IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 18 December 2022, accepted 2 January 2023, date of publication 5 January 2023, date of current version 11 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3234605

RESEARCH ARTICLE

Transformer-Based Named Entity Recognition on Drone Flight Logs to Support Forensic Investigation

SWARDIANTARA SILALAHI, (Member, IEEE), TOHARI AHMAD, (Member, IEEE), AND HUDAN STUDIAWAN, (Member, IEEE)

Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya 60111, Indonesia

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

This work was supported in part by the Institut Teknologi Sepuluh Nopember (ITS); and in part by the Pendidikan Magister Menuju Doktor untuk Sarjana Unggul (PMDSU) Scholarship from the Ministry of Education, Culture, Research and Technology, Indonesia, under Grant 1483/PKS/ITS/2022.

ABSTRACT The increase in drone usage by the public brings the number of drone incident and attack up. Sophisticated preventive mechanisms, as well as post-incident procedures and frameworks, are needed. Forensic investigation is performed upon a drone incident, aiming to uncover the incident scenario, mitigate the risk and report the examination results. Generally, standard drone forensic procedure consists of three stages, i.e., evidence acquisition, evidence analysis, and reporting. Among the existing research, many attempts have been made in framework proposal and evaluation, study case, and tools proposal and evaluation. However, less research focuses on utilizing specific data artifacts from the drone forensic image, such as telemetry, dataflash, and flight log data. Therefore, this research aims to propose the use of log message data to discover and extract some incident-related information using a deep learning-based NLP technique, i.e., named entity recognition using the Transformer. Cosine similarity is proposed as a substitute for dot-product in the self-attention mechanism of the Transformer encoder layer. Additionally, we propose NER architecture built from a mix of several existing methods and report the performance evaluation. We extract the DJI drone forensic image from a publicly available dataset using Autopsy and DJI Phantom Help and collect the decrypted log messages. Six entity types are defined after carefully reading the log message. These entity types are used in the manual annotation process using the IOB2 scheme as the label. The constructed dataset is used to evaluate the proposed model along with several baseline models. The proposed method outperforms the previous baseline model with a 91.348% F1 score. Finally, we conclude the experiment and mention several future directions.

INDEX TERMS Digital forensics, drone flight log, drone forensics, log mining, named entity recognition, transformer encoder, conditional random fields, infrastructure.

I. INTRODUCTION

UAV technology's presence has significantly impacted several sectors, such as industry, film, and advertisement. It can be seen from the increase in the number of consumer drone usage in recent years. A survey from Statista [1] states that the shipments of drone consumers reached approximately 5 million units in 2020 globally. This number is expected to keep increasing to 9.6 million delivery in 2030. The increase in drone employment in many fields brings and opens new challenges to secure drone devices. Other than consumer drones, there are other types of drones, i.e., military, terrorist, and criminal drones [2]. Any failure, error, or malfunction is not tolerated for these types of drones, as in consumer drones. Therefore, it is critical to guarantee the security of the device. To this end, more sophisticated security and forensic procedures are needed to develop to diminish the risk caused by any attack or incident [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Alba Amato.

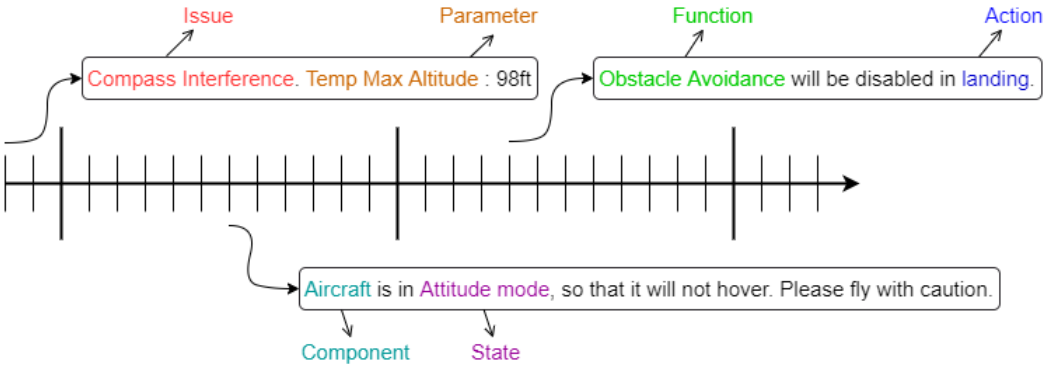

FIGURE 1. An illustration of entity types in a forensic timeline constructed from drone flight log data.

In the digital forensic research area, drone forensics is a quite new research topic. Generally, it is categorized into two sub-topics based on the evidence used to perform the investigation, i.e., digital and physical investigation. Both types of research aim to find relevant information regarding the incident, uncover the attack scenario, and diminish the risk as the effect of an incident [4]. In order to perform a digital forensic investigation, several artifacts can be utilized, such as image, video, telemetry log, and flight log data. The physical examination aims to achieve several objectives but is not limited to identifying any unique identifier, notable features, or damages. Secondly, it determines the model and class of the device, along with the capability of the storage system. Then, it lists the available options to perform extraction [5]. On the other hand, digital evidence is analyzed to achieve other objectives, such as mapping the link between the components of the UAV, identifying and matching the ownership to get a suspect user, and obtaining and inferring some information to prove that the device was used to commit a crime [5].

While the drone is flying, any event that happens to the drone is recorded in a log file, including the component's state, such as sensors, motors, GPS, and links. These data are stored in telemetry and dataflash logs located in the persistent storage attached to the device [6]. Mantas and Patsakis [6] have attempted to utilize telemetry and dataflash logs to perform drone forensic investigation by performing UAV integrity checks, anomaly detection on the visual flight path, command verification, error reporting, and hardware error detection. GRYPHON is proposed as an open-source tool to perform the aforementioned tasks [6]. Previously, DROP, as the first open-source tool for parsing drone flight log data from DJI, was proposed to help the process of acquiring the plain information encrypted in the proprietary.DAT dan.TXT log files of the DJI model [7]. Other than that, most of the drone forensic research is a type of study showcase, which starts from a scenario design, data generation and acquisition, data analysis, and finally, reporting. Several references are using DJI [8], [9], [10], Cheerson [11], Parrot [4], and Yuneec [12] model as experimental devices. However, there is no attempt to utilize specific data, in this case, the log message, to perform a drone forensic investigation. For this reason, we propose a deep learning-based Natural Language Processing (NLP) technique to perform information extraction from the log message to assist the forensic investigation process.

Information Extraction (IE) is one of the sub-topic in the NLP research domain, which aims to infer knowledge from a lake of text data. There are several steps in performing information extraction; after data source collection and preprocessing, Named Entity Recognition (NER) is one of the initial steps in IE [13]. The researcher has taken advantage of NER power to recognize and extract mentioned entities in several domain problems such as agriculture [13], [14], biomedical [15], [16], chemical [16], [17], food and dietary [14], and cybersecurity [18], [19], [20]. Inspired by the success of NER in those domains, we are motivated to investigate the usability of NER in the drone forensic domain, considering the characteristic of the data is unique for every domain specific. Fig. 1 illustrates a forensic timeline constructed from the flight log message along with mentioned entities within. A well-constructed forensic timeline exposes sequential events experienced by a system regarding a particular security incident [21]. In this research, we use flight log data to construct a forensic timeline that consists of the log message and the timestamp.

To perform NER, two common deep learning models can be used, either RNN-based or Transformer-based models. The latest state-of-the-art include BiLSTM-CRF and a pre-trained Transformer-based Language Model. The rise of Transformer-based language models (LM) such as BERT [22], one of the first pre-trained LM models, RoBERTa [23] as an optimized version of BERT, DistilBERT [24], a smaller, faster, cheaper, and lighter version of BERT and GPT [25], types of pre-trained language model that employs only the decoder part of Transformer architecture are significantly impacted the NLP research landscape, including NER. However, every domain-specific problem has its own unique problem and data characteristic. In the general NER, the common entity types are Organization,

Person, or Location. The sentence structure follows the natural language semantics. However, drone flight log messages' sentence structure does not necessarily adhere to that of the natural language in public news, for instance. For this reason, we aim to investigate the success of the Transformer-based technique in recognizing the region of interest in drone flight log messages.

The contributions of this paper are summarized as follows.

1) This research constructs a new NER dataset in the drone forensic domain. To the best of our knowledge, there is no publicly available NER dataset for drone forensic problems yet. We identify and propose six entity types as the tagset in the annotation process. The proposed model achieves a competitive score compared to the state-of-the-art methods, with a 91.146% F1 score. Yet, one of the scenarios achieves high performance, with a 91.348% F1 score.
2) This work showcases how to utilize specific evidence data to perform forensic information extraction to assist a forensic investigation, including a simple framework for data extraction and annotation.
3) We propose and investigate cosine similarity as a substitutive of dot-product in the self-attention sub-layer of the Transformer encoder to model contextual dependency in a sequence. We also propose a new NER architecture consisting of CNN character embedding, BERT word embedding, a Transformer with scaled dot-product attention as the encoder, and CRF as the decoder.

The paper comprises five sections. The remainder of this paper is as follows. Section II reviews the recent related works on drone forensic research, deep learning for named entity recognition, and the use of named entity recognition in cybersecurity. The proposed method is elucidated in Section III. Section IV explains the experimental results and analysis. We conclude the paper in Section V with several future directions.

II. RELATED WORKS

The advancement of the Unmanned Aerial Vehicle (UAV), commonly called a drone, followed by a constantly increasing number of drone usage in society, has brought drone forensic research to the surface and interested researchers. The case study-based paper is the most popular among the published papers on drone forensics. This section briefly discusses and summarizes the published works related to drone forensics. The following sub-section explains the other researchers' works on employing a deep learning model for named entity recognition. Since our research is a sub-field of cybersecurity, we also recap several related attempts at utilizing NER in the cybersecurity domain in the last subsection.

A. DRONE FORENSIC INVESTIGATION

The field of drone forensics is a relatively recent research topic. The growth and development of Unmanned Aerial Vehicle (UAV) technologies bring drone forensics subject to the surface and pique the academics' attention. The case study is the type of most drone forensics published research. In this research category, a forensic examination is performed on a drone device after having a scened flight under a controlled environment. The procedure starts with the data collection stage and ends with the investigation report. Yousef and Iqbal [10] proposed a series of guidelines to help forensic investigators conduct forensic investigations using the DJI Mavic Air drone model. In order to gather the evidence and explain the successfully collected data, various techniques are used, which may aid the investigation process. Several similar studies were done on other drone models, such as the Yuneec Typhoon model [12], DJI Spark [26], and the DJI Phantom [27]. According to those case studies, the drone's controller devices stored valuable data that can be compared to the other artifacts enabling a correlation study between the UAV and the mobile application used to control the drone. Some studies also suggest a technical procedure for drone forensic inquiry. In order to perform an end-to-end analysis from the preparation to the reporting of the findings, ten procedures proposed by Salamh et al. [12] must be followed.

Analyzing the encrypted files is one of the obstacles in the data collection phase. For the DJI models, encrypted evidentiary data is a certainty. However, sometimes we have no access to the DJI proprietary tools. Therefore, the data must be decrypted without using DJI's proprietary tool, even though DJI offers a closed-source and paid decryptor tool. Hence, some studies develop tools to help the researcher and investigator to conduct a forensic analysis. The DROP (Drone Open source Parser) tool developed by Clark et al. [7] is a parser tool for a.DAT file that can also decrypt the encrypted file to obtain the plain data within. Furthermore, DROP can link what the.DAT file holds and match it with the.TXT flight log file contents. After successfully decrypting those two files, GRYPHON [6] can be used for dataflash and telemetry log analysis. The program can perform timeline analysis, analyze flight data to discover an anomaly, map the GPS coordinates, and many other features. Other than the previously mentioned tools, several other tools were identified and described in a survey conducted by Viswanathan et al. [28]. Among the existing tools, Salamh et al. [8] carried out a case study to examine the features of the tools that were found to aid the forensic investigator in selecting the best suitable tools for a particular type of task. In the general digital forensics domain, log2timeline is commonly used to construct a forensic timeline from log records. The result is in.CSV format consisting of log records with corresponding timestamps. Timeline2GUI can be used to parse and analyze the log2timeline output file's contents. It offers an automatic analysis that is too complex if conducted manually. It is also equipped with sophisticated visualization features to highlight critical information and assist the forensic investigator in analyzing, interpreting, and drawing conclusions [29].

