IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.DOI

Severity-oriented multiclass Drone Flight Logs Anomaly Detection

SWARDIANTARA SILALAHI[1], (Student Member, IEEE), TOHARI AHMAD[1], (Member, IEEE), HUDAN STUDIAWAN[1], (Member, IEEE), EIRINI ANTHI[2], LOWRI WILLIAMS[2]

[1]Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, 60111, Indonesia
[2]School of Computer Science and Informatics, Cardiff University, Cardiff CF24 4AG, United Kingdom

Corresponding author: Tohari Ahmad (e-mail: tohari@if.its.ac.id)

This work was supported in part by the Institut Teknologi Sepuluh Nopember (ITS); in part by the Pendidikan Magister Menuju Doktor untuk Sarjana Unggul (PMDSU); and in part by the Peningkatan Kualitas Publikasi Internasional (PKPI) Scholarship from the Ministry of Education, Culture, Research and Technology, Indonesia. This work was conducted by Swardiantara Silalahi whilst visiting the School of Computer Science and Informatics, Cardiff University.

ABSTRACT The availability of log data recorded by computer-based systems such as operating system and network logs, makes it possible for the stakeholder to look after the system for monitoring, evaluation, and improvement purposes. If an incident happens to the system, the log is the first and most important artefact to recover so that investigations may be performed to gather an understanding of why such incidents may have occurred. Log-based anomaly detection is one of the common approaches to uncovering incident scenarios and finding the root cause of such incidents. In the context of drone flight, incidents reported in logs include errors during take-off, flight range issues, and cancellations of actions. Existing studies employ sequence anomaly detection to check whether an event during a drone flight is anomalous. It needs several preceding events and includes deciding if the following event is legitimate or malicious. However, one single log record can have no relationship to other log events and be malicious at the same time. Thus, several studies explored point anomaly detection, where one log record is the only feature needed. Dividing the anomalies into two categories can be overwhelming as the number of logs generated by a system is large. At the same time, it can be helpful to separate critical anomalies from the less severe ones. Therefore, this study proposes **DroLoVe**, a severity-oriented multiclass anomaly detection approach for drone flight log data. In accordance with the dataset characteristics, where the samples from different severity levels share common features, this paper employs a multitask-based label vector representation to train deep neural network models. After an extensive experiment on several baselines, the proposed scenario outperforms other models from existing studies with promising results. The proposed label's representation improves the prediction confidence score on various encoder types with 8.6% and 1.8% from focal and cross-entropy scenarios on average, respectively.

INDEX TERMS Anomaly Detection, Digital Forensics, Drone Forensics, Multitask Learning, Transformer Encoder, Information Security

I. INTRODUCTION

The availability of digital data produced by computer systems continues to increase exponentially. It is followed by the advancement in many research areas to make use of the data, such as natural language processing, which is used to analyse textual data. This type of data can be found in several contexts stored in computer storage devices, such as runtime logs that are constantly generated during the device's operational period. The information recorded in log data is highly valuable for many purposes, including running system monitoring, running process conformance checks, and overall system evaluation [1].

The use of log data is critical when incident cases occur, as the empirical events and incident scenarios can be discovered by analysing the log artefacts. Assuming that the integrity of the log is guaranteed, log data is one of the artefacts with the highest priority for investigating various types of incidents, such as collision, crash landing, cyberattack, payload delivery issue, and weather-related incidents [2]. It is exemplified by the research effort that has been made by the community

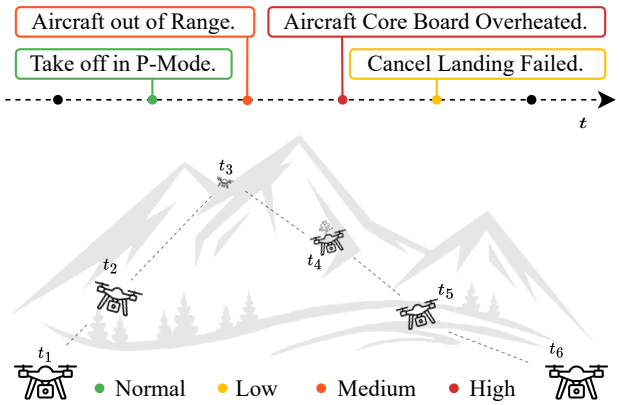


FIGURE 1: An illustration of a flight scenario where several anomalous events with different severity levels happen.

on log-based anomaly detection using various approaches, including machine learning- [3], [4], deep learning- [4]–[6], and graph-based methods [7], [8].

Different types of systems generate log records in different formats and structures, which are strongly affected by the log statements or templates normally used in each of the systems. Certain systems use highly technical log statements, which consist of domain-specific log elements. For instance, operating system logs have common elements such as host names, Internet Protocol (IP) addresses, protocols, ports, and messages [9]. Thus, analysing a particular system log may need different techniques from one type to another.

Generally, log-based anomaly detection can be performed using either point, collective, or contextual approaches. In the log-based anomaly detection literature, most of the existing studies propose either a contextual or collective approach, where the decision is taken by examining a sequence of log events. However, detecting abnormalities in log data can be performed using a point-based approach, where one single log record is the only feature needed. Therefore, several studies explored utilising a point-based approach to detect anomalous events on logs data, such as in operating systems [9], [10], drone devices [11], and distribution systems [9], [10]. In the drone context, a log message contains a description of occurring events which are triggered by various components such as sensors, peripherals, and firmware. Analysing the log message means analysing events from all aspects of the drone including sensors, components and features [2].

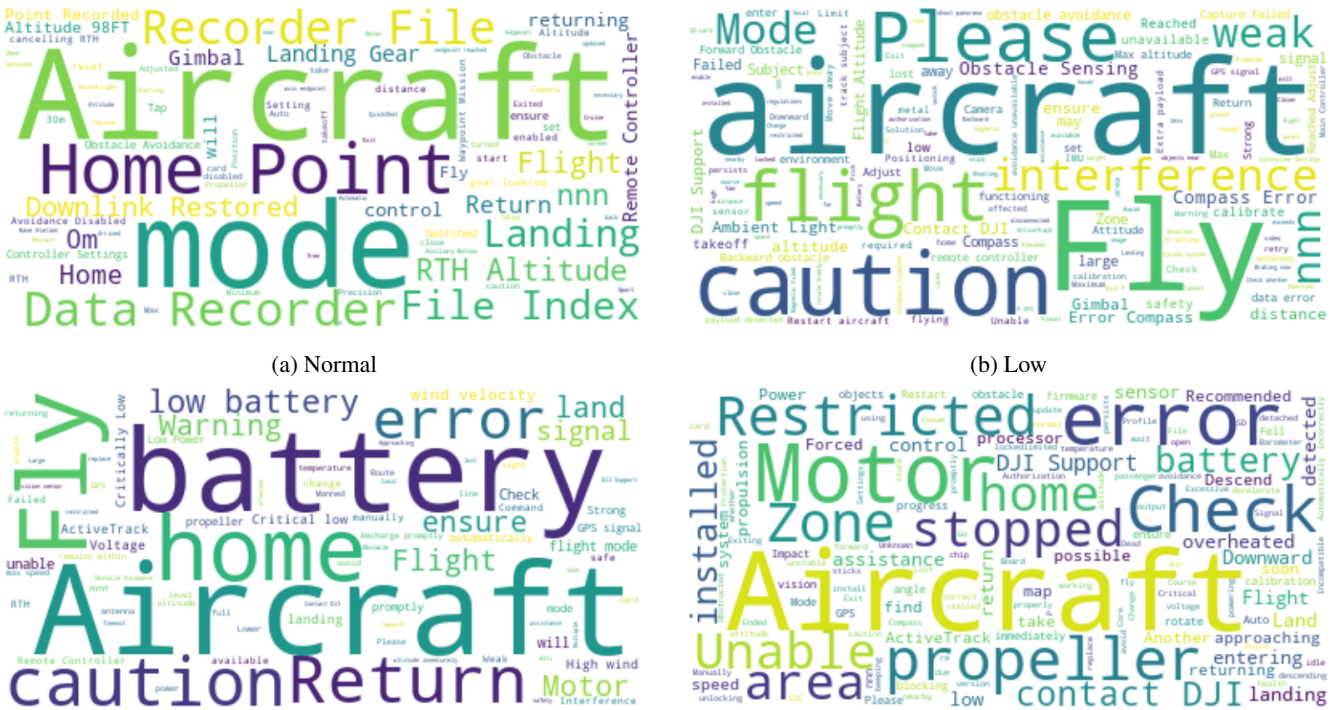
Typically, existing log-based anomaly detection studies classify the log events into two categories: normal and anomaly [3]. This is helpful in an online setting, where the number of detected abnormalities in a certain period of time is considerably small. However, in a digital forensic setting, where the detection is performed on all collected artefacts which might be large in size, the number of the abnormalities is likely to be large [14]. Thus, analysing the detection result in a binary setting is impractical for the investigator. Moreover, out of all detected peculiarities, there might be several negligible ones with less severe impacts on the system or less likely to be related to an incident. Extending binary to multiclass anomaly detection can provide more detailed and contextual detection results to assist in an investigation and analysis [15]–[17]. Considering the severity levels of an anomaly, an investigator can adjust the analysis objectives targeting a certain level only, depending on the needs and cases.

As shown in Fig. 1, the drone experiences several events during a flight. When analysing the flight logs from a forensic perspective, distinguishing the severity levels can help the investigator pinpoint critical anomalies to less severe ones. The challenge is that even though the severity level is different, the samples share a common word or even phrase, as depicted in Fig. 2. It makes it challenging to build a point anomaly detection model that depends on the semantics of the log message only. Fig. 3 shows the visualisation of the semantic feature vectors of the samples in the drone flight log datasets obtained from the pre-trained Bidirectional Encoder Representations from Transformers (BERT) [12] after undergoing a dimensional reduction using the t-SNE [13].

