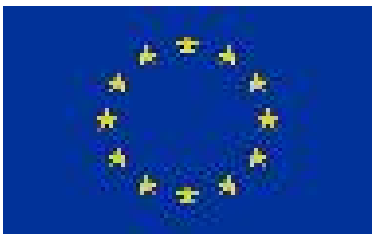
When in Doubt Throw It Out: Building on Confident Learning for Vulnerability Detection

Authors:

Yuanjun Gong, Renmin University of China (China)

Fabio Massacci, University of Trento (Italy),Vrije Universiteit Amsterdam (The Netherlands)



This work has been partly supported by the European Union (EU) under Horizon Europe grant n . 101120393 (Sec4AI4Sec), by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) under grant n. KIC1.VE01.20.004 (HEWSTI), and by the Italian Ministry of University and Research (MUR), under the P.N.R.R. – NextGenerationEU grant n. PE00000014 (SERICS). This paper reflects only the author's view and the funders are not responsible for any use that may be made of the information contained therein.

This is the author accepted version of Gong, Y.J. and Massacci, F. When in Doubt Throw It Out: Building on Confident Learning for Vulnerability Detection. *Proceedings of the International Conference on Software Engineering - New and Emerging Results (ICSE-NIER 2025)*. IEEE Press.

Cybersecurity for AI-Augmented Systems (Sec4AI4Sec) . As artificial intelligence (AI) becomes omnipresent, even integrated within secure software development, the safety of digital infrastructures requires new technologies and new methodologies, as highlighted in the EU Strategic Plan 2021-2024. To achieve this goal, the EU-funded Sec4AI4Sec project will develop advanced security-by-design testing and assurance techniques tailored for AI-augmented systems. These systems can democratise security expertise, enabling intelligent, automated secure coding and testing while simultaneously lowering development costs and improving software quality. However, they also introduce unique security challenges, particularly concerning fairness and explainability. Sec4AI4Sec is at the forefront of the move to tackle these challenges with a comprehensive approach, embodying the vision of better security for AI and better AI for security. More information at **https://sec4ai4sec.eu**.

Hybrid Explainable Workflows for Security and Threat Intelligence (HEWSTI) In research into threats to safety and security, people and AI collaborate to obtain actionable intelligence. Their sources and methods often have significant uncertainties and biases. Experts are aware of these limitations, but lack the formal means to handle these uncertainties in their daily work. This project will invent a 'metadata of uncertainty' for threat intelligence (in both machine-readable and also human-interpretable forms) and validate it empirically. Intelligence agencies will then be able to explicitly consider the trade-off between the accuracy, proportionality, privacy, and cost-effectiveness of investigations. This will contribute towards the responsible use of AI to create a safer, more secure society.

In searCh Of eVidence of stEalth cybeR Threats (COVERT) AT 3 aims to analyze emerging attack methodologies and develop advanced methods for detecting attacks and identifying guidelines for designing IT systems that ensure reduced vulnerability to new attack categories. The detailed objectives can be divided into four macro categories: (i) Development of advanced tools for analyzing malware and software aimed at identifying vulnerabilities that could be exploited by malware; (ii) Development of tools for analyzing network traffic to identify communications related to ongoing attacks; (iii) Development of machine learning systems that are robust to attacks and through which it is possible to extract knowledge aimed at creating more advanced tools for timely analysis and early identification of attacks; (iv) Analysis of the "human factors" involved in an attack with the development of tools for analyzing and correlating information from OSINT (open sources intelligence) and for the defense and prevention of attacks based on social engineering techniques.

Yuanjun Gong (BSc 2018) is PhD candidate at the Renmin University of China, Beijing, China. Her research interests include static analysis, software security and machine learning. Contact her at *gongyuanjun@ruc.edu.cn*.