Understanding the drone device and its parts is a crucial step before beginning a forensic investigation [30].

Accordingly, Jain et al. [30] proposed a framework comprising 12 phases. The first five phases were used to locate and validate the drone's sensors and data. The other seven steps were used to analyze physical evidence like fingerprints and digital evidence from several sources, such as memory cards, flight logs, and network logs. However, the proposed framework has not explained the preparation phase in detail. Therefore, another framework consisting of four investigative phases with a more comprehensive analysis is proposed by Al-Dhaqm et al. [3]. One of the primary distinctions of this framework is a more extensive preparation phase consisting of pre- and post-incident preparation. Pre-incident preparation is an important step that is not yet covered in most forensic frameworks. This step aims to understand several possible indicators of compromise, define potential forensic evidence, and measure a drone device's forensic readiness before a flight. The remaining phases, including post-incident preparation, data acquisition, and data analysis, are likely the same as the other frameworks but more thorough.

Physical and digital evidence is the source of evidence used in a forensic investigation. Analyzing these two types of evidence require diverse technique. For digital evidence, computer-assisted tools are needed to read and present the evidence in a format that humans can comprehend. Study on evidence analysis is dominated by reconstructing and visualizing the flight path taken by the drone during a flight [5]. It is done by utilizing the GPS coordinates recorded in the flight log and with the help of the CsvView tool. A similar study conducted by Kumar and Agrawal [31] utilizes GPS data to reconstruct the flight path with three different drone makes as experimental devices. A tool to convert the.TXT or.JSON flight log file from Parrot make drones into a.CSV file that is easy to understand, named FlyLog Converter Tool, is proposed.

B. NAMED ENTITY RECOGNITION IN CYBERSECURITY

NER plays a vital role in the general NLP study as well as in the domain-specific areas, where it is utilized as an initial task that supports other downstream tasks, such as event extraction and relation extraction [32]. The capability of NER to obtain valuable information from text data can faster an information extraction process with a more accurate result. The extraction is accomplished by processing and recognizing tokens that may be associated with a specific type of entity [33]. NER task is not a new research topic. There have been many studies on the development of NER models. The capability of capturing bidirectional relations among words in a sentence posses by BiLSTM has been around for many years as the state-of-the-art method in the sequence labeling task. Accompanied by a statistical model, CRF, which can maximize the probability of a label sequence, has proven to improve the BiLSTM performance [32]. Many more advanced models with a richer input representation obtained from pre-trained word embedding models, such as Word2vec [34], GloVe [35], ELMo [36], and BERT [13], improved the BiLSTM-CRF architecture. However, the presence of Transformer has revolutionized many NLP tasks, including NER. Since the publication, many variants of pre-trained Transformer-based models have been available. The main advantage of the Transformer architecture is that the attention mechanism in the encoder sub-layer can model the context and relation between the words in a sentence.

TENER [37] attempts to utilize the Transformer encoder to perform NER by incorporating relative positional encoding to make the model distinguish the direction of a particular context. CNN char embedding is added to the embedding vector to represent the char-level feature. The proposed architecture outperformed the RNN-based state-of-the-art models in the benchmark NER dataset, CoNLL2003. Working with deep learning models is primarily a matter of providing decent input to the neural model. Several efforts have been made to increase the performance of the Transformer encoder by incorporating more features into the embedding vector of the word representation. Instead of solely relying on the word embedding as the input source, adding a dictionary feature embedding to the input vector can improve the BiLSTM-CRF model equipped with an attention mechanism [19].

Supervised-based deep learning models can take advantage of the label information attached to the data points in the training process. Besides updating the weight parameters, label information can also be injected into the input vector, as proposed in LUKE [38], to provide a rich input. An entity-aware self-attention mechanism is proposed to separate the token-to-token and token-to-entity context parameter. The masked language model is employed in the pre-training phase to predict some random token and entity. LUKE becomes the state-of-the-art for five well-known entity-related benchmarks, such as CoNLL2003 for NER, Open Entity for entity typing, TACRED for relation classification, ReCoRD for cloze-style question answering, and SQuAD 1.1 for extractive question answering task.

The capability of NER to recognize and extract the region of interest in unstructured text data has been implemented in various domains. Several efforts have been made to utilize NER in the cybersecurity domain. In a process-aware system, valuable information is stored in log files and commonly written in a less human-readable format. Because the records are text data in a large size, the researchers use particular NLP techniques to process the data and perform analysis automatically.

One of the most severe difficulties has been dealing with the complexity of cybersecurity data. Different systems and devices generate logs in different formats. No consistent name system with numerous acronyms, technical terminology, frequent conjunction use, and extensive nesting structure are the main challenges in cybersecurity data [33]. The prior state-of-the-art model employed the XBiLSTM-CRF architecture to conduct NER on a publicly available cybersecurity dataset [18]. The model's performance was enhanced by the concept of concatenating the word's vector representation with the Bidirectional Long Short-Term Memory (LSTM) layer output. The Conditional Random Field (CRF) layer is

used to decode the concatenated output since it can determine how the sequence labels relate to one another.

Most of the work involved in implementing deep learning models is predominantly spent on figuring out how to prepare adequate input representations. Most often, word embedding techniques like GloVe [39], Word2vec [40], BERT [22], and ELMo [41] are used to learn the input representation. Pre-trained language models, such as BERT and ELMo, can generate contextualized representational vectors after going through a pre-trained procedure on a large corpus. Contrary, GloVe employs local and global statistics to build a word vector, yielding a static lookup dictionary after being trained on a relatively large corpus. In order to provide additional information to the word embedding vectors, Gao et al. [19] developed a domain-specific knowledge base and data-driven NER system. Additionally, an attention mechanism is utilized to apply greater weights to more valuable information in a sentence. The experiment demonstrates that the designed model is more capable of identifying rare entities.

Aligning with the rise of the attention-based model, Zhou et al. [20] suggest further NER system development in cybersecurity data using BERT. Instead of taking a random word piece to be masked as used in BERT, Whole Word Mask is employed. Since the masking is applied to the entire word rather than at the word-piece level, this masking mechanism can cope with cybersecurity data better. This solution addressed the issue of conjunctions being frequently used in words like ''buffer-overflow'' or ''man-in-the-middle attack,'' which is one of the main issues in cybersecurity NER.

Among the published literature, there are still not many studies that specifically work on examining a particular type of drone forensic artifacts, especially the human-readable message within a drone flight log, as evidence to perform forensic analysis. Therefore, we are motivated to utilize the log message and perform information extraction to assist in a forensic investigation.

III. PROPOSED METHOD

The model's architecture of the proposed model is depicted in Fig. 2. In this research, we employ the modern deep learning model, Transformer, to encode and model the dependency between words in a sentence and perform named entity recognition in the drone forensic domain. Overall, our proposed method consists of positional encoding, character embedding, word embedding, encoder, and decoder. We further explain the details in the following sub-sections.

The existing studies show that most drone forensic research is based on case studies, tool development, and tool testing and evaluation. There are presently few studies performing analytics against certain drone data artifacts, specifically log message data. Inspired by the success of Transformer-based NER model implementation in various domains, including cybersecurity, this paper intends to take advantage of NER in recognizing mentioned entities in drone flight log messages. In order to fill the research gap, this work investigates the use of information extraction techniques to obtain insight from unstructured evidentiary data. The retrieved information is expected can assist the forensic investigator in pinpointing the critical information related to an incident in the flight logs faster.

A. DATA PREPROCESSING

Drone flight log data contains a number of columns with numerous information regarding the drone's condition and state. From those columns, we take the data from the message, tip, and warning columns. Not every log entry has a message, as the log message is generated and triggered by certain events or incidents. This message contains useful information for the forensic investigator to conduct forensic analysis and investigation. Therefore, the other columns in the drone flight log message are ignored.

After collecting all the log messages from the flight log files, the message is then tokenized to get per token separation without lowercasing. As observed from the dataset, many entities are written in a capital case. To give the model a chance to see the difference between upper and lower case, we preserve the original message without converting them to lowercase. We tokenize the message by keeping the dot and comma, as these two punctuations play the context separator role in a sentence. We use the Spacy[1] tokenizer to tokenize all the messages. The tokenized message is then converted into CoNLL format as a standard NER dataset format. Finally, equal-length tokens and labels are fed to the embedding layer to obtain a representational vector.

B. CHARACTER AND WORD-LEVEL EMBEDDING

Named entity recognition is part of a long process in the Information Extraction pipeline. NER is the initial step in performing information extraction from text data, which recognizes the region of interest and mentioned entities in text data or documents [13]. Since neural networks can not deal with text data, the data must be converted into numerical values. This process is called embedding. There are two levels of embedding used in this research, char-level and word-level embedding. Char-level embedding is used to tackle the out-of-vocabulary problem, which is common in NLP problems. Therefore, each character has its own embedding vector. CNN [43] and LSTM-based [44] char-level embedding are common approaches in NER. Besides CNN and LSTM, Ada-Trans [37] is also used as the char-level embedding in this research to provide rich comparisons.

Despite the parallelism support offered by Transformer architecture, it does not have information about a word's position in a sentence. However, words in a sentence are arranged in sequential order, and the order determines the contextual information. Thus, positional encoding is used to inject the representation of the position of the word. Let t be the index position of a word in a sequence, then $f : t \in \mathbb{N} \rightarrow \mathrm{PE}_t \in \mathbb{R}^d$ is a deterministic function that maps each index position into

[1] https://spacy.io/models

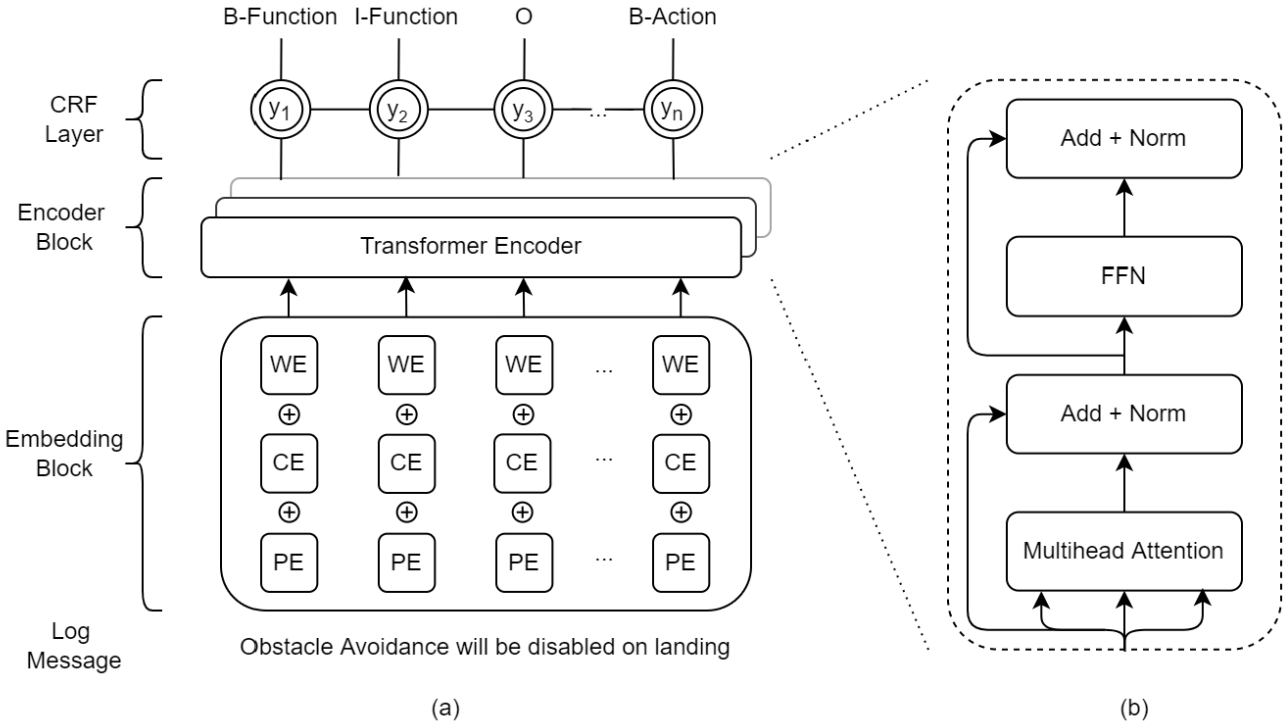

FIGURE 2. (a) Proposed method architecture. (b) Transformer encoder sub-layer [42].

