Journal of Cybersecurity and Privacy

MDPI

Article

An Accuracy-Maximization Approach for Claims Classifiers in Document Content Analytics for Cybersecurity †

Kimia Ameri [1], Michael Hempel [1], Hamid Sharif [1,*], Juan Lopez Jr. [2] and Kalyan Perumalla [2]

[1] Department of Electrical & Computer Engineering, University of Nebraska-Lincoln, Lincoln, NE 68182, USA; kameri2@unl.edu (K.A.); mhempel@unl.edu (M.H.)
[2] Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA; lopezj@ornl.gov (J.L.J.); perumallaks@ornl.gov (K.P.)
* Correspondence: hsharif@unl.edu
† This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The publisher acknowledges the US government license to provide public access under the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

Abstract: This paper presents our research approach and findings towards maximizing the accuracy of our classifier of feature claims for cybersecurity literature analytics, and introduces the resulting model ClaimsBERT. Its architecture, after extensive evaluations of different approaches, introduces a feature map concatenated with a Bidirectional Encoder Representation from Transformers (BERT) model. We discuss deployment of this new concept and the research insights that resulted in the selection of Convolution Neural Networks for its feature mapping aspects. We also present our results showing ClaimsBERT to outperform all other evaluated approaches. This new claims classifier represents an essential processing stage within our vetting framework aiming to improve the cybersecurity of industrial control systems (ICS). Furthermore, in order to maximize the accuracy of our new ClaimsBERT classifier, we propose an approach for optimal architecture selection and determination of optimized hyperparameters, in particular the best learning rate, number of convolutions, filter sizes, activation function, the number of dense layers, as well as the number of neurons and the drop-out rate for each layer. Fine-tuning these hyperparameters within our model led to an increase in classification accuracy from 76% obtained with BertForSequenceClassification's original model to a 97% accuracy obtained with ClaimsBERT.

Keywords: natural language processing; BERT; transfer learning; convolution neural network; classification; cybersecurity; CYVET; accuracy maximization

Citation: Ameri, K.; Hempel, M.; Sharif, H.; Lopez Jr., J.; Perumalla, K. An Accuracy-Maximization Approach for Claims Classifiers in Document Content Analytics for Cybersecurity. *J. Cybersecur. Priv.* **2022**, 2, 418–443. https://doi.org/10.3390/jcp2020022

Academic Editor: Giorgio Giacinto

Received: 14 March 2022
Accepted: 10 June 2022
Published: 15 June 2022

1. Introduction

1.1. Motivation

Operational Technology (OT) audits are vital for determining whether cybersecurity requirements are being met by OT devices, or whether they may actually deteriorate an operator's overall cybersecurity posture. A cybersecurity assessment evaluates that impact, both pre-deployment as well as post-integration into an existing system, for example, when firmware changes occur. Many critical infrastructure sectors, including the energy sector, are seeing an increased reliance on automation and centralized operational controls [1]. While ICS vendors consistently roll out new features in their products to attract customers, consumers are not always aware of the implications of these updated features, or how these feature set changes or updates can alter their cybersecurity posture, as well as potentially impacting their regulatory compliance [2]. In our previous work [1], we have shown that different vendor-supplied features (VSF) can meet, exceed, or degrade corresponding cybersecurity requirements (CR). A vetting system that effectively matches, reconciles, and tallies vendor-supplied features against relevant cybersecurity requirements will help solve this critical challenge in cybersecurity assessment. Both features and requirements are

typically provided in the form of human-readable documents such as PDFs. However, the manifestation of these elements in document form complicates the automation of cybersecurity vetting processes because of their natural language structures and representation formats [1,3]. In order to address this issue, we are in the process of researching a semi-automated cybersecurity vetting engine (CYVET) [1] aimed at cybersecurity assurance for cyber-physical systems. A major objective of CYVET is to improve the current capabilities of the energy sector and other OT systems operators to verify and validate OT infrastructure cybersecurity claims, both prior to deployment and after deployment. In Figure 1, we illustrate the high-level processing tasks involved in CYVET's tallying workflow.

Figure 1. Overall workflow of the CYVET cybersecurity vetting framework.

From this figure we can clearly observe that any errors made by the claims detector impact virtually all of the framework's processing stages. Given its positioning within our framework, a claim classifier is needed to achieve the highest possible classification accuracy so as not to deteriorate downstream performance of subsequent framework tasks. Therefore, it is vitally important for the claims detector to achieve maximum accuracy for the target domain. This challenge forms the core of the research presented in this paper. Throughout the entire framework we make extensive use of Natural Language Processing (NLP). It utilizes automated methods to evaluate vendor claims that were identified from the text content extracted from device documentation in order to achieve an unbiased and objective evaluation of cybersecurity ramifications related to the ICS device being evaluated. Through the implementation of our CYVET vetting system, we can significantly simplify ICS compliance analysis as well as preparing for mandated reporting, providing insights into the capabilities and drawbacks of ICS systems, as well as conducting a capabilities comparison against stated operational requirements [1]. As seen in Figure 1 and described in our previous work [2], our framework utilizes an extensive document library, curated from vendor product documents across the OT vendor space, with a particular focus on vendors supplying the energy sector. In [2], we provided extensive details on how our frame constructs its ICS device information repository. This repository contains documents related to ICS devices, including manuals, catalogs, and brochures. An important challenge of the vetting process is the analysis of this data repository and the identification of sequences extracted from vendor-supplied documents that represent the vendor's *stated claims* related to specific product features. The results obtained from this processing step are vital to all downstream tasks within our framework. Thus, any errors in claims classification negatively impact our overall system performance. This motivated out effort to research a methodology that helps us maximize the claims detector's accuracy through variations both in its architecture and in its hyperparameters.

1.2. Goals and Contributions

The purpose of this article is to introduce our work in developing a unique and efficient method of classifying *cybersecurity feature claims* (henceforth simply referred to as "claims"). The final architecture resulting from the research presented in this paper that is aimed at determining a model achieving maximum classification accuracy is based on a pre-trained BERT language model combined with a Convolution Neural Network (CNN) to create a feature map for extracting informative features from transformers and then assigning them to a classifier. Specifically, for our work we define a claim as any sequence that expresses a claim in regards to the availability, or lack of availability, of a cybersecurity feature in a given product. Sequences that make unrelated claims or that are not making any such claims, are subsequently labeled as not containing a cybersecurity claim. As part of this research we studies the impact of various different NN feature map approaches, different activation functions, and more.

The resulting ClaimsBERT is a classification model enabling our framework to automatically identify any cybersecurity claims present in ICS device documents. The contributions of this work are summarized in the following items:

1. We introduce a new concept of ClaimsBERT to significantly enhance base model BertForSequenceClassification, by integrating CNN feature maps in order to improve performance and accuracy;
2. We present our findings obtained using extensive experiments for optimizing our model's overall architecture and its hyperparameters;
3. We discuss the effectiveness of a suitable feature map in parameter selection in our proposed architecture for ClaimsBERT;
4. We show that our model achieves a high classification accuracy of 97 percent;
5. We discuss the results of our extensive evaluation of ClaimsBERT and its performance comparison with other network-based BERT models. The results indicate that significantly higher accuracy is obtained when integrating CNN into original BERT classifier and fine-tune all layers.

2. Related Works

A pre-trained language model refers to a model of NLP that has been trained using an unsupervised training approach on a large text corpus that represents a general domain of language. Among the most well-established pre-trained language models are Embeddings from language Models (ElMo) [4], the Universal Language Model with Fine-Tuning (ULMFiT) [5], Bidirectional Encoder Representations from Transformers (BERT) [6], and the Generative Pre-Training (GPT) model [7].

Among all mentioned language models, we chose BERT for our work because it is an open-source model with a very strong tokenizer and word-embedding matrix. In our previous work we fine-tune BERT with neural network to build cybersecurity claim sequence classifier CyBERT [8], and show that its design (BERT+NN) improves upon the performance obtainable from other language models such as GPT2 and ULMFiT. From Figure 1 it is readily apparent that due to its very early positioning within the overall processing workflow the overall accuracy of our vetting system is highly dependent on the accuracy obtained by the claims classifier. Any errors made during classification are propagated downstream through all subsequent processing tasks. Therefore, in this is research we focus on maximizing the claim classifier's accuracy.

BERT uses a contextualized embedding technique that is designed to capture word semantics in a context based on its surrounding text [9]. BERT uses the WordPiece embedding method, which divides each word into a limited set of common sub-words [10], down to the individual letter level. This technique is very flexible and eliminates the need to deal with unfamiliar words contained within a dataset.

BERT is a multi-layer bidirectional transformer-stacked encoder based on an attention-based model architecture [11]. Each encoder is comprised of multi-head self-attention and feed-forward neural network (FFN) sub-layers [11]. The self-attention layer of the first encoder is initialized with the embedding matrix from each word (tokens). Subsequently, the Query, Key, and Value matrices are calculated for this embedding by the attention mechanism. Each matrix articulates a different representation of the same initial embedding. The self-attention matrix formula is given by Equation (2):

$$\text{Self-Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \quad (1)$$

where Q is the Query matrix, K is the Key matrix and V is the Value matrix. The parameter d_k represents the Key matrix's dimension. The softmax score in the attention equation determines how much each word will be expressed at this position. Multiplying the softmax score with the Value matrix (V) produces the attention value. As a result of this multiplication, the values of the words we want to focus on are maintained, and irrelevant

words are dropped out. This process is repeated eight times, with eight different randomly initiated matrices for Q, V and K, as shown in [6,12]. The multi-head attention is then calculated by concatenating all eight self-attentions, then performing the dot product to multiply it with another random weight matrix. The multi head attention value shows which attention head is more important for the meaning of a given words. Each self-attention head focuses on a different aspect of how the tokens interact with each other, which makes BERT aware of the context of the given sentence [6,11].

