

On the Effect of Ruleset Tuning and Data Imbalance on Explainable Network Security Alert Classifications

Citation for published version (APA):

Teuwen, K. T. W., Baggen, S., Zambon, E., & Allodi, L. (2025). On the Effect of Ruleset Tuning and Data Imbalance on Explainable Network Security Alert Classifications: a Case-Study on DeepCASE. In *10th IEEE European Symposium on Security and Privacy Workshops, EuroS&PW 2025* (pp. 16-29). Article 11129503 Institute of Electrical and Electronics Engineers. <https://doi.org/10.1109/EuroSPW67616.2025.00009>

Document license:
CC BY

DOI:
[10.1109/EuroSPW67616.2025.00009](https://doi.org/10.1109/EuroSPW67616.2025.00009)

Document status and date:
Published: 01/09/2025

Document Version:
Accepted manuscript including changes made at the peer-review stage

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy

If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

On the Effect of Ruleset Tuning and Data Imbalance on Explainable Network Security Alert Classifications: a Case-Study on DeepCASE

Koen T. W. Teuwen, Sam Baggen, Emmanuele Zambon and Luca Allodi

Eindhoven University of Technology

Eindhoven, The Netherlands

{k.t.w.teuwen, s.a.m.baggen, e.zambon, l.allodi}@tue.nl

Abstract—Automation in Security Operations Centers (SOCs) plays a prominent role in alert classification and incident escalation. However, automated methods must be robust in the presence of imbalanced input data, which can negatively affect performance. Additionally, automated methods should make explainable decisions. In this work, we evaluate the effect of label imbalance on the classification of network intrusion alerts. As our use-case we employ DeepCASE, the state-of-the-art method for automated alert classification. We show that label imbalance impacts both classification performance and correctness of the classification explanations offered by DeepCASE. We conclude tuning the detection rules used in SOCs can significantly reduce imbalance and may benefit the performance and explainability offered by alert post-processing methods such as DeepCASE. Therefore, our findings suggest that traditional methods to improve the quality of input data can benefit automation.

Index Terms—Security Operations Center (SOC), Network Intrusion Detection System (NIDS), Network Security Alerts, Alert Reduction, Intrusion Detection Ruleset Tuning, Network Intrusion Detection Rules

1. Introduction

Security Operations Centers (SOCs) are important to an organization's security strategy. A SOC monitors network and/or host activity to detect malicious events [1]. SOCs employ a variety of tools to aid in monitoring efforts, including *Network Intrusion Detection Systems (NIDS)*, which have been known to generate an unmanageable volume of security events for suspicious activities [2]. *Suricata* [3] is a common signature-based NIDS that employs *rules* defining the malicious behavior to detect and raise *alerts*. SOC analysts use related events and contextual information to determine whether an incident has occurred [4]. A substantial part of the resulting workload can be considered unnecessary due to the high prevalence of False Positives (FPs, i.e., false alerts) [5].

Previous research improves detectors using traditional methods, such as studying effective rule design [6]. Tuning of IDS rulesets [1] is commonly applied in SOCs but can negatively affect IDS coverage [7]. A second stream of work reduces the workload required to process incoming alerts [8]–[10]. Similarly to the analyst process [4], these methods leverage contextual events to determine which alerts are related to incidents.

The trustworthiness of automated solutions in SOCs depends on their robustness in light of data characteristics

such as concept drift [11], [12] or the emergence of previously unseen classes [13], and on the explanations offered by these methods [14] to help analysts understand potential cybersecurity incidents. Although it is well known that network intrusion data is highly imbalanced [5], [6], [15], and related studies suggest that imbalanced data [16], [17] negatively affect Machine-Learning (ML) performance, the effect of high imbalance on the performance of alert post-processing solutions remains unclear. For example, if a malicious connection overlaps with multiple benign connections in the feature space, a classifier might classify an attack as irrelevant, favoring performance on the majority class [18]. Previous work on imbalance in the security domain has not focused on the effects of imbalance on performance itself, but rather on perceived performance through the choice of appropriate evaluations methods (i.e., how imbalance may result in misleading performance evaluations that are not representative of performance in real-world usage scenarios) [11], [19]–[21]. Although it is reasonable to assume that imbalance impacts ML performance beyond misleading evaluations, it remains unclear what the magnitude of the impact is and how imbalance impacts performance compared to other dataset characteristics such as dimensionality [22] or heterogeneity [23]. The robustness of post-processing methods on highly imbalanced SOC data remains an open problem.

In this work, we study the effect of NIDS tuning on (semi-)automated alert post-processing methods. One recent method to automatically filter alerts is DeepCASE, which has proven promising in initial evaluations [8]. In this paper, we take DeepCASE as a case-study to evaluate the impact of data imbalance on post-processing performance. Further, we also investigate the explainability of post-processing methods, the lack of which can be concerning [24].

Concretely, we make the following contributions.

- To the best of our knowledge, we are the first to study the effects of imbalance on the post-processing of security alerts using real SOC data.
- We find that current post-processing methods like DeepCASE might not be robust against high imbalance, worsening classification performance and explanations.
- Further, our findings suggest that tuning is effective in improving the quality of data input to these methods, increasing classification performance and explainability.

First, we discuss the relevant background and related work in Section 2. Then, we present the methodology of our study in Section 3, and our findings in Section 4.

Finally, we discuss the implications and limitations of our findings in Section 5 and conclude our work in Section 6.

2. Background and Related Work

In Section 2.1, we discuss the background on DeepCASE, the topic of our case-study on alert post-processing in SOC, and discuss related work on imbalance and machine learning for intrusion detection in Section 2.2.

2.1. Background on DeepCASE

DeepCASE [8] belongs to the group of methods to reduce SOC workloads through post-processing using ML methods, and is designed to classify alerts to distinguish attacks from FPs by evaluating alerts in the *context* of preceding alerts regarding the same host (victim). To this end, it uses two components: a context builder to encode alerts and their context together into a *sequence*, and an interpreter to cluster similar sequences. Clusters are manually labeled, and new alerts close to an existing cluster are automatically assigned the cluster label. The behavior of both components can be tuned using hyperparameters described in Table 2 of Appendix A. DeepCASE was evaluated on the Lastline dataset, consisting of 10.5M NIDS alerts from a real-world SOC, with 45.1k attacks.

The context builder is based on a Recurrent Neural Network (RNN) with attention mechanism [25], and utilizes a sliding window to collect the context of each alert. Preceding alerts are selected based on the involved (victim) host and only within a fixed time window. The events in the resulting sequence are represented by one-hot encoding. The context builder then derives an *attention vector* from the sequence that estimates how relevant the context events are. To derive this attention vector, the RNN is trained to predict alert e_i given its context $[e_0, \dots, e_{i-1}]$. A confidence score associated with this prediction is used as a parameter to decide whether the attention vector is reliable enough to proceed with the classification.

After creating the attention vectors for the sequences, DeepCASE’s interpreter clusters the sequences together using both the attention vector and the context, which are combined into a *vector of total attention*. The interpreter uses DBSCAN [26] to cluster these vectors. After clustering, a manual step is taken in which the clusters are labeled by evaluating a sample of sequences assigned to the cluster. Thereafter, a cluster and all sequences assigned to it automatically receive the label of the sequence with the highest risk. We note that similar contexts result in similar vectors of total attention per event, which implies that DeepCASE implicitly assumes that the contexts have similar relevant contextual events and thus are likely to have the same output label.

In addition to the labels assigned during the manual cluster analysis, alerts can be classified into one of three rejection labels. Firstly, it is possible that the confidence of the context builder is insufficient to use a resulting attention vector for clustering. Secondly, a sequence may contain events of a type that was not present in the training data. Lastly, vectors of total attention may be too far from any cluster to be assigned a label. In all of these cases, an analyst must manually classify the alert.

The attention vectors derived from the context builder are argued to be useful for explaining DeepCASE’s automatic classification of alerts [8], providing insight into DeepCASE’s internals. Nadeem *et al.* present a SoK study on explainable AI for cybersecurity [24]. One of their takeaways is that involving humans in the evaluation of the quality of explanations is crucial, and that only 14% of the reviewed literature performs such user studies. Before our work, no evaluation of the correctness of the explanations offered by DeepCASE had been carried out.