a d dimensional vector, for $d \equiv 0 \mod 2$. This function is formulated in (1), where i is the index of the vector element.

$$f(t)_i = \begin{cases} \sin\left(\frac{1}{\omega_i} \times t\right), & i \equiv 0 \mod 2 \\ \cos\left(\frac{1}{\omega_i} \times t\right), & i \equiv 1 \mod 2 \end{cases} \quad (1)$$

$$\omega_i = 10000^{2i/d} \quad (2)$$

Word embedding is a representational vector used in NLP tasks to represent the features in text data. This vector not only contains the features in text data but also mimics the behavior of text data, such as semantics. A well-constructed word embedding vector can be used to estimate the similarity between two words having $\mathbf{u}$ and $\mathbf{v}$ as the embedding vector with d dimension using cosine similarity [45] as defined in (3).

$$\cos\ \theta = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}||\ ||\mathbf{v}||} = \frac{\sum_{i=1}^{d} u_i v_i}{\sqrt{\sum_{i=1}^{d} u_i^2}\sqrt{\sum_{i=1}^{d} v_i^2}} \quad (3)$$

There are two types of word embedding, static and contextual embedding. Word2vec [40], fasttext [46] and GloVe [39] are the common static embedding. While ELMo [41] and BERT [22] are an example of contextual embedding. The difference between static and contextual embedding is in the way of the lookup process. Static embedding has a static dictionary that maps the word into a vector. Therefore, a word will have exactly one representational vector, no matter what the context is. Contrary, contextual embedding generates a different representational vector for each distinct context of a certain word has.

In this research, GloVe, $E \in \mathbb{R}^{d_{vocab} \times d_{glove}}$ is used as a static embedding, and BERT, $E_s \in \mathbb{R}^{d_s \times d_{bert}}$ is used as the contextual embedding for sequence s. The final embedding vector is the concatenation of the positional embedding vector, char-level features extracted by the AdaTrans, and the pre-trained word embeddings GloVe or BERT.

C. TRANSFORMER ENCODER LAYER

The development of research on the topic of natural language processing reached a significant stage after the presence of an attention-based deep learning architecture called Transformer in 2017 [47]. This architecture was first introduced by the Google research team for English-German and English-French translation problems. The ability to understand and model the language is the main advantage of the Transformer. Transformer architecture is divided into two major parts: the Encoder and the Decoder. In this study, the only part used was the Encoder. In general, the elements that build the Encoder block include Input Embedding, Positional Encoding, Multi-head Attention, and Feed-forward Networks [42].

The attention mechanism in Transformer architecture tries to model the way some data in a database system are retrieved. Previously, the attention mechanism was introduced by Bahdanau et al. [48] in 2015 as additive attention, which was then modified by Luong et al. [49] in 2015 by proposing dot-product attention. These two papers use language translation

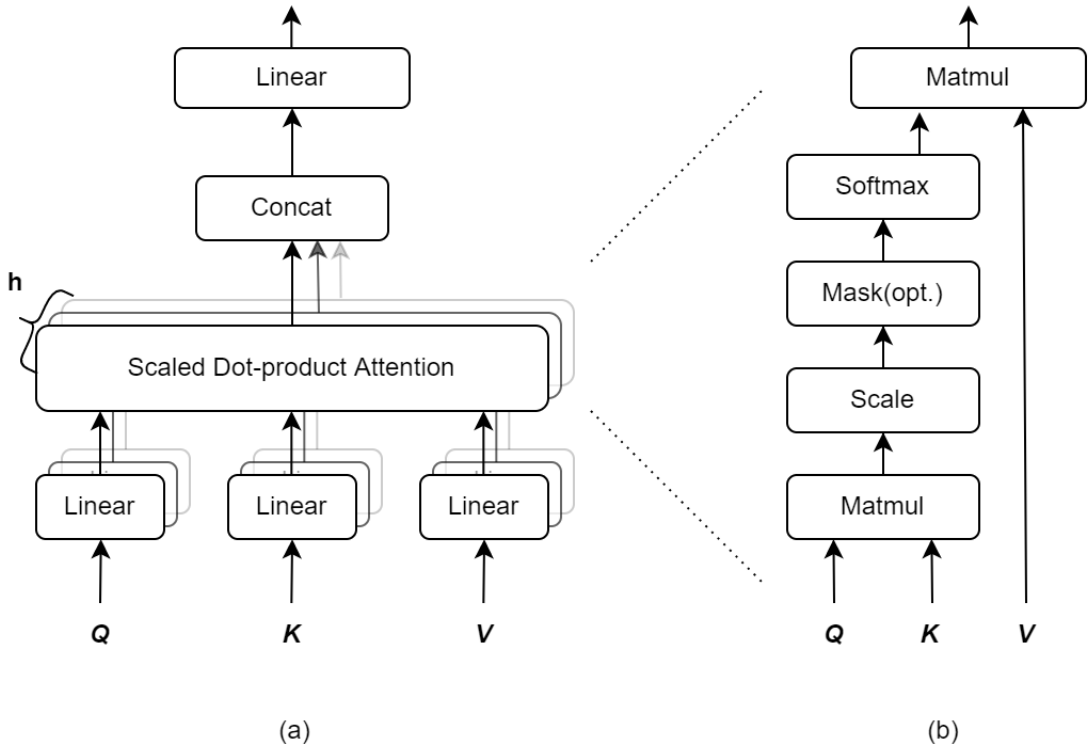

FIGURE 3. (a) The inner structure of the multi-head attention sub-layer. The shadow represents the attention heads arranged in parallel. (b) The inner structure of the self-attention mechanism. [42].

as the experimental case and model contextual learning using the attention mechanism. In the dot-product attention, each token in the sequence is transformed into three different representational vectors, i.e., query, key, and value, as shown in Fig. 3. In order to obtain the contextual representation of the currently processed token, there are five steps to follow [42].

1) Project each of the token's vectors in the sequence into three representational vectors, i.e., $\mathbf{q} \in \mathbb{R}^{d_k}$, $\mathbf{k} \in \mathbb{R}^{d_k}$, and $\mathbf{v} \in \mathbb{R}^{d_v}$. These three vectors are computed by multiplying the embedding vector $\mathbf{e} \in \mathbb{R}^{d_{model}}$ with three weight matrices $W^q \in \mathbb{R}^{d_{model} \times d_k}$, $W^k \in \mathbb{R}^{d_{model} \times d_k}$, and $W^v \in \mathbb{R}^{d_{model} \times d_v}$ which randomly initialized.
2) Take the dot-product between the vector of the current token $\mathbf{q}_t$ to each vector of the context token $\mathbf{k}_j$ in the sequence, yields vector $\mathbf{s}_t = \langle s_{t1}\ s_{t2}\ s_{t3}\ \ldots\ s_{tj} \rangle$ for $j = 1, 2, 3, \ldots, n$, where n is the number of token in a sequence.

$$s_{tj} = \mathbf{q}_t \cdot \mathbf{k}_j \tag{4}$$

3) Scale the output of the dot-product by dividing it with $\sqrt{d_k}$. This is the main difference between Loung's attention with the Vaswani's attention.

$$\hat{\mathbf{s}}_t = \mathbf{s}_t \times \frac{1}{\sqrt{d_k}} \tag{5}$$

4) Normalize the scaled dot-product output using softmax. The output of this step is a probability distribution to weigh the $\mathbf{v}$ vector as the target context, yielding a vector $\mathbf{w}_t = \langle w_{t1}\ w_{t2}\ w_{t3}\ \ldots\ w_{tj} \rangle$. Therefore, $\sum_{j=1}^{n} w_{tj} = 1$.

$$w_{tj} = \frac{\exp(\hat{s}_{tj})}{\sum_{j=1}^{n} \exp(\hat{s}_{tj})} \tag{6}$$

5) Finally, perform Hadamard Product ($\odot$) between the probability distribution with the $\mathbf{v}$ vector to get the weighted value vector, as the weight indicates the amount of attention that exists between the query and key vector.

$$\mathbf{y}_t = \sum_{j=1}^{n} w_{tj} \mathbf{v}_j \tag{7}$$

Vector $\mathbf{y}_t \in \mathbb{R}^{d_v}$ is the output of the scaled dot-product self-attention mechanism, as explained previously, which contains the contextual representation of the current token. Mathematically, the self-attention score of $\mathbf{q}_t$ against each of $\mathbf{k}_j$ and $\mathbf{v}_j$ in a sequence with n number of tokens is formulated as (8).

$$\text{Attn}(\mathbf{q}_t, \mathbf{k}_j, \mathbf{v}_j) = \sum_{j=1}^{n} \text{softmax}(\frac{\mathbf{q}_t \cdot \mathbf{k}_j}{\sqrt{d_k}}) \mathbf{v}_j \tag{8}$$

Practically, the computation of forward propagation in neural networks is in a matrix multiplication nature. Instead of taking the dot-product between the vectors one by one, the

whole self-attention mechanism can be wrapped into a single matrix multiplication operation by building Q, K, and V for query, key, and value matrices, respectively. These matrices are obtained by multiplying the word embedding matrices of a sequence $E_s \in \mathbb{R}^{d_s \times d_{model}}$, with the weight matrices $W^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W^K \in \mathbb{R}^{d_{model} \times d_k}$, and $W^V \in \mathbb{R}^{d_{model} \times d_v}$. The projected matrices $Q \in \mathbb{R}^{d_s \times d_k}$, $K \in \mathbb{R}^{d_s \times d_k}$, and $V \in \mathbb{R}^{d_s \times d_v}$ are the representational matrices for the sequence. Therefore, the self-attention mechanism for a single sequence can be formulated as (9).

$$\text{Attn}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{9}$$

In the self-attention mechanism, each token has one self-attention score against each token in the sequence. It makes the output vector tends to contain only a single context for each token in the sentence. However, it is possible for certain word has several contextual relations with more than one word in the sentence. Therefore, multi-head attention comes as a solution to learning several contexts for each token, which is modeled in each attention head weight. The attention head is a hyperparameter in Transformer architecture. We can set the number of attention heads as needed based on our data and case. In this paper, the sequence length is mostly (more than 80%) less than ten words, and the longest sequence is 33, so it is less likely that a word has several contexts in a sequence. To keep the model's complexity simple, the d_k and d_v are taken from $d_{model}/H = 128$. Thus, the complexity of multi-head attention with $d_k = d_{model}/H$ is the same as a single head with $d_k = d_{model}$. The multi-head attention mechanism can be formulated in a matrix multiplication operation, as shown in (10), where h denotes the attention head, H is the number of the attention head, a is the index of attention head, and W^O is a weight matrix for the concatenated output from each attention head.

$$\text{MultiHead}(Q, K, V) = \text{Concat}(h_1, \ldots, h_H)W^O \tag{10}$$

$$h_a = \text{Attn}(QW_a^Q, KW_a^K, VW_a^V) \tag{11}$$

The $W_a^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_a^K \in \mathbb{R}^{d_{model} \times d_k}$ and $W_a^V \in \mathbb{R}^{d_{model} \times d_v}$ matrices are different weight matrices for each attention head. In order to obtain the multi-head attention score, the result of each attention head is concatenated, then multiplied by a weight matrix $W^O \in \mathbb{R}^{hd_v \times d_{model}}$. The resulting matrix is then passed as an input to the next sub-layer, which is a fully-connected layer. The overall multi-head attention mechanism is depicted in Fig. 3.

