDR).

Experiment	DeepGini (%)	DeepAbstraction (%)	DeepAbstraction++ (%)	Δ
Exp.0	45.26	58.75	64.26	↑+5.51
Exp.1	48.62	56.39	59.90	↑+3.51
Exp.2	44.86	59.13	63.79	↑+4.66
Exp.3	46.53	53.19	59.21	↑+6.02
Exp.4	51.08	65.38	67.76	↑+2.38
Exp.5	45.54	60.20	62.79	↑+2.59
Exp.6	45.00	67.59	75.30	↑+7.71
Exp.7	46.55	63.79	69.70	↑+5.91

RQ 2 Answer :

DeepAbstraction++ demonstrates considerably higher effectiveness than other algorithms. Therefore, the new additions to the framework greatly enhances the performance.

6.3 Performance Stability

As demonstrated in Fig. 6, DeepAbstraction++ model exhibits remarkable stability in performance. Regardless of the β value, which ranges from 1 to 5000, the performance remains constant for each experiment. This consistent performance across a broad spectrum of β values indicates a high level of stability in DeepAbstraction++ performance. Similarly, when the parameter β is held constant at a value such as 1, and γ varies within a range from 1 to 5000, we consistently observe the stable performance of DeepAbstraction++ model across all γ values for every experiment. For the sake of brevity, the corresponding graph is omitted as it is highly similar to Fig. 6.

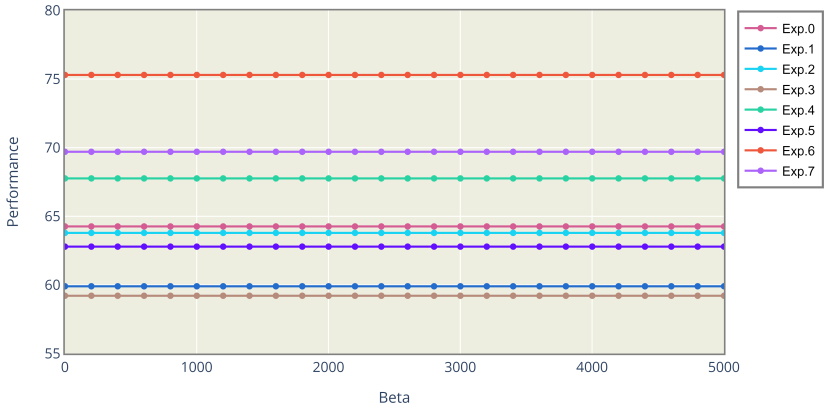


Fig. 6: The Impact of β on the performance stability when $\gamma = 1$.

RQ 3 Answer :

DeepAbstraction++ consistently maintains the stability, unaffected by the values of γ and β, indicating that γ and β do not impact the performance.

6.4 Combination Strategy Selection

Table 4 compares the effectiveness of different strategies where γ and β are 1 and 3. The evaluation is based on the WFDR percentage, incorporating four strategies: mean, product, mode, and max. The mean strategy generally outperforms the other strategies in all the experiments, with the highest mean value seen in Exp.6 at 75.30%. However, there are exceptions to this pattern. For instance, Exp.4 presents a noteworthy difference between the mean (67.76%) and the max (57.04%). Moreover, in Exp.5, the mean and product values are identical at 62.79%. Other combination strategies tend to perform badly because they need to include all monitors in their final combined verdict. For instance, the max strategy only chooses the maximum verdict over the six verdicts. Also, the mode strategy is biased towards the majority verdicts rather than incorporating all.Additionally, the product strategy prioritizes the rejection verdict above other verdicts when determining the final combined verdict. However, product strategy works better when all verdicts are rejection according to Fig. 4. On the other hand, the mean strategy manages to incorporate all monitors when determining the final verdict.

Table 4: Comparative analysis of strategies based on WFDR (%).

Experiment	mean (%)	Product (%)	mode (%)	max (%)
Exp.0	**64.26**	58.76	59.17	60.30
Exp.1	**59.90**	59.43	55.45	59.95
Exp.2	**63.79**	63.45	58.09	60.68
Exp.3	**59.21**	58.42	52.51	56.70
Exp.4	**67.76**	65.38	64.19	57.04
Exp.5	**62.79**	**62.79**	61.07	52.44
Exp.6	**75.30**	73.41	67.19	65.15
Exp.7	**69.70**	68.98	62.02	66.41

RQ 4 Answer :

The results suggest a prevailing superiority of the mean strategy in merging the verdicts from multiple monitors over other strategies.