This paper is a further experiment of the previous initial study in multiclass anomaly severity detection on drone flight logs data [18]. It is strongly inspired by the work in [9], [10] where detecting anomalies on logs data is seen as a sentiment analysis task. In this work, the distinctive feature lies within the employment of a sequence classification model with a multitask label to train a detection model for better performance. An extensive experiment is conducted on several different encoders commonly used in log-based anomaly studies, including long short-term memory (LSTM), gated recurrent unit (GRU), transformer, and fine-tuned large language model (LLM). To provide an objective performance comparison of the proposed framework, several baselines are constructed, including those that are proposed in published works.

This paper proposes a transformer encoder-based log-based anomaly detection optimised using multitask label representation to help the model learn from the overlapped features shared by the samples from different severity levels. A domain-specific decoding procedure is also proposed to infer the prediction result. In-depth analyses are performed to evaluate the proposed framework thoroughly. The main **contributions** of this paper are summarised as follows:

1) Propose **Dro**ne **Lo**g Sev**e**rity (DroLoVe), a severity-oriented multiclass anomaly detection approach on drone flight log data.
2) Propose a data-driven label's vector representation and a severity-oriented predicted label's decoding procedure to perform multiclass anomaly severity detection within drone flight log messages.
3) Provide in-depth discussions and analyses of the model's performance supported by an extensive experiment on a large number of hyperparameter search



(a) Normal (b) Low

(c) Medium (d) High

FIGURE 2: A word cloud visualisation to see that the samples from different severity levels share common words and phrases.

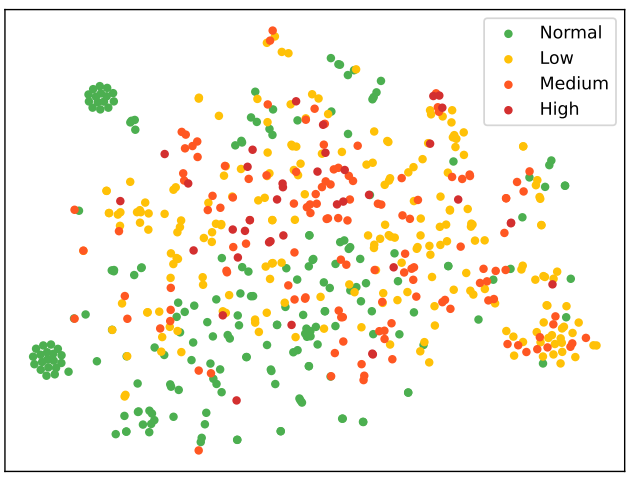


FIGURE 3: A 2D visualisation of the dataset using BERT [12] and t-SNE [13]. It shows that the samples are not linearly separated; samples belong to different classes are close to one another.

spaces.

4) Open the experimental code and results publicly available on a GitHub repository[1] to promote transparent, verifiable, and reproducible research.

The remainder of the paper is structured as follows: Section II discusses relevant previous studies related to log-based anomaly detection and how to deal with class imbalance problems in log-based anomaly detection task, the proposed framework is explained in detail in Section III, Section IV provides the experimental setup, followed by Section V where the experimental results along with comprehensive analyses and discussions are presented. The conclusion and future works are elaborated in Section VII.

[1]https://github.com/swardiantara/DroMoLog

II. RELATED WORK

This section discusses closely relevant prior studies on log-based anomaly detection and different approaches to handle class imbalance problems.

A. ANOMALY DETECTION ON LOGS DATA

Log data recorded by a system store valuable information that can be utilised to perform analysis for investigative purposes, such as anomaly detection which is commonly referred to as log-based anomaly detection. Recently, with the advancements in Machine Learning (ML) and Deep Learning (DL) models, many studies proposed ML-based [3], [4], DL-based [4]–[6], and graph neural network (GNN)-based [7] solutions for detecting abnormalities in log data. Among the published studies, various modelling techniques were used, including point, contextual, and collective. These modelling methods refer to the input needed by the model to perform anomaly detection [9]. In a point setting, the detection is performed against individual log records. The model depends solely on the features from a single log entry. On the contrary,

TABLE 1: Summary of Related Works in Log-based Anomaly Detection

Ref.	Parsing	Pre-process	Input	Model	Handle CIP	Multi Class
DeepSyslog [19]	✓	Cleansing	Sequence	LSTM	✗	✗
DronLomaly [20]	✗	Normalisation	Sequence	LSTM	✗	✗
LayerLog [21]	✗	Cleansing	Sequence	LSTM	✗	✗
LogEncoder [22]	✓	✗	Sequence	BiLSTM	✗	✗
LogGraph [8]	✓	✗	Sequence	GNN	✗	✗
SwissLog [23]	✓	✗	Sequence	BiLSTM	✗	✗
Loader [24]	✗	✗	Sequence	Transformer	✗	✗
NeuralLog [25]	✗	✗	Sequence	Transformer	✗	✗
Pylogsentiment [9]	✓	✗	Point	GRU	Tomek Link	✗
SentiLog [26]	✗	Cleansing	Point	BiLSTM	✗	✗
TransSentLog [10]	✓	✗	Point	Transformer	Tomek link	✗
DroLoVe (Ours)	✗	✗	Point	Transformer	Loss Weight	✓

contextual and collective settings utilise a group of log events to detect the presence of abnormalities.

Common problems encountered in log-based anomaly detection include the unstructured nature of log messages, each system having its own log characteristics, being less human-readable, and containing many special or technical terms [22], [23]. To deal with these problems, a typical ML/DL-based log-based anomaly detection comprises several stages, i.e., log cleansing, log parsing, feature extraction, model training, model testing, and model evaluation. The role of log parsing in log analysis and anomaly detection has been critical to perform both online and offline detection [27]. Performing log parsing aims to extract the core features of logs and reduce the noise. However, employing parsing can remove valuable information within the log messages [25]. To prevent parsing errors from being propagated to the next detection phase, utilising a contextual embedding, such as BERT, to extract the semantic features of logs data without performing parsing can be a solution [21]. Therefore, the whole features extracted from each log record are preserved as it is. Nevertheless, similar but contradicting log events ended up having features that are close to one another in the latent space. Dealing with such an issue, Qi et al. [22] propose a contrastive-based approach which consists of a representation learning model to provide a decent input to a one-class classifier to distinguish the normal from the abnormal log samples. Instead of removing the parameter values in a log message when performing log parsing, the information can be used as an additional feature along with the metadata of the logs to improve the model's performance [19].

Log-based anomaly detection has been applied widely to various systems, such as operating systems [9], [10], parallel file systems [26], drones [11], [18], [20], internet of things [28], and industrial control systems [29]. Among these studies, a sequence-based approach is the common modelling technique used, where the detection is observed on a collection of log events. For that reason, a recurrent-based network is employed to capture the sequential dependency and relationship between the log events' occurrences. For instance, an LSTM [22] and GRU [9] model is used to capture the contextual features of the log sequence. The same approach can also be used in a point-based setting, where the sequence of words or parameters in a log message is the features [9]. Since part of the log record contains a human-readable message, the transformer model can be utilised to extract the semantic information between the words and parameters within a log record to perform point anomaly detection [10], [24]. In this case, given that drone log data also includes natural language, a point-based anomaly detection approach is adopted in this paper. Table 1 presents the summary of the previous related works.

B. CLASS IMBALANCE PROBLEM (CIP) IN LOG-BASED ANOMALY DETECTION

It is assured that in log-based anomaly detection, the number of anomalous samples is significantly less than the normal ones. Employing a supervised-based technique is prone to bias, as the model tends to learn from the majority samples. To overcome CIP, several studies proposed data-level and algorithm-level solutions [30]. Moreover, in a certain case, there are no anomalous samples available. In this particular situation, a one-class approach can be used to construct a normal baseline model. During the detection, an anomaly score is used to decide if an input event is anomalous based on a threshold value [20], [22].

A practical solution to overcome CIP is by controlling the class distribution in the dataset, either by oversampling the minority class, undersampling the majority class, or adding more data to the existing dataset [31]. Overall, either approach can improve the model's performance, depending on the dataset characteristics [32], [33]. For instance, generating more samples of minority classes using the Synthetic Minority Oversampling Technique (SMOTE) can improve the detection performance of a deep Q network (DQN)-based [34] and deep neural network (DNN)-based [35] models. Instead of duplicating the minority class to add more samples, as in random oversampling, SMOTE uses the k-nearest neighbour as an anchor and creates a new sample that is close to those k samples. It helps the model to learn from minority classes better, instead of from redundant samples produced by random oversampling [35]. However, in other

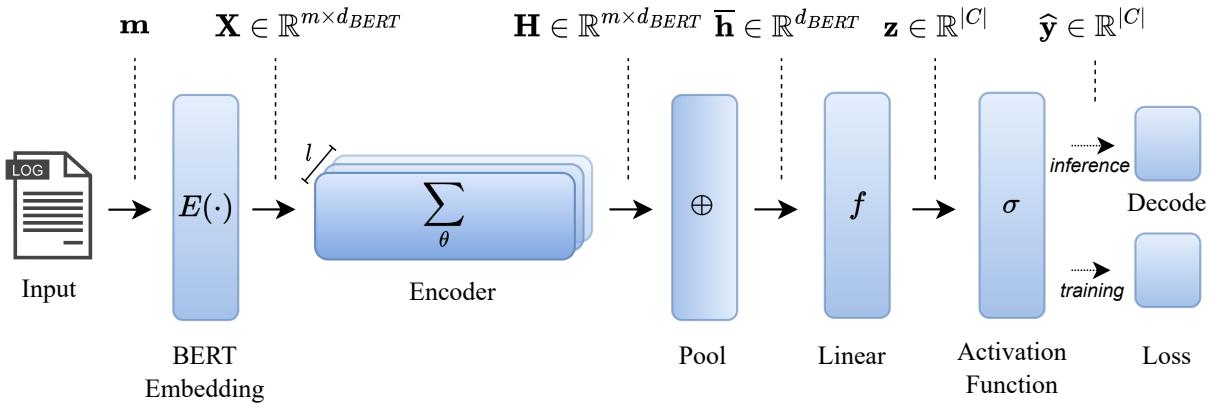


FIGURE 4: The proposed approach overall architecture.

cases, normal samples can be unnecessarily abundant, where removing a certain amount of samples does not reduce the informational value from the dataset. Random undersampling (RUS) is a simple way to eliminate several majority samples to balance the class distribution [36]. Instead of picking random samples to remove, Tomek Link can be used to choose majority samples that are close to minority samples to remove. It can increase the class separability of the dataset, which then helps the model to distinguish between the normal and anomalous events and increases the model's performance [9]. Performing minority oversampling and majority undersampling simultaneously also makes it possible to balance the class proportion, which then helps increase the model's performance [37].