The fixed sinusoidal positional encoding proposed in Transformer is not representative enough since it only represents the distinct position and distance but lacks direction information. Inspired by the success of bidirectional LSTM, TENER incorporates direction-aware positional encoding to give the attention mechanism ability to model which direction of a certain context comes from [50] and [51], which is then called AdaTrans. Therefore, the modified formula to obtain the attention score between query and key vector is shown in (13) [37], where t is the index position of the current token and j is the index position of the context token. Fixed sinusoidal positional encoding PE_t in (1) becomes R_{t-j} in (12) to represent the relative positional encoding, and $R_{t-j} \in \mathbb{R}^{d_k}$ to make it compatible with the word embedding vector dimension. $\mathbf{u}$ and $\mathbf{v}$ are learnable parameters to give the model the ability to distinguish the representation of $\mathbf{e}_{t,j}$ and $\mathbf{e}_{t+1,j}$ from different distances, and ω_i is the same term as (2).

$$R_{t-j} = \left[\ldots \sin\left(\frac{t-j}{\omega_i}\right) \cos\left(\frac{t-j}{\omega_i}\right) \ldots\right]^T \tag{12}$$

$$A_{t,j}^{rel} = Q_t K_j^T + Q_t R_{t-j}^T + \mathbf{u}K_j^T + \mathbf{v}R_{t-j}^T \tag{13}$$

$$\text{Attn}(Q, K, V) = \text{softmax}(A^{rel})V \tag{14}$$

Several attention mechanism modifications focus on injecting more linguistic features into the embedding vector and the attention computation. However, to the best of our knowledge, there is no attempt to control the attention output's behavior yet. Inspired by [52] where cosine is used as a normalization function in neural network architecture, we intended to use cosine similarity to normalize the attention score. Originally, the output of the attention is scaled by $\sqrt{d_k}$ [42], then fed to the softmax function to get the probability distribution. However, the resulting probability distribution has only one significant element, which is then used to weigh the context value vector. Thus, the attention score will represent exactly one context only. Multihead attention overcomes this issue by projecting the key, query, and value vector into several distinct attention heads which do not share their parameters. Since cosine can smoothen the probability distribution from the softmax output, we aim to investigate the use of cosine normalization as a substitute for the dot-product operation in the self-attention mechanism. As illustrated in Fig. 4, the probability distribution on the smaller scale tends to have several significant values compared to the larger one. This slope probability distribution will capture several attention from the context words' vector. Additionally, from the existing NER architecture, we explore several possible arrangements to find an architecture with the best performance evaluated on our dataset.

Before performing matrix multiplication between the key and query vector in the self-attention mechanism, our proposed method first divides these two vectors with their respective norm and constructs the matrix back. The modified self-attention mechanism is depicted in Fig. 5. Since (3) can be written in the form of (15), then $\hat{Q}$ and $\hat{K}$ are the query, and key matrices constructed from the vectors that have been divided by their respective norm. Consequently, we can fully exploit the optimizable matrix multiplication operation as in the vanilla Transformer architecture. Thus, the forward propagation is slightly the same, except for the additional step for dividing the key and query vector by its norm before performing the matrix multiplication. Therefore, the attention score between the query and key vector using

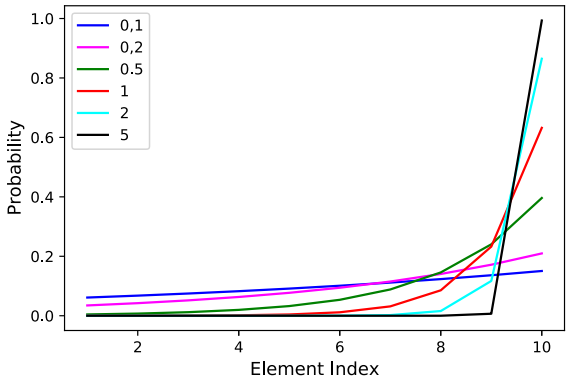

FIGURE 4. Softmax behavior on vectors in different scales.

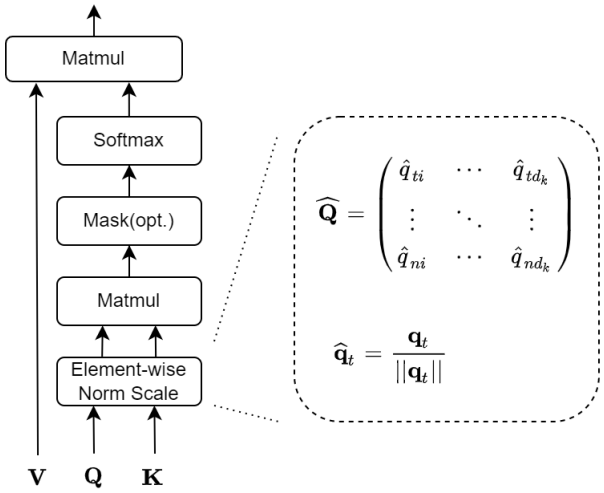

FIGURE 5. Cosine-based self-attention mechanism. The respective vector norm is used to scale each row element of *Q* and *K* matrices.

cosine similarity can be computed using (16).

$$\text{cosine}(\hat{\mathbf{u}}, \hat{\mathbf{v}}) = \frac{\mathbf{u}}{||\mathbf{u}||} \cdot \frac{\mathbf{v}}{||\mathbf{v}||} \tag{15}$$

$$\text{Attn}\left(\mathbf{q}_t, \mathbf{k}_j, \mathbf{v}_j\right) = \sum_{j=1}^{n} \text{softmax}(\text{cosine}(\hat{\mathbf{q}}_t, \hat{\mathbf{k}}_j))\mathbf{v}_j \tag{16}$$

The output of the multi-head attention sub-layer is then passed to the Add + Norm sub-layer, as shown in Fig. 2 (b). The term Add in Add + Norm sub-layer means a residual connection [53] between the previous sub-layer output and the current sub-layer output before being propagated to the next sub-layer. This residual connection retains the positional information from the embedding layer during the computation to the upper layer of the architecture. The term Norm refers to LayerNorm [54] to control the value of each sub-layer output. Afterward, the next sub-layer is the Feed Forward Network (FFN) which consists of two linear transformations with ReLU [55] activation function in between. This sub-layer is formulated in (17) as follows:

$$\text{FFN}(x) = \max(0,\ xW_1 + b_1)W_2 + b_2 \tag{17}$$

where W and b is the weight and bias parameter for each linear layer in FFN, and x is the input vector. The output of this sub-layer is then passed to a linear layer before being propagated to the decoder.

D. CONDITIONAL RANDOM FIELD LAYER

In sequence labeling tasks, such as NER, CRF is a common method used. According to studies, the Hidden Markov Model and the Maximum Entropy Markov Model (MEMM) are ineffective at analyzing sentence-level sequences compared to the CRF approach [15]. The main CRF features that can compute cross-position label combination probability grab the researchers' attention to apply this method to the NER problem. Combining the previous state-of-the-art NER model, BiLSTM, with CRF has proven to improve performance [56]. In this paper, CRF is used as the decoder for all encoder combinations in our experiment. For an observed sequence $\mathbf{x} = \langle x_1\, x_2\, \ldots\, x_n\rangle$ with the corresponding target label $\mathbf{y} = \langle y_1\, y_2\, \ldots\, y_n\rangle$, let $\mathbb{Y}$ be the set of all valid sequence of labels in the dataset. The probability of the predicted label from the encoder is computed using (18), where $f(\mathbf{x},\ y_{t-1},\ y_t,\ t)$ is an arbitrary feature function to compute the transition score from y_{t-1} to y_t in the sequence $\mathbf{x}$. Let d be the number of feature functions used, $\text{Feat}(\mathbf{x},\ y_{t-1},\ y_t,\ t)$ is the weighted sum of all transition scores from y_{t-1} to y_t in the sequence $\mathbf{x}$ from each feature function. After getting all possible paths and their corresponding probability, the Viterbi algorithm is used to discover $\hat{\mathbf{y}}$, which denotes the path with the highest probability, as written in (21).

$$P\left(\mathbf{y} \mid \mathbf{x}\right) = \frac{1}{Z} \exp\left[\sum_{t=1}^{n} \text{Feat}(\mathbf{x}, y_{t-1}, y_t, t)\right] \tag{18}$$

$$Z = \sum_{\tilde{\mathbf{y}} \in \mathbb{Y}} \exp\left[\sum_{t=1}^{n} \text{Feat}(\mathbf{x}, \tilde{y}_{t-1}, \tilde{y}_t, t)\right] \tag{19}$$

$$\text{Feat}(\mathbf{x}, y_{t-1}, y_t, t) = \sum_{j=1}^{d} w_j f_j(\mathbf{x}, y_{t-1}, y_t, t) \tag{20}$$

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y} \mid \mathbf{x}) \tag{21}$$

IV. EXPERIMENTAL RESULT AND ANALYSIS

In this section, we give the details of the long process of dataset preparation which consists of data collection, decryption, extraction, cleansing, entity type identification, annotation rules definition, data annotation, and train test splitting. We then describe the experiment settings we used to get the experimental results. Furthermore, we discuss the performance of our proposed method with several attention mechanism arrangements. We then compare the performance of our proposed method with other baseline models. Finally, we disclose the research challenges and limitations we encounter throughout the experiment. The experimental code along with the dataset is available on a GitHub repository.[2]

[2]https://github.com/swardiantara/droner-cosine

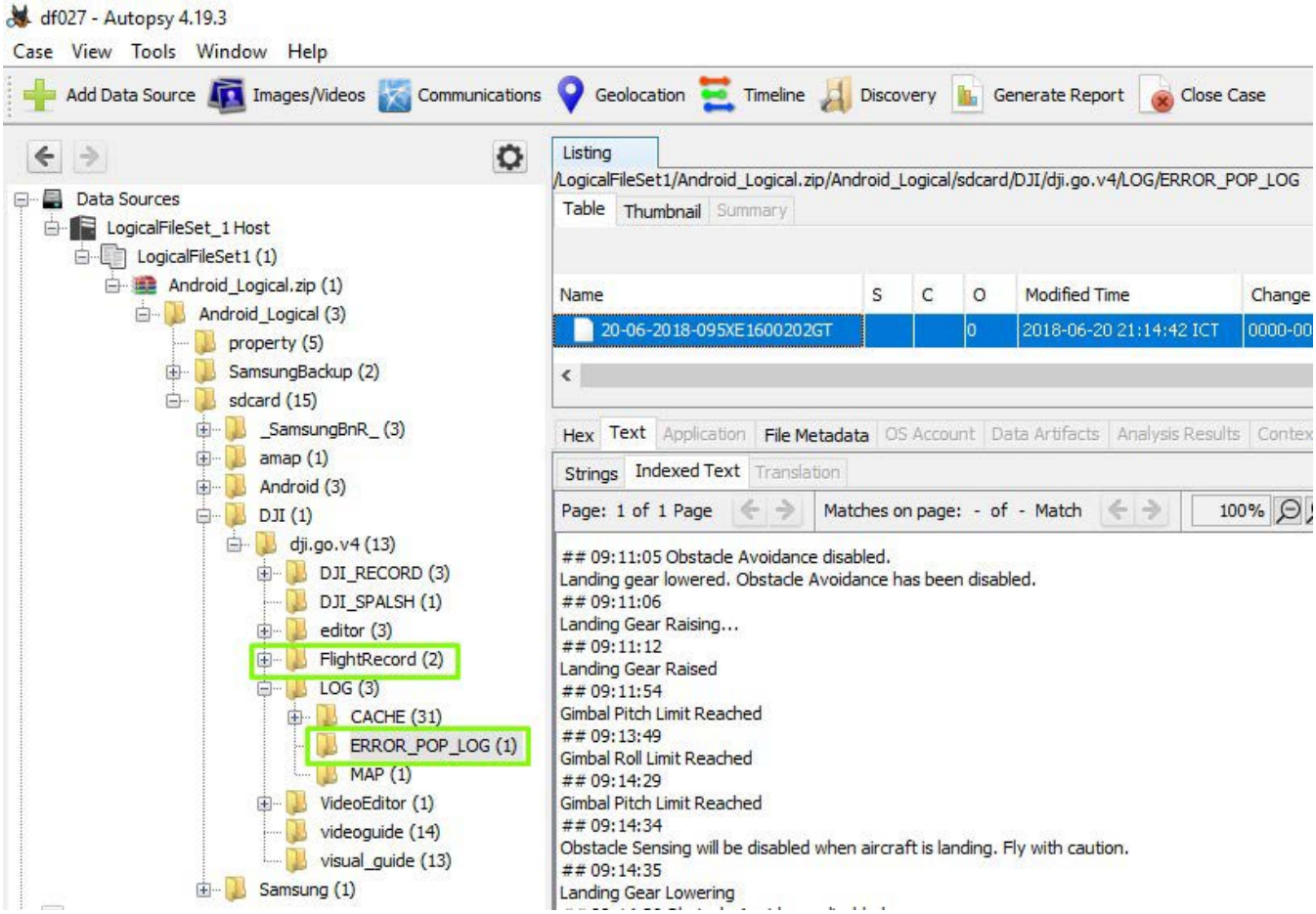

FIGURE 6. Data collection and extraction using Autopsy. The highlighted folders are the location of the flight log files containing human-readable messages.