2.2. Related work

Several definitions for label imbalance and related metrics exist, such as entropy. For this study, we use the definition of *Imbalance Ratio (IR)* from Lorena *et al.* [27] that generalizes to multi-class settings and is limited to the interval $[0..1]$. The definition of IR is given in Equation (1) where n_c is the number of classes, n_{c_i} is the number of samples in class i , and n is the number of samples in the complete dataset. An IR close to 1 indicates an imbalanced dataset, and an IR close to 0 indicates a balanced dataset. To characterize typical IDS data, we note that the dataset described by Yang *et al.* [5] has an IR of $1 - 4e-6$.

$$IR = 1 - \frac{1}{\frac{n_c-1}{n_c} \sum_{i=1}^{n_c} \frac{n_{c_i}}{n-n_{c_i}}} \quad (1)$$

Recent work focuses on NIDS explainability [14] and robustness, specifically against concept drift [11], [12] (i.e., changes in data distributions) that can affect performance. Some of these changes, such as those caused by adversarial behavior, can be considered inherent to the context in which IDS operate, like imbalance is to the context in which post-processing methods operate [5]. *Label imbalance* is a phenomenon in which the relative size of data classes differs significantly [27] and can negatively affect ML performance [16], [17]. Similarly, other data(set) characteristics such as dataset size [28], heterogeneity [23], and dimensionality [22] can affect performance. In this work, we study the robustness of post-processing methods against label imbalance and also relate this to the quality of explanations.

From the literature, we identify two semi-supervised methods with a purpose similar to DeepCASE: (1) NoDoze [29], and (2) AlertPro [10]. NoDoze ranks alerts and relies on the assumption that rare events are anomalous. Using contextual information, NoDoze constructs large dependency graphs that include many FPs and thus must merge paths with similar anomaly scores to reduce the number of irrelevant paths retrieved. AlertPro uses a variety of features, some of which are extracted from the context (i.e., surrounding alerts) to estimate the risk associated with alerts. Previous research on SOC workflows [7] has proposed that post-processing research could benefit from improving or tuning detectors, as this would improve the quality of the data used to train alert classifiers. Since the tuning of rulesets involves disabling noisy rules whilst not affecting low-noise rules, the proportion of generated noise, and therefore the imbalance, is reduced. In this study, we explore the effects of tuning as a method for reducing imbalance and benefiting post-processing methods.

Past research has examined work on ML methods and datasets in the security domain and found several bad practices that threaten the validity of research results. Arp *et al.* [21] identified common pitfalls of ML in computer security, of which most can affect the validity of evaluations. Two notable pitfalls are spurious correlations and data snooping. *Spurious correlations* refer to a phenomenon by which features unrelated to the security problem (e.g., resulting from the mixing of malicious and benign data from different origins) artificially inflate the performance of ML to be unreflective of real-world performance. *Data snooping* is a pitfall that refers to ML training using data that would normally be unavailable for training (i.e., incorporation of test data into the training set) and can lead to inflated performance due to unrealistic ‘foreknowledge’. They also mention the base rate fallacy, stressing the importance of using metrics robust against label imbalance for performance evaluation and their interpretation but do not focus on the fundamental effects on performance if suitable metrics were appropriately interpreted. Other previous research also discussed the impact of imbalance on perceived performance [11], [19], [20], but, to the best of our knowledge, the impact on the performance itself has not been studied yet, especially in the context of ML approaches for alert post-processing in SOC. Flood *et al.* [30] focus specifically on benchmark dataset designs and their application. Liu *et al.* [23] conducted a similar study on error prevalence in NIDS datasets and found several problems in commonly used datasets. Among the issues highlighted by all these works, a common concern is the suitability of the data (i.e., realism, variety, volume, correctness). In this work, we account for relevant pitfalls from these works and explore effects of label imbalance on the performance itself using suitable metrics.

3. Methodology

In order to study the effects of imbalance on the classification performance and explainability of alert post-processing methods, we devise an experimental methodology in which we build datasets that vary in imbalance using data from a cooperating SOC and compare the performance of DeepCASE on those datasets. We address the following research questions.

RQ1 *How does label imbalance affect the classification performance of DeepCASE in the context of the cooperating SOC?*

RQ2 *How does label imbalance affect the correctness of the explanations offered by DeepCASE?*

Our methodology accounts for several of the bad practices described in Section 2.2 to mitigate adverse effects on the validity of the research.

3.1. Data Provisioning

The cooperating SOC, monitoring several organizations using a combination of network- and host-based intrusion detection systems, has made their data available

for this study¹. The data shared were collected from the network of an educational institution. The monitored assets include institutional desktop computers and servers, as well as personal devices in a Bring-Your-Own-Device (BYOD) setting. We focus on Suricata [3] alerts the SOC collected from network traffic.²

The SOC dataset consists of 24.5M alerts generated by 489 unique Suricata rules. It was collected over a period of approximately five weeks between March 29 and May 4, 2022. The alerts collected involve approximately 1.3k distinct hosts in the monitored environment. Due to the lack of incidents during the collection period, none of the alerts generated by the rules are related to security incidents. Therefore, we use the same dataset as [31], in which traffic from 10 replayed successful attacks is integrated in the SOC dataset, generating 616 alerts. The traffic captures are replayed in the SIEM environment in such a way that they are indistinguishable from the live traffic, in accordance with existing recommendations for traffic injection [32]. Concretely, injected traffic consists of both benign and malicious traffic that is mapped to the same IP range as the existing traffic in the environment. The introduction of this traffic is analogous to the introduction of a new device in the BYOD setting from which the SOC dataset is collected. All protocols present in the injected traffic are also present in the SOC dataset. The injection results in the same alerts raised by the signature-based IDS as would otherwise have been raised if the attack and benign traffic were exhibited by a host in the environment where the SOC dataset was collected. We manually asserted that no alerts were raised that we believe to be the result of the manner in which the attacks were injected by inspecting every alert raised alongside the corresponding attack PCAP and verifying that its trigger conditions are explainable by the nature of the attacks and not due to the traffic not being part of the monitored network. We also verify that the alert timestamp coincides with the data collection period of the SOC dataset, and the IP addresses are consistent with the nature of the injected incident and the environment whose data should be matched. Note that DeepCASE is not concerned with low-level details of the injected traffic since it operates on the level of the alerts, which describe high-level security events in relation to hosts. After injection, the 616 events are manually inspected and all found to be related to the injected (successful) attack scenarios. The collaborating SOC uses only two labels for the data: *Non-Incident* and *Incident*, where *Incident* alerts are alerts related to successful attacks which should be escalated and *Non-Incident* alerts are all alerts considered FPs or otherwise not worth escalating.

3.2. Experiment Design

Data preparations To prepare DeepCASE for the SOC dataset, we perform hyperparameter optimization using the first 1% of the sequences, similar to the original

1. Ethical considerations regarding the data handling have been pre-approved by the research institution at the time of the instantiation of the collaboration with the involved SOC and include usage of (pseudo-)anonymized data to prevent identification of individuals.

2. A comparison with the Lastline dataset employed in DeepCASE is provided in Table 5 of Appendix B.

work [8].³ The 99% of sequences chronologically following the optimization set is used as the experiment set. The experiment set is further split into a train and a test set, with 20% of the data in the train and 80% in the test set. To increase DeepCASE chance at learning and classifying the Incident alerts, we copy to the training set some sequences from the Incident class in the test set. DeepCASE would otherwise struggle since all Incident alerts would be in the test set. To this end, we randomly sample sequences from the test set with the Incident label and copy them to the train set until the percentages of Incident alerts in both sets are similar. We consider this to be permissible since the primary goal of this work is not to assess the performance of DeepCASE w.r.t. the original evaluation, but to assess the impact of imbalance on its performance, which is not obscured. Although the incorporation of incidents from the test set into the training set constitutes a form of temporal data snooping that may artificially inflate performance on the minority class, this is equally present in all experiments we compare and therefore does not inhibit our ability to study the effects of imbalance on DeepCASE. We remind the reader that the goal of this research is not to suggest the expected level of performance of DeepCASE when deployed in real-world environments.

To study the effects of imbalance, we require datasets that vary in IR. We create tuned datasets by simulating the tuning process commonly employed by SOC [7]: SOC commonly disable noisy rules to suppress alerts generated by those rules. We reproduce various versions of the rule filter that the SOC implemented after data collection. To this end, we deactivate rules retroactively inline with the SOC filter, and drop all alerts associated with selected rules from the tuned datasets. According to the responsible SOC engineers, the rule filter became more refined over time, hence making datasets from more recent filters (High IR + Medium IR) include fewer FPs. We generate an additional dataset (Low IR) using more aggressive filtering, whereby per-host rule filters are changed to global rule filters to filter a rule not only for a single host but for all hosts. Both filter methods are normally applied by the SOC. Critically, rule filtering is a process that automated methods such as DeepCASE should be able to cope with, since rule filtering is a common low complexity practice for SOC to reduce analyst workloads and is considered best practice in the literature [1]. All tuned datasets have a label IR of a different order of magnitude, varying between $1 - 5e-5$ for the unfiltered dataset and $1 - 2e-2$ for the low IR dataset.