Overcoming CIP can also be achieved by using a method-level solution, which revolves around selecting an appropriate model or designing a training procedure. For instance, Qi et al. [38] utilised a bidirectional generative adversarial network to obtain the reconstruction loss and discrimination loss as the features for an n-stacked ensemble classifier to perform anomaly detection., which resulted in an increase in recall. Ensemble models have been shown to perform better than a single classifier, exemplified by [39], who proposed an isolation forest (IF)-based method to perform anomaly detection with various contamination ratios. Similar to the ensemble, training two detection models for detecting the normal and anomalous events separately can reduce false positive cases [40]. A common challenge in using ensemble-based models is how to perform final predictions, considering each sub-model has its own predictions. Therefore, employing self-supervised contrastive learning can be a solution to pre-train a model that can produce distinct features of normal to abnormal samples. Using the decent features from the pre-trained model, a standard clustering is performed to decide if a test event is anomalous based on the Mahalanobis distance score with respect to a threshold value [41]. Clustering can also be used to estimate the unlabeled samples' label probability, which is then used to train a discriminative model. A semi-supervised approach also shows a positive impact on the model's performance [42].

Unlike the previously discussed studies, inspired by an existing study from another domain [43], this paper employs loss weighting to train a neural model using an imbalanced dataset. Performing data augmentation at the message level to overcome CIP is impractical in this study as the augmented messages do not exist in the actual scenario. Therefore, this paper explores the effect of using various class weighting schemas and loss functions.

III. PROPOSED FRAMEWORK: DROLOVE

This section presents the proposed method towards a severity-oriented multiclass anomaly detection approach for drone flight log data. The overall flow of the proposed framework is depicted in Fig 4. The following section describes the approach in more detail.

A. OVERVIEW

Performing point log-based anomaly detection that takes human-readable log messages as input is similar to conducting sequence or text classification. As mentioned in Section I, this task may be interpreted as a sentiment analysis task that aims to detect negative sentiments within log messages. Taking an n-length log message $\mathbf{m} = [w_1, w_2, ..., w_n]$ from the dataset $\mathbf{D} = \{(\mathbf{m}_i, c_i)\}_{i=1}^{|\mathbf{D}|}$ as the input, where c is the label class name, BERT is used to tokenise the input into a fixed length sequence with m maximum length and retrieve the contextual embeddings, resulting in an input matrix $\mathbf{X} \in \mathbb{R}^{m \times d_{BERT}}$ which is then paired with the encoded label $\mathbf{y}$. Thus, the numerical input becomes $(\mathbf{X}, \mathbf{y})$. This study aims to train a neural network model $\mathcal{M}_\theta$ to classify the message $\mathbf{m}$ into one of the predefined classes $c \in C = \{\texttt{high}, \texttt{medium}, \texttt{low}, \texttt{normal}\}$, or can be written as $\mathcal{M}_\theta : \mathbf{m} \rightarrow c$.

Based on past literature discussed in Section II, processing the sequence of words and performing classification based on the sequence-level features is best performed using modern

deep learning models such as LSTM, GRU, and Transformer. In this study, these models are used in the experiment to identify the best-performing one. The encoder takes the input matrix $\mathbf{X}$ to learn the contextual dependencies among the tokens in the input sequence, yielding the encoder hidden states $\mathbf{H} \in \mathbb{R}^{m \times 768}$, as the BERT-base model produced a 768-dimensional vector. Note that the encoder can be stacked in layers depending on the needs. In this study, the layer's number varies between one and three to prevent the model from being too complex.

Before passing the encoder's hidden states to the final linear classifier, a pooling is performed to aggregate the features from each token within the sequence to get the final sequence-level representation. To this end, various pooling techniques are used: maximum, average, CLS (Classify token), and *last* as depicted in Fig. 5. Max and average pooling are performed at an element-wise manner, meaning that each of the token's vector elements in the corresponding position are aggregated. CLS is a special token from the BERT pre-training task used to represent the sequence features. Pooling CLS means taking the CLS' vector representation as the final feature. As for the *last* refers to the last token's representation in the sequence; this only applies to LSTM and GRU models. This aligns with the nature of the recurrent model where the last token's hidden state is considered to be the sequence representation. When the model employs bi-directionality, the last token's forward hidden state and the first token's backward hidden state are concatenated to form the same size of the final hidden state as the unidirectional one. The pooling layer takes the hidden state $\mathbf{H}$ as the input and resulting the vector $\bar{\mathbf{h}} \in \mathbb{R}^{768}$.

The next step after performing pooling is to feed the vector $\bar{\mathbf{h}}$ to the linear layer, yielding the unnormalised logits $\mathbf{z} \in \mathbb{R}^{|C|}$, where $|C|$ denotes the number of target class in the dataset. During the training phase, these vectors are used to compute the loss and update the model's parameter after passing through a normalisation function. For the typical one-hot encoding label's representation, the standard cross-entropy is used to compute the loss, as defined in the following equation:

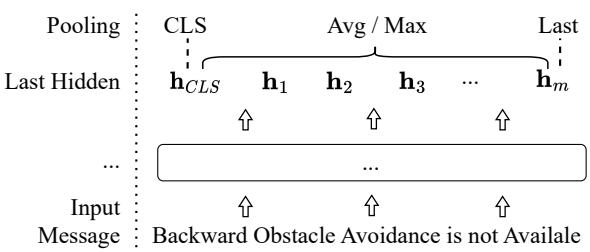


FIGURE 5: An illustration for the pooling mechanism used in the experiment. Note that the *last* pooling is only applicable to the LSTM and GRU encoder.

$$\mathcal{L}^{CE} = -\sum_{i=1}^{|C|} y_i \cdot \log(\hat{y}_i) \tag{1}$$

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z}) \tag{2}$$

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_j^{\dim(\mathbf{z})} \exp(z_j)} \tag{3}$$

where $\mathbf{y}$ and $\hat{\mathbf{y}}$ are the true label vector and the prediction probability distribution, respectively. The prediction probability $\hat{\mathbf{y}}$ is obtained from Eq 2, where each of the element in $\hat{\mathbf{y}}$ is computed using Eq 3.

B. HANDLING CLASS IMBALANCED PROBLEM

One of the common problems when training a neural network is when a dataset has an imbalanced proportion between the classes, which happens to be the case in this study. The distribution of the sample in the dataset is shown in Table 2. As discussed in Section II-B, several techniques can be used to deal with this situation, either by balancing the sample distribution or incorporating class weights during the model's training [30]. In this study, it is impractical to perform oversampling on the dataset, especially in the form of natural language, as is the case in other domains where techniques such as synonym replacement or structure rearrangement are applied. This is due to the nature of the domain, where oversampling the drone flight log messages does not reflect the real-world case and situation. Therefore, incorporating a weighting schema during the training is a feasible option.

Preventing a neural model from learning from the majority of samples can be achieved by weighting the loss of each class based on the frequency distribution. Therefore, the importance of the minority class is considered equal [43] to the majority class. Incorporating a class weight into the cross-entropy loss as defined in Eq 1 where weighting the per sample loss is written as the following:

$$\mathcal{L}^{CE^*} = -\sum_{i=1}^{|C|} \alpha_i \cdot y_i \cdot \log(\hat{y}_i) \tag{4}$$

where α_i denotes the weight for the i-th class. Another way of performing weighting to loss function is to penalise the model's prediction with a low confidence score, called Focal loss [43]. This is achieved by transforming the ratio of the loss value between high-confidence and low-confidence prediction probabilities. Focal loss is computed using the following equation:

$$\mathcal{L}^{Focal} = -\sum_{i=1}^{|C|} \alpha_i \cdot (1 - \hat{y}_i)^{\gamma} \cdot \log(\hat{y}_i) \tag{5}$$

where γ is a hyperparameter to control the range of the prediction probability value being penalised. Increasing $\gamma > 0$ weakens the relative loss from samples with high prediction confidence scores. Therefore, the model is forced to focus more on hard-to-classify samples [43].

	One-Hot Encode	Multitask-110
High	[0 0 0 1]	[0 0 1 1]
Medium	[0 0 1 0]	[0 1 1 0]
Low	[0 1 0 0]	[1 1 0 0]
Normal	[1 0 0 0]	[1 0 0 0]
	(a)	(b)

FIGURE 6: Proposed label's vector representation based on the dataset characteristics.

Computing the class weight can be challenging, as the importance of a particular class could be vague relative to the other classes in the label set. A common practice is using frequency-based weighting, where the class weight is computed based on the frequency distribution of the class in the dataset. In this study, three class weights are explored and used to train all encoder types: uniform, inverse, and balanced. Uniform refers to equal weighting on all classes, which means no weighting is used. While inverse and balanced can be calculated using the following equations:

$$\alpha_c^{inv} = \left(\frac{|\mathbf{D}_c|}{|\mathbf{D}|}\right)^{-1} \tag{6}$$

$$\alpha_c^{bal} = \frac{|\mathbf{D}|}{|C| \cdot |\mathbf{D}_c|} \tag{7}$$

where $|\mathbf{D}|$ denotes the total samples in the dataset and $|\mathbf{D}_c|$ represents the number of the samples belong to class c with $\mathbf{D}_c \subset \mathbf{D}$.