A. DATASET PREPARATION

To the best of our knowledge, there are no publicly available NER datasets in the drone forensic domain yet. For this reason, a new dataset is constructed for the experiment in this paper. However, there is an open drone forensic image dataset publicly available provided by VTO Labs Drone Forensic Program.[3] Therefore, the first step in dataset preparation was the data extraction process. From the total of 82 drone images from 10 different models, we choose 60 drone images from three drone models, i.e., DJI, Parrot, and Yunnec, to extract, simply because these three models are the majority among the available models. As of March 2021, DJI had a market share of 76%, based on the sale volume. Thus, most of the consumer and commercial drones in the market are DJI-made [57].

The drone images are stored in several different formats, such as.ZIP,.001, and.BIN. These images are acquired from the controller devices, which are considered the primary evidence close to the owner and contain incident-related information [58]. After exploring the drone images with the help of Autopsy[4] and DJI Phantom Help[5] for extracting and decrypting, Autopsy was used to extract the drone images file from the Android-based controller with.001 and.BIN extensions. The Autopsy is also used to decrypt the files inside the.ZIP files obtained from the iOS-based controller devices. Fig. 6 shows the Autopsy interface when extracting a drone forensic image acquired from an Android-based controller device. The green boxes denote the path of the flight log files stored. Sometimes, `/dji.go.v4/` appear in a different folder name, i.e., `/dji.pilot/`. Both of the folders possibly exist at the same time in a single drone forensic image.

The only data taken from the drone images were human-readable log messages in order to perform entity recognition. To find this kind of data, we explore the drone images directory, which potentially contains human-readable log data. We found it in the flight log data. Then, we try to find all the locations of flight log data in all directories of every drone image we have downloaded and extracted. Unfortunately, we did not find the expected data from Yuneec

[3]https://www.vtolabs.com/drone-forensics

[4]https://www.autopsy.com/

[5]https://www.phantomhelp.com/LogViewer/upload/

Message	You	can	reset	the	RTH	Altitude	in	Remote	Controller	Settings	after	cancelling	RTH	.
IOB2	O	O	O	O	B-Parameter	I-Parameter	O	B-Function	I-Function	I-Function	O	O	B-Action	O
BIOES	O	O	O	O	B-Parameter	E-Parameter	O	B-Function	I-Function	E-Function	O	O	S-Action	O

FIGURE 7. Difference between IOB2 and BIOES annotation scheme for the same log message. The labels are assigned using the contextual tagging procedure.

TABLE 1. Number of extracted messages from every drone model.

Drone Model	Dataset	# of Message
DJI Mavic 2 Pro	DF069	28
DJI Mavic 2 Zoom	DF068	117
	DF067	31
DJI Matrice 210	DF060	49
	DF059	34
DJI Mavic Air	DF050	85
	DF048	101
DJI S1000+	DF043	17
DJI Matrice 600	DF034	96
DJI Inspire 2	DF027	249
	DF026	109
	DF025	83
DJI Mavic Pro	DF021	54
	DF020	161
	DF019	92
DJI Inspire 1	DF011	44
	DF010	62
DJI Phantom 4 Pro V2	DF063	64
	DF062	63
	DF061	46
DJI Spark	DF008	31
	DF007	3
DJI Phantom 3	DF002	34
	DF001	59
DJI Phantom 4	DF006	62
	DF005	76
	Total	1850

and Parrot models. Therefore, the only model that contains the data is the DJI. After collecting the flight log files, we use DJI Phantom Help tools to decrypt the files and get the plain data, which then is parsed to get the log message data. The number of messages from every drone image is shown in Table 1.

The next step is identifying the entity type mentioned in the drone log messages. Before reading the log messages, we filtered the duplicate message and got the unique message to read. After carefully reading the unique log message and comprehensively studying what every log message indicates, we categorized the entity types mentioned in the flight log message into six groups, i.e., Component, Action, Parameter, Function, State, and Issue. These entity types are used as the label for every word in a message after performing data annotation.

Annotation is a process of assigning a label to each data point in order to train a supervised model. In this case, the log message is the data that will be annotated. To demonstrate the power of contextual learning in the Transformer encoder, two annotation procedures are used to label the data, i.e., contextual tagging and consistent tagging. Consistent tagging refers to assigning a label to a word by only considering the token and ignoring the context within the sentence. Contrary, contextual tagging assigns a label to each word in a sentence by considering the present context. The following are several criteria for the annotation process on each entity type.

A ''Component'' label will be assigned to a span that indicates drone components, such as motors, sensors, and batteries. If the span is indicating of an action taken by the drone, then it will be assigned the ''Action'' label. The ''parameter'' label is assigned to the span, which indicates some variables stored in the drone, such as maximum flight distance, maximum flight altitude, and battery temperature. Every drone type has features or functions supporting the task given to it. Some example of span indicates function is obstacle avoidance, obstacle sensing, and remote controller settings. This type of span is assigned the ''Function'' label. Some spans indicate a drone's mode, such as sport mode, auto landing mode, and quick shot mode. These spans get the ''State'' label. Lastly, the ''Issue'' label is assigned to a span that indicates flight issues that happen to the drone during a flight.

Before assigning the label to each word, we first tokenize the sentence into tokens using tecoholic[6] tools. Then, the same tool is used to perform the data annotation process. IOB2 is used as the annotation scheme since IOB2 is one of the typical schemes in the NER task [59]. However, the BIOES scheme is proven can improve the NER model's performance [37]. Therefore, after finishing the annotation using the IOB2 scheme, a python script is used to convert the annotation into a BIOES scheme. We manually annotate the unique message only by carefully reading the context of the sentence first. Fig. 7 shows a sample of annotated data using the IOB2 and BIOES scheme in CoNLL format. Sometimes, a particular span belongs to two or more alternative entity types' tags. For this confusing span, we chose the longest span as the context of the mentioned entity. The ''battery temperature'' span is given the Parameter label for contextual tagging. However, for consistent tagging, the Component label for the word ''battery'' and the Outside label for the word ''temperature'' is assigned, respectively. Additionally, for the ''battery signal error'' span, the Issue is assigned for those three tokens considering the context. Nevertheless, consistent tagging assigns each token the Component, Outside, and Issue labels, respectively. After completing the label for all unique messages, we do the annotation for all messages

[6]https://tecoholic.github.io/ner-annotator/

Message	Max	Flight	Distance	Reached	.	Adjust	in	Main	Controller	Settings	if	necessary	.
Contextual	B-Parameter	I-Parameter	I-Parameter	O	O	O	O	B-Function	I-Function	I-Function	O	O	O
Consistent	O	B-Parameter	I-Parameter	O	O	O	O	B-Component	I-Component	B-Function	O	O	O

FIGURE 8. Results of using contextual and consistent tagging procedure on a log message using the IOB2 scheme. The striking difference lies in the "Main Controller Settings" span.

TABLE 2. Number of message, token, and entities in the consistent and contextual dataset.

Dataset	Split Set	Message	Token	Entity
Consistent	Train	1412	9103	3843
	Test	438	4005	953
	Total	1850	13108	4796
Contextual	Train	1412	9103	4184
	Test	438	4005	1118
	Total	1850	13108	5302

TABLE 3. Per entity type token distribution.

Consistent Dataset

Split	Component	Parameter	Function	State	Issue	Action
Train	566	1042	830	134	791	480
Test	175	206	123	124	125	200
Total	741	1248	953	258	916	680

Contextual Dataset

Split	Component	Parameter	Function	State	Issue	Action
Train	503	1828	490	125	955	283
Test	147	322	167	107	246	129
Total	605	2150	657	232	1201	412

by using the labeled unique message as a lookup dictionary. Fig. 8 shows the annotation results using consistent and contextual tagging procedures.

After completing the annotation process, the dataset is split into train and test sets. Unlike the usual splitting method, we split the dataset based on the drone types. The first nine drone models in Table 1 are the train set, while the last four are used as the test set. By doing this, the train and test sets are generated from completely different drone models. Assuming that every model has its own features and functionalities, which vary among them, then the generated log messages will be different as well. However, since all the drones are DJI make, the test set is chosen from the most advanced type to make the test set contains log messages that do not exist in the train set. Because the features and functionalities in a more advanced model will not be in a less advanced model, so do with the generated log messages.

As the final result of data preparation, the distribution of every entity type in the train and test set of the annotated datasets is shown in Table 2 and Table 3. The final composition of the dataset is 76:24 for train and test sets, respectively, from a total of 1850 log messages. The final proportion is uncontrollable since the splitting is done based on the drone models instead of directly dividing the message into a certain common ratio used in existing research.

B. EXPERIMENT SETTINGS

The experiment was conducted using the publicly available code provided by TENER's original paper [37]. Therefore, the only requirement to install is fastNLP library.[7] We modified the code to implement the proposed method. While the hardware specification is as follows: Intel Core i7-8700 @ 3.2GHz, 16GB RAM, NVIDIA GeForce GTX 1060 6GB, and Ubuntu 20.04 LTS operating system.

The dimension of the input vector is 768, divided into eight attention heads with 96 as the dimension for each head and three encoder layers in the Transformer architecture. Both train and test batch size is 8, with a learning rate of 0.001 and with a warm-up step of 0.01. We set the dropout to 0.15 except for the fully-connected layers, which used 0.4. The intermediate fully-connected layers are sized 1536 dimensions. These parameters are inspired by Transformer [42] and TENER [37] original papers. The number of epochs we used is 50 because it has already provided convergence, as shown in Fig. 11. Three char-level embeddings, such as LSTM, CNN, and AdaTrans, were combined with two word-level embeddings, GloVe and BERT, to provide input for the encoder layer. In the Transformer encoder, three different attentions are employed combined with options whether to scale or unscale the attention score in the self-attention mechanism. Finally, the CRF is the only decoder used.

Several scenarios which used BiLSTM as the encoder are designed for the experiment based on the published reference as the baseline methods for comparison. The combination of arrangements from the available options of word embedding, char embedding, and the attention type are presented in the following subsection, along with the results. We freeze the BERT embedding to avoid the domination of the attention mechanism used in the BERT pretraining phase. Therefore, BERT parameters will not be updated during the training. We ran the experiment three times for each scenario and took the average as the final evaluation score.

The evaluation mechanism used in this experiment is the span-oriented paradigm. It means the predicted tag is evaluated on the entity type level instead of on the tag level. Therefore, if the predicted entity type is correct, even if the tag is not strictly correct, the predicted token is considered True Positive. For example, if the true label is B-Component, while the predicted label is I-Component or vice versa, we count the predicted label as True Positive.