Fabio Massacci (Phd 1997) is a professor at the University of Trento, Italy, and Vrije Universiteit Amsterdam, Fabio Massacci is a professor at the University of Trento, Trento, Italy, and Vrije Universiteit Amsterdam, 1081 HV Amsterdam, The Netherlands. His research interests include empirical methods for the cybersecurity of sociotechnical systems. For his work on security and trust in sociotechnical systems, he received the Ten Year Most Influential Paper Award at the 2015 IEEE International Requirements Engineering Conference. He is named co-author of CVSS v4. He leads the Horizon Europe Sec4AI4Sec project and the Dutch National Project HEWSTI. He is past chair of the Security and Defense Group of the Society for Risk Analysis, and IEEE CertifAIEd Lead Assessor. Contact him at *fabio.massacci@ieee.org*.

How to cite this paper:

- Gong, Y.J. and Massacci, F. When in Doubt Throw It Out: Building on Confident Learning for Vulnerability Detection. *Proceedings of the International Conference on Software Engineering - New and Emerging Results (ICSE-NIER 2025)*. IEEE Press.

When in Doubt Throw It Out: Building on Confident Learning for Vulnerability Detection

Yuanjun Gong
Renmin University of China, Beijing, CN
gongyuanjun@ruc.edu.cn

Fabio Massacci
University of Trento, IT and Vrije Universiteit Amsterdam, NL
fabio.massacci@ieee.org

***Abstract*—[Context:] Confident learning's intuition is that a good model can be used to identify mislabelled data. By swapping mislabeled samples that are not confidently predicted, the performance of model can be further improved. [Problem:] Unfortunately, vulnerability detectors are generally under-performing models and confidence learning would conclude that the bulk of the dataset is mislabelled. [New Idea:] We extend confidence learning by identifying a type of training samples that appear in presence of under-performing models: *confusing samples*. [Emerging Result:] We analyze the formal constraints for confusing samples and perform preliminary experiments that show that the model's performance is effectively improved after *deleting* confusing samples entirely from the training set.**

***Index Terms*—Vulnerability Detection, Confident Learning, Confusing Samples**

I. Introduction

Confident learning [1] (CL for short) has been proposed as a method for identifying the latent true labels of dataset samples, by bootstrapping the performance of a good model on the dataset itself. The key idea is that the label confidently predicted by a reliable model could be right and the given label could be wrong. By changing labels during training we could improve the performance of the model. CL assumes that the model is reasonably good to start with.

Unfortunately, vulnerability detection models don't perform well. When Wen et al. [2] compared seven vulnerability detectors on three dataset: FFMPeg+Qemu [3], Reveal [4], and Fan et al. [5], the precision of the models was rarely above 50% (Table I). Even LineVul, which is one of the most accurate, recent detector doesn't systematically performs well [6]. Datasets are noisy as well and up of two third of vulnerability labels are wrong in many so called 'real-world' datasets [7]. If dataset are noisy at this level, then even a good detector such as LineVul [8] can hardly be a good detector since it is good at predicting wrong values. If models and datasets are so bad, how would confidence learning fare?

Only Nie et al. [13] have used confident learning on CWE-specific vulnerability detection datasets and shown some improvement but this result is not confirmed by independent experiments. When we applied by-the-book CL strategies to under-performing models, between 28%-38% of the training set would be identified as *Mislabeled* samples (Table III). This would call into question either the nature of datasets argued to provide a reliable ground truth based on developers' actions or the very idea behind confidence learning.

This work was done when the first author was a visiting Ph.D. student at The University of Trento.

TABLE I
VULNERABILITY DETECTION PERFORMANCE OF VARIOUS ML TOOLS [2]

Dataset	FFMPeg+Qemu		Reveal		Fan et al.	
Tools	P	R	P	R	P	R
VulDeePecker [9]	46.05	32.55	21.13	13.10	38.44	12.75
Russell et al. [10]	54.76	40.72	16.21	52.68	14.86	26.97
SySeVR [11]	46.06	58.81	40.07	24.94	30.91	14.08
Devign [3]	52.50	64.67	31.55	36.65	30.61	15.96
Reveal [12]	55.50	70.70	31.55	61.14	17.22	34.04
IVDetect [4]	52.37	57.55	-	-	-	-
AMPLE [2]	55.64	83.99	51.06	46.15	29.98	34.58

To debug this unsatisfactory conflict, we studied the distribution of samples' prediction, the confident thresholds, and all possible formal combinations of out-of-sample predicted probabilities of a (poor) model's training set and we have identified an alternative approach.