C. SEVERITY-ORIENTED LABEL'S VECTOR REPRESENTATION

Training a neural network to perform a multiclass classification task typically converts the class names into a one-hot encoding vector, where each vector element represents a particular class which relies on an assumption that the samples from different classes are mutually exclusive [44]. As discussed in Section I, and shown in Fig. 2 and Fig. 3, the samples from distinct classes in the dataset are instead mutually inclusive. Therefore, this study proposes a label's vector representation inspired by the multitask learning paradigm. Fig. 6 shows the proposed label along with the standard one-hot encoding. The label relies on the assumption that samples belonging to a higher severity level with one class share common low-level features with samples in a lower severity level. We call it a severity-oriented label as the model is trained on two alternative labels instead of one exclusive label only. Therefore, in case of misclassification, the prediction is expected to fall one level under the true label.

Aligning to the nature of the label's vector representation, the loss function used in the training is log loss for multiclass classification, which can be computed using the following equation:

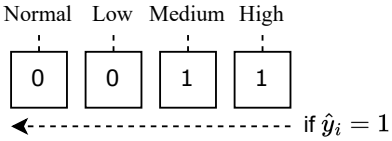


FIGURE 7: Severity-oriented decoding process of the predicted label's vector into the class name which is done in a backward direction.

$$\mathcal{L}^{Log} = -\sum_{i=1}^{|C|} \alpha_i \cdot [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{8}$$

where α_i is the same term as in Eq 4. With this loss function, the model is forced to learn from the shared features among the two neighbouring classes. This assumption originated from the multi-object detection task, where a single image contains multiple objects to detect and those different objects share low-level features. Therefore, predicting a certain object benefited from other's object features within the same input sample [45]. The same assumption underlies the design for the label's vector representation in this study.

Inferring the prediction result with the proposed label is different from the typical one-hot encoded label. The index of the element with the highest probability value that is obtained from $\arg\max$ is used as the final prediction. Then, the integer index is decoded back to the class name using the same function used to convert the class name into the integer index in the label encoding process. Conversely, each vector element in the proposed label represents a particular class, as shown in Fig. 6. Therefore, to normalise the logits from the model into probability, sigmoid is used on each of the prediction vector elements and is written as the following:

$$\hat{y}_i = \frac{1}{1 + \exp(-z_i)} \tag{9}$$

where $\hat{y}_i$ is the prediction probability for the i-th class and $\hat{y}_i \in [0, 1]$. To get the prediction label's vector, a prediction threshold $\lambda = 0.5$ is applied, making the element with a value more than or equal to the threshold equal to 1, otherwise it is 0. Finally, severity-oriented decoding is used to get the label class name back by checking the prediction label's vector from the last element and moving backward to the first element, as illustrated in Fig. 7. If the current element is marked as 1, the class name in that position becomes the predicted class.

IV. EXPERIMENTAL SETUP

This section provides details about the dataset used in the experiment presented herein, as well as describes the baseline models used for performance comparison, the experiment environment, and the hyperparameter settings used.

TABLE 2: Summary of the distribution of the class in the dataset

Dataset	Split	High	Medium	Low	Normal	Total
Filtered	Train	30	104	173	161	468
	Test	8	26	43	40	117
	Total	38	130	216	201	585
Unfiltered	Train	32	632	433	696	1,793
	Test	8	159	108	174	449
	Total	40	791	541	870	2,242

A. DATASET

In this study, one of the artefacts of drone devices, the flight log message, is used as the source of the evidence to perform multiclass anomaly severity detection. These human-readable messages, such as "Aircraft Core Board Overheated" and "Compass error, calibration required", can be found in most of DJI-make's flight logs in three columns: warning, tip, and message. Originally, the flight log was in encrypted .CSV files [46]. Thus, the drone phantom help[2] is needed to decrypt the flight log files and then extract the contents. The dataset used in this study is also used in [18], where the log messages are collected from two sources: VTO Labs [46] and Drone Wiki [47]. Instead of taking the average over 5 folds, one of the folds (5^{th} fold) is directly used as the standard deviation signified the performance evaluation on 5 different folds is insignificant [18]. The filtered dataset refers to the log messages after performing a unique filtering process, while the unfiltered is the actual collection of drone flight log messages gathered from the two sources mentioned previously. The dataset is split into training and testing with an 80:20 ratio. The summary of the per-class distribution in the dataset is presented in Table 2.

B. BASELINES

In this study, an extensive experiment with various types of encoders is conducted to construct strong baselines for performance comparison. Ensuring the objectivity of the comparison, these baselines' hyperparameters are fine-tuned using grid search on a finite search space, as shown in Table 3. The details of the baseline construction are described in the following subsections.

1) Baseline from Diverse Suitable Encoders

The distinction between the baseline and the proposed method is based on the pair of label's vector representation and the loss function used during the training. Two baselines are defined: one-hot encoding with cross-entropy loss and one-hot encoding with focal loss. The proposed scenario employs multitask encoding with log loss. Using these three scenarios, a significant number of models using the following encoders are trained:

1) **None** refers to fine-tuning the BERT's [12] parameter on the dataset. Different pooling mechanisms and class weighting strategies are explored whilst using this baseline, resulting in 54 scenarios.
2) **Transformer** implies the transformer encoder submodule in the transformer architecture, proposed in [48]. From the search space in Table 3, bi-directionality is the only irrelevant hyperparameter during the grid search, generating 486 scenarios.
3) **LSTM** [49] and **GRU** [50] allude to recurrent neural network families that are common in sequence classification tasks. In these two models, the number of attention heads is the only irrelevant hyperparameter during the grid search, yielding 324 scenarios each.

Therefore, the overall scenarios are 1,188 in total. BERT is chosen as the embedding model for all scenarios as it is widely used in diverse domains and proven to be better compared to other contextual embedding models [51].

2) Baseline from Previous Works

Among the relevant published studies in the log-based anomaly detection space, sequence-based is one of the most commonly used approaches. Therefore, there are limited relevant references as this study employs a point-based approach. Below are the relevant baselines from previous research:

1) **Pylogsentiment** [9] is the first study which employs sentiment analysis-based anomaly detection on operating system logs using GRU and GloVe embedding.
2) **SentiLog** [26], similar to [9], use a two-layered BiLSTM and GloVe embedding model to perform anomaly detection on parallel file system logs.
3) **TransSentLog** [10] is a further development of [9] and uses a two-layered transformer encoder which used two attention heads and GloVe embedding along with integrated gradients to add explainability to the trained model.
4) **NeuralLog** [25] is a one-layered transformer encoder-based model trained on various log anomaly benchmarks using 12 attention heads and BERT as the embedding. Contrary to the other three baselines, this study performed the detection on the sequence of log records, instead of point-based.

BERT has been demonstrated to be a better embedding compared to GloVe [51], as BERT produces a contextual feature vector based on the relationship among the words within a sentence. In this study, BERT is used as the embed-

TABLE 3: The search space for the hyperparameter tuning

Variable	Value Space			
	None	Transformer	LSTM	GRU
Loss	Cross-Entropy, Focal, Log			
Num. of Layers	-	1, 2, 3		
Attention Heads	-	4, 6, 8	-	
Pooling	Max, Avg, CLS		Max, Avg, Last	
Class Weight	Uniform, Balanced, Inverse			
Bidirectionality	-		True, False	

[2]https://www.phantomhelp.com/logviewer/upload

TABLE 4: The Best Performing Model for Each Encoder Type and Loss Function Based on Accuracy and F1 Score Tested on the Unfiltered Dataset After Performing Hyperparameter Tuning.

Encoder	Loss	#Layer	#Head	Pooling	Class Weight	Best Epoch	Acc (%)	Pre (%)	Rec (%)	F1 (%)	Conf (μ_σ)
BiGRU*	CE	1	-	Last	Uniform	9	96.429	96.527	96.429	96.470	$0.985_{0.07}$
BiGRU	Focal	2	-	Last	Uniform	12	96.205	96.430	96.205	96.303	$0.968_{0.07}$
BiGRU	Log	3	-	Max	Uniform	4	96.429	96.556	96.429	96.449	$0.984_{0.05}$
LSTM	CE	1	-	Avg	Uniform	12	96.205	96.486	96.205	96.315	$0.995_{0.03}$
LSTM*	**Focal**	**2**	**-**	**Last**	**Uniform**	**10**	**96.429**	**96.604**	**96.429**	**96.498**	$\mathbf{0.965_{0.07}}$
LSTM	Log	1	-	Avg	Balanced	11	96.205	96.535	96.205	96.328	$0.991_{0.05}$
None*	CE	-	-	Avg	Balanced	6	95.759	96.370	95.759	95.985	$0.988_{0.05}$
None	Focal	-	-	Avg	Uniform	15	95.759	95.837	95.759	95.778	$0.974_{0.09}$
None	Log	-	-	Avg	Uniform	7	95.759	95.864	95.759	95.786	$0.993_{0.05}$
Transformer	**CE**	**3**	**6**	**Avg**	**Uniform**	**11**	**96.429**	**96.608**	**96.429**	**96.487**	$\mathbf{0.993_{0.04}}$
Transformer	Focal	3	4	Max	Uniform	10	96.429	96.400	96.429	96.366	$0.967_{0.08}$
Transformer*	**Log**	**1**	**6**	**Avg**	**Balanced**	**7**	**96.875**	**96.856**	**96.875**	**96.851**	$\mathbf{0.995_{0.03}}$

Best-performing model from each loss function
*Best-performing model from each encoder type

ding model when reproducing the baselines' performance. Additionally, hyperparameter tuning is also performed to make the performance comparison as objective and fair as possible.

C. EXPERIMENT SETTINGS

The experiment is conducted on a personal computer equipped with a 16GB VRAM GPU. The method is implemented in Python version 3.10.13 with the help of the Pytorch version 2.0.1 library. The model is trained and tested only once by setting the random seed value to guarantee reproducibility over multiple runs and on different devices. During the experiment, as each encoder type has its hyperparameter, the search space is not the same as one another. The details of the search space are presented in Table 3. Other than the ones listed in the table, the hyperparameter values are set as follows: learning rate is $\eta = 2e-5$, 15 epochs, $\gamma = 2$ [43] for the focal loss, and 8 batch size on train and test.