Therefore, the attention mechanism is used to calculate the importance of any word in the sentence. In BERT, each attention layer is followed by a fully connected FFN. The FFN function consists of two linear transformations and a ReLU function, as shown in Equation (2):

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2. \quad (2)$$

This feed-forward layer enhances the deep non-linearity in the model. The robustness of the algorithm is also enhanced by residual connections from previous states. The encoders map each input sequence to a continuous representation and then reprocesses then using the same layer structure [11]. The output of the final encoder will be a vector that can be supplied to the NN layers of any downstream task. In the case of ClaimsBERT, this output is used by a classifier stacked on top of this BERT transformer architecture.

There are two different versions of BERT. Both models have been trained utilizing a dataset comprised of the text content of the BookCorpus and the collection of pages obtained from English Wikipedia, which combined constitute a dataset of more than 3.5 billion words [6]. The specifics of both versions of BERT are:

- **BERT-Base**: consists of 12 encoder layers, utilizes an embedding size of 768 dimensions, 12 multi-head attentions, and is comprised of 110M tunable parameters in total;
- **BERT-Large**: consists of 24 encoder layers, utilizes an embedding size of 1024 dimensions, 16 multi-head attentions, and a total of 340M tunable parameters.

For completeness, we also mention the concepts of Vision Transformers (ViT) [13], Image GPT [14], and Multiscale Vision Transformers (MViT) [15], which are the analogous forms of their respective NLP models, adapted specifically to Computer Vision (CV) tasks such as in the domain of autonomous vehicles. Recently, ViT models were also used to detect and extract text from images, akin to an advanced generalized form of optical character recognition (OCR), for example from image object labels, instructions and road signs, within the broader application of Scene Text Recognition (STR). An example is shown in [16], in which a ViT model was used to find text in complex environments such as product labels, signboards, road signs and markers to help machines and agents make informed decisions. These works, however, use the transformer approach to extract text from image, not to extract meaning and context from within the text, as would be the case in NLP applications. Thus, the use of transformers in the CV domain, while in many cases a crucial technique, does not constitute natural language processing.

Another application of transformers is for so-called Vision and Language (V&L) tasks, which utilize multi-modal transformer models, comprise of separate processing streams that in one stream focus on text encoders, and in another stream on vision encoders [17–23]. These V&L approaches focus on determining links and relationships between visual and textual content and learn their joint features [20]. These models leverage links between text and image content, for example for navigation tasks of autonomous vision-based vehicles or robots, or for lip reading and associating visual information about facial expressions with textual information. As a result, their application domain is essentially independent of ClaimsBERT's application domain. Therefore, they are not considered to be feasible choices for our research presented in this paper, which represents a specific application with emphasis on BERT within the NLP domain.

Recently, a new multi model for news classification was introduced in [24]. The authors used RoBERTa to obtain text features, and separately applied a Vision Transformer for

image feature extraction from news. Only after these two separate processes, the resulting features from both were concatenated together to build a fusion feature. All these features then feed into Multilayer Perceptron (MLP) layers to predict the text, image, and fusion labels separately. The fusion label is considered to be the final prediction label for the news. However, these models are not applicable to purely text-based processing, which is the focus of this paper representing aspects of the research related to our vetting framework.

In recent years the use of MLPs for text classification tasks has seen increased popularity [25–27]. Even though the use of MLP techniques in these models is shown to achieve generally comparable performance to that of self attention-based transformers (BERT) [28], in most applications self attention-based transformers are shown to achieve better accuracy [29]. Therefore, the foundation of our architecture presented in this paper was chosen to be BERT.

The performance of a pre-trained language model can subsequently be further increased by adapting it to a target downstream task [30–34]. Adapting a language model to domain-specific downstream tasks can be divided into the pre-training of language models and fine-tuning of language models. Pre-training essentially involves a full retraining of the model on a new corpus with randomly initialized weights, whereas fine-tuning starts with established weights and alters them to better suit the model when training on a smaller additional corpus.

Although the pre-training of BERT into domain-specific models, such as for biomedical language BioBERT [35] or scientific text (SciBERT) [36], improves downstream performance significantly, the process is computationally expensive and typically requires large extra corpora [35,37]. Therefore, we are focusing on the fine-tuning process for the ClaimsBERT model, which can achieve very similar performance improvements. To evaluate the size of the dataset required specifically for the fine-tuning of language models, Sun et al. [33] in their work evaluated the impact that the dataset size has on the fine-tuning outcome, and showed that fine-tuning an existing language model can successfully be achieved using only a few training shots from a small dataset. This reduces or even eliminates the need to produce large-corpus datasets when fine-tuning models.

Fine-tuning enables NLP language models to be applied to many different tasks. For some NLP applications, however, a language model by itself is not sufficient for accomplishing a given downstream task, and it becomes necessary to expand the language model's overall architecture by stacking it with another form of neural network, for example using a convolutional neural network for language models targeting classification NLP tasks. For such application scenarios, the combination of the BERT language model and deep learning models such as recurrent neural networks or convolutional neural networks were shown to be effective in recent studies for capturing meaningful features from the available data [8,38–40]. We utilize a similar approach for our ClaimsBERT classifier. In our approach, we train the entire pre-trained model on our dataset and use the transformer encoders stacked with neural networks to feed the output to a softmax layer for back-propagation through the entire architecture, which updates the pre-trained weights based on our dataset. Adding a feature mapping stage on top of the transformers architecture delivers useful features that can improve the performance for our downstream task. As we will show in our paper, our research showed that a CNN network provided the best results in comparison with other approaches for this feature mapping stage, specifically NN, LSTM, BiLSTM and MLP.

In this paper, we focus on maximizing the accuracy of our claim classifier without significantly impacting its efficiency. We apply our novel framework ClaimsBERT on our curated cybersecurity claim sequence database. To the best of our knowledge, no other cybersecurity-related classifier using BERT has been published in the scientific literature. Furthermore, our approach presented herein marks a significant improvement upon the original BERT classifier. The new ClaimsBERT classifier can detect claim sequences from the large dataset of vendor documents. These sequences then will be used in a vetting approach against industry standard requirements in our cybersecurity vetting engine (CYVET).

Language Models in Cybersecurity

Using language models such as BERT for cybersecurity applications is a growing research area in recent years [8,41–49]. Fine tuning language models such as BERT for cybersecurity domain tasks can provide many benefits to the cybersecurity community.

One example of using language models in this domain is the fine tuning of BERT for Name Entity Recognition (NER) applications for different languages, such as Chinese [50], Russian [46] and English [42–44]. NER models provide cybersecurity professionals with an efficient way to extract specific entity information about attacks and vulnerabilities [48]. In another study, BERT was fine tuned on Android source code applications to identify and classify existing malware [41]. Fine-tuning BERT for classification tasks such as attack classification [48], cybersecurity claim classification [8], knowledge graph [51] and vulnerability classification [52]. ExBERT is another example of fine-tuning BERT for vulnerability exploitability prediction using sentence-level sentiment analysis [52]. An effective evaluation of evolving risks can be accomplished with the help of semantically connected text graphs using the Construction Cybersecurity Knowledge Graph (CKG) and Graph Convolutional Network (GCN) based on BERT [51]. Analysts who are usually required to sort through attack details to categorize various types of attack vectors may benefit from Cybersecurity Knowledge Graph (CKG) [51].

Our review shows the benefits of fine tuning language model such as BERT for the downstream tasks in the cybersecurity domain. Therefore, we developed and present ClaimsBERT, which can accurately identify feature claims based on claim sequences from our cybersecurity domain database. The focus here is on how NLP can be leveraged for cybersecurity applications.

3. Dataset Curation

When we initiated this research effort, there was no dataset available for NLP tasks specific to cybersecurity literature [2]. As a result, we proposed a framework for curating a large repository of device information for ICS in our previous work [2]. Using this framework, we are able to identify ICS vendor websites, collect website-accessible documents, and identify documents relevant to the dataset using web scraping, data analytics, and natural language processing (NLP). Our framework starts by determining ICS vendor names by scraping CISA's ICS-Cert website at https://us-cert.cisa.gov/ics (Access on 1 March 2021). These results are combined with the vendor name results obtained utilizing predefined keyword search queries applied to different search engines, including Google, Bing, AOL, and Baidu. The framework then expands vendor names into vendor websites by conducting web searches to identify their most likely website URLs. The resulting names and websites are then classified using a web content-based scoring metric along with Latent Dirichlet Allocation (LDA) to identify vendors-of-interest. This process evaluates the home page content of each vendor's website to determine if that vendor falls within the category of an ICS solutions vendor. Finally, our framework conducts a comprehensive web site crawl through all relevant pages to curate a list of downloadable documents, and proceeds to download each document it located. All downloaded documents are then associated with the corresponding ICS vendor. Based on our results from processing the identified vendor websites, we found that [2]:

- 3% of the documents were unreadable;
- 5% were scanned documents;
- 29% where not related to ICS products;
- 63% of the downloaded documents are ICS product-related documents.

When analyzing the 12,581 ICS product-related documents that were found, we could determine that

- 25% were classified as "manuals";
- 69% were classified as "brochures";
- 6% were classified as "catalogs".

Downloaded documents may contain regular paragraphs, tables, lists, and images. Text from readable documents was extracted via the `PyMuPDF` python package, and we leveraged python's `Pytesseract` package for performing optical character recognition (OCR) with any scanned PDFs. With this approach, we managed to extract 2,160,517 sequences with 41,073,376 words across our curated dataset of ICS documents [8].