Intuitively, if a model is poor and labels aren't good either, there can be samples where we can't find a clear indication on whether the model's prediction is wrong or the sample is mislabeled. So, better to throw those samples out.

- We identify two types of training samples where the label with the highest prediction probability is not a confident prediction. We name them *Confusing* samples.
- We identified the necessary and sufficient conditions for *Confusing* samples to exist.
- Our preliminary experiments shows that by deleting *Confusing* samples from the training set, the model's performance on the testing set is effectively improved. In one case, even better than relabelling or pruning mislabeled samples as suggested by the CL algorithm.

These samples only materialize when the model is under-performing (as vulnerability detection models are), and therefore works on CL for images have rarely witnessed them.

II. Background on Confident Learning

CL initially applies K-fold cross validation on the training set to obtain the out-of-sample prediction $pred(k, \ell)$ for each sample k on class ℓ. The function $label(k, \ell) = 1$ if the given label of sample k is ℓ and zero otherwise. We denote a label that is different from the label ℓ as ℓ^*. Then, CL calculates

TABLE II
THE POSSIBLE SITUATION OF THE PREDICTIONS

ID	Confidence in ℓ	Confidence in ℓ^*	better prediction
1	$pred(k,\ell) > t_\ell$	$pred(k,\ell^*) > t_{\ell^*}$	$pred(k,\ell) > pred(k,\ell^*)$
2	$pred(k,\ell) > t_\ell$	$pred(k,\ell^*) > t_{\ell^*}$	$pred(k,\ell) < pred(k,\ell^*)$
3	$pred(k,\ell) > t_\ell$	$pred(k,\ell^*) < t_{\ell^*}$	$pred(k,\ell) > pred(k,\ell^*)$
4	$pred(k,\ell) > t_\ell$	$pred(k,\ell^*) < t_{\ell^*}$	$pred(k,\ell) < pred(k,\ell^*)$
5	$pred(k,\ell) < t_\ell$	$pred(k,\ell^*) > t_{\ell^*}$	$pred(k,\ell) > pred(k,\ell^*)$
6	$pred(k,\ell) < t_\ell$	$pred(k,\ell^*) > t_{\ell^*}$	$pred(k,\ell) < pred(k,\ell^*)$
7	$pred(k,\ell) < t_\ell$	$pred(k,\ell^*) < t_{\ell^*}$	$pred(k,\ell) > pred(k,\ell^*)$
8	$pred(k,\ell) < t_\ell$	$pred(k,\ell^*) < t_{\ell^*}$	$pred(k,\ell) < pred(k,\ell^*)$

the confident threshold t_ℓ for each class as the average of the model's predicted probability for samples with given label ℓ.

$$t_\ell = \frac{1}{\sum_k label(k,\ell)} \sum\nolimits_k pred(k,\ell) \cdot label(k,\ell) \quad (1)$$

CL assumes that a predicted label is valid only if the prediction is confident, i.e. the prediction on label ℓ should be greater than the confident threshold t_ℓ. The latent true label of sample k is assumed by CL to be

$$trueLabel(k) = \text{argmax}_\ell\{pred(k,\ell) \mid pred(k,\ell) > t_\ell\} \quad (2)$$

CL pruning strategies relabel the database according to the confident joint or the joint distribution between the given labels and the true labels. For example, given $label(k,\ell) = 1$ and $label(k,\ell^*) = 0$, the Off-Diagonals CL Strategy decides whether the sample k is *Mislabeled* following Eq. 3 below

$$mislabel(k,\ell) = \begin{cases} 1 & \text{if } \exists \ell^* \neq \ell : label(k,\ell) = 1 \wedge \\ & pred(k,\ell^*) > t_{\ell^*} \wedge \\ & pred(k,\ell^*) > pred(k,\ell) \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