During the training, the best-performing checkpoint is saved and used in the evaluation. The first criterion for choosing the best checkpoint is by comparing both the accuracy and F1 score of the current execution with the previous best run. If the accuracy and F1 score of the current execution is not higher than the previous score, then the second criterion is to check if the F1 score is improved while the accuracy is stagnant. If the second criterion does not hold, it is observed if the accuracy is improved while the F1 score is stagnant. Other than these criteria, the current execution is ignored. The position of the best checkpoint is recorded as the best epoch for performance evaluation. This value can be affected by the loss computation that uses unnormalised logits during the training. Therefore, an ablation study is performed to investigate the effect.

The accuracy, weighted average precision, recall, and F1 scores are reported as the performance evaluation metrics. Additionally, the mean prediction probability is recorded to measure how confident the model is in predicting the test data on average. To verify the importance of each component in the model, an ablation study is performed on several aspects. To provide an in-depth analysis from a domain-specific perspective, error analyses are also conducted by investigating the the misclassification cases. Finally, the learned representation of the dataset is examined to check if the model successfully learns a decent representation during the training.

V. PERFORMANCE EVALUATION AND DISCUSSION

Following the procedure and details in the previous section, this section reports the experimental results along with in-depth discussions and analyses.

A. BEST PERFORMING SCENARIO

Evaluation Metric: After experimenting with all the designed scenarios, the models' performance is evaluated on the test dataset. The best-performing model from each pair of encoder type and loss function is reported in Table 4 and Table 5 for unfiltered and filtered datasets. Based on the evaluation metrics shown in both tables, the proposed scenario outperforms all baseline scenarios on all encoder types tested on the unfiltered dataset, with an accuracy of 96.875 and an F1 score of 96.851. As for the filtered dataset, the transformer encoder trained on the baseline scenario outperforms the other scenarios on all encoder types with an accuracy of 83.761 and an F1 score of 82.967. The proposed scenario only performs better on the GRU and LSTM encoders compared to the baseline scenarios.

Prediction Confidence: Evaluating a neural model's performance strongly depends on the task and domain problem. In a particular task, accuracy can be the only important metric. However, in some other domains, having a high accuracy does not suffice. For example, in this study, the model needs to be sure when predicting if a log is not an anomaly. To check if a model is certain of the prediction, we can investigate the prediction probability. Therefore, we record the prediction probability of each scenario during the testing, taking the mean (μ) and the standard deviation (σ) to perform the model's prediction confidence analysis. The distribution

TABLE 5: The Best Performing Model for Each Encoder Type and Loss Function Based on Accuracy and F1 Score Tested on the Filtered Dataset After Performing Hyperparameter Tuning.

Encoder	Loss	#Layer	#Head	Pooling	Class Weight	Best Epoch	Acc (%)	Pre (%)	Rec (%)	F1 (%)	Conf (μ_σ)
GRU	CE	3	-	Avg	Uniform	9	81.197	81.176	81.197	80.863	$0.973_{0.07}$
GRU	Focal	3	-	Avg	Uniform	9	81.197	80.738	81.197	80.723	$0.893_{0.11}$
GRU*	Log	1	-	Avg	Uniform	9	82.051	81.950	82.051	81.429	$0.978_{0.06}$
LSTM	CE	3	-	Avg	Uniform	9	82.051	81.621	82.051	81.325	$0.931_{0.11}$
BiLSTM	Focal	3	-	Last	Inverse	12	82.051	81.816	82.051	81.618	$0.869_{0.11}$
LSTM*	**Log**	**2**	**-**	**Last**	**Inverse**	**13**	**82.051**	**81.699**	**82.051**	**81.859**	$\mathbf{0.976_{0.07}}$
None*	CE	-	-	CLS	Balanced	11	80.342	82.251	80.342	80.873	$0.958_{0.10}$
None	Focal	-	-	CLS	Balanced	7	80.342	80.914	80.342	80.602	$0.833_{0.16}$
None	Log	-	-	Avg	Uniform	14	78.632	79.110	78.632	78.846	$0.975_{0.07}$
Transformer*	**CE**	**3**	**4**	**CLS**	**Uniform**	**13**	**83.761**	**84.699**	**83.761**	**82.963**	$\mathbf{0.970_{0.09}}$
Transformer	**Focal**	**1**	**4**	**CLS**	**Inverse**	**12**	**82.906**	**82.663**	**82.906**	**82.315**	$\mathbf{0.876_{0.15}}$
Transformer	Log	2	8	Max	Balanced	11	81.197	80.756	81.197	80.876	$0.968_{0.09}$

Best-performing model from each loss function
*Best-performing model from each encoder type

of the mean prediction confidence is presented in Fig. 8 based on the loss function. From the figure, it can be observed that the confidence score of log loss is significantly higher and more stable than the other two losses, indicating that the proposed label increases the model's prediction confidence overall for encoder types on average.

Convergence Speed: The best epoch in the Table 4 and Table 5 indicate the iteration position when the model reaches the best performance from a total of 15 epochs. It can be used to evaluate the model's training behaviour in different scenarios. From the unfiltered evaluation, the proposed scenario reached the best performance at the 7^{th} iteration, relatively faster than the other two best-performing scenarios, which need 11 and 10 iterations for cross-entropy and focal, respectively. On the contrary, the result of the filtered dataset shows an opposite tendency. The higher-performing scenarios tend to take more iterations than the lower-performing ones. However, the proposed scenario outperforms the other two baseline scenarios on the GRU encoder with the same number of iterations. When using a transformer encoder, the proposed scenario underperforms the other two baselines significantly while taking an almost similar number of iterations. Fig. 9 shows the distribution of the best epoch on each loss function, where log loss tends to take more epochs to reach the best checkpoint.

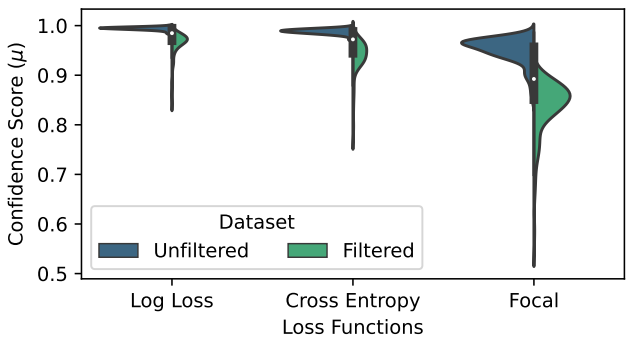


FIGURE 8: The distribution of the mean (μ) prediction confidence score from each loss function. Log loss is significantly higher and more stable than CE and focal on both datasets.

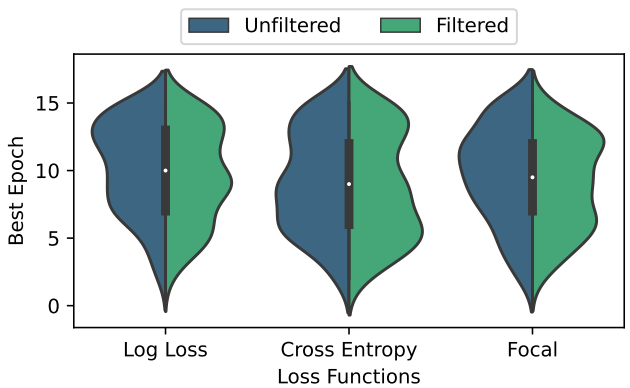


FIGURE 9: The distribution of the best epoch from each loss function. Log loss tends to need more epochs compared to cross-entropy and focal loss.

B. HYPERPARAMETER ANALYSIS ON EACH ENCODER

Considering the total number of experimented scenarios, it is not possible to discuss all the details in this paper. Therefore, we perform a chi-square test to determine which hyperparameter is significant towards the accuracy of the models. Note that this test measures the difference in the mean of accuracy among different groups based on the categorical value in each hyperparameter. Thus, the significant hyperparameter can be different from one encoder to another. Fig. 10 shows the test result, where the heat map colour indicates the statistic test score and the annotation in each cell indicates the corresponding p-value. Chi-square can be computed using the following formula:

$$\chi^2 = \sum \frac{(O_{ij} - E_{ij})^2}{E_{ij}} \tag{10}$$

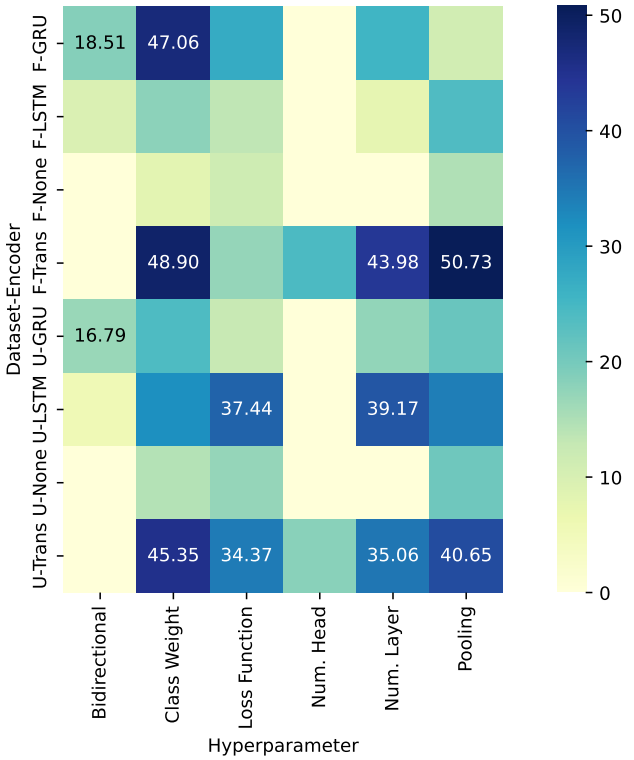


FIGURE 10: A significance test using the chi-square test of independence to check which hyperparameter is significant towards the accuracy. Only those with p-values less than 0.05 are shown.

where O_{ij} and E_{ij} are the observed and expected frequencies in cell (i, j) in a contingency table constructed from each hyperparameter paired with the accuracy. Since the accuracy is a continuous variable, the value is converted into several groups by binning the value into several ranges. Having the chi-square test result presented in Fig 10, it can be seen that the significant hyperparameters differ from one encoder to another. Note that this test does not reflect the direction of the dependency. Instead, it shows which hyperparameter the accuracy depends on. The dependency can be either increasing or decreasing the accuracy. Nevertheless, it can help choose which hyperparameter to modify in the next experiments.