In order to label the sequences in our curated dataset we developed a new mobile cross-platform application based on Xamarin that allows us to manually label sequences extracted from these ICS documents. We first identified three types of claim sequences: generic, device, and cybersecurity claims. Specifying individual claim types facilitates future investigations into claim type detection. We categorized all of these types of claims as "claim" labels in this study and removed the sequences with the "Not Sure" label from the classification dataset. For more details on this labelling process please refer to [8].

Figure 2 illustrates the labeled dataset we used to train our classifier models for the Tally-Vet project. In this paper we focus only on identifying "Claim" sequences. Figure 3 presents the final class count and distribution of all classes.

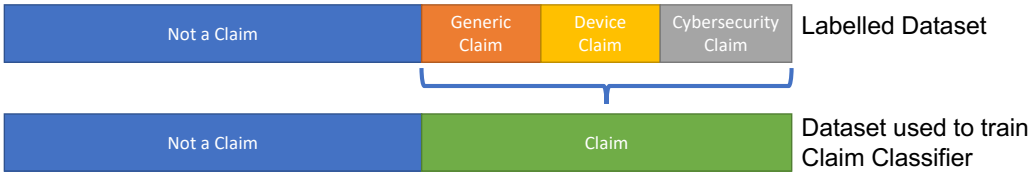

Figure 2. An illustration of the labeled dataset and classes used to train our classifiers.

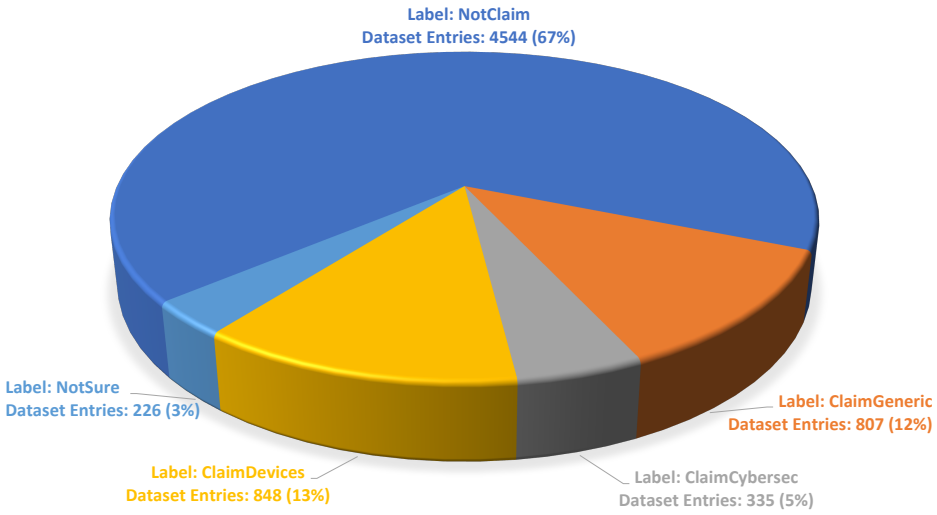

Figure 3. Class label distribution in each class contained in our curated dataset.

4. NLP Model Optimization for ClaimsBERT

In this section we detail the process we followed to maximize the accuracy for our claim sequence classifier. To achieve this goal we first utilized the BertForSequenceClassification on our curated dataset. This allows us to establish a performance baseline. To improve upon this BertForSequenceClassification baseline model's accuracy in our cybersecurity claims classification, our initial objective was to research and select the best component to add to BERT that aids in its achievable accuracy, and then fine tune the resulting models. Next, we focused on extensive experiments to optimize hyperparameters, including the learning rate (LR), type of activation function, the number of convolution layers and their corresponding filter sizes, as well as the number of dense layers and their configuration such as drop-out rate and number of neurons. After we determined the best model configuration and optimized its hyperparameters we finally tested the resulting model for its susceptibility to the randomness effect.

4.1. BERT Baseline (BertForSequenceClassification)

The Google research team released a pre-trained classifier BertForSequenceClassification in [6]. This model is based on BERT with an added drop-out layer on top of the stacked encoders, as well as a softmax classification/regression layer. To predict each class label in this classifier, the label probability is therefore determined by:

$$P(c|h) = \text{softmax}(Wh), \tag{3}$$

where W indicates the matrix for task-specific parameters and h is the final hidden state of the first token in the BERT token embedding matrix (CLS). The highest accuracy we could achieve in any of our tests using BertForSequenceClassification applied against our curated cybersecurity claim dataset without any fine-tuning applied first was 76%. Therefore, in order to evaluate how far we could improve the accuracy of the base BERT classifier we then fine-tuned this baseline architecture, without any architectural modifications, by unfreezing all layers and allowing the model to adjust its weights during the training process. This is described in the section below.

4.2. Fine-Tuning BERT

Although adapting a pre-trained language model such as BERT for a specific task can improve performance significantly, an appropriate fine-tuning strategy must be developed [5]. In this paper we followed the same techniques we utilized in our previous paper [8] in order to maximize the performance we gain from the fine-tuning process. These techniques are utilized for any of the model improvements presented throughout the remainder of this paper in order to ultimately successfully train our ClaimsBERT classifier. Specifically, these techniques are employed to avoid the risk of catastrophic forgetting, overfitting and any randomness effects, which can occur when training a language model on a small dataset.

To ensure that the model will be able to modify BERT's pre-trained weights during the training process based on our given downstream task, the chosen hyperparameters, and our curated dataset, we unfreeze all of the transformer layers during the fine-tuning process [33].

Hyperparameter Selection

The two primary hyperparameters that affect fine-tuning BERT are the learning rate (LR) to avoid catastrophic forgetting, and selecting an appropriate epoch limit in order to avoid overfitting. More detailed information on these are available in our previous paper [8], but both are summarized below for convenience:

- **Catastrophic Forgetting**: The cyclic method we utilized was first presented by Smith in [53], and it allows us to determine the optimal learning rate for our model training that avoids catastrophic forgetting. This method initially selects a low learning rate, which is then increased exponentially for each subsequent batch. The LrFinder function [53] was used to determine the best learning rate for each architecture.
- **Overfitting**: A common problem when training a neural network is the determination of an appropriate number of training epochs to use. An overfitted training dataset can result from too many epochs, while underfitting may be caused by a lack of iterations. If the monitored metric does not improve after a certain number of epochs, then we can stop the training process through selection of an appropriate early-stopping method. By implementing a data-driven automation approach, it eliminates the need to manually select the number of epochs. Our model monitors validation loss values, and if it does not show any improvement after two epochs, we stop training.

Fine-tuning BERT allowed us to improve its performance from the 76% maximum accuracy obtained prior to fine-tuning, to a significantly higher 92% accuracy [8]. This is a very significant improvement. It allowed our research to progress on the implementation of all downstream tasks for the CYVET framework. However, we needed to explore how to further improve the accuracy of this claims classifier. The achieved accuracy of 95% still

indicated that 1 out of every 20 sequences would be misclassified. It was our goal, therefore, to explore architectural changes that would help us further improve this accuracy.

4.3. Classification Optimization Using Feature Map

One significant way to aid the classifier is by providing it with information on specific features that are not readily apparent from the dataset. This is achieved through the use of a feature map component that is added to the overall architecture prior to the classifier stage. In our proposed architecture for this study, we therefore added a new component to the original BertForSequenceClassification model, indicated by the feature map box shown in Figure 4. This component is responsible for extracting informative features from the output of the transformers, and connecting the resulting feature map to the classifier. More specifically, the role of the feature map is to reduce the sequence dimensions into a form that is easier to process, without losing features that are critical for obtaining a reliable prediction. This component concentrates on three main steps:

1. **Reshaping**: This involves reshaping the output of the transformers' NSP layer from BERT, in order for the batch size and the sequence length to be compatible with the convolutional layer's input;
2. **Convolution**: This involves connecting multiple convolutional layers with different filter sizes, and then using the max pooling to learn higher-order representations of the data while reducing the number of parameters;
3. **Flattening**: This involves converting the matrix from the final pool to a single array. This flattened vector is then connected to a fully connected neural network for the classification task.

Determining the most informative layer of BERT that is then connected to the feature map, as well as determining the filter size, activation function and number of convolution layers are additional important considerations for fine-tuning our new classifier architecture.

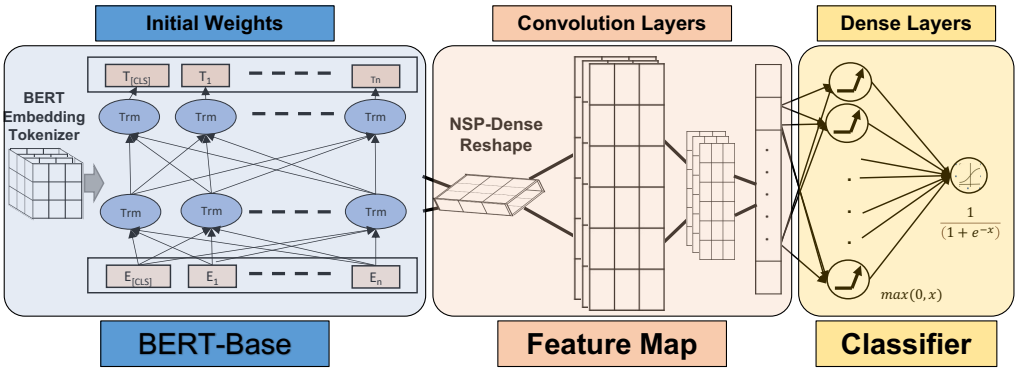

Figure 4. ClaimsBERT framework components.