III. IDENTIFYING *Confusing* SAMPLES

To simplify the presentation, we only focus on the vulnerability detection scenario with a binary classifier. For each sample k, the prediction for its given class $pred(k,\ell)$ can be either greater or less than the confident threshold t_ℓ. Likewise, the predicted probability of the wrong label $pred(k,\ell^*)$ can be greater or less than t_{ℓ^*}, and the predict result is one of the two. In Table II we list all the possible situations a sample can belong to (we omit the case of exact equivalence since values come from floating-point operations on tensors).

- Samples in situation #1 and #3 are predicted with high confidence, therefore they are neither *Mislabeled* samples nor *Confusing* samples.
- Samples #2 and #6 are *Mislabeled* according to CL, because the predicted probability $pred(k,\ell^*)$ for the wrong label ℓ^* is greater than t_{ℓ^*} (a better than average, i.e. confident prediction) and greater than $pred(k,\ell)$. So, we are confident of the prediction of the sample, which just has a 'wrongly given' label.
- Sample #7 is not considered as *Mislabeled* sample by CL because it has low confidence predictions.
- Sample #5 would be attributed to ℓ^* by CL, because $pred(k,\ell^*)$ is higher than the threshold t_{ℓ^*}, even if ℓ is a better prediction. This is a type of *Confusing* samples. We denote it as GIVENWINSCONFIDENTALTERNAT sample.
- Samples #4 and #8 are not *Mislabeled* samples according to CL, since the prediction on ℓ^* is not confident, but wrong prediction si still wrong, so we categorize them as *Confusing* ssamples of type ALTERNATNOTCONFIDENTYETWINS.

A. Capturing GIVENWINSCONFIDENTALTERNAT *samples*

A *Confusing* sample k is GIVENWINSCONFIDENTALTERNAT if

$$t_\ell > pred(k,\ell) > pred(k,\ell^*) > t_{\ell^*} \quad (4)$$

Necessary Conditions. Assume $label(k,\ell) = 1$. Let N_ℓ be the number of samples such that $label(k,\ell) = 1$ for all samples $k = 1 \ldots N$. We use $Pr_T(\ell^*)$, to denote the average predicted probability of the True Negative samples towards label ℓ. Likewise, $Pr_T(\ell)$, $Pr_F(\ell)$, $Pr_F(\ell^*)$ represent for the average predicted probability of the True Positive samples, the False Negative samples, and the False Positive samples, accordingly.

Since $pred(k,\ell) + pred(k,\ell^*) = 1$, we need to make sure $t_{\ell^*} < \frac{1}{2}$ and $t_\ell > \frac{1}{2}$ to get *Confusing* samples.

Denoting the true positive rate (recall) of the model for label ℓ as TPR_ℓ, we have $\text{TPR}_\ell \cdot N_\ell$ samples having a correct prediction with average predicted probability $Pr_T(\ell) > \frac{1}{2}$. Similarly, $(1 - \text{TPR}_\ell)N_\ell$ samples have a wrong prediction with average predicted probability as $Pr_F(\ell) < \frac{1}{2}$.

The confident threshold t_ℓ can be calculated as

$$t_\ell = \text{TPR}_\ell \cdot Pr_T(\ell) + (1 - \text{TPR}_\ell) \cdot Pr_F(\ell)\} \quad (5)$$

Since t_ℓ should be grater than $\frac{1}{2}$, we get

$$\text{TPR}_\ell > \frac{\frac{1}{2} - Pr_F(\ell)}{Pr_T(\ell) - Pr_F(\ell)} \quad (6)$$

Similarly, we get the constraint for False Positive Rate(FPR_ℓ) from $t_{\ell^*} < \frac{1}{2}$ as

$$\text{FPR}_\ell > \frac{Pr_T(\ell^*) - \frac{1}{2}}{Pr_T(\ell^*) - Pr_F(\ell^*)} \quad (7)$$

Eq. 6 and Eq. 7 are the necessary conditions for GIVENWINSCONFIDENTALTERNAT samples.