Effects on classification performance The NIDS rule adjustment process described in Section 3.1 results in three datasets to evaluate the effect of the rule adjustment process on classification performance. However, they alone are insufficient to attribute any performance difference solely to label imbalance, because tuning also affects other dataset characteristics that may simplify decision boundaries. Therefore, we build four control datasets in

which we vary other characteristics that may affect performance, to determine whether imbalance is indeed more influential on classification performance than these other characteristics (RQ1). We determine which characteristics to control for by: (i) collecting a preliminary list from the existing literature (see Section 2.2) and our understanding of DeepCASE; (ii) completing the list by interviewing authors of DeepCASE and (iii) ensuring that the identified characteristics are indeed affected by the tuning process. The four characteristics resulting from this process are: (1) dataset size [28], (2) event imbalance (i.e., classes are defined by event type instead of label), (3) context heterogeneity [23] (i.e., number of unique context vectors in the data) and (4) context builder input dimensionality [22] (i.e., number of unique event types). In an effort to prevent spurious correlations [21], we construct datasets by retroactively removing rules, sequences, or alerts to vary the identified factors, rather than injecting additional events. Concretely, we randomly sample data points to remove according to varying probabilities for each control dataset. These probabilities are derived from dataset statistics (i.e., alert/context frequencies) to influence a subset of all identified factors whilst maintaining others at levels similar to those of the unfiltered or tuned datasets. We generate control datasets for every tuned dataset to match some of its properties and vary other characteristics. These control processes and their effects on the identified characteristics are further detailed in Appendix C. To distinguish the effects of each controlled characteristic, we run DeepCASE on the control datasets for each of the high, medium, and low IR. Each run uses the hyperparameters obtained from the hyperparameter optimization described in Appendix A. Each run is executed five times to account for the stochasticity of neural network outputs. Since the control datasets vary label imbalance, dataset size, event imbalance, context heterogeneity, and context builder input dimensionality in manners uniquely distinct from the tuning process, these allow us to attribute performance improvements to specific properties of the dataset such as imbalance.

To analyze the effect of the IR on classification performance, we use a combination of descriptive statistics and regression analysis. We use a robust linear model to assess which characteristic most significantly affects classification performance. To account for scale differences between variables, all variables are scaled to have mean 0 and standard deviation 1. To ensure that no collinear variables are used in the regression, we compute the Variance Influence Factor (VIF) and remove any variables with a VIF greater than 10 [33]–[35] from the regression. We report statistical significance for each estimated effect size from a t-test; p-values below 0.05 are considered significant. In addition, we use a Kolmogorov-Smirnov (KS) Test as a goodness-of-fit test to understand how much variance can be explained by the model [36]. Rejection of the null hypothesis (H_0) of a KS-test implies poor fit. The KS test is chosen over the usual R^2 due to its robustness against outliers.

Effects on correctness of explanations To assess the correctness of DeepCASE’s explanations (RQ2), we asked an expert from the collaborating SOC to indicate relevance of each contextual event (range [0, 1]) to the investigated alert (i.e. manually replicating DeepCASE’s ‘explanation’

3. Details are provided in Appendix A. We reproduce DeepCASE results in [8] with our dataset as described in Appendix B and confirm that the performance of DeepCASE is similar to the performance reported for the Lastline dataset. We also perform a control experiment (see Section C.1) using the default DeepCASE hyperparameters to verify that the hyperparameter optimization was successful.

of its classification). Considering the large size of the dataset, we build this baseline only for `Incident` alerts, labelling hundreds of contexts. To validate the correctness of the expert-labeled vectors, a researcher executes the same task, after which we validate the vectors of the expert using the secondary labeling conducted by another expert, as described in Appendix D. Both experts rely solely on the same sequences of alert messages that are presented to DeepCASE but may interpret them using tacit knowledge [37] from their experience working with SOC. After confirming high agreement between the experts, the labeling of the first expert is kept as ground truth to evaluate DeepCASE.

We compare the expert explanations with DeepCASE’s explanations for the unfiltered and tuned datasets. By examining the cumulative distributions of similarities between explanations of different raters (i.e., DeepCASE and an expert) over these datasets, we can infer the effect of label imbalance on the correctness of DeepCASE’s explanations. We group individual runs belonging to the same dataset together to obtain a single distribution of similarities for each dataset/rater. Note that we do not leverage the control datasets to assess the effect on explainability since the sequences manually labeled by experts can include different events and require a new manual labeling for every run of a control dataset, leading to an unreasonable amount of work.

3.3. Metrics

Classification performance metrics In the original work *precision* and *recall* (PR) are used to evaluate, respectively, how many classifications are correct and how many classified alerts are classified correctly. The F1-score combines precision and recall into a single metric using a harmonic mean. One common method to calculate PR scores is *micro-averaging*, in which equal weight is assigned to every alert. Alternatively, when *macro-averaging*, the PR scores are computed separately for each class, after which the per-class scores are combined with equal weight. The macro F1-score is a more informative metric in highly imbalanced environments where the minority class performance is (more) important than the majority class performance. Thus, we use the macro F1-score for hyperparameter optimization to encourage DeepCASE to perform well on the minority class.

To reproduce and compare with the original results, we follow the same approach when calculating the macro F1-score. However, we observe that rejected alerts impact SOC operations as those alerts must be analyzed manually, and hence contribute to workload. Therefore, we develop an alternative metric, the *relaxed* F1-score, to capture this. The *relaxed* precision and recall, respectively, evaluate how many alerts are correctly presented and how many interesting alerts are presented to an analyst. Contrary to the original work, we consider rejected `Incident` alerts as True Positive (TP) (because worthy of analysis from a human analyst) and consider rejected `Non-Incident` alerts as FP (because they waste analyst time). For all other possible alerts/classifications, the definition of TP/FP remains identical to that from the original work [8]. Note that we do not change how the PR is computed, but only the definition of what constitutes a TP/FP to better

capture the alert classification’s impact on the analyst workload. As a result, rejected `Non-Incident` alerts reduce precision, and rejected `Incident` increase precision and recall. The relaxed F1-score therefore more accurately reflects operational impact of post-processing and is therefore used to assess the effects of data characteristics like imbalance on the classification performance (e.g., regression).

Correctness of explanations metrics To assess the correctness of explanations, we need to compare explanations (i.e., vectors of total attention) from different raters (e.g., expert or DeepCASE). In line with common practice, we use cosine similarity to compare vectors due to the well-understood semantics of the cosine similarity, resulting in a score in the range $[0, 1]$ ⁴ [38]. If the cosine similarity is close to 1, then the vectors of total attention are similar, and raters agree on the correct explanation. If the cosine similarity is close to 0, then the vectors indicate different relevant events, and raters disagree on the explanation.

4. Results

We discuss the effects on classification performance in Section 4.1 after which we turn to the effects on the correctness of the explanations in Section 4.2.

4.1. Classification performance

Figure 1 shows the relaxed F1-score. Our first observation is that the performance of DeepCASE for the tuned datasets (IR Experiments) is higher than that in the hyperparameter control experiment, especially for the low IR dataset. This suggests that hyperparameter optimization was indeed successful in tweaking DeepCASE’s performance on alternate data. For the various tuned datasets, and also for most control experiments, we observe a clear pattern by which DeepCASE obtains higher relaxed F1 scores on Low- and Medium-IR datasets, suggesting that a lower IR, as a result of enhanced ruleset tuning, leads to higher classification performance. Table 7 (Appendix E) shows the confusion matrix for the unfiltered and tuned datasets. From this confusion matrix, we can see that the Low- and Medium-IR datasets succeed in retrieving more relevant `Incident` alerts, and more importantly, retrieve less irrelevant `Non-Incident` alerts, which for the High-IR and unfiltered datasets were shown more often to analysts because they were rejected by DeepCASE.

One notable outlier is the performance on the Filtering Method control dataset. We attribute this outlier to a significant decrease in heterogeneity due to the filtering approach. Upon inspection of the confusion matrix, shown in Table 8 (Appendix E), we observe that DeepCASE rejects fewer `Incident` alerts, likely due to a decrease in data complexity that makes the problem addressed by DeepCASE less complex. This decrease in data complexity is further detailed in Section C.3 (Appendix C). Concretely, the control dataset has a lower heterogeneity and dimensionality compared to the tuned datasets. We note that the performance of DeepCASE on the control datasets is not necessarily representative of the performance that

4. The minimum cosine similarity is 0 instead of -1 because the vectors of total attention are non-negative and thus cannot be antiparallel.