Delving into the feature map component itself, we first needed to determine the best possible network-based model to be used as part of the feature map. Our research focused on five different types: Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), Multilayer Perceptron (MLP), Convolution Neural Network (CNN) and Neural Network (NN). All five are types of Artificial Neural Networks (ANN) that can be used for extracting abstract features from complex data. Essentially, the difference between these architectures is the type of neurons that form them, and the manner in which information flows through them [54]. The NN consists of multiple neurons or perceptrons in each layer, which enables the NN to capture the correlation distribution between inputs [55]. MLP operates very similar to the NN. CNN is a multilayer neural network with convolution, which is able to detect complex features within data that are not readily apparent. Convolution is responsible for identifying the most significant features [56]. LSTM networks benefit from recognizing the relationship between values at the beginning and end of a sequence [54].

BiLSTM networks utilize information from both directions in sequences, which helps to produce meaningful information, especially when applied in NLP tasks [57]. NN, MLP and CNN models are hierarchical, while BiLSTM and LSTM have a sequential architecture. For NLP tasks such as text classification, CNNs and LSTMs are often preferred. However, since they provide somewhat complementary information, there is no consensus within the scientific community on which deep neural network is best suited for any specific NLP problem [56]. This is also precisely what influenced our research effort presented here.

This necessitated a comprehensive evaluation of the different ANN architecture choices presented above, in the context of our specific NLP application, to determine which architecture would provide the best performance. The results of this evaluation are presented further below in the results chapter. Those results indicate that the best choice for our claims classifier's feature map component was the CNN type. Once the most appropriate model architecture was determined we could then focus on researching the optimization of our hyperparameter selection of the revised architecture. A detailed illustration of the architecture for BERT + CNN (ClaimsBERT) is shown in Figure 5, where the model starts with initial weights established using a general corpus for the BERT-base model. This is followed by a CNN model and a classifier, used to further fine-tune BERT using supervised data to target tasks for text classification.

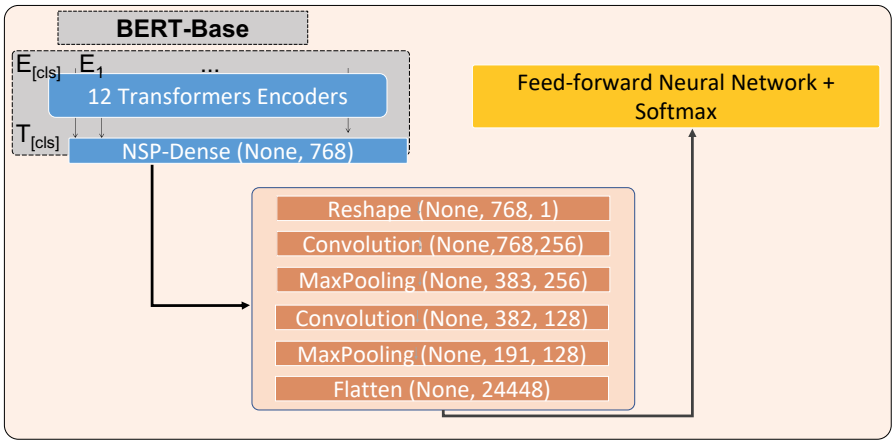

Figure 5. Optimal architecture for ClaimsBERT components.

We conducted extensive research and experimentation to jointly optimize the hyperparameters for the feature map and the classifier, including the number of convolution and dense layers as well as their corresponding configuration, such as filter and kernel size, activation function type, drop-out rate and number of neurons. The final optimal architecture and hyperparameters are shown in Table 1. The corresponding classification performance results are shown and discussed in the results section below.

Table 1. Optimal architecture for ClaimsBERT model.

Layer	Input Shape	Output Shape	Activation Function	Parameters
NSP-Dense	(None, 768)	(None, 768)	–	590,592
Reshape	(None, 768)	(None, 768, 1)	–	0
Convolution1D	(None, 768, 1)	(None, 767, 256)	ReLU	768
MaxPooling1D	(None, 767, 256)	(None, 383, 256)	–	0
Convolution1D	(None, 383, 256)	(None, 382, 128)	ReLU	65,664
MaxPooling1D	(None, 382, 128)	(None, 191, 128)	–	0
Flatten	(None, 191, 128)	(None, 24448)	–	0
Drop-out	(None, 24448)	(None, 24448)	–	0
Dense	(None, 24448)	(None, 64)	ReLU	1,564,736
Drop-out	(None, 64)	(None, 64)	–	0
Dense	(None, 64)	(None, 2)	Softmax	130

Note: Total parameters: 110,769,474 (all are trainable parameters).

5. Comparative Analysis Results and Discussion

All training and testing tasks related to any models presented in this paper were conducted utilizing our university's high-performance computing infrastructure, HCC Crane [58]. All compute nodes we utilized are equipped with NVIDIA Tesla V100 GPUs with 16 GB of RAM per GPU. Our sequence dataset was first divided into appropriate sets for training, validation, and testing. This ensures an unbiased estimate of out-of-sample accuracy. The respective set sized were 70%, 10%, and 20% of or our total dataset, for training, validation, and testing.

Considering the label ratios in a dataset for the average F1 score results in the expression for the macro-weighted F1 score. This macro-weighted F1 score, together with the prediction accuracy for the test set are the parameters we used to compare the models.

5.1. Comparison of ClaimsBERT against the Pre-Trained BertForSequenceClassification

We use the pre-trained BertForSequenceClassification as the baseline for our cybersecurity claim classifier. In the BertForSequenceClassification model, all transformer layers are frozen and only the classifier layer weights are tuned during training process. Table 2 compares the classification performance for BertForSequenceClassification and ClaimsBERT.

Table 2. Comparing classification performance of ClaimsBERT with BertForSequenceClassification.

Model	Architecture	Accuracy	F1 Score	Precision	Recall
BERT + CNN (ClaimsBERT)	12 Encoder 2 Convolution 2 Dense	0.973	0.96	0.963	0.966
BERTSequence Classifier	12 Encoder 1 Dense	0.764	0.751	0.743	0.741

5.2. Comparison of ClaimsBERT against Other Network-Based Classifier Models

In order to evaluate our proposed architecture and fine-tuning strategy on BERT, we compare the results of ClaimsBERT with our previously trained classifier CyBERT (BERT + NN) [8] and other network-based models.

5.2.1. LSTM

In Long Short-Term Memory (LSTM) neural networks learn features at the phrase-level by using a convolutional layer. Then the convolutional layers used to learn how such higher-layer representations relate to long-term features through this convolutional layer [12,59].

The output of this convolutional layer are higher-layer representations of the processed content, and is then provided to the LSTM in order to learn relationships regarding long-term features.

In order to determine the optimal hyperparameters and the resulting architecture, we assessed the different numbers of LSTMs and dense layers. With the LrFinder function [53], we found that the best LR for each model architecture varied based on the different numbers of LSTM and Dense layers. Table 3 reports the highest accuracy and F1 for each architecture. The best LSTM network-based model contains a single LSTM layer with 50 hidden units and a 0.3 dropout rate, linked to two dense layers with 150 and 124 neurons, respectively. Our experiments clearly demonstrated that this configuration best adapts to the information in our dataset compared to other LSTM-based models.

Table 3. Comparing BERT + LSTM Classifier results. (Bold text indicates the model parameters achieving the highest accuracy.)

Architecture		LR	Accuracy	F1 Score
	1 Dense	7×10^{-5}	0.92	0.9
BERT + 1 LSTM	**2 Dense**	2×10^{-5}	**0.94**	**0.93**
	3 Dense	4×10^{-5}	0.93	0.91
	1 Dense	3×10^{-5}	0.92	0.91
BERT + 2 LSTM	2 Dense	7×10^{-5}	0.94	0.92
	3 Dense	4×10^{-5}	0.93	0.91
	1 Dense	2×10^{-5}	0.93	0.91
BERT + 3 LSTM	2 Dense	1×10^{-4}	0.92	0.91
	3 Dense	7×10^{-5}	0.91	0.9

5.2.2. BiLSTM

Bidirectional LSTM (BiLSTM) models are especially convenient for sequential modeling [60]. BiLSTM models have therefore found widespread application in retrieving contextual knowledge. This knowledge is the results of convolution layer feature maps [60]. In bidirectional LSTMs, the embeddings are obtained by concatenating two one-way actions [61].

For this research we also explored the relative impact provided by varying the number of BiLSTM layers and their hidden units, the dense layers and their neurons, and the dropout rates for each layer stacked on top of BERT. The LrFinder function [53] was used to determine the optimum LR for each architecture. Table 4 reports the highest accuracy and F1 for each architecture. The highest accuracy for the combined BERT+BiLSTM-based model was achieved by stacking a BiLSTM with 150 hidden units and a 0.3 dropout rate on top of BERT, linked to one dense layer with 64 neurons and a 0.5 dropout rate.

Table 4. Comparing BERT + BiLSTM Classifier results. (Bold text indicates the model parameters achieving the highest accuracy.)

Architecture		LR	Accuracy	F1 Score
	1 Dense	5×10^{-5}	0.94	0.93
BERT + 1 BiLSTM	**2 Dense**	6×10^{-5}	**0.95**	**0.94**
	3 Dense	4×10^{-5}	0.94	0.93
	1 Dense	8×10^{-5}	0.92	0.91
BERT + 2 BiLSTM	2 Dense	5×10^{-5}	0.93	0.92
	3 Dense	4×10^{-5}	0.91	0.91
	1 Dense	1×10^{-4}	0.91	0.91
BERT + 3 BiLSTM	2 Dense	8×10^{-5}	0.94	0.92
	3 Dense	4×10^{-5}	0.91	0.9

5.2.3. Multilayer Perceptron

A Multilayer Perceptron (MLP) is a feedforward artificial neural network. For this research, each MLP block consists of two fully-connected NN layers, connected by a GELU nonlinearity, which matches the configuration of the MLP modules used in the MLP-Mixer paper [28].