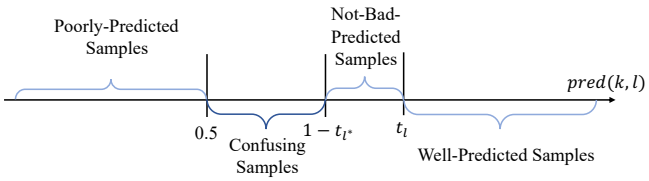
When a model works well on positive reports but less well on negative reports, *Confusing* samples can be in the dataset.

Sufficient Conditions. Given $t_\ell > \frac{1}{2}$ and $t_{\ell^*} < \frac{1}{2}$, for GIVENWINSCONFIDENTALTERNAT samples not to exist we need

$$\begin{aligned} \forall k, \neg(t_\ell > pred(k,\ell) \vee pred(k,\ell) > pred(k,\ell^*) \\ \vee pred(k,\ell^*) > t_{\ell^*}) \end{aligned} \quad (8)$$

By reasoning by cases there will be no GIVENWINSCONFIDENTALTERNAT samples in the dataset if only the following types of samples exist:

- Well-Predicted. $pred(k,\ell) > t_\ell > \frac{1}{2}$
- Poorly-Predicted. $pred(k,\ell) \leq \frac{1}{2}$
- Not-Bad-Predicted. $\frac{1}{2} \leq 1 - t_{\ell^*} < pred(k,\ell) < t_\ell$



For example the vulnerable function *qemu_chr_open_socket* in the QEMU datatset has been scored 0.523 by our classifier,and the confidence threshold for vulnerable samples is 0.682 and that for not vulnerable samples is 0.476. So this function is correctly predicted but not confidently, instead, the alternative prediction is confident.

Fig. 1. Sufficient Condition for GIVENWINSCONFIDENTALTERNAT

Figure 1 shows the relationship between the predicted probability of the sample and its prediction category. The sufficient constraint for GIVENWINSCONFIDENTALTERNAT samples is $\exists k, \frac{1}{2} < pred(k,\ell) < 1 - t_{\ell^*}$.

B. Capturing ALTERNATNOTCONFIDENTYETWINS

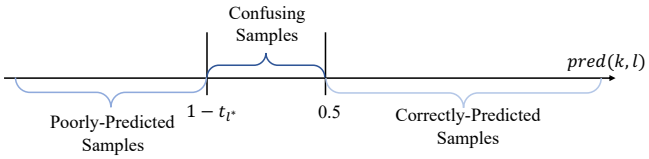
The formal definition of ALTERNATNOTCONFIDENTYETWINS is the following one

$$t_{\ell^*} > pred(k,\ell^*) > pred(k,\ell) \tag{9}$$

With a reasoning process similar to the other type we conclude that the necessary condition for ALTERNATNOTCONFIDENTYETWINS samples is the following one:

$$\text{FPR}_\ell < \frac{Pr_T(\ell^*) - \frac{1}{2}}{Pr_T(\ell^*) - P_F(\ell^*)} \tag{10}$$

The sufficient conditions of the ALTERNATNOTCONFIDENTYETWINS samples is $\exists k, \frac{1}{2} > pred(k,\ell) > 1 - t_{\ell^*}$. The positions of samples are shown in Figure 2. The condition that $t_{\ell^*} < 1/2$, used in the necessary condition for the first type of samples, does not hold here because we have $pred(k,\ell^*) > 1/2$ and $pred(k,\ell^*) < t_{\ell^*}$.



For example the non-vulnerable function *sprintf_len* in the QEMU dataset has been scored 0.439 by our classifier, and the confidence threshold for vulnerable samples is 0.682. Therefore, this function is wrongly predicted and none of the predictions is confident.