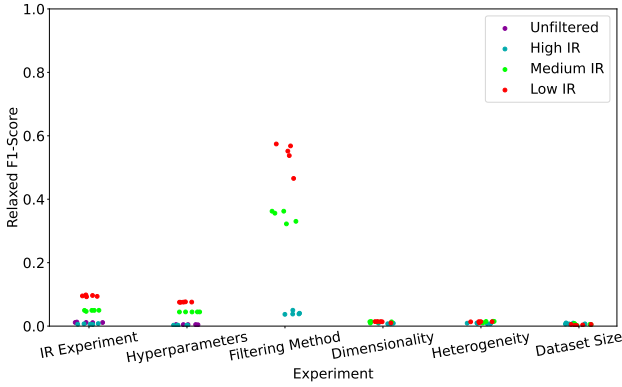


Figure 1. Relaxed F1-score for all experiments. Each point represents the relaxed F1-score for one run.

TABLE 1. REGRESSION RESULTS SHOWING THE EFFECT SIZE OF VARIOUS CHARACTERISTICS ON THE RELAXED F1-SCORE USING A ROBUST LINEAR MODEL.

Dimension	Coefficient	Confidence Interval	<i>P</i> -value
Constant	0.09	[0.07, 0.10]	< 0.01
Label IR	-1.03	[-1.04, -1.02]	< 0.01
Dataset Size	-0.01	[-0.01, 0.03]	0.24
Heterogeneity	-0.04	[-0.06, -0.03]	< 0.01
Unique Events	0.01	[-0.01, 0.03]	0.27
N.obs.			80
KS Test	0.13		0.15

could be obtained by DeepCASE in practice, but it does serve a purpose in assessing the effect of the characteristics of the data on the classification performance.

To attribute differences in performance to imbalance, we use regression over the various controlled characteristics following the methodology described in Section 3.2. The outcome is shown in Table 1. The event IR, was dropped from the regression due to its VIF exceeding the threshold we set. The test statistic of the KS-test is 0.13 and the corresponding *p*-value is 0.15. Given the *p*-value, we fail to reject H_0 and lack indications that the model poorly explains the performance differences between the various datasets. The test statistic indicates that the cumulative distribution function resulting from the regression is at most 0.13 off from the cumulative distribution function of true relaxed F1-score, suggesting a non-negligible portion of the effect is successfully explained by the model.

The significant regression coefficients inform on characteristics that significantly influence the relaxed F1-score. We conclude the label IR appears to have the largest impact with a coefficient of -1.03 , compared to the other characteristics we control for. Hence, the relaxed F1-score is higher for datasets with a lower label IR. The rest of the characteristics appear to have little or inconclusive effect on the relaxed F1-score, with coefficients between -0.04 and 0.01 . However, heterogeneity appears to have a small but significant effect, where a reduction in heterogeneity leads to a small improvement of the relaxed F1-score. To contextualize this, we note that the effect size of context heterogeneity on DeepCASE’s performance is less than 5% of the effect size of label imbalance.

4.2. Explainability

The cumulative distributions of the cosine similarities between the vectors of total attention per event provided by DeepCASE and the ground truth for the tuned datasets

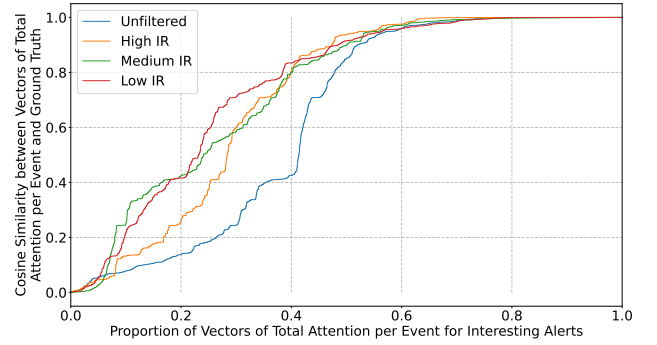


Figure 2. Cumulative distribution plots of the cosine similarity comparing of explanations from DeepCASE and the expert for the unfiltered and tuned datasets.

are shown in Figure 2. We see that approximately 25% of the vectors of total attention per event is identical to the ground truth and about 50% of the vectors per event have a cosine similarity of 0.75 or higher, suggesting high similarity. However, about 30% of the vectors have a cosine similarity of 0.25 or lower for the unfiltered dataset, suggesting DeepCASE provides incorrect explanations for these alerts.

As we compare the unfiltered set to the tuned sets with different IR, we observe that the vectors provided by DeepCASE on the tuned datasets are usually more similar to the expert rating. Interestingly, the vectors from the Low-IR dataset also appear to outperform those from the High-IR dataset, which was also tuned to some extent. Hence, we conclude that ruleset tuning effectively assists DeepCASE in providing correct explanations.

5. Discussion

We focus on DeepCASE as a case-study for the broader class of alert post-processing methods in the context of SOC. To support this, we compare DeepCASE with two related methods: (1) NoDoze [29], and (2) AlertPro [10]. DeepCASE, NoDoze, and AlertPro all use information from surrounding alerts, similar to what analysts would do [4]. NoDoze employs mitigation methods, such as merging paths with similar anomaly scores, to reduce the number of irrelevant paths retrieved using a threshold to decide on the similarity of anomaly scores. Similarly to DeepCASE, where frequently occurring alerts can evict relevant alerts from contexts, in AlertPro excessive alerts can suppress relevant information from other alerts in the analyzed contexts. Thus, both NoDoze and AlertPro rely on methods that are prone to the effects of label imbalance, similar to DeepCASE.

Implications. Although it is well known that imbalance can negatively affect ML performance [39], prior to this work it was unclear what magnitude of this effect is in the context of network intrusion detection. The results of Section 4.1 indicate that label imbalance is, by far, the most important factor for the performance of DeepCASE. Regression over results derived from the datasets controlling for other characteristics that may affect decision boundaries shows that label imbalance outweighs other factors such as the size of the dataset or the heterogeneity of the sequences.

We envision two directions to increase robustness of alert post-processing methods such as DeepCASE in

light of the imbalance inherent to the network intrusion detection domain: (1) reducing the imbalance in the data provided as input to such methods; and (2) making such methods more robust against imbalance. Tuning rulesets [7] and increasing the quality of IDS rules are examples in the first direction. Researchers in the ML domain have already made steps towards the second direction by applying methods such as focal loss [40], cost-sensitive learning [18], or by applying different methods for pre-processing alert data (e.g., constructing sequences).

Regardless of whether post-processing methods are used, deterministic alternatives can be utilized to reduce SOC workloads. Ruleset tuning has been discussed as a method to improve the performance of DeepCASE, but also reduces the number of irrelevant alerts by 99.6%, without the application of other post-processing methods. Hence, research on more traditional methods can still benefit ML methods for SOC.

During our experiments, we computed various performance metrics, including the micro- and macro-averaged F1-scores in addition to the relaxed F1-score used in Section 4.1. Some metrics like micro F1-score would not show significant differences even though the performance on the minority class was significantly affected. In light of the high imbalance, which is common in NIDS research, and the relative importance of the minority class, we consider micro-averaging inadequate for assessing NIDS or alert post-processing systems and encourage researchers to use appropriate performance measures [21] that account for the imbalance and differences in cost associated with misclassifications between classes. We also encourage researchers to use robust metrics that account for rejection classes in a manner corresponding to the intended application context of the systems.

Previous research has stressed the importance of user studies in evaluating the explainability of decision support tools [24]. The original work on DeepCASE suggested that vectors of total attention per event can serve as an explanation but has not evaluated its quality in that context. In Section 4.2 we make a first step towards evaluating the explainability of DeepCASE by involving two experts. Our results show that the classification for a non-negligible portion of vectors of total attention are poorly explained by DeepCASE. However, a reduction of the label imbalance through tuning resulted in a promising improvement in explanation correctness. Further research on the robustness of post-processing methods in the presence of imbalance can therefore enhance interpretability of these methods and encourage trust in SOC automation.