For this research we also explored the relative impact provided by the number of MLP layers and their hidden units, the dense layers and their neurons, and the dropout rates for each layer stacked on top of BERT. The LrFinder function [53] was used to determine the optimum LR for each architecture. Table 5 reports the highest accuracy and F1 for each architecture. The highest accuracy for the combined BERT + MLP model was achieved by stacking three MLP blocks with 256 and 128 neuron sizes with a GELU activation function and a 0.5 dropout rate. These MLP blocks then connected to a dense layer with 64 hidden units and a 0.5 dropout rate on top of BERT, linked to one dense layer with the softmax activation function for the classifier.

Table 5. Comparing BERT + MLP Classifier results. (Bold text indicates the model parameters achieving the highest accuracy.)

Architecture		LR	Accuracy	F1 Score
BERT + 1 MLP	1 Dense	4×10^{-5}	0.936	0.925
	2 Dense	6×10^{-5}	0.942	0.932
	3 Dense	3×10^{-6}	0.941	0.935
BERT + 2 MLP	1 Dense	4×10^{-6}	0.943	0.935
	2 Dense	3×10^{-6}	0.932	0.921
	3 Dense	2×10^{-5}	0.945	0.931
BERT + 3 MLP	1 Dense	4×10^{-6}	0.93	0.923
	2 Dense	5×10^{-6}	**0.952**	**0.942**
	3 Dense	3×10^{-5}	0.925	0.911

5.2.4. Neural Network

In order to evaluate the effect of NN-based model on BERT, we compare the results of ClaimsBERT with our previously trained classifier CyBERT [8]. Different numbers of dense layers on top of stacked encoders in BERT were studied. For our initial BERT + NN model we studied the impact of selecting different dropout rates and numbers of neurons within each dense layer. The best results were achieved when the model was comprised of two dense layers of 64 and 16 neurons, respectively, and corresponding dropout rates of 0.5 and 0.3 [8].

As shown in Table 6, ClaimsBERT maximizes the accuracy of the claims classifier. As it is shown in Table 6, ClaimsBERT achieves an accuracy of the claim classifier of 97%.

Table 6. Comparing classification performance of ClaimsBERT with other network-based models. The bold row indicates best model parameters with highest accuracy.

Model	Architecture	Accuracy	F1 Score	Precision	Recall
BERT + CNN (ClaimsBERT)	12 Encoder 2 Convolution 2 Dense	**0.973**	**0.96**	**0.963**	**0.966**
BERT + NN (CyBERT [8])	12 Encoder 3 Dense	0.954	0.93	0.914	0.943
BERT + BiLSTM	12 Encoder 1 BiLSTM 2 Dense	0.951	0.941	0.951	0.949
BERT + LSTM	12 Encoder 1 LSTM 2 Dense	0.947	0.937	0.947	0.947
BERT + MLP	12 Encoder 3 MLP 2 Dense	0.952	0.942	0.947	0.938

5.3. Hyperparameter Finetuning Results

In the following subsections, we detail the methodology we used to determine the optimal values for the hyperparameters associated with our ClaimsBERT claim classification model.

5.3.1. Convolution Layer

There are several layers of filtering and pooling in the initial part of a CNN. Feature maps are created by applying these layers to an array of multi-dimensional features [62]. These features represent abstractions of the most significant characteristics contained in the input data. This is followed by a fully connected network that maps the extracted features to their respective targets [63]. In classification problems, this target represents

data that share common characteristics. CNNs flatten out the multi-dimensional output of convolutional components and pass it on to the fully connected network.

For the CNN hyperparameters, we investigated the effects of different values, such as the number of convolutions, as well as different filter and kernel sizes for each convolution layer. The filter size in the convolution layer determines the dimension of the output space, whereas the kernel size specifies the length of the convolution window. The output of the convolution layer has a shape of:

$$(m - k + 1, n), \tag{4}$$

where n is the number of filters, k is the kernel size of the filters, and m indicates the number of words in each sentence [62]. The number of filter weights (i.e., parameters) in the convolution layer are:

$$n * (k * w) + n, \tag{5}$$

where n is the number of filters, k is the kernel size, and w is the embedding dimension [62].

We also investigated the effects of filter size on the training process and the classifier model accuracy. Different architectures with different filter sizes, including 256, 128, 64 and 32, were tested to find the most effective model. We experimented starting with a BERT base model, expanded it with different numbers of convolution layers, followed by dense layers for each model. In order to determine the configuration maximizing the accuracy of the classifier we utilized a parameter sweep methodology. With this approach we started at a configuration of one convolution layer and one dense layer and swept both parameters upwards until we found the peak accuracy. We validated the peak by observing the configuration that showed a reduction in accuracy. The resulting sweep spanned the range of one to four dense layers and one to three convolutional layers. The results presented in Table 7 show the results from that sweep operation.

The input sequence length is 68 for all models. Using the LrFinder function [53], the best learning rate was determined for each architecture. The training process initiates with 100 epochs and uses the early-stopping method to avoid the overfitting problem, as described earlier.

The best results were achieved when we utilized two convolutions, with filter sizes of 256 and 128, respectively. We found that a kernel size of two performed best for both convolutions. Figure 6 compares the highest accuracy based on the number of convolutions and filter sizes. Figure 6 illustrates the effect of filter size on training time and accuracy for different BERT + CNN models.

According to the results, if the filter size is small, the model takes longer to train. This can be attributed to the fact that the model takes more epochs to converge. Figure 6 illustrates how the accuracy increases as the filter size increases. Input parameters are more informative for models, which start building their feature map with a filter size of 256. As a result, the model will start with significantly better features and therefore can converge more quickly. The two plots on the right side shows the distribution of training time and accuracy for models with same filter size but different convolution and dense layers. In the bottom plot, for instance, we see that the accuracy distribution for models with a filter size of 128 is between 91 and 96, whereas that of models with a filter size of 32 is between 89 and 93. However, the same model training time with filter size 32 is twice as long as a model with filter size 128.

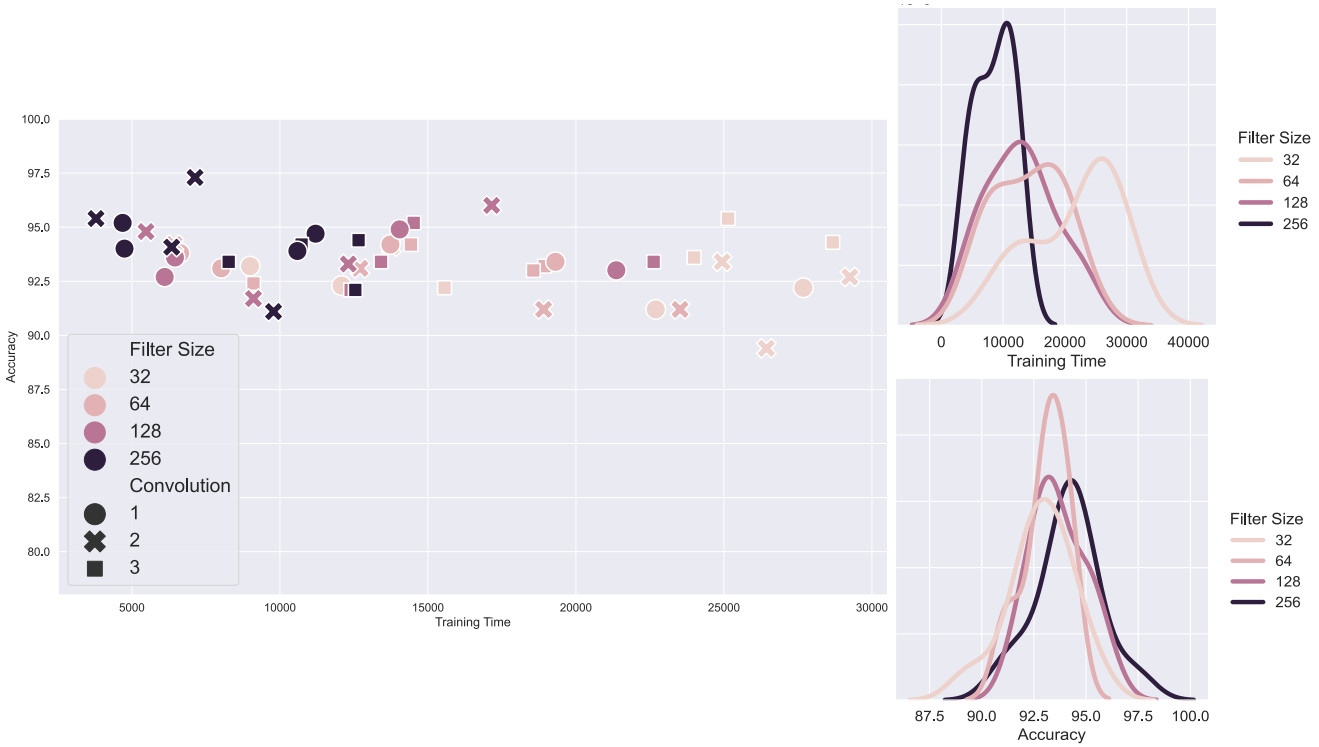

Figure 6. Filter size effect on training time and accuracy when fine-tuning BERT with different number of convolution and dense layers.

5.3.2. Max Pooling

In our model, the convolutional layer is followed by a pooling layer. The primary aim of the pooling layer is to decrease the size of the convolved feature map, and therefore reduce the computational costs. It involves dividing the output layers into subsections and calculating the value that best represents the output. In addition to reducing the number of parameters, this helps the algorithm learn higher-order representations of the data [64]. Max pooling reduces the dimensionality of the input by reducing the number of parameters in the output from the previous convolutional layer. This can be seen in the input and output shapes, shown in Table 1.