7 Conclusion and Future Work

This paper tackles a key challenge in DeepAbstraction framework: the tau selection, a factor that significantly shapes the box size and impacts the model stability and performance. To address this problem, we introduced an innovative solution, the combined parameterized boxes. This approach relies on the collective verdicts from monitors with different tau values to assess network predictions. These verdicts are assigned careful weights to prevent any one type of verdict from dominating and to maintain a balanced decision-making process. Various combination strategies, including mean, max, product, and mode, are proposed to unify the weighted verdicts of different monitors into a final verdict. Our proposed approach holds significant potential to enhance the performance of DeepAbstraction framework.

A potential future direction could involve formulating a tau selection procedure that is dynamic and influenced by the specific characteristics of each dataset, thus tailoring the process to the unique data distribution. Similarly, we foresee advancements in a more sophisticated verdict weighting scheme that could adjust adaptively the verdict weights contingent on the local data distribution. Lastly, DeepAbstraction++ lays the groundwork for future studies to fine-tune and further expand these techniques for an even more efficient and trustworthy monitoring system.

References

1. Al-Qadasi, H., Falcone, Y., Bensalem, S.: Difficulty and severity-oriented metrics for test prioritization in deep learning systems. In: 2023 IEEE International Conference On Artificial Intelligence Testing (AITest). IEEE (2023), in press
2. Al-Qadasi, H., Wu, C., Falcone, Y., Bensalem, S.: DeepAbstraction: 2-level prioritization for unlabeled test inputs in deep neural networks. In: 2022 IEEE International Conference On Artificial Intelligence Testing (AITest). pp. 64–71. IEEE (2022)
3. Byun, T., Sharma, V., Vijayakumar, A., Rayadurgam, S., Cofer, D.D.: Input prioritization for testing neural networks. CoRR **abs/1901.03768** (2019)
4. Chen, Y., Cheng, C.H., Yan, J., Yan, R.: Monitoring object detection abnormalities via data-label and post-algorithm abstractions. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 6688–6693. IEEE (2021)
5. Dang, X., Li, Y., Papadakis, M., Klein, J., Bissyandé, T.F., Traon, Y.L.: Graphprior: Mutation-based test input prioritization for graph neural networks. ACM Transactions on Software Engineering and Methodology (2023)
6. Deng, L.: The mnist database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Processing Magazine **29**(6), 141–142 (2012)
7. Feng, Y., Shi, Q., Gao, X., Wan, J., Fang, C., Chen, Z.: Deepgini: prioritizing massive tests to enhance the robustness of deep neural networks. In: Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis. pp. 177–188 (2020)

8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
9. Henzinger, T.A., Lukina, A., Schilling, C.: Outside the box: Abstraction-based monitoring of neural networks. In: ECAI. Frontiers in Artificial Intelligence and Applications, vol. 325, pp. 2433–2440. IOS Press (2020)
10. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. University of Toronto (2009)
11. Kueffner, K., Lukina, A., Schilling, C., Henzinger, T.: Into the unknown: active monitoring of neural networks (extended version). International Journal on Software Tools for Technology Transfer (2023)
12. Li, Y., Li, M., Lai, Q., Liu, Y., Xu, Q.: Testrank: Bringing order into unlabeled test instances for deep learning tasks. Advances in Neural Information Processing Systems **34**, 20874–20886 (2021)
13. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. NIPS (2011)
14. Quinlan, J.R.: Induction of decision trees. Machine learning **1**(1), 81–106 (1986)
15. Shmelova, T., Yatsko, M., Sierostanov, I., Kolotusha, V.: Artificial intelligence methods and applications in aviation. In: Handbook of Research on AI Methods and Applications in Computer Engineering, pp. 108–140. IGI Global (2023)
16. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. CoRR **abs/1409.4842** (2014)
17. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning (ICML) (2019)
18. Wang, Z., You, H., Chen, J., Zhang, Y., Dong, X., Zhang, W.: Prioritizing test inputs for deep neural networks via mutation analysis. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). pp. 397–409. IEEE (2021)
19. Wu, C., Falcone, Y., Bensalem, S.: Customizable reference runtime monitoring of neural networks using resolution boxes. CoRR **abs/2104.14435** (2021)
20. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. CoRR **abs/1708.07747** (2017)
21. Yang, K., Tang, X., Qiu, S., Jin, S., Wei, Z., Wang, H.: Towards robust decision-making for autonomous driving on highway. IEEE Transactions on Vehicular Technology (2023)
22. Ziegler, C.: A google self-driving car caused a crash for the first time. `https://www.theverge.com/2016/2/29/11134344/google-self-driving-car-crash-report` (2016), accessed: 2023-07-27