Threats to validity. Data realism and representativeness are important issues [21] and past research has raised concerns regarding the generation of benchmark datasets, specifically through artificial means [30], [32]. Since the methods described in Section 3.2 and Appendix C introduce no new data, artificial diversity or incorrect labels have not been introduced in the tuned datasets. We believe that injection of attack traffic as described in Section 3.1 and incorporation of test data into the training set as described in Section 3.2 have not led to spurious correlations or data snooping threatening the validity of our conclusions. These pitfalls would artificially inflate the performance on the minority class, whereas the performance on the minority class remains distinctly lower than on the ma-

jority class, as shown in Table 7 (Appendix E). Moreover, these pitfalls are present equally in all performed experiments and do therefore not prevent comparisons between performance on different datasets to assess the effects of imbalance, although the mentioned performance need not be reflective of real-world deployments. Note that datasets generated by means other than mere collection or tuning may not be representative of the real world but serve their purpose well in assessing the impact of effects in the data on the performance of alert post-processing methods. Therefore, the results of our control datasets should not be interpreted as reflecting real-world performance but only relative to the results on the datasets with different IR. The choice of appropriate metrics in light of imbalanced data [21] has been addressed in Section 3.3.

We believe to have optimized DeepCASE well for the presence of label imbalance using the macro F1-score. Thereafter, we have used the relaxed F1-score, since it is considered more representative of the problem that a SOC would address using DeepCASE. Although the macro F1-score used to optimize hyperparameters and the relaxed F1-score used to evaluate DeepCASE’s performance are different, the macro and relaxed F1-scores are statistically correlated. Therefore, we deem it unlikely that optimizing for the macro F1-score has negatively affected conclusions drawn based on results using the relaxed F1-score. When comparing the macro F1-scores shown in Figure 3 (Appendix B) and the relaxed F1-scores shown in Figure 1, we observe that both metrics rank the performance of DeepCASE similarly for the different levels of imbalance for each of the tuned and control datasets used in the regression from Section 4.1. We consider the relaxed F1-score to be more indicative of real-world performance and operational implications, since it accounts for the impact of alerts rejected (i.e., unclassified) by DeepCASE on time spent investigating by security analysts.

The regression presented in Section 4.1 shows a significant effect on the relaxed F1-score, despite the absence of other variables that we aimed to control for due to the high collinearity of those variables. Since event IR and label IR are correlated, it is possible that (part of) the effect of label IR in the regression is actually the result of event IR instead. Despite this ambiguity, we remain confident that imbalance, be it event or label, has a significant effect on the classification performance.

Since expert 1 and expert 2 agreed with each other about as much as DeepCASE agreed with expert 1, we cannot draw strong conclusions from the correctness of DeepCASE’s explanations. From feedback acquired from the experts on the provided explanations, the experts struggled the most with the ambiguous interpretation of attention assigned to events in the input sequence when no relevant events exist in the input sequence. Explanations offered by DeepCASE therefore remain a topic requiring future research, as it apparently is not trivial for experts to agree on what constitutes correct explanations.

6. Conclusion

In this work, we evaluated the effect of label imbalance on DeepCASE and showed that classification performance increases when label imbalance is reduced as a result of tuning. Furthermore, we evaluated the correctness of

explanations offered, and found that the correctness also improves by tuning. We found that tuning of rules in SOC can reduce the imbalance and can effectively improve classification performance and correctness of explanations offered by alert post-processing methods like DeepCASE.

Data Availability

To promote reproducibility, we archive an anonymized copy of the SOC dataset, the code used to create the tuned datasets and control datasets, and the code used to run and evaluate DeepCASE's performance at our institution. The archived dataset and the code used in this research can be shared upon request with interested researchers.

Acknowledgment

Parts of this work were derived from a prior M.Sc. graduation project [41]. The authors thank the unnamed SOC for their cooperation and for making data available for analysis, and Thijs van Ede discussing DeepCASE with us. This publication is part of the CATRIN, INTERSECT, and SeReNity projects (with numbers NWA.1215.18.003, NWA.1160.18.301, and CS.010) which are (partly) financed by the Dutch Research Council (NWO). For the purpose of Open Access, a CC-BY 4.0 public copyright license is applied to any Author Accepted Manuscript version arising from this submission.

References

- [1] K. Knerler, I. Parker, and C. Zimmerman, *11 Strategies of a World-Class Cybersecurity Operations Center*, 2022. [Online]. Available: <https://www.mitre.org/news-insights/publication/11-strategies-world-class-cybersecurity-operations-center>
- [2] M. Vielberth, F. Bohm, I. Fichtinger, and G. Pernul, "Security operations center: A systematic study and open challenges," *IEEE Access*, vol. 8, p. 227756–227779, 2020. [Online]. Available: <https://doi.org/10.1109/ACCESS.2020.3045514>
- [3] Open Information Security Foundation, "Suricata," November 2024. [Online]. Available: <https://web.archive.org/web/20241001122450/https://suricata.io/>
- [4] L. Kersten, T. Mulders, E. Zambon, C. Snijders, and L. Allodi, "'give me structure': Synthesis and evaluation of a (network) threat analysis process supporting tier 1 investigations in a security operation center," in *Nineteenth Symposium on Usable Privacy and Security (SOUPS 2023)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 97–111. [Online]. Available: <https://www.usenix.org/conference/soups2023/presentation/kersten>
- [5] L. Yang, Z. Chen, C. Wang, Z. Zhang, S. Booma, P. Cao, C. Adam, A. Withers, Z. Kalbarczyk, R. K. Iyer, and G. Wang, "True attacks, attack attempts, or benign triggers? an empirical measurement of network alerts in a security operations center," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 1525–1542. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity24/presentation/yang-limin>
- [6] K. Teuwen, T. Mulders, E. Zambon, and L. Allodi, "Ruling the unruly: Designing effective, low-noise network intrusion detection rules for security operations centers," in *Proceedings of the 2025 ACM on Asia Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, Inc, 2024. [Online]. Available: <https://doi.org/10.1145/3708821.3710823>
- [7] M. Vermeer, N. Kadenko, M. van Eeten, C. Gañán, and S. Parkin, "Alert alchemy: Soc workflows and decisions in the management of nids rules," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2023, p. 2770–2784. [Online]. Available: <https://doi.org/10.1145/3576915.3616581>
- [8] T. van Ede, H. Aghakhani, N. Spahn, R. Bortolameotti, M. Cova, A. Continella, M. van Steen, A. Peter, C. Kruegel, and G. Vigna, "DeepCASE: Semi-Supervised Contextual Analysis of Security Events," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2022.
- [9] W. U. Hassan, M. A. Noureddine, P. Datta, and A. Bates, "Omega-log: High-fidelity attack investigation via transparent multi-layer log analysis," in *Network and distributed system security symposium*, 2020.
- [10] X. Wang, X. Yang, X. Liang, X. Zhang, W. Zhang, and X. Gong, "Combating alert fatigue with alertpro: Context-aware alert prioritization using reinforcement learning for multi-step attack detection," *Computers & Security*, vol. 137, p. 103583, 2024. [Online]. Available: <https://doi.org/10.1016/j.cose.2023.103583>
- [11] G. Andresini, F. Pendlebury, F. Pierazzi, C. Loglisci, A. Appice, and L. Cavallaro, "Insomnia: Towards concept-drift robustness in network intrusion detection," in *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*. New York, NY, USA: Association for Computing Machinery, 2021, p. 111–122. [Online]. Available: <https://doi.org/10.1145/3474369.3486864>
- [12] F. Barbero, F. Pendlebury, F. Pierazzi, and L. Cavallaro, "Transcending transcend: Revisiting malware classification in the presence of concept drift," in *2022 IEEE Symposium on Security and Privacy (SP)*, 2022, pp. 805–823. [Online]. Available: <https://doi.org/10.1109/SP46214.2022.9833659>
- [13] D. Han, Z. Wang, W. Chen, K. Wang, R. Yu, S. Wang, H. Zhang, Z. Wang, M. Jin, J. Yang, X. Shi, and X. Yin, "Anomaly detection in the open world: Normality shift detection, explanation, and adaptation," in *30th Annual Network and Distributed System Security Symposium, NDSS 2023, San Diego, California, USA, February 27 - March 3, 2023*. The Internet Society, 2023.
- [14] F. Wei, H. Li, Z. Zhao, and H. Hu, "xNIDS: Explaining deep learning-based network intrusion detection systems for active intrusion responses," in *32nd USENIX Security Symposium (USENIX Security 23)*. Anaheim, CA: USENIX Association, Aug. 2023, pp. 4337–4354. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity23/presentation/wei-feng>
- [15] M. Vermeer, M. van Eeten, and C. Gañán, "Ruling the rules: Quantifying the evolution of rulesets, alerts and incidents in network intrusion detection," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2022, p. 799–814. [Online]. Available: <https://doi.org/10.1145/3488932.3517412>
- [16] G. Ho, A. Sharma, M. Javed, V. Paxson, and D. Wagner, "Detecting credential spearphishing attacks in enterprise settings," *Proc. of 26th USENIX Security*, 2017.
- [17] H. Kaur, H. S. Pannu, and A. K. Malhi, "A systematic review on imbalanced data challenges in machine learning: Applications and solutions," *ACM Comput. Surv.*, vol. 52, no. 4, Aug. 2019. [Online]. Available: <https://doi.org/10.1145/3343440>
- [18] V. López, A. Fernández, S. García, V. Palade, and F. Herrera, "An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics," *Information Sciences*, vol. 250, pp. 113–141, 2013. [Online]. Available: <https://doi.org/10.1016/j.ins.2013.07.007>
- [19] S. Axelsson, "The base-rate fallacy and the difficulty of intrusion detection," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 3, p. 186–205, 8 2000. [Online]. Available: <https://doi.org/10.1145/357830.357849>
- [20] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: Eliminating experimental bias in malware classification across space and time," in *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 729–746. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity19/presentation/pendlebury>