Fig. 2. Sufficient Condition for ALTERNATNOTCONFIDENTYETWINS

IV. EXPERIMENTS AND RESULTS

In this section, we use a pre-trained sentenceBERT [14] encoder along with two activated linear layers to construct a vulnerability detector that is comparable to the state-of-the-art in Table I, but it is still not satisfactory (Precision = 63%, Recall = 47%). We use 4,000 positive samples and 4,000 negative samples from Devign's Qemu dataset [3] for the training set, as well as 1,000 positive samples and 1,000 negative samples to form the testing set. Training with entire dataset yields a precision of 63.35% and a recall of 47.7%.

TABLE III
THE CONFIDENT LEARNING RESULT OF THE TRAINING SET

	$\ell = 0$	$\ell = 1$
Total Samples	4,000 (50.00%)	4,000 (50.00%)
CL Method *Misclassified*	2,265 (28.31%)	795 (9.93%)
CL Method *Off-Diagonals*	1,370 (17.14%)	795 (9.93%)
CL Method *Prune-by-Class*	1,708 (21.35%)	795 (9.93%)
CL Method *Prune-by-Noise-Rate*	1,708 (21.35%)	795 (9.93%)
CL Method *Prune-by-Both*	1,704 (21.30%)	795 (9.93%)

TABLE IV
THE PERFORMANCE AFTER PRUNING OR INVERTING MISLABELED SAMPLES AND PRUNING CONFUSING SAMPLES

Dataset	Samples Influenced	Precision	Recall
Original Dataset	0	63.35	47.70
Remove *Mislabeled*	2,165	59.07	76.50
Invert *Mislabeled*	2,165	60.07	69.50
Rem. **ALTERNATNOTCONFIDENTYETWINS**	900	59.62	75.30
Rem. **GIVENWINSCONFIDENTALTERNAT**	94	59.25	77.20

The confidence thresholds are $t_0 = 0.476$ and $t_1 = 0.682$. While $pred(k,\ell) + pred(k,\ell) = 1$, the two thresholds do not necessarily sum to one. The values are comparable to Table I: we have a worse recall but a better precision.

CL proposes five rules to process *Mislabeled* samples, namely *Misclassified*, *Off-Diagonals*, *Prune-by-Class*, *Prune-by-Noise-Rate*, and *Prune-by-Both* (See [1] for further details).

Intuitively, the only thing that changes are the fraction of *Mislabeled* samples according to Eq. 3 that are changed. Table III shows the number of mislabeled training samples identified by the various pruning methods. Essentially more than $\frac{1}{3}$ of the dataset are wrongly labeled. This would call into question the very nature of QEMU as a realiable dataset allegedly based on actual, real world commits.

We analyze the out-of-sample predicted probabilities of the training samples and identify the two types of confusing samples with the aforementioned definition. A total of 900 **ALTERNATNOTCONFIDENTYETWINS** samples (10%) and a total of 94 **GIVENWINSCONFIDENTALTERNAT** samples (1%) are recognized. These amounts look an acceptable compromise. After removing the confusing samples from the training set, we retrain the model and test the model's performance. We also test the model after pruning the *Mislabeled* samples identified by CL algorithm and after inverting the *Mislabeled* samples labels according to the *Off-Diagonals* strategy.

The resulting precision and recall are shown in Table IV. Removing confusing samples from the training set improves the performance. In particular, removing GIVENWINSCONFIDENTALTERNAT samples has the least influence to the training set, while getting better result on recall with no impact on precision (60% to 59%).