C. COMPARISON WITH STATE-OF-THE-ART MODELS

In this study, several baselines from previous studies are reproduced on the dataset to perform performance comparisons. The reproduced performance from an experimental procedure explained in subsection IV-B2 is presented in Table 6. On both datasets, the proposed scenario outperforms all the baselines with an improvement of 0.423 and 0.665 in the F1 score on the filtered and unfiltered datasets, respectively. Considering that Pylogsentiment used a GRU-based encoder and SentiLog used an LSTM-based encoder, the proposed scenarios on these two encoders consistently outperform the baselines' performance tested on the filtered dataset. Note that we also perform hyperparameter tuning on these baselines to ensure the validity of the performance comparison. As for transformer-based baselines which are NeuralLog and TransSentLog, the proposed scenario achieves lower performance tested on the filtered dataset. However, on the unfiltered dataset, the transformer-based proposed scenario achieves the highest performance. Based on these findings, it is verified that the proposed scenario can improve the detection model's performance.

D. ABLATION STUDY

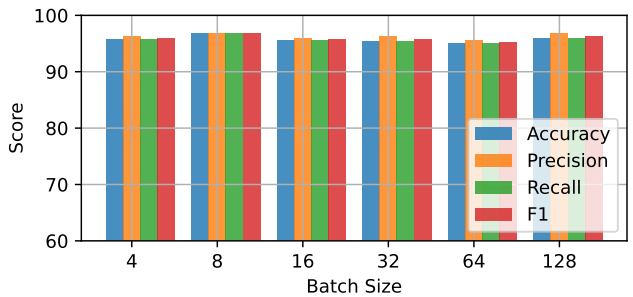
Following the rapid advancement in neural network research, more complex and sophisticated architectures are emerging. A model can consist of layers of components that have a specific role in the learning process. After conducting an experiment on a certain dataset and task, it is crucial to verify and check which component in the model contributes positively to the model's performance. Therefore, we perform an ablation study to explain which part of the proposed approach has a significant impact on the performance. Note that only the best-performing scenario on each dataset is explored. First, we investigate the significance of the CLS token from the BERT embedding. Secondly, we analyse the impact of freezing the BERT's parameter during training. Thirdly, we vary the batch size during the training. Finally, we examine the effect of increasing the prediction threshold used in our proposed approach on the accuracy and F1 scores.

As shown in Table 6, excluding the CLS token's embedding before feeding the input matrix to the encoder decreases the performance significantly. Also, freezing the BERT's parameter during the training caused a striking drop in the performance scores.

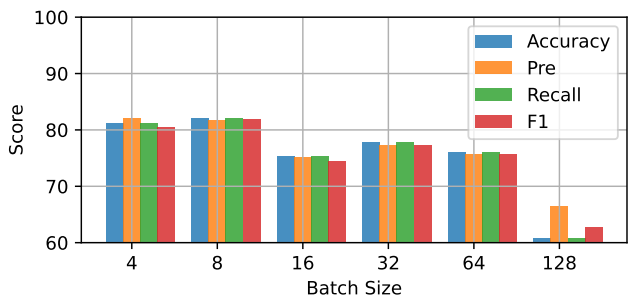
Training a neural model using a batched sample is a common practice to shorten the training time and enforce the model to learn from representative samples. The size of a batch can have an impact on the model's performance. As shown in Fig 11, varying the batch size has a notable impact on the filtered dataset but insignificant on the unfiltered dataset. From the figure, it can be seen that the 8 is the best option to use. During the inference when using multitask encoding and severity-oriented decoding, the threshold plays a crucial role in the prediction evaluation. Fig 12 shows the effect of increasing the threshold on the accuracy and F1 scores of the best-performing model tested on both datasets. An interesting case is shown in Fig 12a when $\lambda = 0.65$, the accuracy reaches 97%, exceeding the best model's accuracy when $\lambda = 0.5$. This happens when the true label is lower than the predicted label, while the prediction confidence of the higher label surpasses the threshold. Thus, following the decoding procedure in section III-C, the predicted label is incorrect. This phenomenon also happens on the filtered dataset when $\lambda > 0.94$. Overall, the decrement in the accuracy and F1 scores is insignificant until $\lambda = 0.9$ and $\lambda = 0.8$ on the unfiltered and filtered datasets, respectively.

TABLE 6: Performance Comparison with Several Baselines from Previous Works

Ref.	Filtered Dataset					Unfiltered Dataset				
	Acc (%)	**Pre** (%)	**Rec** (%)	**F1** (%)	**Conf** (μ_σ)	**Acc** (%)	**Pre** (%)	**Rec** (%)	**F1** (%)	**Conf** (μ_σ)
Pylogsentiment [9]	80.342	81.400	80.342	80.604	$0.944_{0.11}$	95.536	95.575	95.536	95.534	$0.988_{0.07}$
SentiLog [26]	79.487	79.621	79.487	79.458	$0.968_{0.08}$	95.759	95.962	95.759	95.842	$0.993_{0.04}$
NeuralLog [25]	82.051	81.506	82.051	81.436	$0.819_{0.17}$	96.205	96.208	96.205	96.186	$0.985_{0.07}$
TransSentLog [10]	81.197	81.120	81.197	81.127	$0.957_{0.11}$	95.982	96.289	95.982	96.104	$0.989_{0.05}$
DroLoVe (GRU)	82.051	81.950	82.051	81.429	$0.978_{0.06}$	96.429	96.556	96.429	96.449	$0.984_{0.05}$
DroLoVe (LSTM)	**82.051**	**81.699**	**82.051**	**81.859**	$\mathbf{0.976_{0.07}}$	96.205	96.535	96.205	96.328	$0.991_{0.05}$
w/o CLS vector	78.632	77.957	78.632	78.206	$0.978_{0.06}$	95.536	96.047	95.536	95.717	$0.996_{0.03}$
freeze BERT's params	64.103	71.658	64.103	66.186	$0.864_{0.15}$	89.063	94.805	89.063	91.414	$0.976_{0.07}$
DroLoVe (Transformer)	81.197	80.756	81.197	80.876	$0.968_{0.09}$	**96.875**	**96.856**	**96.875**	**96.851**	$\mathbf{0.995_{0.03}}$
w/o CLS vector	78.632	78.491	78.632	78.107	$0.952_{0.10}$	95.759	96.096	95.759	95.877	$0.992_{0.04}$
freeze BERT's params	70.940	70.038	70.940	70.391	$0.845_{0.15}$	93.080	94.284	93.080	93.541	$0.972_{0.08}$



(a) Unfiltered Dataset



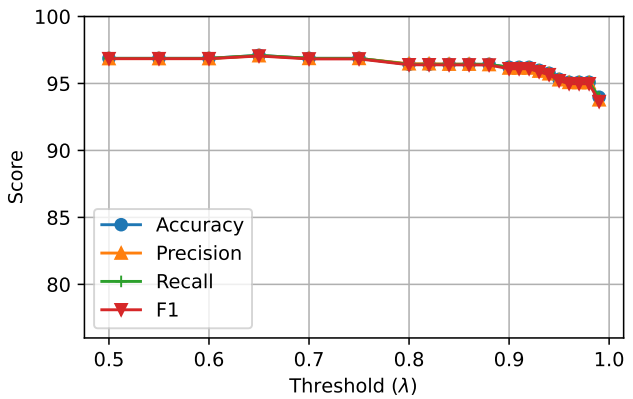
(b) Filtered Dataset

FIGURE 11: Analysis of the effect of different batch sizes on the accuracy and F1 scores.

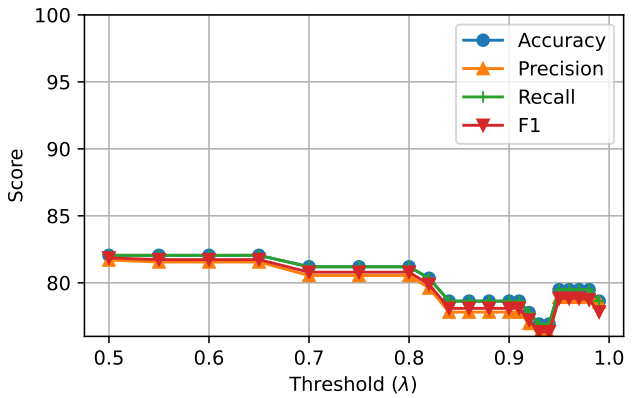
Thus, it confirms the prediction confidence distribution that is depicted in Fig 8.

E. ERROR ANALYSIS

In the previous section, the models' performance has been discussed thoroughly from a quantitative perspective. Here, we investigate the samples from the test set that are predicted incorrectly by the best-performing model from each scenario. In a binary setting, let the Normal class be positive and the Anomaly is negative. Then, False Positive (FP) refers to samples with a true label negative but predicted as positive. At the same time, a False Negative (FN) is a case when a sample belongs to a positive class but is predicted as negative. However, in this study, we define the misclassification cases differently:



(a) Unfiltered Dataset (Transformer)



(b) Filtered Dataset (LSTM)

FIGURE 12: Analysis of the effect of increasing the prediction threshold on the accuracy and F1 scores.

- **False Positive** refers to a case when the true class is higher than the predicted class.
- **False Negative** is a misclassification case where the true class is lower than the predicted class.

Table 7 shows the position of FP and FN based on the above definition in a multiclass confusion matrix.