Precision, Recall, and F1 score are used as the evaluation metrics after counting the true positive (TP), false positive

[7]https://fastnlp.readthedocs.io/

TABLE 4. Performance evaluation of all scenarios on the dataset annotated using consistent tagging with AdaTrans as the char-level embedding. The best score is indicated in bold font. BERT and GloVe are used as word embedding.

Model	F1-Score	Precision	Recall
BERT - Scaled - AdaTrans	86.497	84.640	88.439
BERT - Scaled - Cosine	85.233	82.964	87.629
BERT - Scaled - Transformer	85.515	82.663	**88.574**
BERT - Unscaled - AdaTrans	**86.586**	84.857	88.394
BERT - BiLSTM	86.193	86.607	85.785
BERT - Unscaled - Cosine	85.408	83.017	87.944
BERT - Unscaled - Transformer	85.712	83.314	88.259
GloVe - Scaled - AdaTrans	84.981	84.080	85.965
GloVe - Scaled - Cosine	83.429	83.062	83.806
GloVe - Scaled - Transformer	84.283	83.301	85.290
GloVe - Unscaled - AdaTrans	85.489	84.769	86.235
GloVe - BiLSTM	86.311	**86.990**	85.650
GloVe - Unscaled - Cosine	83.714	83.490	83.941
GloVe - Unscaled - Transformer	84.538	83.892	85.200

TABLE 5. Performance evaluation of all scenarios on the dataset annotated using consistent tagging with LSTM as the char-level embedding. The best score is indicated in bold font. BERT and GloVe are used as word embedding.

Model	F1-Score	Precision	Recall
BERT - Scaled - AdaTrans	86.183	84.995	87.404
BERT - Scaled - Cosine	85.406	82.576	**88.439**
BERT - Scaled - Transformer	85.330	83.970	86.775
BERT - Unscaled - AdaTrans	**86.543**	86.247	86.865
BERT - BiLSTM	85.931	86.511	85.425
BERT - Unscaled - Cosine	84.978	82.683	87.404
BERT - Unscaled - Transformer	85.377	83.865	86.955
GloVe - Scaled - AdaTrans	85.275	83.849	86.775
GloVe - Scaled - Cosine	84.829	83.553	86.145
GloVe - Scaled - Transformer	85.730	83.537	88.079
GloVe - Unscaled - AdaTrans	86.254	84.259	88.349
GloVe - BiLSTM	86.506	**87.254**	85.785
GloVe - Unscaled - Cosine	85.718	84.265	87.224
GloVe - Unscaled - Transformer	85.535	83.825	87.359

TABLE 6. Performance evaluation of all scenarios on the dataset annotated using consistent tagging with CNN as the char-level embedding. The best score is indicated in bold font. BERT and GloVe are used as word embedding.

Model	F1-Score	Precision	Recall
BERT - Scaled - AdaTrans	86.778	84.810	88.844
BERT - Scaled - Cosine	85.465	82.297	88.934
BERT - Scaled - Transformer	86.139	83.400	89.069
BERT - Unscaled - AdaTrans	85.324	83.267	87.494
BERT - BiLSTM	84.635	**86.903**	82.501
BERT - Unscaled - Cosine	85.436	83.368	87.629
BERT - Unscaled - Transformer	85.977	83.945	88.124
GloVe - Scaled - AdaTrans	**87.771**	85.604	**90.058**
GloVe - Scaled - Cosine	85.338	82.967	87.854
GloVe - Scaled - Transformer	86.179	84.131	88.349
GloVe - Unscaled - AdaTrans	87.441	86.726	88.169
GloVe - BiLSTM	85.766	85.883	85.650
GloVe - Unscaled - Cosine	84.742	82.444	87.224
GloVe - Unscaled - Transformer	85.081	84.356	85.830

(FP), and false negative (FN) for each label. Since there are seven labels in the dataset with an imbalance proportion, we use the micro-average approach to compute the final evaluation score. The formula for per entity type precision, recall, and F1 are shown in (22), (23), and (24), respectively, where c is the entity type, and C is the total number of entity types exists in the dataset. Micro-average for the precision and recall are identical to the per class formula, but the TP, FP, and FP are the sum from all classes as in (25) and (26). While (27) is used to calculate the micro-average F1 score. For all of these evaluation metrics, we used the pre-defined function `SpanFPreRecMetric` in fastNLP library.[8] The ε symbol is a small number to avoid division by zero error, while β is a term to weigh between precision and recall in order to obtain the F1 score. In this paper, we use $\varepsilon = 1e-13$ and $\beta = 1$.

$$\text{pre} = \frac{\text{TP}}{\text{TP} + \text{FP} + \varepsilon} \tag{22}$$

$$\text{rec} = \frac{\text{TP}}{\text{TP} + \text{FN} + \varepsilon} \tag{23}$$

$$\text{F1} = \frac{2 \times (\text{pre} \times \text{rec})}{\text{pre} + \text{rec} + \varepsilon} \tag{24}$$

$$\text{pre}_{\text{micro}} = \frac{\sum_{c=1}^{C} \text{TP}_c}{\sum_{c=1}^{C} \text{TP}_c + \sum_{c=1}^{C} \text{FP}_c + \varepsilon} \tag{25}$$

$$\text{rec}_{\text{micro}} = \frac{\sum_{c=1}^{C} \text{TP}_c}{\sum_{c=1}^{C} \text{TP}_c + \sum_{c=1}^{C} \text{FN}_c + \varepsilon} \tag{26}$$

$$\text{F1}_{\text{micro}} = \frac{(1+\beta) \times \text{pre}_{\text{micro}} \times \text{rec}_{\text{micro}}}{\beta \times \text{pre}_{\text{micro}} \times \text{rec}_{\text{micro}} + \varepsilon} \tag{27}$$

C. RESULTS ON DIFFERENT ANNOTATION RULES

This subsection presents all the possible arrangements from the available options of character embedding, word embedding, and attention mechanism. Since two types of datasets are constructed, every architectural arrangement is tested on these two datasets. Table 4 to 6 shows the first dataset's evaluation scores, which were annotated using a non-contextual tagging procedure. The model with the best performance is highlighted a bold font. The presented scores in the tables contain both proposed and baseline models. Each table represents a scenario that is grouped based on the character embedding used, as Table 4 shows the models employing AdaTrans for extracting character embedding. Subsequently, Table 5 and 6 show the models' architecture where LSTM and CNN were used as the character embedding, respectively.

From the evaluation score presented in Table 4 to 6, the best performance was achieved by GloVe – Scaled AdaTrans combination with an 87.771% F1 score. AdaTrans attention consistently achieves the highest score for all character embedding and word embedding options. GloVe outperforms the BERT embedding evaluated on the non-contextual dataset for all scenarios. This is because the first dataset has consistent tagging, meaning that a word has a consistent tag for all different contexts in the dataset. It complies with the GloVe behavior, where each word has exactly one representational vector. BERT – Scaled Transformer achieved the best overall performance for the second dataset with a 91.348% F1 score. The annotation procedure in the second dataset complies with the contextual representation resulting from

[8] https://fastnlp.readthedocs.io/zh/latest/fastNLP.core.metrics.html

TABLE 7. Performance evaluation of all scenarios on the dataset annotated using contextual tagging with AdaTrans as the char-level embedding. The best score is indicated in bold font. BERT and GloVe are used as word embedding.

Model	F1-Score	Precision	Recall
BERT - Scaled - Transformer	90.669	88.789	92.633
BERT - Scaled - AdaTrans	90.514	88.970	92.117
BERT - Scaled - Cosine	90.661	88.631	92.787
BERT - Unscaled - Transformer	90.873	**89.232**	92.581
BERT - Unscaled - AdaTrans	90.139	88.294	92.066
BERT - Unscaled - Cosine	**91.146**	88.808	**93.612**
BERT - BiLSTM	88.387	88.737	88.047
GloVe - Scaled - Transformer	86.849	86.939	86.759
GloVe - Scaled - AdaTrans	87.378	87.906	86.862
GloVe - Scaled - Cosine	85.341	85.484	85.214
GloVe - Unscaled - Transformer	87.567	87.570	87.584
GloVe - Unscaled - AdaTrans	87.529	87.374	87.687
GloVe - Unscaled - Cosine	87.247	87.435	87.069
GloVe - BiLSTM	85.320	86.547	84.132

TABLE 8. Performance evaluation of all scenarios on the dataset annotated using contextual tagging with LSTM as the char-level embedding. The best score is indicated in bold font. BERT and GloVe are used as word embedding.

Model	F1-Score	Precision	Recall
BERT - Scaled - Transformer	90.382	**89.253**	91.551
BERT - Scaled - AdaTrans	90.233	88.191	92.375
BERT - Scaled - Cosine	**90.885**	88.872	**92.993**
BERT - Unscaled - Transformer	90.207	88.813	91.654
BERT - Unscaled - AdaTrans	88.246	85.902	90.726
BERT - Unscaled - Cosine	90.846	89.180	92.581
BERT - BiLSTM	88.382	88.783	87.996
GloVe - Scaled - Transformer	85.662	85.596	85.729
GloVe - Scaled - AdaTrans	86.107	85.778	86.450
GloVe - Scaled - Cosine	84.923	83.691	86.193
GloVe - Unscaled - Transformer	85.550	84.640	86.502
GloVe - Unscaled - AdaTrans	87.501	87.276	87.738
GloVe - Unscaled - Cosine	84.326	83.319	85.368
GloVe - BiLSTM	85.117	86.452	83.823

TABLE 9. Performance evaluation of all scenarios on the dataset annotated using contextual tagging with CNN as the char-level embedding. The best score is indicated in bold font. BERT and GloVe are used as word embedding.

Model	F1-Score	Precision	Recall
BERT - Scaled - Transformer	**91.348**	**89.239**	**93.560**
BERT - Scaled - AdaTrans	90.447	88.983	91.963
BERT - Scaled - Cosine	90.665	88.404	93.045
BERT - Unscaled - Transformer	90.502	88.994	92.066
BERT - Unscaled - AdaTrans	88.795	87.144	90.520
BERT - Unscaled - Cosine	90.881	88.909	92.942
BERT - BiLSTM	88.055	87.612	88.511
GloVe - Scaled - Transformer	85.127	84.496	85.781
GloVe - Scaled - AdaTrans	84.501	84.993	84.029
GloVe - Scaled - Cosine	85.322	84.719	85.935
GloVe - Unscaled - Transformer	85.808	86.290	85.368
GloVe - Unscaled - AdaTrans	87.519	87.768	87.275
GloVe - Unscaled - Cosine	85.384	84.693	86.090
GloVe - BiLSTM	86.088	87.581	84.647

BERT embedding, where every word has one representational vector for each distinct context in the dataset. This claim is supported by the experimental results shown in Table 7 to 9, where the scenarios that utilized BERT as the word embedding outperform the models that employed GloVe as the word embedding. As the dataset annotated using contextual tagging procedures better represents the semantics of a span, the following subsection discusses only the results of the contextual dataset.

D. ATTENTION MECHANISM IN COMPARISON

An attention-based model has been widely used in NLP research to model the context between words within a sentence. In this experiment, we investigate three different attention mechanisms to recognize mentioned entities in drone log messages. After conducting an extensive experiment, we obtain the results as shown in Table 4 to 9. The architecture arrangements are inspired by the TENER paper, which proposed AdaTrans to extract character-level features and incorporate relative positional to the attention layer. However, several scenarios have not been reported yet. Therefore, we experiment to find the best architecture to use on our dataset. The details explanation of every scenario is as follows.

Overall, the model's architecture consists of five layers, i.e., positional encoding, char embedding, word embedding, encoder, and decoder. Relative positional encoding proposed in TENER is used to reproduce the AdaTrans' unexplored scenarios. To obtain character-level embedding, either CNN, LSTM, or AdaTrans is utilized in every scenario as listed in Table 7 to 9. For the pre-trained word embedding, either GloVe or BERT is used to obtain the words' vector representation. Unscaled attention is reported to be better used in NER since mentioned entities commonly consist of a few words only [37]. Thus we experiment with every attention type with the scaled and unscaled scenario in the encoder layer, including the AdaTrans encoder. The employment of CRF can undoubtedly improve the performance of a NER model [56]. Thus CRF is used as the decoder for all scenarios arrangements.