To explore the composition of the training dataset (when a sample is classified as a mislabeled sample and when it is classified as a confusing sample), we further analysed the training dataset in Figure 3. Figure. 3(a), respectively Figure. 3(b), shows the model's predicted probability on the given label 0, respectively 1. Since the model's confidence

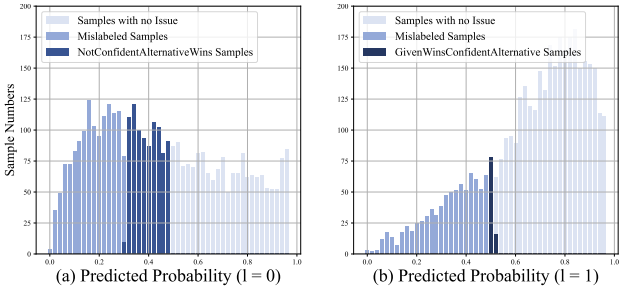


Fig. 3. The Distribution of Mislabeled and Confusing Samples

thresholds are $t_0 < 0.5$ and $t_1 > 0.5$, in Figure. 3(a) where $\ell = 0$, only ALTERNATNOTCONFIDENTYETWINS samples exist. In Figure. 3(b) where $\ell = 1$, only GIVENWINSCONFIDENTALTERNAT samples exist. Confusing samples are present when the model does not have high confidence in its predictions for both categories. In comparison to *Mislabeled* samples, *Confusing* samples make up a small fraction of the total samples, and processing *Confusing* samples will be a less invasive modification of the original dataset than changing the alleged *Mislabeled* samples.

We further analyzed the distribution of out-of-sample predicted probabilities for the training set before and after pruning. As shown in Figure 4(a), the improvement to the model by pruning the GIVENWINSCONFIDENTALTERNAT samples results in increasing the proportion of correct predictions for the negative samples. The improvement to the model by pruning the ALTERNATNOTCONFIDENTYETWINS samples results in increasing the proportion of correct predictions for the positive samples, as shown in Figure 4(b).

V. DISCUSSION AND FUTURE PLANS

In this paper, we discuss an alternative approach to confident learning for vulnerability detection. If the model is poor, mislabeled samples identified by confident learning may become too many for relabeling to be a sound approach. Our proposal is to identify and delete *Confusing* samples. Our preliminary experiments show that a model can gain performance after pruning those samples from the training. While we have given necessary and sufficient conditions for *Confusing* samples to exist we have not proved that their removal always improves the model. Such evidence requires more experimental analysis, such as combining both mislabeled and confusing samples as well as analyzing differences between them.

Real-World Vulnerability Detection. We plan to apply the *Confusing* sample pruning strategy to more datasets with real-world vulnerabilities and a variety of vulnerability detection models, to test whether the phenomenon reported here is widespread, and whether the performance of state-of-the-art detectors is always improved. Among the various models, LineVul [8] has emerged as one the most promising and its evaluation could lead to useful insights.

Unbalanced Training Set. We used a balanced training set with equal positive and negative samples. Real-world vulnerability datasets have unbalanced samples, e.g., Reveal [4],

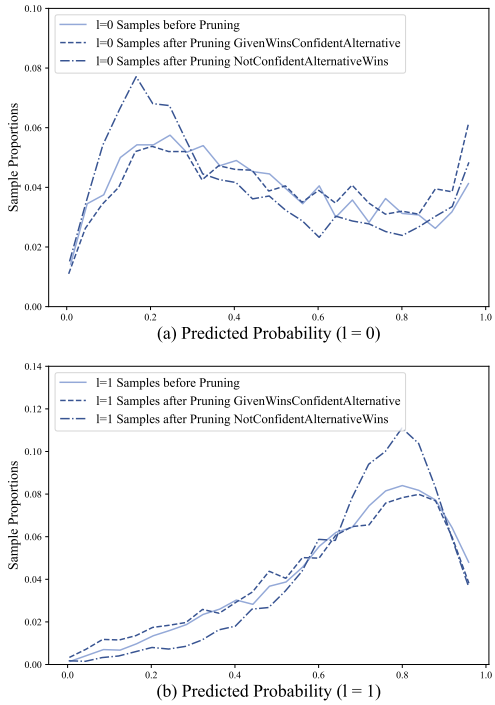


Fig. 4. Probabilities before and after Pruning Confusing Samples

Fan et al. [5] have far more non-vulnerable samples than vulnerable ones. Further analysis would be focused the impact of unbalanced datasets on the fraction of *Confusing* samples, and the impact of their removal on the model's performance.