Figure 7 shows an example of max pooling in CNN, which reduces the dimensionality and also represents the highest value in each kernel window.

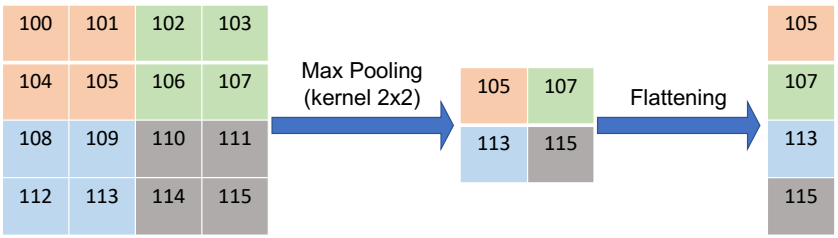

Figure 7. An example of max pooling in CNN with a 2 × 2 kernel size and the flattening function.

The final pooling layer then is flattened. The flattening layer involves taking the pooled feature map that is generated in the pooling step and transforming it into a one-dimensional vector. An example visual representation of the flattening process is illustrated in Figure 7. The flattened layer then is connected to a fully connected neural network for the classification task.

5.3.3. Activation Function

The other important parameter in CNN networks is the activation function. The choice of activation function has a significant impact on the neural network's performance, and most notably the accuracy. Hence, in our research to maximize the overall accuracy of our classifier, studying various different activation functions was a vital aspect.

An activation function is the feature of activated neurons that can be maintained and mapped by a nonlinear function, which can be used to solve nonlinear problems. The activation function enhances the expression capacity of the neural network model [65]. The most common activation functions include: sigmoid, tanh, Rectified Linear Unit (ReLU) and softplus.

Mathematically, these activation functions are defined as:

$$ReLU(x) = max(0, x) \tag{6}$$

$$tanh(x) = (1 - exp(-2x))/(1 + exp(-2x)) \tag{7}$$

$$sigmoid(x) = 1/(1 + exp(-x)) \tag{8}$$

$$softplus(x) = log(1 + exp(x)). \tag{9}$$

Figure 8 shows the different activation function mentioned above.

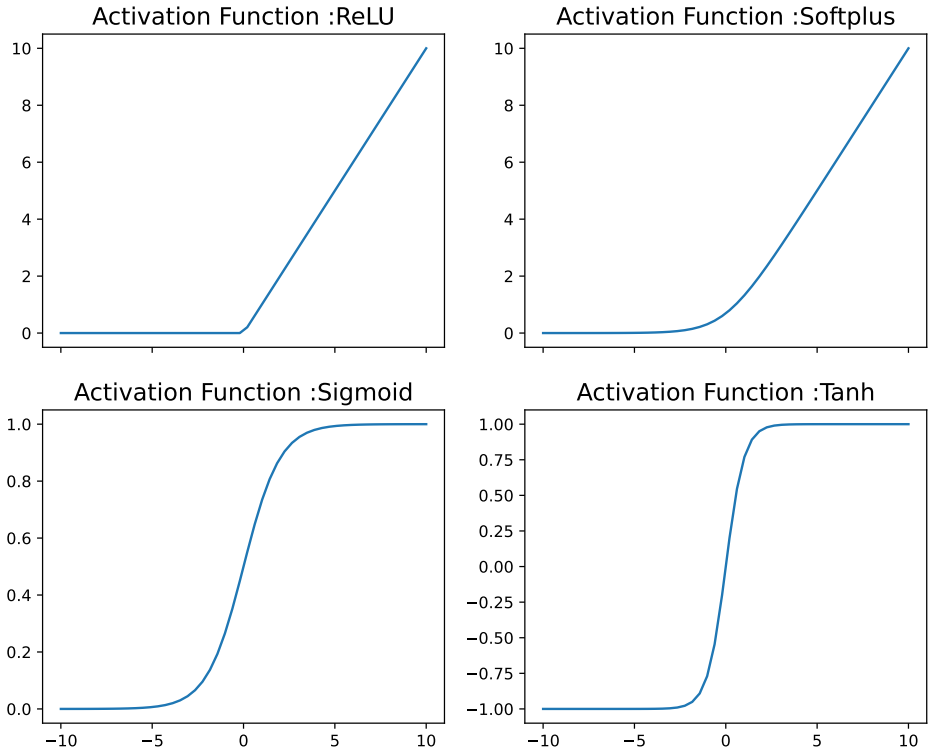

Figure 8. Different activation functions visualization.

As shown in Figure 8 and Equation (6), the ReLU function will return 0 if it receives any negative input. The softplus is a smooth approximation to the ReLU activation function. When the softplus function is near 0, it is smooth and differentiable (Equation (9)). The sigmoid activation function values lies between zero and one (Equation (8)) while the tanh activation function outputs outputs between -1 and 1 (Equation (7)). Both sigmoid and tanh suffer from gradient problem near the boundaries.

ReLU and its derivative are both monotonic functions. The main advantage of the ReLU function is its simplicity. It does not require heavy processing, and because a smaller number of neurons is activated in the ReLU function it is also more computationally

efficient than the sigmoid and tanh functions [66]. That is a result of the fact that for any positive input, the derivative of ReLU is of value 1.

Another significant advantage of using the ReLU activation function is its sparsity [67]. Finally, ReLU can produce true zero values. In contrast, the tanh and sigmoid activation functions both approximate zero outputs, i.e. a close value to zero, but not the true zero value. Therefore, negative input can result in true zero values in the hidden layers of neural networks, allowing one or more true zero values to be reflected. This is called a sparse representation and can have a significant impact on accelerating learning and simplifying a model in representational learning [67].

Figure 9 illustrated the effect of activation function on training time and accuracy for different BERT + CNN models. According to the results, sigmoid, tanh and softplus activation functions failed to understand and learn separable features to distinguish between class labels. According to Figure 9, the ReLU activation function manages to enhance the claim classifier significantly.

The conventional activation function in the neural network of our ClaimsBERT model is the ReLU function [65]. Based on the aforementioned advantages of the ReLU activation function and our evaluation results, we selected this function for the hyperparameter selection process in our model.

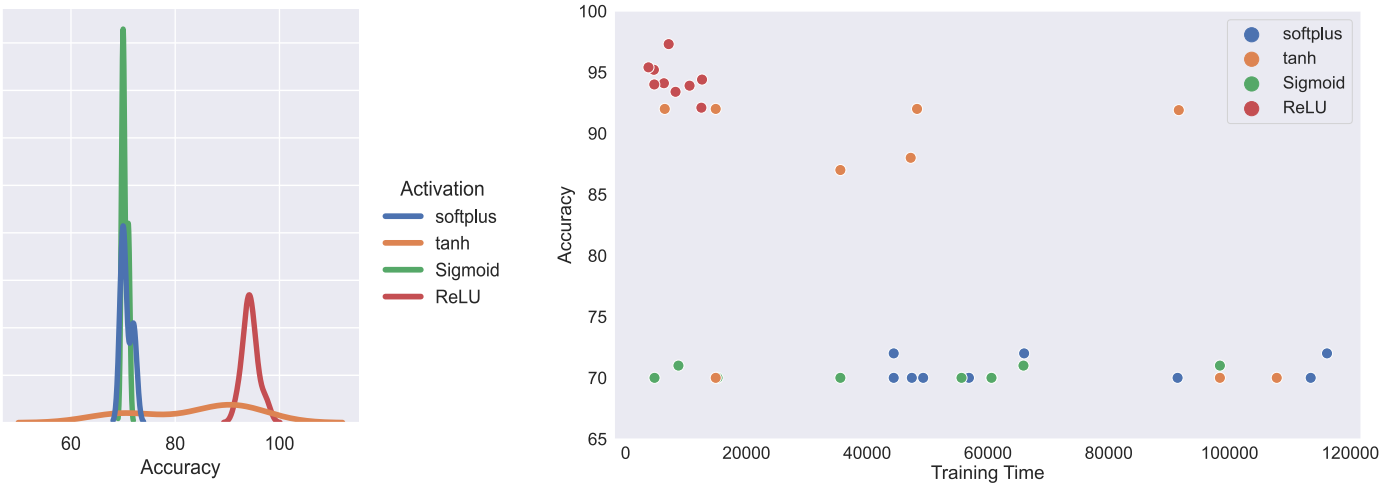

Figure 9. Activation function effect on training time and accuracy when fine-tuning BERT with different number of convolution and dense layers.

5.3.4. BERT Classifier Layer

The BERT model consists of an embedding layer, whose operation is based on calculating the sum of the embeddings of the tokens, segmentation, and positions, respectively. For the BERT base model, the embedding matrix is followed by 12 encoders stacked on top of it, plus an additional pooling layer. The special classification token (CLS) is the aggregation of the last hidden state weights after the final encoder, and is used for sequence representation. The CLS token is an input for the next-sentence prediction (NSP) layer. This NSP layer is the first layer after the encoders, and is used to determine what relationship exists between neighboring sentences [6]. The NSP layer then is reshaped to be connected to the convolutional layer (Figure 4) to build the feature map.

An artificial neural network is capable of collecting different levels of syntactic structure information from different layers, based on the defined downstream task, such as a text classifier [5]. For this paper, we researched the impact of different convolution and dense layers linked to the output of the stacked BERT encoders. For each evaluated configuration we then trained and tested the model in order to select the highest-performing model for our given dataset.