The CNN-BERT-Scaled Transformer outperforms the other scenario with a 91.348% F1 score. This score is slightly higher than the unscaled Transformer. We assume this slight difference is because of the dataset size, so the effect of either using scaling or not is insignificant. The AdaTrans-based encoder is considered a baseline model, which will be discussed in the following subsection. Our proposed model that uses cosine similarity instead of dot-product operation in the self-attention mechanism underperforms the Transformer with a competitive F1 score of 91.146%. This shows that the cosine similarity is able to model the context between words in a sentence, just like the dot-product intuition in the self-attention mechanism.

The presence of contextual pre-trained word embedding has positively impacted NLP research recently. The main advantage of contextual over static pre-trained word embedding is the ability to generate a unique representational vector of a word for each distinct context within two or more different sentences. In this experiment, these two types of pre-trained word embeddings were employed. From the

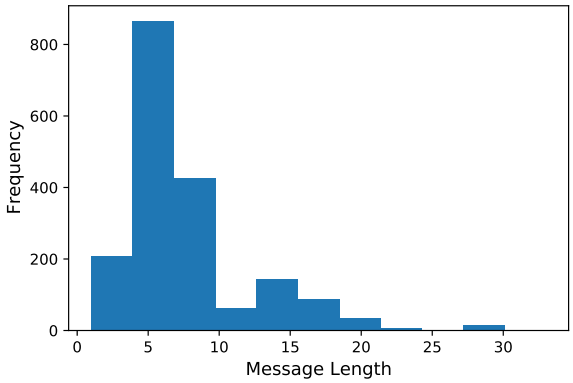

FIGURE 9. Message length distribution in the dataset.

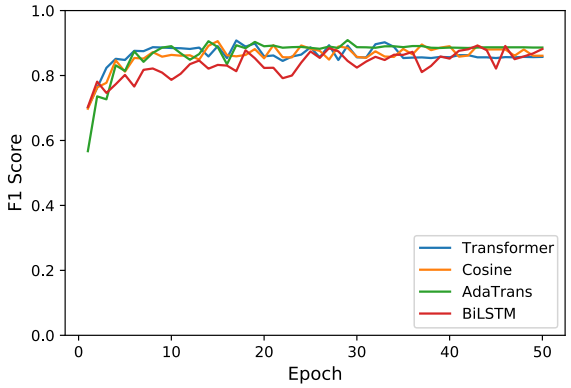

FIGURE 11. Convergence speed of the best model for each encoder on the train set of the contextual dataset.

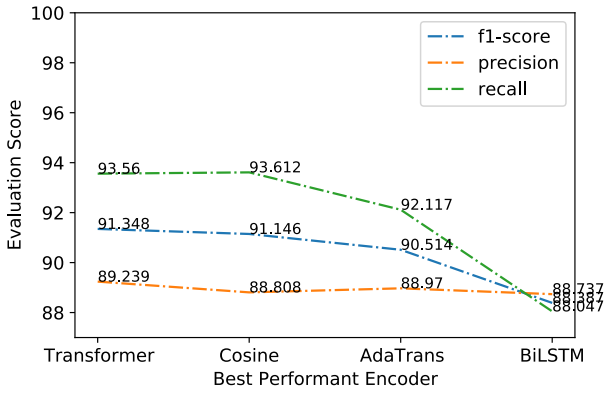

FIGURE 10. Comparison between different encoders with the best performance on the test set of the contextual dataset.

results reported in Table 7 to 9, the involvement of either contextual or static embedding significantly affects the model's performance evaluation score. This can be seen from the dot-product attention performance, with a 3.782% difference between the best dot-product attention that uses BERT and GloVe. This significant difference is also happening to the cosine attention, with a 3.899% difference. The other scenarios show consistent results, where a model with static and contextual pre-trained word embedding has a significant difference in the evaluation score. The process of capturing context that exists between words in a sentence also occurs in the encoder layer, which is performed by the self-attention mechanism. Therefore, the representational vector obtained from the static embedding is undergoing a refinement process in the encoder layer. Eventually, the words' vectors from the embedding layer went through a contextual learning pipeline, just like what the contextual pre-trained word embedding has done. Contextual vector representation from BERT fits the intuition of contextual learning in the self-attention mechanism by means that the contextually related words within a sentence are close to one another in the representational space. Therefore, using a static and contextual embedding in a Transformer-based model resulted in significantly different evaluation scores. Nevertheless, the experimental results show an insignificant effect of using different character-level embedding on the models' performance.

The scale factor implemented in scaled dot-product attention, as proposed in the vanilla Transformer, has been argued better not be used in the NER task [37]. The reason is to sharpen the probability distribution yielded by the softmax in the self-attention mechanism. The sharper the attention, the fewer contexts are captured by the attention, aligning to the span length of mentioned entities commonly exist [37]. However, the experimental results in Table 8 and 9 show a contradictive point. Consider the following points. First, the average sentence length is 6.3 and 8.8 in the train and test sets, respectively. Secondly, the sentence length is dominated by lengths ranging from one to ten, with more than 80% of the portion, as shown in Fig. 9. Thus, it is unlikely that the sentence contains several contexts. Therefore, unscaled attention is supposedly better if used instead of scaled attention. However, the experimental results demonstrated the contrary on dot-product and AdaTrans attention. The scaled attention for dot-product and AdaTrans attention achieved better performance than unscaled ones. Contrary, the cosine has better performance with unscaled attention. As shown in Fig. 5, the element-wise norm scale operation played the same role as the scale factor in scaled dot-product attention. Therefore, a scaling factor is needed in the self-attention mechanism and has proven to improve the model's performance compared to unscaled attention.

E. COMPARISON WITH OTHER BASELINE MODELS

To verify the superiority of our proposed methods, we compare the proposed models with several baseline models, as shown in Fig 10. The detailed architecture for each encoder is as follows: CNN-BERT-Scaled Transformer, AdaTrans-BERT-Unscaled Cosine, AdaTrans-BERT-Scaled AdaTrans, and AdaTrans-BERT-BiLSTM. For all of these encoders, CRF is used as the decoder. In terms of convergence speed, as depicted in Fig. 11, the proposed method converges as fast as Transformer and AdaTrans. Moreover, cosine attention outperforms the BiLSTM model. From the F1 score, our proposed method achieves the second-best performance

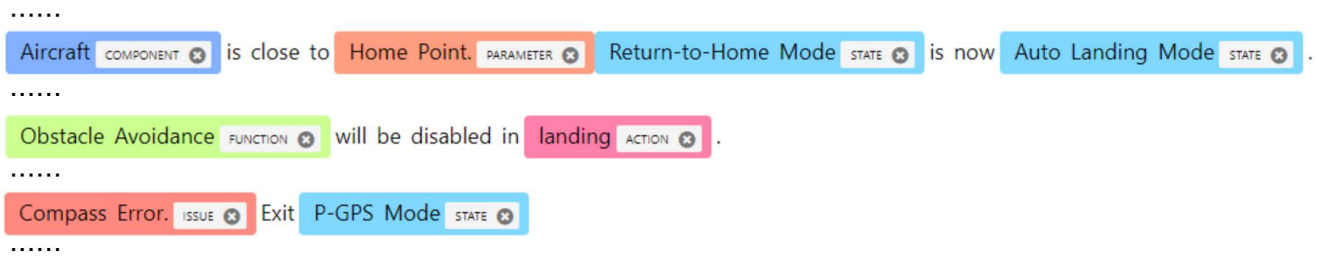

FIGURE 12. Illustration of highlighted entities mentioned in a flight log message. The color highlight is used to help the investigator pinpoint the critical information within a log message.

with a 91.146% F1 score. This model uses the AdaTrans as the character-level feature extractor, concatenated with the output of BERT as the pre-trained word embedding to get a word-level feature vector. The unscaled cosine attention is used as the encoder and CRF as the decoder. The scaled Transformer model achieved the best performance, with a 91.348% F1 score. This scenario consists of CNN char embedding, BERT word embedding, scaled dot-product attention, and CRF as the decoder. In comparison, the unscaled AdaTrans attention is in the third position, accompanied by AdaTrans as the character embedding and BERT as the word embedding, with a 90.514% F1 score. This proves that the relative attention mechanism is unsuitable for our case since our dataset has a relatively short sequence, with 6.3 and 8.8 words in length on average in train and test data, respectively.

Our proposed model underperforms the scaled dot-product attention with a 0.202% difference in the F1 score. However, from the recall score, our proposed model outperforms all the baseline models with a 93.612% recall score. This shows that the proposed method has the lowest False Negative rate, where the number of misclassification on entities are small. It means that the mentioned entities in the datasets are mostly correctly classified. Therefore, the proposed method successfully recognizes the region of interest in the log message.

The evaluation score indicates that the proposed model can recognize mentioned entities in flight log message data. When a forensic investigator conducts an evidence analysis process, plenty of evidence must be examined, analyzed, and evaluated. To this end, presenting the NER result in a sophisticated visualization can help the investigator pinpoint the region of interest in flight log data faster. Fig. 12 shows a sample log message that has been fed to the NER model in a visualization form to assist the forensic investigation. Having the mentioned entities highlighted with a particular color, the investigator can ignore the message with no highlights and focus only on those with color highlights. The color can be set to represent a level of importance. For instance, red can be used to highlight the Issue entity type. The highlight can help the forensic investigator find a message containing words or phrases with the Issue label.

F. CHALLENGES AND LIMITATIONS

After conducting the experiment, we described several challenges in the following. Since drone forensics is a relatively new research domain, a few open drone image datasets are available. We only found one drone image dataset, the VTO Labs Drone Forensic dataset. Even from 15 different models and more than 20 datasets, we only discover less than 2000 log messages. Moreover, the dataset does not contain any specific drone incident scenario. It implies that no ground truth can be used to test the proposed method regarding the forensic investigation, finding, and reporting view. Since this is an initial attempt on NER for drone forensics, there are few references, datasets, and domain-specific knowledge, such as entity types related to incidents and regions of interest in drone log messages. Therefore, many opportunities are opened by this attempt in the future, which will be our next project. Considering the time needed to perform a thorough analysis for one drone model, it is unrealistic to include other drone models. Besides, DJI has the largest market share, and the availability of a public dataset is one of the considerations for this research to be verifiable and reproducible.

V. CONCLUSION AND FUTURE WORKS

In this research, we have experimented with the employment of cosine similarity as a substitute for dot-product self-attention in the encoder sub-layer of Transformer architecture. To evaluate our proposed approach, we construct our own NER dataset by manually extracting several drone forensic image datasets that are publicly available from the VTO Labs. For a relatively small dataset, we obtain a good result indicated by the F1 score of 91.348% achieved by the dot-product attention supported by CNN character embedding and BERT word embedding. Our proposed approach outperforms the RNN-based state-of-the-art by achieving the F1 score of 91.146%. The proposed model can achieve high scores even if the test data are generated from different drone models. This proves that NER can be used as an extraction tool to assist the forensic investigation by only utilizing the log message data to recognize some incident-related information.

We plan to further analyze the trade-off between the convergence speed with the decrease in the number of parameters

in the simpler architecture when using cosine as the attention type. Since the number of epochs is a one-time cost, and the inference time is a repetitive cost, we plan to explore further how many epochs are needed to train the simpler model having fewer parameters after employing the cosine in the self-attention sublayer without losing performance. As this research is still an initial step in information extraction, we plan to deploy the NER model so that it can be used as a practical solution for the forensic investigator.

REFERENCES

[1] F. Laricchia. (2022). *Consumer Drone Unit Shipments Worldwide From 2020 to 2030*. [Online]. Available: https://www.statista.com/statistics/1234658/worldwide-consumer-drone-unit-shipments

[2] J.-P. Yaacoub, H. Noura, O. Salman, and A. Chehab, "Security analysis of drones systems: Attacks, limitations, and recommendations," *Internet Things*, vol. 11, Sep. 2020, Art. no. 100218.