- [21] D. Arp, E. Quiring, F. Pendlebury, A. Warnecke, F. Pierazzi, C. Wressnegger, L. Cavallaro, and K. Rieck, "Dos and don'ts of machine learning in computer security," in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 3971–3988. [Online]. Available: <https://www.usenix.org/conference/usenixsecurity22/presentation/arp>
- [22] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *Computational Intelligence and Bioinspired Systems*, J. Cabestany, A. Prieto, and F. Sandoval, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 758–770.
- [23] L. Liu, G. Engelen, T. Lynar, D. Essam, and W. Joosen, "Error prevalence in nids datasets: A case study on cic-ids-2017 and cse-cic-ids-2018," in *2022 IEEE Conference on Communications and Network Security (CNS)*, 2022, pp. 254–262. [Online]. Available: <https://doi.org/10.1109/CNS56114.2022.9947235>
- [24] A. Nadeem, D. Vos, C. Cao, L. Pajola, S. Dieck, R. Baumgartner, and S. Verwer, "Sok: Explainable machine learning for computer security applications," in *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, 2023, pp. 221–240. [Online]. Available: <https://doi.org/10.1109/EuroSP57164.2023.00022>
- [25] Z. Niu, G. Zhong, and H. Yu, "A review on the attention mechanism of deep learning," *Neurocomputing*, vol. 452, pp. 48–62, 2021. [Online]. Available: <https://doi.org/10.1016/j.neucom.2021.03.091>
- [26] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.
- [27] A. C. Lorena, L. P. F. Garcia, J. Lehmann, M. C. P. de Souto, and T. K. Ho, "How complex is your classification problem? A survey on measuring classification complexity," *CoRR*, vol. abs/1808.03591, 2018. [Online]. Available: <http://arxiv.org/abs/1808.03591>
- [28] J. Prusa, T. M. Khoshgoftaar, and N. Seliya, "The effect of dataset size on training tweet sentiment classifiers," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 96–102. [Online]. Available: <https://doi.org/10.1109/ICMLA.2015.22>
- [29] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, "Nodoze: Combatting threat alert fatigue with automated provenance triage," in *network and distributed systems security symposium*, 2019.
- [30] R. Flood, G. Engelen, D. Aspinall, and L. Desmet, "Bad design smells in benchmark nids datasets," in *2024 IEEE 9th European Symposium on Security and Privacy (EuroS&P)*, 2024, pp. 658–675. [Online]. Available: <https://doi.org/10.1109/EuroSP60621.2024.00042>
- [31] L. Kersten, T. Mulders, E. Zambon-Mazzocato, C. Snijders, and L. Allodi, "'give me structure': Synthesis and evaluation of a (network) threat analysis process supporting tier 1 investigations in a security operation center," in *Proceedings of the 19th Symposium on Usable Privacy and Security, SOUPS 2023*. Usenix Association, Aug. 2023, pp. 97–111, 19th Symposium on Usable Privacy and Security, SOUPS 2023 ; Conference date: 07-08-2023 Through 08-08-2023.
- [32] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an intrusion detection dataset: the cicids2017 case study," in *2021 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2021, pp. 7–12.
- [33] B. K. Slinker and S. A. Glantz, *Primer of applied regression and analysis of variance*. McGraw-Hill, 1990.
- [34] G. S. Maddala and K. Lahiri, *Introduction to econometrics*. Macmillan New York, 1992, vol. 2.
- [35] J. B. Gray, "Regression with graphics: A second course in applied statistics," 1994.
- [36] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.
- [37] S. Y. Cho, J. Happa, and S. Creese, "Capturing tacit knowledge in security operation centers," *IEEE Access*, vol. 8, pp. 42 021–42 041, 2020.
- [38] R. Connor, "A tale of four metrics," in *Similarity Search and Applications*, L. Amsaleg, M. E. Houle, and E. Schubert, Eds. Cham: Springer International Publishing, 2016, pp. 210–217.
- [39] M. S. Santos, P. H. Abreu, N. Japkowicz, A. Fernández, and J. Santos, "A unifying view of class overlap and imbalance: Key concepts, multi-view panorama, and open avenues for research," *Information Fusion*, vol. 89, p. 228–253, Jan. 2023. [Online]. Available: <https://doi.org/10.1016/j.inffus.2022.08.017>
- [40] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2999–3007. [Online]. Available: <https://doi.org/10.1109/ICCV.2017.324>
- [41] S. Baggen, "Deepdive: Evaluating deepcase on dataset imbalance & validity of explanations," Master's thesis, Mathematics and Computer Science, 2024. [Online]. Available: <https://research.tue.nl/en/studentTheses/deepdive-evaluating-deepcase-on-dataset-imbalance-validity-of-exp>
- [42] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research*, vol. 13, no. 2, 2012.

Appendix A.

Hyperparameter optimization

DeepCASE has several hyperparameters that can be optimized as discussed in Section 2.1 and detailed in Table 2. The first 1% of the dataset that is selected only for hyperparameter optimization and is excluded from the remaining runs. Within this 1% of the data, we make a 50/50 split to obtain train and test sets for hyperparameter optimization. Similar to the approach described in Section 3, we copy over Incident samples from the test set to the train set until both sets have similar proportions of Incident alerts.

The optimal parameters are selected using random grid search, where a point on the grid is randomly selected for trial, combining the advantages of grid search and manual search [42] while avoiding an excessively large search space. Contrary to the original work, we optimize the macro F1-score instead of the micro F1-score [8] to favor performance on the minority class.

The hyperparameter optimization experiment is run before any data analysis is performed. Every time the dataset presented to DeepCASE changes, we first run a hyperparameter optimization experiment. This ensures DeepCASE can perform optimally for the dataset, such that the conclusions we reach from the data analysis are independent of the chosen hyperparameters, considering the optimal hyperparameters are typically different when a model is deployed on different data.

For discovering suitable hyperparameter values, we use random search on a grid with a limit on the amount of possible trials to find the optimal performing set of hyperparameters. The hyperparameters and their possible choices are shown in Table 3. Random search is chosen because it generally outperforms grid search in both the time spent on computations and the performance of the model [42]. However, if we rely purely on random search with the upper bound and lower bound of each parameter set to the highest and lowest values in the original paper, then the possible search space is so large (409 600 000 combinations) that an excessive number of trials would be required. Considering the required time and hardware, as well as the potential impact on the environment, we reduce

TABLE 2. OVERVIEW OF DEEPCASE’S HYPERPARAMETERS.

Hyperparameter	Function
Context Length (n)	Maximum amount of events considered to be part of the context.
Context Timeout (t)	Maximum duration between an event and the corresponding alert for the event to be considered part of the context.
Context Builder Hidden Nodes	Amount of nodes that are part of the hidden layer of the context builder’s neural network.
δ (Label Smoothing Regularization)	Adjustment of target output label probabilities during training of context builder.
$\tau_{confidence}$ (Confidence Threshold)	Minimum confidence required before an attention vector and the vector of total attention per event is used in the clustering algorithm.
ϵ (DBSCAN Maximum Distance)	Maximum distance between vectors of total attention where the DBSCAN algorithm considers the different vectors to be part of the same cluster.
Minimum Cluster Size	Minimum amount of vectors required to be considered a cluster, all within a maximum distance of ϵ .

TABLE 3. POSSIBLE VALUES FOR DEEPCASE’S HYPERPARAMETERS.