For this paper, we next focused on adapting different hyperparameters, including the different filter sizes for convolution layers, the drop-outs, and the number of neurons for

each dense layer. We achieved the best results for two convolutions with 256 and 128 filter sizes, respectively. Both convolution layers are followed by a Max pooling layer, which capture the largest element of the feature map based on the kernel size. A fully connected neural network with a 64 neuron dense layer is followed by a drop-out layer with rate 0.5. This feed forward network is followed by a final dense layer with the "softmax" activation function for the classification task. Table 7 illustrates the results we obtained for different configurations of convolution and dense layers.

Table 7. The impact of the number of convolution and dense layers on fine-tuning ClaimsBERT. The bold row indicates best model parameters with highest accuracy.

Architecture		Filter Size	LR	Accuracy	F1-Score
BERT + 1 Convolution	1 Dense	(256)	6×10^{-5}	0.95	0.94
	2 Dense	(256)	2×10^{-7}	0.94	0.93
	3 Dense	(256)	3×10^{-5}	0.94	0.92
	4 Dense	(256)	7×10^{-5}	0.94	0.93
BERT + 2 Convolution	1 Dense	(256,128)	3×10^{-5}	0.94	0.93
	2 Dense	**(256,128)**	9×10^{-5}	**0.97**	**0.96**
	3 Dense	(256,128)	2×10^{-7}	0.93	0.93
	4 Dense	(256,128)	3×10^{-7}	0.91	0.89
BERT + 3 Convolution	1 Dense	(256,128,64)	7×10^{-5}	0.93	0.91
	2 Dense	(256,128,64)	2×10^{-6}	0.94	0.92
	3 Dense	(256,128,64)	5×10^{-5}	0.92	0.91
	4 Dense	(256,128,64)	1×10^{-6}	0.94	0.92

5.3.5. Randomness Impact

Devlin et al. suggested that when fine-tuning BERT utilizing a small dataset, it can result in an unstable model [6]. This stems from the fact that the training outcome can be heavily influenced by the randomly selected initial weights and biases [68]. The training process for our model utilizing the optimized hyperparameters was therefore repeated for 100 different random seeds. The purpose of this was to accomplish a statistically reliable result in regards to the accuracy of our model's performance measurements.

The resulting standard deviation, confidence interval (CI), mean, and the margin of error for the 100 random seeds are presented in Table 8. The Figure 10 displays the distributions of the training accuracy, validation accuracy, and testing accuracy for all 100 random seeds. According the results from Figure 10 and Table 8, the ClaimsBERT's true accuracy was within an interval of 0.952 to 0.956 with a 95% CI. As the results shows in Table 8, the mean accuracy value for both validation and testing improved significantly in ClaimsBERT model.

The obtained results also suggested that the random seed value selection can impact the ClaimsBERT's accuracy nominally. Hence, based on the literature recommendation [6], the random seed that results in the greatest validation accuracy for ClaimsBERT was selected.

Table 8. ClaimsBERT Accuracy Performance Results for Investigating the Seed Randomness Effect.

Model	Dataset	SD	Mean	CI (95%)	Margin of Error
BERT + CNN (ClaimsBERT)	Training	0.007	0.991	0.99 to 0.993	0.00142
	Validation	0.009	0.953	0.952 to 0.956	0.0.0018
	Testing	0.009	0.951	0.95 to 0.953	0.00183

Despite the fact that our dataset contained only a relatively small number of labelled entries, the results we are showing in Figure 10 and Table 8 clearly illustrate that ClaimsBERT, with optimized configuration, achieves high classification accuracy at a high degree of confidence when identifying cybersecurity feature claims.

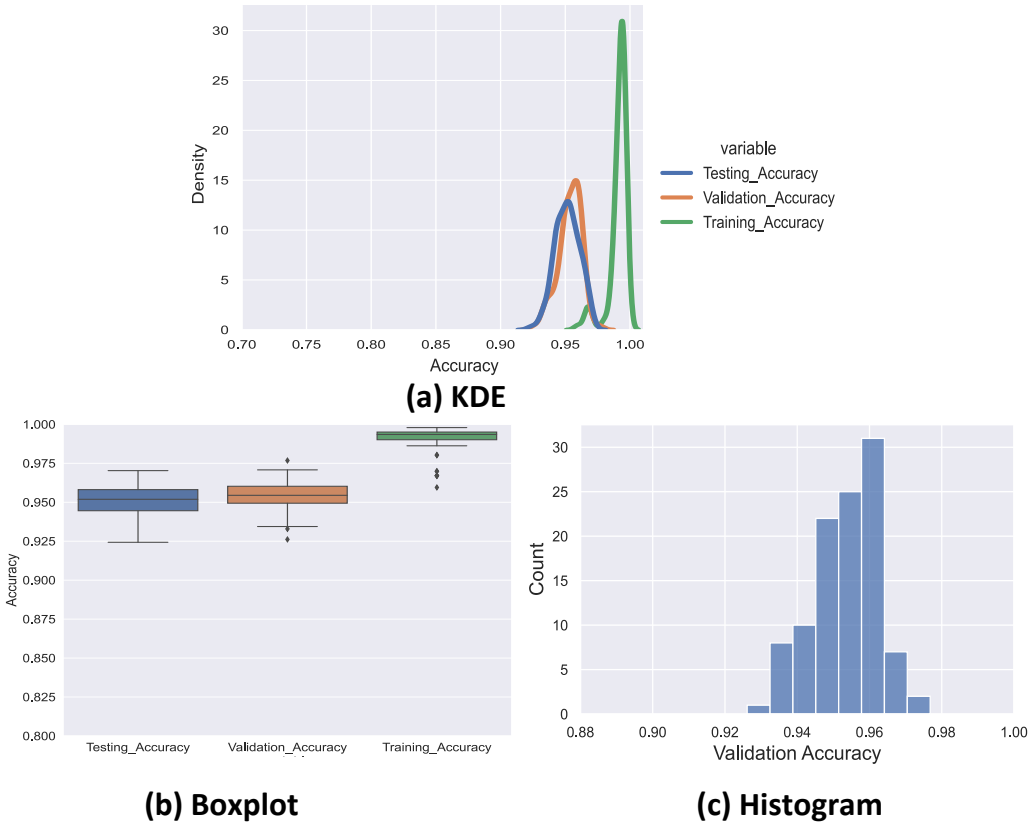

Figure 10. Kernel density estimation (**a**) and box plot (**b**) for all sets accuracy; histogram only for validation set accuracy (**c**).

5.4. Performance Comparison for All Models

Figure 11 compared the validation and training and accuracy for all BERT models stacked with different network-based models. The highest validation accuracy was 97.3% attained by ClaimsBERT, which is a classifier we proposed based on BERT followed by optimal feature map obtained from CNN, fully connected NN and a classifier.

Figure 11 compares the training and validation loss for our new model ClaimsBERT against our previous model CyBERT, BERT + MLP, BERT + BiLSTM, BERT + LSTM and BertForSequenceClassification. Based on the loss plots, we can see how well the learning rate function initialized both models. Furthermore, the loss plots illustrate how early-stopping avoids overfitting when the validation loss increases in the training phase after four epochs.

We also evaluated and compared the classification performance of our ClaimsBERT and other network-based models, with the results shown in Table 9. To evaluate the time complexity for these models, we utilized the same machine when training both models and measured the time needed to complete the training process. We then also evaluated the classification time, again utilizing the same machine, with the results for training and classification time measurements shown in Table 9. All time measurements are shown in seconds.

The classification time refers to the time required to complete the entire training process, from building the model architecture, loading the best trained model checkpoint, and training the model on the set of sequences. The classification time measurements we report in this table represent the time that the respective model required in order to test the classifier on each of the 6420 sequences in the dataset. The classification time indicates that for the exact same set of data on the same machine, the ClaimsBERT model is only 19 s, or 2.96%, slower than CyBERT, while the ClaimsBERT accuracy is two percent higher than CyBERT. We believe that the significant boost in accuracy 97% more than justifies the slight increase in classification time.

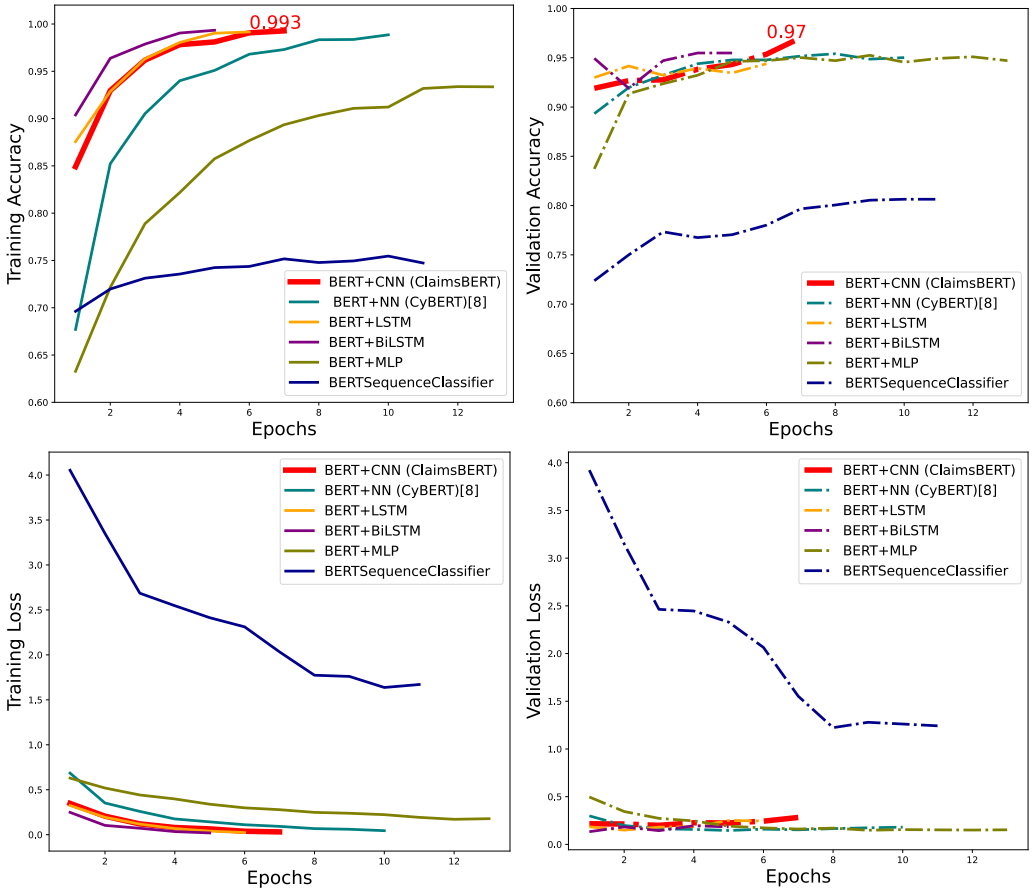

Figure 11. Comparing training accuracy (**left**), validation accuracy (**right**) for all tested models.