Multi-Class Classification. We analyzed necessary and sufficient constraints for the existence of *Confusing* samples for a binary-classification scenario. To design a complete methodological approach, we plan to generalize the analysis to multi-class classification such as CWE-specific classes with a not-vulnerable class.

VI. ARTIFACT AVAILABILITY STATEMENT

The replication package is available on Zenodo at https://doi.org/10.5281/zenodo.14641942.

VII. ACKNOWLEDGEMENTS

This work has been partly supported by the European Union (EU) under Horizon Europe grant n. 101120393 (Sec4AI4Sec), by the Italian Ministry of University and Research (MUR), under the P.N.R.R. – NextGenerationEU grant n. PE00000014 (SERICS), and by the Nederlandse Organisatie voor Wetenschappelijk Onderzoek (NWO) under grant n. KIC1.VE01.20.004 (HEWSTI).

VIII. CREDIT AUTHOR STATEMENT

Conceptualization YG, FM; Methodology YG, FM; Software YG; Validation YG, FM; Formal analysis FM, YG; Investigation YG; Data Curation YG; Writing - Original Draft YG; Writing - Review & Editing YG, FM; Visualization YG; Supervision FM; Project administration FM; Funding acquisition FM.

REFERENCES

[1] C. Northcutt, L. Jiang, and I. Chuang, "Confident learning: Estimating uncertainty in dataset labels," *Journal of Artificial Intelligence Research*, vol. 70, pp. 1373–1411, 2021.

[2] X.-C. Wen, Y. Chen, C. Gao, H. Zhang, J. M. Zhang, and Q. Liao, "Vulnerability detection with graph simplification and enhanced graph representation learning," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 2275–2286.

[3] Y. Zhou, S. Liu, J. Siow, X. Du, and Y. Liu, "Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks," *Advances in neural information processing systems*, vol. 32, 2019.

[4] Y. Li, S. Wang, and T. N. Nguyen, "Vulnerability detection with fine-grained interpretations," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 292–303.

[5] J. Fan, Y. Li, S. Wang, and T. N. Nguyen, "Ac/c++ code vulnerability dataset with code changes and cve summaries," in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020, pp. 508–512.

[6] E. Imgrund, T. Ganz, M. Härterich, L. Pirch, N. Risse, and K. Rieck, "Broken promises: Measuring confounding effects in learning-based vulnerability discovery," in *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, 2023, pp. 149–160.

[7] R. Croft, M. A. Babar, and M. M. Kholoosi, "Data quality for software vulnerability datasets," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, pp. 121–133.

[8] M. Fu and C. Tantithamthavorn, "Linevul: A transformer-based line-level vulnerability prediction," in *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022, pp. 608–620.

[9] Z. Li, D. Zou, S. Xu, X. Ou, H. Jin, S. Wang, Z. Deng, and Y. Zhong, "Vuldeepecker: A deep learning-based system for vulnerability detection," *arXiv preprint arXiv:1801.01681*, 2018.

[10] R. Russell, L. Kim, L. Hamilton, T. Lazovich, J. Harer, O. Ozdemir, P. Ellingwood, and M. McConley, "Automated vulnerability detection in source code using deep representation learning," in *2018 17th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, 2018, pp. 757–762.

[11] Z. Li, D. Zou, S. Xu, H. Jin, Y. Zhu, and Z. Chen, "Sysevr: A framework for using deep learning to detect software vulnerabilities," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, pp. 2244–2258, 2021.

[12] S. Chakraborty, R. Krishna, Y. Ding, and B. Ray, "Deep learning based vulnerability detection: Are we there yet?" *IEEE Transactions on Software Engineering*, vol. 48, no. 9, pp. 3280–3296, 2021.

[13] X. Nie, N. Li, K. Wang, S. Wang, X. Luo, and H. Wang, "Understanding and tackling label errors in deep learning-based vulnerability detection (experience paper)," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023, pp. 52–63.

[14] N. Reimers, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.