Hyperparameter	Possible values
Context Length (n)	10, 15, 20
Context Timeout (t)	1 day, 1 week, 1 month
Context Builder Hidden Nodes	32, 64, 128
δ (Label Smoothing Regularization)	[0, 1] with steps of 0.05
$\tau_{confidence}$ (Confidence Threshold)	[0.05, 1] with steps of 0.05
ϵ (DBSCAN Maximum Distance)	[0.05, 1] with steps of 0.05
Minimum Cluster Size	5, 10, 20, 50

the amount of possible combinations by randomly sampling from a grid, combining the advantages of random search with applying domain knowledge to ensure random search will avoid insensible parameters like a sequence length of 1. This approach significantly reduces the search space (779 760 combinations), making hyperparameter optimization feasible. The resulting optimal hyperparameters for each of the different sets are shown in Table 4.

Appendix B. Reproduction

To further support the interpretation of our experiments and enable comparison with other datasets, we describe the relevant aspects of the data and make a comparison with the dataset used by Van Ede *et al.* [8]. Respectively, we refer to these datasets as the SOC and Lastline datasets and their comparison is summarized in Table 5.

The SOC dataset contains twice the amount of fired alerts and nearly two times more distinct rules compared to the Lastline dataset, generated from less than 300 times as many hosts, and was gathered during a shorter time period. Based on these numbers, it seems that the environment from which the SOC dataset originates raises more FPs than that of the Lastline dataset, an observation explained by the fact that the rules used by the Suricata sensor generating the alerts had not been tuned by the cooperating SOC. This also implies that the SOC dataset has a higher imbalance and offers more complexity to post-processing methods such as DeepCASE.

We have opted to use only two labels for the SOC data: *Non-Incident* and *Incident*, where *Incident* alerts are alerts related to successful attacks and *Non-Incident* alerts are all alerts considered FPs. In the original work, DeepCASE was used to estimate threat levels associated with alerts, which can be considered a generalization of the task for which we employ it: distinguishing successful from unsuccessful attacks. Our less granular labeling presents DeepCASE with a less complex task.

One may raise concerns about the difference in the number of incidents between the Lastline dataset and the dataset used in this work. We attribute this to a difference in the interpretation of what constitutes an incident and the corresponding labeling methodology. The cooperating SOC labels an alert as an attack if it is related to a malicious action that is *successful*. Therefore, the SOC does not consider alerts related to malicious but unsuccessful actions as attacks. We suspect that in the Lastline dataset, the label *ATTACK* might also have been given to malicious but unsuccessful actions. As a result, the SOC dataset is more imbalanced. This high imbalance is in line with a previous characterization of SOC data [5], whose dataset was collected over four years from a real-world SOC and contained 115 million alerts and only 227 attacks, where their definition of “attack” aligns with ours.

As shown in Figure 3 (note the Figure also contains results on the control datasets described in Appendix C), the unfiltered dataset has a mean macro F1-score of 0.502 for the IR experiment. We note that the Macro F1-scores obtained in the original work⁵ are similar to the Macro F1-scores obtained in this work. Thus, the performance of DeepCASE on the Lastline and SOC datasets is comparable despite differently defined attacks.

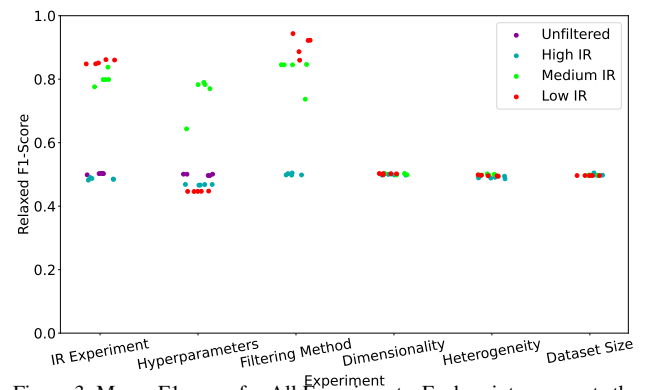


Figure 3. Macro F1-score for All Experiments. Each point represents the Macro F1-score of one run.

Looking at the confusion matrix for the unfiltered dataset in Table 7 of Section E, we can see that nearly all Non-Incident alerts are classified correctly, whereas nearly all Incident alerts are rejected. Among the few classified Incident alerts, most are classified incorrectly. Due to the imbalance in the dataset, the micro F1-score would almost only reflect the performance of the majority class and hence hide the poor performance on the minority class. This reaffirms our choice for the macro F1-score and relaxed F1-score discussed in 3.3.

5. https://github.com/Thijsvanede/DeepCASE/blob/sp/experiments/baseline/results/deepcase_after_automatic.txt#L58

TABLE 4. OPTIMAL HYPERPARAMETERS FOR THE DIFFERENT DATASETS.

Hyperparameter	Unfiltered	High Label IR	Medium Label IR	Low Label IR
Context Length	15	10	20	15
Context Timeout	1 Day	1 Month	1 Day	1 Week
Context Builder Hidden Nodes	64	32	32	64
δ	0.25	0.5	0.35	0
$\tau_{confidence}$	0.05	0.05	0.05	0.05
ϵ	0.8	0.75	0.9	1
Minimum Cluster Size	50	5	5	5

TABLE 5. PROPERTIES OF THE SOC AND LASTLINE DATASETS.

Property	SOC Dataset	Lastline
Fired Alerts	24.4 million	10.5 million
Unique Rules	474	291
Collection Time	5.5 weeks	5 months
Distinct Hosts	1.3k	388k
Attacks	616	45.1k

This poor performance on the Incident alerts is despite the copying of some Incident alerts from the test set to the train set as described in Section 3.

The macro F1-score generally increases as label IR is reduced. The high IR dataset has a mean macro F1-score of 0.486, the medium IR dataset has a mean macro F1-score of 0.802 and the low IR dataset has a mean macro F1-score of 0.854. This is in-line with our findings from Section 4.1 using the relaxed F1-score.

The macro F1-score for the low IR dataset for the hyperparameter control experiment described in Section C.1 of Appendix C is surprisingly not as high as its IR counterpart, with a mean macro F1-score of 0.447. However, it seems to be the case that the default hyperparameters only work as expected for the unfiltered, high IR and medium IR datasets, due to their scores being similar with their IR experiment counterparts, with mean macro F1-scores of 0.499, 0.468 and 0.754 respectively. Since simply tuning the hyperparameters seems to increase the macro F1-score significantly for the low IR dataset, we can conclude that hyperparameter optimization for each dataset can significantly increase DeepCASE’s performance.

Appendix C.

Control experiments and datasets

This appendix describes how we control for various experimental conditions such as hyperparameters (Section C.1) and characteristics other than imbalance. Concretely, we describe how we derive control datasets in Section C.2 and make some comparisons between datasets in Section C.3. The outcome of running DeepCASE controlling for these other characteristics is discussed in Section 4.1.

C.1. Hyperparameter control experiment

While not a property of the dataset, we did identify different sets of hyperparameters (see Appendix A) for the different datasets, which are also different compared to the set of hyperparameters supplied by the original work [8]. To verify whether the optimization is performed correctly and to assess the effect of hyperparameter tuning on DeepCASE, we perform a hyperparameter control experiment. As such, we run the different datasets through DeepCASE using the selection of hyperparameters as suggested by

the original work [8]. The effect of hyperparameter optimization and the outcome of this control experiment is discussed in relation to the original work in Appendix B.

C.2. Control datasets

According to the methodology presented in Section 3.2, we require control datasets to distinguish the effects of different characteristics that may affect classification performance. The characteristics we control for are: dataset size, event imbalance, context heterogeneity, and context builder input dimensionality. In the following, we describe the derivation of four types of control datasets, each intended to vary the aforementioned characteristics w.r.t. the unfiltered dataset as depicted in Table 6. We generate a control dataset of each type for every tuned dataset to match some of its properties and vary other characteristics. Runs on the control dataset use the same hyperparameter values as used for the corresponding tuned dataset.

C.2.1. Filtering Method. To derive this control dataset, we filter out sequences until we match the dataset size of the corresponding tuned dataset. Sequences are selected at random with uniform probability of selecting a specific sequence, with the exception that sequences with the Incident label are never removed. As a result, the label imbalance and dataset size are reduced from the unfiltered dataset. By chance, all sequences associated with a rule are randomly selected to be filtered for many infrequent rules, reducing the dimensionality as a result. Contrary to the tuned datasets, the event IR and heterogeneity remain largely unaffected.

C.2.2. Dataset Size. To derive this control dataset, we follow the same approach as for the Filtering Method experiment but also aim to fix the label imbalance to be equal to that of the unfiltered dataset. This allows us to better understand the effects of dataset size on classification performance. To this end, we perform an additional step where we filter out sequences with the Incident label uniformly at random to increase the label IR until it matches that of the unfiltered dataset. As a result, the dataset size is reduced from the unfiltered dataset. Similarly to the Filtering Method control dataset, the dimensionality is also reduced as a result. Contrary to the tuned datasets, label IR, event IR, and heterogeneity of these control datasets remain similar to those of the unfiltered dataset.