[3] A. Al-Dhaqm, R. A. Ikuesan, V. R. Kebande, S. Razak, and F. M. Ghabban, "Research challenges and opportunities in drone forensics models," *Electronics*, vol. 10, no. 13, p. 1519, Jun. 2021.

[4] F. Iqbal, B. Yankson, M. A. AlYammahi, N. AlMansoori, S. M. Qayed, B. Shah, and T. Baker, "Drone forensics: Examination and analysis," *Int. J. Electron. Secur. Digit. Forensics*, vol. 11, no. 3, pp. 245–264, 2019.

[5] G. Thornton and P. B. Zadeh, "An investigation into unmanned aerial system (UAS) forensics: Data extraction & analysis," *Forensic Sci. Int., Digit. Invest.*, vol. 41, Jun. 2022, Art. no. 301379.

[6] E. Mantas and C. Patsakis, "GRYPHON: Drone forensics in dataflash and telemetry logs," in *Advances in Information and Computer Security*, N. Attrapadung and T. Yagi, Eds. Cham, Switzerland: Springer, 2019, pp. 377–390.

[7] D. R. Clark, C. Meffert, I. Baggili, and F. Breitinger, "DROP (DRone open source parser) your drone: Forensic analysis of the DJI phantom III," *Digit. Invest.*, vol. 22, pp. S3–S14, Aug. 2017.

[8] F. E. Salamh, M. M. Mirza, and U. Karabiyik, "UAV forensic analysis and software tools assessment: DJI phantom 4 and matrice 210 as case studies," *Electronics*, vol. 10, no. 6, p. 733, Mar. 2021.

[9] M. Maarse, L. Sangers, J. van Ginkel, and M. Pouw, "Digital forensics on a DJI phantom 2 Vision + UAV," Univ. Amsterdam, Amsterdam, The Netherlands, Tech. Rep., 2016.

[10] M. Yousef and F. Iqbal, "Drone forensics: A case study on a DJI Mavic air," in *Proc. IEEE/ACS 16th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Nov. 2019, pp. 1–3.

[11] I. McAteer, P. Hannay, M. Malik, and Z. Baig, "Forensic analysis of a crash-damaged Cheerson CX-20 auto pathfinder drone," *J. Digit. Forensics, Secur. Law*, vol. 13, no. 4, pp. 5–22, 2018.

[12] F. E. Salamh, U. Karabiyik, and M. K. Rogers, "RPAS forensic validation analysis towards a technical investigation process: A case study of Yuneec Typhoon H," *Sensors*, vol. 19, no. 15, p. 3246, Jul. 2019.

[13] Q. H. Ngo, T. Kechadi, and N.-A. Le-Khac, "Domain specific entity recognition with semantic-based deep learning approach," *IEEE Access*, vol. 9, pp. 152892–152902, 2021.

[14] N. Perera, T. T. L. Nguyen, M. Dehmer, and F. Emmert-Streib, "Comparison of text mining models for food and dietary constituent named-entity recognition," *Mach. Learn. Knowl. Extraction*, vol. 4, no. 1, pp. 254–275, Mar. 2022.

[15] M. Asghari, D. Sierra-Sosa, and A. S. Elmaghraby, "BINER: A low-cost biomedical named entity recognition," *Inf. Sci.*, vol. 602, pp. 184–200, Jul. 2022.

[16] Y. Tong, F. Zhuang, D. Wang, H. Ying, and B. Wang, "Improving biomedical named entity recognition with a unified multi-task MRC framework," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2022, pp. 8332–8336.

[17] Z. Zhai, D. Q. Nguyen, S. A. Akhondi, C. Thorne, C. Druckenbrodt, T. Cohn, M. Gregory, and K. Verspoor, "Improving chemical named entity recognition in patents with contextualized word embeddings," 2019, *arXiv:1907.02679*.

[18] P. Ma, B. Jiang, Z. Lu, N. Li, and Z. Jiang, "Cybersecurity named entity recognition using bidirectional long short-term memory with conditional random fields," *Tsinghua Sci. Technol.*, vol. 26, no. 3, pp. 259–265, Jun. 2021.

[19] C. Gao, X. Zhang, and H. Liu, "Data and knowledge-driven named entity recognition for cyber security," *Cybersecurity*, vol. 4, no. 1, pp. 1–13, May 2021.

[20] S. Zhou, J. Liu, X. Zhong, and W. Zhao, "Named entity recognition using BERT with whole world masking in cybersecurity domain," in *Proc. IEEE 6th Int. Conf. Big Data Anal. (ICBDA)*, Mar. 2021, pp. 316–320.

[21] H. Studiawan, F. Sohel, and C. Payne, "Sentiment analysis in a forensic timeline with deep learning," *IEEE Access*, vol. 8, pp. 60664–60675, 2020.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, MN, USA, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[23] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.

[24] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," 2019, *arXiv:1910.01108*.

[25] T. B. Brown et al., "Language models are few-shot learners," 2020, *arXiv:2005.14165*.

[26] D.-Y. Kao, M.-C. Chen, W.-Y. Wu, J.-S. Lin, C.-H. Chen, and F. Tsai, "Drone forensic investigation: DJI spark drone as a case study," *Proc. Comput. Sci.*, vol. 159, pp. 1890–1899, Jan. 2019.

[27] T. E. A. Barton and M. A. H. B. Azhar, "Forensic analysis of popular UAV systems," in *Proc. 7th Int. Conf. Emerg. Secur. Technol. (EST)*, Sep. 2017, pp. 91–96.

[28] S. Viswanathan and Z. Baig, "Digital forensics for drones: A study of tools and techniques," in *Applications and Techniques in Information Security*, L. Batina and G. Li, Eds. Singapore: Springer, 2020, pp. 29–41.

[29] M. Debinski, F. Breitinger, and P. Mohan, "Timeline2GUI: A Log2Timeline CSV parser and training scenarios," *Digit. Invest.*, vol. 28, pp. 34–43, Mar. 2019.

[30] U. Jain, M. Rogers, and E. T. Matson, "Drone forensic framework: Sensor and data identification and verification," in *Proc. IEEE Sensors Appl. Symp. (SAS)*, Mar. 2017, pp. 1–6.

[31] R. Kumar and A. K. Agrawal, "Drone GPS data analysis for flight path reconstruction: A study on DJI, parrot & Yuneec make drones," *Forensic Sci. Int., Digit. Invest.*, vol. 38, Sep. 2021, Art. no. 301182.

[32] Y. Lou, T. Qian, F. Li, and D. Ji, "A graph attention model for dictionary-guided named entity recognition," *IEEE Access*, vol. 8, pp. 71584–71592, 2020.

[33] T.-M. Georgescu, B. Iancu, A. Zamfiroiu, M. Doinea, C. E. Boja, and C. Cartas, "A survey on named entity recognition solutions applied for cybersecurity-related text processing," in *Proc. 5th Int. Congr. Inf. Commun. Technol.*, X.-S. Yang, S. Sherratt, N. Dey, and A. Joshi, Eds. Singapore: Springer, 2021, pp. 316–325.

[34] H. Gasmi and A. Bouras, "LSTM recurrent neural networks for cybersecurity named entity recognition," in *Proc. 13th Int. Conf. Softw. Eng. Adv.*, 2018, pp. 1–15.

[35] H. Studiawan, F. Sohel, and C. N. Payne, "Automatic log parser to support forensic analysis," in *Proc. 16th Austral. Digit. Forensics Conf.*, 2018, pp. 1–10.

[36] M. Affi and C. Latiri, "BE-BLC: BERT-ELMO-based deep neural network architecture for English named entity recognition task," *Proc. Comput. Sci.*, vol. 192, pp. 168–181, Jan. 2021.

[37] H. Yan, B. Deng, X. Li, and X. Qiu, "TENER: Adapting transformer encoder for named entity recognition," 2019, *arXiv:1911.04474*.

[38] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto, "LUKE: Deep contextualized entity representations with entity-aware self-attention," 2020, *arXiv:2010.01057*.

[39] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, vol. 19, no. 5, pp. 1532–1543.

[40] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. Int. Conf. Learn. Represent.*, Jun. 2013, pp. 1–12.

[41] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *J. Assoc. Comput. Linguistics*, vol. 1, pp. 2227–2237, Mar. 2018.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*.

[43] H. Chen, Z. Lin, G. Ding, J. Lou, Y. Zhang, and B. Karlsson, "GRN: Gated relation network to enhance convolutional neural network for named entity recognition," 2019, *arXiv:1907.05611*.
[44] A. Ghaddar and P. Langlais, "Robust lexical features for improved neural network named-entity recognition," in *Proc. 27th Int. Conf. Comput. Linguistics*. Santa Fe, NM, USA, Aug. 2018, pp. 1896–1907.
[45] M. Maryamah and I. T. S. Nopember, "Pseudo-relevance feedback combining statistical and semantic term extraction for searching Arabic documents," *Int. J. Intell. Eng. Syst.*, vol. 14, no. 5, pp. 238–246, Oct. 2021.
[46] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," 2016, *arXiv:1607.04606*.
[47] J. Wang, A. Jatowt, and M. Yoshikawa, "TimeBERT: Extending pre-trained language representations with temporal information," 2022, *arXiv:2204.13032*.
[48] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. 3rd Int. Conf. Learn. Represent.*, Y. Bengio and Y. LeCun, Eds. San Diego, CA, USA, 2015, pp. 1–15.
[49] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," 2015, *arXiv:1508.04025*.
[50] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," 2018, *arXiv:1803.02155*.
[51] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," 2019, *arXiv:1901.02860*.
[52] C. Luo, J. Zhan, X. Xue, L. Wang, R. Ren, and Q. Yang, "Cosine normalization: Using cosine similarity instead of dot product in neural networks," in *Artificial Neural Networks and Machine Learning—ICANN 2018*. Cham, Switzerland: Springer, 2018, pp. 382–391.
[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.
[54] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
[55] A. F. Agarap, "Deep learning using rectified linear units (ReLU)," 2018, *arXiv:1803.08375*.
[56] Z. Huang, W. Xu, and K. Yu, "Bidirectional LSTM-CRF models for sequence tagging," 2015, *arXiv:1508.01991*.
[57] D. Slotta. (2021). *Global Market Share of Consumer and Commercial Drone Manufacturers in March 2021, Based on Sales Volume*. [Online]. Available: https://www.statista.com/statistics/1254982/global-market-share-of-drone-manufacturers/
[58] E. Mantas and C. Patsakis, "Who watches the new watchmen? The challenges for drone digital forensics investigations," *Array*, vol. 14, Jul. 2022, Art. no. 100135.
[59] N. Alshammari and S. Alanazi, "The impact of using different annotation schemes on named entity recognition," *Egyptian Informat. J.*, vol. 22, no. 3, pp. 295–302, Sep. 2021.

SWARDIANTARA SILALAHI (Member, IEEE) was born in Pekanbaru, Riau, Indonesia, in 1997. He received the B.Ed. degree in informatics education from Universitas Negeri Jakarta, Jakarta, Indonesia, in 2021. He is currently pursuing the Ph.D. degree in computer science with the Institut Teknologi Sepuluh Nopember (ITS), Indonesia.

He was a Production Support Engineer at Danamon Bank, from 2020 to 2021. Since September 2021, he has been a Research Assistant with the Net-Centric Computing Laboratory. He has several published works in the digital forensics area. His research interests include digital forensics, log mining, natural language processing, and deep learning. He is an awardee of the PMDSU Scholarship by the Ministry of Education, Culture, Research and Technology, Indonesia.

TOHARI AHMAD (Member, IEEE) received the bachelor's degree in computer science from the Institut Teknologi Sepuluh Nopember (ITS), Indonesia, the master's 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 currently a Professor. His research interests include network security, information security, data hiding, and computer networks.

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

HUDAN STUDIAWAN (Member, IEEE) received the bachelor's and master's degrees from the Institut Teknologi Sepuluh Nopember, Indonesia, in 2009 and 2011, respectively, and the Ph.D. degree from Murdoch University, Australia, in 2021. He is currently a Lecturer with the Institut Teknologi Sepuluh Nopember. His current research interests include digital forensics and natural language processing.