In an anomaly detection setting, FP is more important than FN since detecting anomalous events as normal is a

TABLE 7: Custom Confusion Matrix for Error Analysis

	Normal	Low	Medium	High	
Normal	TP	FN(1)	FN(2)	FN(3)	True Class
Low	FP(1)	TP	FN(1)	FN(2)	
Medium	FP(2)	FP(1)	TP	FN(1)	
High	FP(3)	FP(2)	FP(1)	TP	
	Predicted Class				

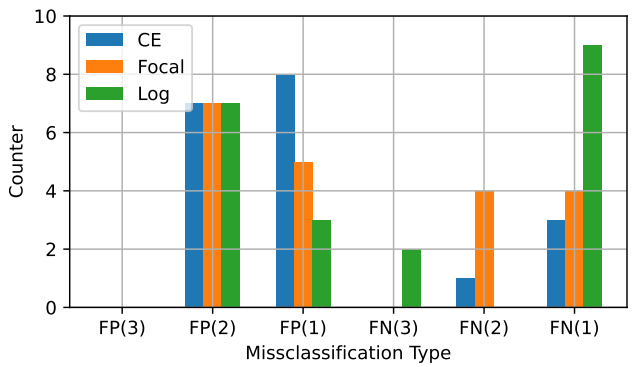


FIGURE 13: The number of misclassified samples from the best-performing model on each loss function tested on the filtered dataset.

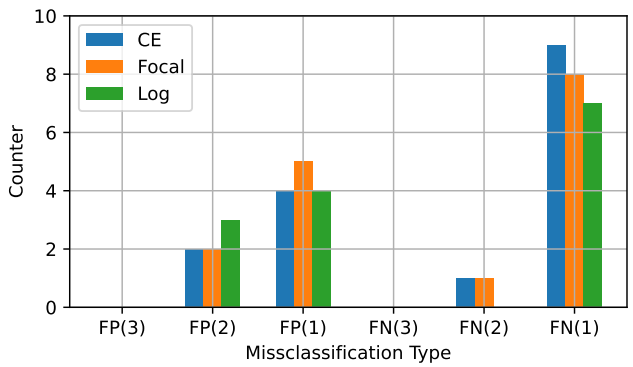


FIGURE 14: The number of misclassified samples from the best-performing model on each loss function tested on the unfiltered dataset.

critical error. Having a high number of FPs can endanger the system with a serious impact, especially for FP(3) cases, which means the true label is High and the predicted label is Normal. At the same time, having a high number of FNs can cause many false alarms, but they are not as critical as FPs. Following the definition in the previous paragraph, we plot the frequency of each misclassification type in Fig 13 and Fig 14. From the figures, it can be seen that the proposed approach performs better on the unfiltered dataset, despite having a high number of FN(1) and FN(3). However, on the filtered dataset, the proposed approach has more FP(2) than the other two scenarios, even though it has a smaller number of FP(1), FN(1), and FN(2).

Other than analysing the misclassified samples, we also analyse the learned representation of the dataset by each of the best-performing models. The filtered dataset is used to investigate the samples' representation plotted into 2D graphs, as shown in Fig 3. From the visualisation, it can be seen that the samples that belong to High class are still close to the Normal class in Fig 15a and Fig 15b. While in Fig 15c, the High class is far from the Normal class but even further from the Medium class. Even though all scenarios result in a well-separated representation, none of the scenarios reflects the nature of the label. One would expect the High class to be far from the Low class, representing the distance between the class' severity.

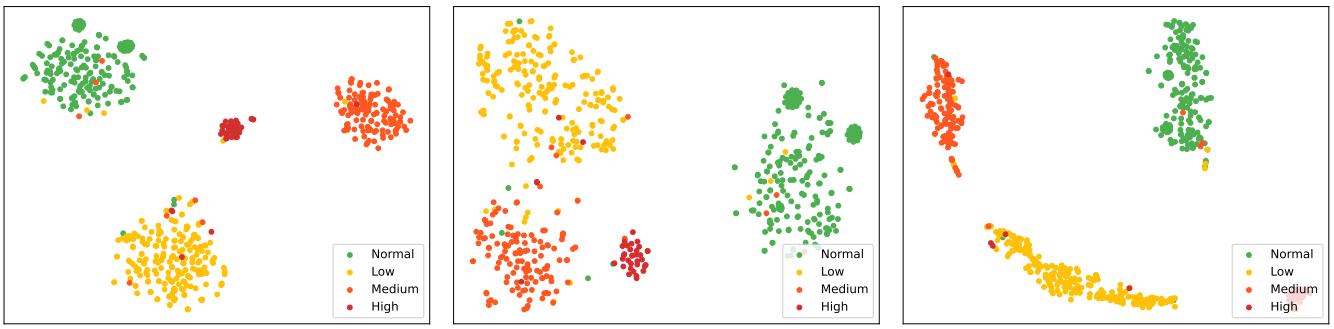
VI. CHALLENGES, LIMITATIONS, AND THREATS TO VALIDITY

In the previous sections, the performance of the proposed framework has been discussed and analysed. In this section, challenges that are encountered during the study are disclosed, along with limitations and threats to validity. As drone forensics is an emerging topic, there are very limited public datasets available. The log messages used in this study are mainly acquired from DJI-made devices. There is yet to come a publicly available dataset of log messages from other drone manufacturers. Several publicly available datasets are mainly about sensor data and multimedia artefacts [52]. While in this study, we solely depend on the human-readable messages generated by the drone during a flight. Given the condition where a small number of unique messages and most of them are acquired from DJI drones, the proposed model's generality remains untested. Considering the performance evaluation score, where the highest accuracy is under 85%, the model needs further improvement so that the validity of the detection can be enhanced so that the model's detection results can be convincing and accountable enough to the investigator to be included in the investigation report.

VII. CONCLUSION AND FUTURE WORK

In this paper, we demonstrate how to train a log-based anomaly detection model that can prioritise the prediction of higher severity anomalies on drone flight logs while increasing the prediction confidence score at the same time. Considering the nature of the dataset, where samples that belong to different severity levels share common features, a multitask label's vector representation along with severity-oriented decoding is proposed.

An extensive experiment proved that the proposed approach is better than the previous work baselines while being inferior to our baseline scenario based on the accuracy and F1 scores. From the anomaly detection perspective, the proposed method achieves higher prediction confidence and can prioritise higher severity levels during the inference on the test dataset. Despite the promising results, the proposed model is tested only on messages acquired from DJI-made devices, leading to untested generality. Moreover, the testing



(a) Cross Entropy (b) Focal (c) Log Loss

FIGURE 15: A 2D visualisation of the filtered dataset representation obtained from the best model from each loss function tested on the filtered dataset. t-SNE [13] is used to reduce the dimension of the embeddings.

set does not reflect an incident scenario, making the resulting model untested in a real-case environment.

Further future studies may include exploring the possibility of doing high-level oversampling to introduce the models with more log message patterns, producing a dataset with incident scenarios to perform case studies for verifying the proposed method's performance and making it publicly available.

DATA AVAILABILITY

The dataset used in the experiment is available on a reasonable request. The code for the experiment, the resulting performance evaluation, the scripts for data analysis, and the figures in this paper are made publicly available on the GitHub[3] to promote transparent, reproducible, and verifiable research.

REFERENCES

[1] W. Aalst, A. Adriansyah, A. Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. B. R.P., P. Brand, R. Brandtjen, J. Buijs, A. Burattin, J. Carmona, M. Castellanos, J. Claes, J. Cook, N. Costantini, F. Curbera, E. Damiani, M. de Leoni, and M. Wynn, "Process mining manifesto," in *Lecture Notes in Business Information Processing*, vol. 99, 2011, pp. 169–194.

[2] E. Mantas and C. Patsakis, "Who watches the new watchmen? The challenges for drone digital forensics investigations," *Array*, vol. 14, p. 100135, 2022.

[3] A. B. Nassif, M. A. Talib, Q. Nasir, and F. M. Dakalbab, "Machine Learning for Anomaly Detection: A Systematic Review," *IEEE Access*, vol. 9, pp. 78 658–78 700, 2021.

[4] X. Zhao, Z. Jiang, and J. Ma, "A Survey of Deep Anomaly Detection for System Logs," in *International Joint Conference on Neural Networks (IJCNN)*, 2022, pp. 1–8.

[5] M. Landauer, S. Onder, F. Skopik, and M. Wurzenberger, "Deep learning for anomaly detection in log data: A survey," *Machine Learning with Applications*, vol. 12, p. 100470, 2023.

[6] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep Learning for Anomaly Detection: A Review," *ACM Computing Surveys*, vol. 54, no. 2, 2021.

[7] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A Comprehensive Survey on Graph Anomaly Detection With Deep Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 012–12 038, 2023.

[8] J. Li, H. He, S. Chen, and D. Jin, "LogGraph: Log Event Graph Learning Aided Robust Fine-Grained Anomaly Diagnosis," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–15, 2023.

[3]https://github.com/swardiantara/DroMoLog

[9] H. Studiawan, F. Sohel, and C. Payne, "Anomaly Detection in Operating System Logs with Deep Learning-Based Sentiment Analysis," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 5, pp. 2136–2148, 2021.

[10] T.-A. Pham and J.-H. Lee, "TransSentLog: Interpretable Anomaly Detection Using Transformer and Sentiment Analysis on Individual Log Event," *IEEE Access*, vol. 11, pp. 96 272–96 282, 2023.

[11] S. Silalahi, T. Ahmad, and H. Studiawan, "Transformer-based Sentiment Analysis for Anomaly Detection on Drone Forensic Timeline," in *International Symposium on Digital Forensics and Security (ISDFS)*, 2023, pp. 1–6.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *NAACL-HLT*, 2019, pp. 4171–4186.

[13] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[14] Z. Zhao, C. Xu, and B. Li, "A LSTM-Based Anomaly Detection Model for Log Analysis," *Journal of Signal Processing Systems*, vol. 93, no. 7, pp. 745–751, 2021.

[15] M. Memarzadeh, B. Matthews, and T. Templin, "Multiclass Anomaly Detection in Flight Data Using Semi-Supervised Explainable Deep Learning Model," *Journal of Aerospace Information System*, vol. 19, no. 2, pp. 83–97, 2022.

[16] F. Shahzad, A. Mannan, A. R. Javed, A. S. Almadhor, T. Baker, and D. Al-Jumeily OBE, "Cloud-based multiclass anomaly detection and categorization using ensemble learning," *Journal of Cloud Computing*, vol. 11, no. 1, p. 74, 2022.