C.2.3. Dimensionality. With context builder input dimensionality, we refer to the amount of unique alerts in the dataset. Filtering out a rule will remove all alerts generated by the said rule from the dataset, which affects multiple

TABLE 6. COMPARISON OF EXPECTED EFFECTS OF TUNED DATASETS AND CONTROL DATASETS ON CONTROLLED CHARACTERISTICS WITH RESPECT TO THE UNFILTERED DATASET.

Control Dataset	Label Imbalance	Event Imbalance	Dataset Size	Heterogeneity	Dimensionality
SOC Ruleset Tuning	-	-	-	+	-
Filtering Method	-	-	-	-	-
Dataset Size	-	-	-	-	-
Dimensionality	-	-	-	-	-
Heterogeneity	-	-	-	+	-

contexts in the data and reduces how many unique contexts can be constructed. To derive this control dataset, we randomly filter out alerts with probabilities inversely related to the frequency of alerts in the data. The probability of filtering out an alert i is shown in Equation (2), where a_c is the number of alerts in the dataset and a_i is the number of occurrences of the alert i . The probability is normalized to sum to 1. The square is included to encourage more aggressive, and hence faster, filtering. These probabilities make it more likely that infrequent alerts are filtered out and less likely that common alerts are filtered. Note that sequences with the `Incident` label are never removed. The dimensionality of the resulting control dataset is reduced from the unfiltered dataset to match that of the corresponding tuned dataset. Contrary to the tuned datasets, label IR, event IR, dataset size, and heterogeneity of these control datasets remain similar to those of the unfiltered dataset.

$$\left(\frac{a_c}{a_i}\right)^2 \quad (2)$$

C.2.4. Heterogeneity. Context heterogeneity refers to how heterogeneous the contexts of a dataset are. The more unique contexts a dataset contains, the more heterogeneous the dataset is. To assess the effect of heterogeneity, we introduce another control dataset with the aim of only varying the heterogeneity and not other characteristics such as label imbalance, dimensionality, or dataset size. As such, we randomly filter out common sequences until we match the desired context heterogeneity of the corresponding tuned dataset. The probability of filtering out a sequence i is shown in Equation (3), where s_c is the number of contexts in the dataset and s_i is the amount of occurrences of context i . The probability is normalized to sum to 1. The square is included to encourage more aggressive, and hence faster, filtering. We perform upsampling to match the dataset size of the unfiltered dataset. The resulting control dataset has the same heterogeneity as its tuned counterpart (hence increased from the unfiltered dataset) but label imbalance and dimensionality similar to that of the unfiltered dataset. The event IR is also reduced from the unfiltered dataset as a result of the filtering.

$$\left(\frac{s_i}{s_c}\right)^2 \quad (3)$$

C.3. Filtering Method Control Dataset Comparison with Tuned Datasets

Looking deeper into the Filtering Method (FM) control experiment specifically and its comparison to the tuned datasets which is summarized in Table 6, we see that all controlled characteristics are significantly different compared to the tuned datasets, except for the label IR. For

all FM control experiment datasets, the dimensionality, and heterogeneity are lower than in the corresponding tuned datasets. These properties seem to indicate that the composition of the filtering method control dataset is more homogeneous and less varied w.r.t. the event types than the tuned datasets. This is also reflected in the number of unique contexts, which are generally lower for the filtering method experiment compared to the IR experiment. This corresponds to the intuition that if you give a neural network largely similar data, its ability to classify those data correctly improves, hence explaining the significant difference in performance.

Appendix D. Expert labeling validation

Another expert (also recruited from the SOC, and referred to as expert 2) independently assessed a random selection of the same vectors of total attention per event, the size of 11% of the total set in-line with the approach described in Section 3.2.

When comparing the manually created vectors of total attention per event, as shown in the form of a cumulative distribution in Figure 4, we see that the experts have created similar vectors for the majority of the set. More specifically, for approximately 40% of the compared vectors, the labeling of both experts are identical. More than 60% of vectors have a cosine similarity higher than 0.8, suggesting high agreement between experts on what constitutes correct labeling. Only about 20% of vectors have a cosine similarity below 0.4, suggesting a low agreement. Considering that expert 2 has independently created similar vectors of total attention per event for the majority of the randomly selected vectors, we conclude that the ground truth created by expert 1 provides a good baseline for which events in the input sequence should be assigned attention.

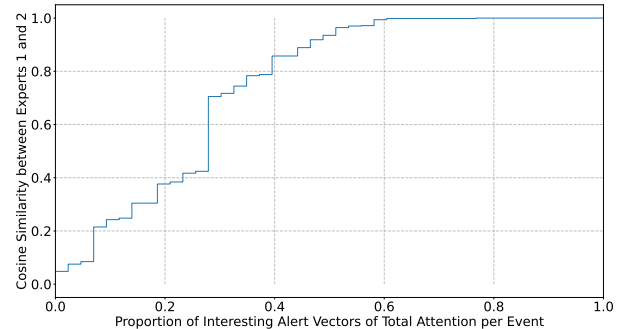


Figure 4. Cumulative Distribution of the Cosine Similarity of Expert Created vectors of total attention per event between Expert 1 and 2

Appendix E. Additional results

This appendix lists confusion matrices from our experiments described in Section 3 and Section 4. Table 7 reports on the performance on the unfiltered and tuned datasets. Table 8 reports on the performance on the filtering method control datasets.

As explained in Section 4.1, the difference in performance between runs of DeepCASE on the tuned datasets and the Filtering Method (FM) control datasets is attributed to the lower data complexity of the FM control datasets. When comparing the aggregated classification reports on the tuned datasets and FM control datasets, shown in Table 7 and Table 8, we see this reflected in the results. When comparing the performance on the tuned datasets with the performance on the FM control datasets, both the mean amount and the standard deviation of `Non-Incident` alerts that are not classified are consistently lower for the FM control datasets, with a mean of 5.0% versus 0.8% for the high IR dataset, 11.0% versus 1.1% for the medium IR dataset and 14.9% versus 1.3% for the low IR dataset. Moreover, both the mean amount and the standard deviation of misclassified `Incident` alerts is also about the same, indicating that the number of alerts investigated on the FM control datasets is smaller than on the tuned datasets. Since the relaxed F1-score punishes rejected alerts, the lower amount of rejected `Non-Incident` alerts explains why the relaxed F1-score is higher on the FM control datasets. The runs of DeepCASE on these control datasets show that DeepCASE performs better on datasets that are largely repetitive in nature.

TABLE 7. AGGREGATED CONFUSION MATRIX FOR THE UNFILTERED AND TUNED DATASETS.

Dataset	Predicted True	Not Classified		Non-Incident		Incident	
		μ	σ	μ	σ	μ	σ
Unfiltered	Non-Incident	104133	5369	19294023	5235	183	159
		0.5%		99.5%		0.0%	
	Incident	610	1	3	1	2	1
High IR		99.2%		0.5%		0.3%	
	Non-Incident	171200	19912	3202255	30326	21643	24987
		5.1%		94.3%		0.6%	
Medium IR	Incident	606	0	7	2	3	2
		98.4%		1.1%		0.5%	
	Non-Incident	23855	699	193760	699	57	0
Low IR		11.0%		89.0%		0.0%	
	Incident	281	30	0	0	334	30
		45.7%		0.0%		54.3%	
	Non-Incident	11642	275	66395	275	61	2
		14.9%		85.0%		0.1%	
	Incident	174	11	0	0	441	11
		28.3%		0.0%		71.7%	

TABLE 8. AGGREGATED CONFUSION MATRIX FOR THE FILTERING METHOD (FM) CONTROL DATASETS.

Dataset	Predicted True	Not Classified		Non-Incident		Incident	
		μ	σ	μ	σ	μ	σ
High FM	Non-Incident	27665	3407	3366150	3086	1407	2681
		0.8%		99.1%		0.1%	
	Incident	608	2	4	1	2	2
Medium FM		99.0%		0.7%		0.3%	
	Non-Incident	2321	174	215473	174	1	0
		1.1%		98.9%		0.0%	
Low FM	Incident	311	52	1	1	302	53
		50.7%		0.1%		49.2%	
	Non-Incident	1035	129	77160	182	28	56
		1.3%		98.6%		0.1%	
	Incident	167	33	0	0	447	33
		27.2%		0.0%		72.8%	