[17] G. O. Anyanwu, C. I. Nwakanma, J. M. Lee, and D.-S. Kim, "Novel hyper-tuned ensemble Random Forest algorithm for the detection of false basic safety messages in Internet of Vehicles," *ICT Express*, vol. 9, no. 1, pp. 122–129, 2023.

[18] S. Silalahi, T. Ahmad, and H. Studiawan, "Drone Flight Log Anomaly Severity Classification via Sentence Embedding," in *International Conference on Artificial Intelligence, Blockchain, Cloud Computing, and Data Analytics (ICoABCD)*, 2023, pp. 100–105.

[19] J. Zhou, Y. Qian, Q. Zou, P. Liu, and J. Xiang, "DeepSyslog: Deep Anomaly Detection on Syslog Using Sentence Embedding and Metadata," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3051–3061, 2022.

[20] L. K. Shar, W. Minn, N. B. D. Ta, J. Fan, L. Jiang, and D. L. W. Kiat, "DronLomaly: Runtime detection of anomalous drone behaviors via log analysis and deep learning," in *29th Asia-Pacific Software Engineering Conference (APSEC)*, 2022, pp. 119–128.

[21] C. Zhang, X. Wang, H. Zhang, J. Zhang, H. Zhang, C. Liu, and P. Han, "LayerLog: Log sequence anomaly detection based on hierarchical semantics," *Applied Soft Computing*, vol. 132, p. 109860, 2023.

[22] J. Qi, Z. Luan, S. Huang, C. Fung, H. Yang, H. Li, D. Zhu, and D. Qian, "LogEncoder: Log-Based Contrastive Representation Learning for Anomaly Detection," *IEEE Transactions on Network and Service Management*, vol. 20, no. 2, pp. 1378–1391, 2023.

[23] X. Li, P. Chen, L. Jing, Z. He, and G. Yu, "SwissLog: Robust Anomaly Detection and Localization for Interleaved Unstructured Logs," *IEEE*

Transactions on Dependable and Secure Computing, vol. 20, no. 4, pp. 2762–2780, 2023.

[24] T. Xiao, Z. Quan, Z.-J. Wang, Y. Le, Y. Du, X. Liao, K. Li, and K. Li, "Loader: A Log Anomaly Detector Based on Transformer," *IEEE Transactions on Services Computing*, vol. 16, no. 5, pp. 3479–3492, 2023.

[25] V. Le and H. Zhang, "Log-based Anomaly Detection Without Log Parsing," in *36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2021, pp. 492–504.

[26] D. Zhang, D. Dai, R. Han, and M. Zheng, "SentiLog: Anomaly Detecting on Parallel File Systems via Log-Based Sentiment Analysis," in *13th ACM Workshop on Hot Topics in Storage and File Systems*, 2021, p. 86–93.

[27] S. Lupton, H. Washizaki, N. Yoshioka, and Y. Fukazawa, "Literature Review on Log Anomaly Detection Approaches Utilizing Online Parsing Methodology," in *28th Asia-Pacific Software Engineering Conference (APSEC)*, 2021, pp. 559–563.

[28] J. Singh and S. Gupta, "Evaluating the Impact of Local Data Imbalance on Federated Learning Performance for IoT Anomaly Detection," in *14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2023, pp. 1–7.

[29] F. Kong, J. Li, B. Jiang, H. Wang, and H. Song, "Integrated Generative Model for Industrial Anomaly Detection via Bidirectional LSTM and Attention Mechanism," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 541–550, 2023.

[30] S. Das, S. S. Mullick, and I. Zelinka, "On Supervised Class-Imbalanced Learning: An Updated Perspective and Some Key Challenges," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 6, pp. 973–993, 2022.

[31] N. T. Anh, L. H. Hoang, V. D. Minh, and T. H. Hai, "BKIDset - A New Intrusion Detection Dataset To Mitigate The Class Imbalance Problem," in *15th International Conference on Advanced Computing and Applications (ACOMP)*, 2021, pp. 106–111.

[32] H. Studiawan and F. Sohel, "Performance Evaluation of Anomaly Detection in Imbalanced System Log Data," in *4th World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, 2020, pp. 239–246.

[33] J. Singh and S. Gupta, "Evaluating the Impact of Local Data Imbalance on Federated Learning Performance for IoT Anomaly Detection," in *14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 2023, pp. 1–7.

[34] X. Ma and W. Shi, "AESMOTE: Adversarial Reinforcement Learning With SMOTE for Anomaly Detection," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 943–956, 2021.

[35] M. M. Rashid, F. Sabrina, B. Ray, A. Morshed, S. Gordon, and S. Wibowo, "Anomaly Detection in IoT Applications using Deep Learning with Class Balancing," in *IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, 2022, pp. 1–6.

[36] T. Sutthipanyo, T. Lamsan, W. Thawornsusin, and W. Susutti, "Log-Based Anomaly Detection Using CNN Model with Parameter Entity Labeling for Improving Log Preprocessing Approach," in *IEEE Region 10 Conference (TENCON)*, 2023, pp. 914–919.

[37] O. Elghalhoud, K. Naik, M. Zaman, and R. M. S, "Data Balancing and CNN based Network Intrusion Detection System," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2023, pp. 1–6.

[38] J. Qi, Z. Luan, S. Huang, Y. Wang, C. Fung, H. Yang, and D. Qian, "Adanomaly: Adaptive Anomaly Detection for System Logs with Adversarial Learning," in *IEEE/IFIP Network Operations and Management Symposium*, 2022, pp. 1–5.

[39] T. Al-Shehari, M. Al-Razgan, T. Alfakih, R. A. Alsowail, and S. Pandiaraj, "Insider Threat Detection Model Using Anomaly-Based Isolation Forest Algorithm," *IEEE Access*, vol. 11, pp. 118 170–118 185, 2023.

[40] P. Chand, M. Moh, and T.-S. Moh, "An Approach to Improving Anomaly Detection Using Multiple Detectors," in *16th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2022, pp. 1–8.

[41] S. Yan, S. Wang, Z. Chen, X. Jiang, and X. Cao, "CSLog: Anomaly Detection for Syslog Based on Contrastive Self-Supervised Representation Learning," in *24th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2023, pp. 165–170.

[42] X. Ma, J. Keung, P. He, Y. Xiao, X. Yu, and Y. Li, "A Semisupervised Approach for Industrial Anomaly Detection via Self-Adaptive Clustering," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 2, pp. 1687–1697, 2024.

[43] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *ICCV*, 2017, pp. 2999–3007.

[44] P. Cerda, G. Varoquaux, and B. Kégl, "Similarity encoding for learning with dirty categorical variables," *Machine Learning*, vol. 107, no. 8, pp. 1477–1494, 2018.

[45] X. Zhou, Y. Gao, C. Li, and Z. Huang, "A Multiple Gradient Descent Design for Multi-Task Learning on Edge Computing: Multi-Objective Machine Learning Approach," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 1, pp. 121–133, 2022.

[46] S. Silalahi, T. Ahmad, and H. Studiawan, "DroNER: Dataset for Drone Named Entity Recognition," *Data in Brief*, p. 109179, 2023.

[47] "Drone Error and Warning Codes - World's Most Comprehensive List," 2023. [Online]. Available: https://app.airdata.com/wiki/Notifications/

[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is All you Need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017, pp. 6000—-6010.

[49] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, p. 1735–1780, 1997.

[50] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1724–1734.

[51] C. Wang, P. Nulty, and D. Lillis, "A Comparative Study on Word Embeddings in Deep Learning for Text Classification," in *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval*, 2020, p. 37–46.

[52] H. Studiawan, G. Grispos, and K.-K. R. Choo, "Unmanned Aerial Vehicle (UAV) Forensics: The Good, The Bad, and the Unaddressed," *Computers & Security*, p. 103340, 2023.

SWARDIANTARA SILALAHI received his bachelor's in informatics education and his master's in informatics in 2021 and 2023, respectively. Currently, he is pursuing a PhD in computer science at the Institut Teknologi Sepuluh Nopember (ITS), Indonesia. His research interests include digital forensics, log mining, natural language processing, and deep learning.

TOHARI AHMAD received the Bachelor's degree in computer science from Institut Teknologi Sepuluh Nopember (ITS), Indonesia, master degree in information technology from Monash University, Australia, and the Ph.D. degree in computer science from RMIT University, Australia in 2012.

From 2001 to 2003, he was a consultant for some international companies. In 2003, he moved to ITS, where he is now a Professor. His research interests include network security, information security, data hiding, and computer networks. He is a reviewer of a number of journals.

Prof. Ahmad is a member of ACM, IEEE. His awards and honors include the Hitachi Research Fellowship and JICA Research Program to conduct research in Japan.

HUDAN STUDIAWAN is a Lecturer at Institut Teknologi Sepuluh Nopember, Indonesia. He graduated from Institut Teknologi Sepuluh Nopember, Indonesia, receiving bachelor's and master's degrees in 2009 and 2011, respectively. He received a Ph.D. degree from Murdoch University, Australia, in 2021. His current research interests are digital forensics and natural language processing.

EIRINI ANTHI is a Lecturer in cybersecurity at the School of Computer Science & Informatics, Cardiff University. She teaches Operating Systems Security and Cybersecurity Operations. In addition, her research interests revolve around the security of the Internet of Things (IoT) and Industrial Control Systems (ICS). More particularly, her research examines the security issues that come along with these devices/systems and focuses on developing intelligent and more robust cyber-attack detection mechanisms for such networks using machine learning and adversarial machine learning techniques. As part of her doctorate, she developed state-of-the-art tools to detect and defend against network-based cyber-attacks in such infrastructures.

LOWRI WILLIAMS is a Lecturer at the School of Computer Science & Informatics, Cardiff University. Her work focuses on the development of novel approaches towards automated cyber defence. In particular, her interest is on how to apply text mining and machine learning techniques in defence methodologies and different cybersecurity contexts.