Table 9. Comparing ClaimsBERT and other tested models performance.

Model	Training Time ⁑	Classification Time ⁑	Trainable Parameters
BERT + CNN (ClaimsBERT)	12,873	727	110,769,474
BERT + NN (CyBERT [8])	32,970	708	108,647,026
BERT + BiLSTM	51,211	1125	112,915,970
BERT + LSTM	47,176	989	109,482,240
BERT + MLP	10,478	821	109,380,482
BERTSequence Classifier	7832	335	109,483,778

⁑ Times are in seconds.

Determining and reviewing the receiver operating characteristic (ROC) curves is a useful technique for evaluating binary classification algorithms. The ROC curve demonstrates a Dual dimensional presentation of the classifier efficiency. The False Positive Rates (FPR) versus the True Positive Rates (TPR) in classification are plotted in ROC. We would like to observe a high TPR and a low FPR in the ideal classifier scenario [69].

The True Positive Rate, which is also known as sensitivity is calculated as:

$$TPR = \frac{TP}{(TP + FN)}, \tag{10}$$

where TP measures the probability of an actual positive instance being classified as positive, and where FN is a measure of the probability that an actual positive instance will be classified as negative.

The False Positive Rate is calculated as:

$$FPR = \frac{FP}{(TN + FP)}, \tag{11}$$

where FP is a measure of the probability that an actual negative instance will be classified as positive. TN is the probability of an actual negative instance classified as negative.

The ROC curve comparison for all the models we evaluated for this study is presented in Figure 12. Compared to the other models we tested, our ClaimsBERT classifier performed better, because ClaimsBERT ROC curve is closer to the upper-left corner than the other models (Figure 12). The Area Under the ROC curve (AUC) is another important parameter for analyzing classifier models. In general, a higher AUC score indicates a better classifier performance [69]. Table 10 and Figure 12 presenting the AUC value for all language models, which indicate that ClaimsBERT had the best AUC value amongst all language models we have tested for this study.

Figure 12 shows that the highest AUC belongs to ClaimsBERT, which was 0.968. This demonstrate that ClaimsBERT is better at identifying classes than any other language models we evaluated in this study. This figure also shows that our model (ClaimsBERT) improves the AUC by more than 19 percentage points compared to BertForSequenceClassification's AUC.

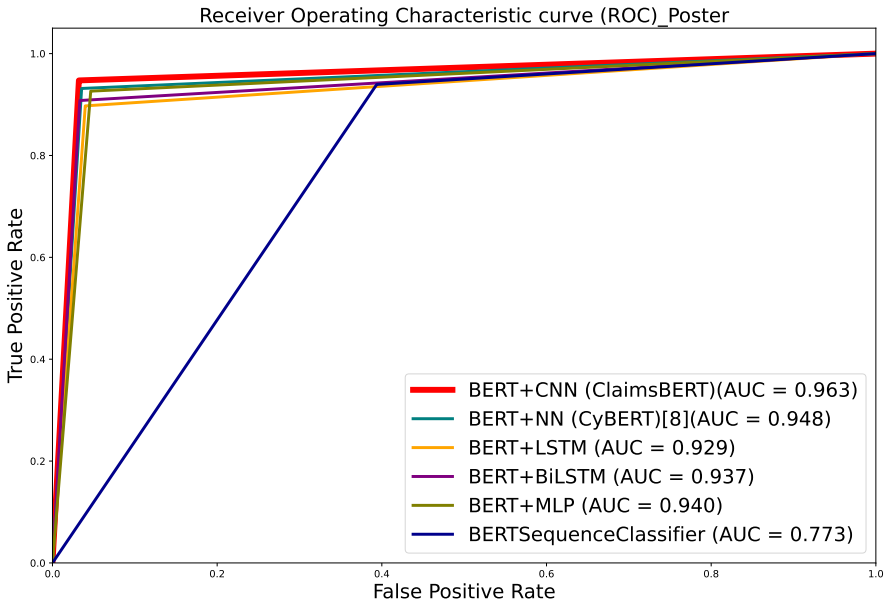

Figure 12. ROC comparison for all language models.

The Table 10 reports the required for completing model training after optimum learning rate determination for each evaluated model architecture. We found that BERT + BiLSTM training took the longest of all the models, followed by BERT + LSTM. We can also observe that the ClaimsBERT model training time is smaller than CyBERT's training time, even though it has more trainable parameters (Table 10). An explanation for the faster training process in the ClaimsBERT classifier is observable in Figure 11: Because of the effectiveness of feature map selection via CNN, the model starts out with more informative parameters and can therefore converge faster than CyBERT. Training these models does not need to be repeated, unless changes to the training dataset are necessary. Based on results reported in Table 10, we observe that adding LSTM and BiLSTM on top of BERT encoders increase the training time, and does not improve the accuracy compare with adding neural network to BERT (CyBERT model).

Table 10. Comparison across all tested models.

Model	Best Architecture	Accuracy	Macro Weighted F1	AUC	Training Time ‡	Testing Time ‡	Trainable Parameters
BERT + CNN (ClaimsBERT)	**12 Encoder 2 Convolution 2 Dense**	**0.97**	**0.96**	**0.968**	12,873	108	110,769,474
BERT + NN (CyBERT) [8]	12 Encoder 3 Dense	0.954	0.93	0.948	32,970	97	108,647,026
BERT + BiLSTM	12 Encoder 1 BiLSTM 2 Dense	0.951	0.941	0.937	51,211	142	116,693,570
BERT + LSTM	12 Encoder 1 LSTM 2 Dense	0.947	0.937	0.929	47,176	135	112,915,970
BERT + MLP	12 Encoder 3 MLP 2 Dense	0.952	0.942	0.940	10,487	85	1,093,800,482
BERTSequence Classifier	12 Encoder 1 Dense	0.76	0.72	0.773	7832	77	109,482,240

‡ Times are in seconds.

6. Conclusions

In this paper, we introduced a new concept of ClaimsBERT, a classifier model generated by incorporating feature maps via CNN into BERT, which resulted in significantly improved accuracy and performance. The proposed classifier was established by fine-tuning BERT using two convolution layers—a fully connected dense layer and a classification layer stacked on transformers.

This classifier model marks a significant improvement over BertForSequenceClassification, which is a sequence classifier introduced by BERT. We use our curated cybersecurity claim sequences dataset to train our claim classifier. Our ClaimsBERT model increases the accuracy of the claim sequence classifier compared to BertForSequenceClassification, improving it from 72 to 97 percent. The ClaimsBERT model is established based on the feature map generated via CNN. These convolution layers are able to generate a feature map with smaller overall dimensionality, which helps the model understand the input features more efficiently. We also provided an in-depth comparative analysis of ClaimsBERT to show the effectiveness of our fine-tuning strategies and our hyperparameter selection method.

The extensive experimental results demonstrate the effectiveness, efficiency and robustness of our ClaimsBERT classifier. The results demonstrate that the performance of our ClaimsBERT is better than all other language models we tested for this paper.

The development of our novel classification model was inspired by our ongoing research activities that aim to create a new unbiased, objective, and semi-supervised cybersecurity vetting approach named CYVET that focuses on feature set claims for ICS devices, in order to obtain insights into the impact these ICS devices have on an operator's overall cybersecurity posture. This is based on the detection and verification of vendor claims that are found in device documentation, identified using NLP. In this context, the research advancement presented here represents a fundamental cornerstone of our CYVET program for vetting cybersecurity claims in the broad domain of industrial control systems.

We wish to clarify that while our claims classifier is highly successful in identifying sequences that contain claims, it does not provide an indication of whether this is a claim of interest, or a generic claim unrelated to cybersecurity features. To accomplish this finer distinction, our research efforts are also now focusing on a claims type classifier, which requires a subsequent NLP processing step that follows the use of our ClaimsBERT classifier to detect claim sequences.

Furthermore, this classification does not provide an indication of whether the claim represents a statement in support of this feature, or expressing the lack of this feature. For this support indication, our research has contributed a sentiment analysis capability to our vetting framework that provides this indication. Nevertheless, we note that these steps

require additional processing and it would be preferable for ClaimsBERT to provide all of these as an output vector.

7. Future Work

Beyond the expansion of our current ClaimsBERT to incorporate claims type and support indication into its generated output, we are also actively pursuing research into alternate approaches to BERT. One example is the use of Gated Multi-Layer Perceptrons (gMLP), which have recently been introduced by the team at Google Brain [27]. The gMLP deep-learning model is reported to achieve better performance with comparable accuracy on some benchmark datasets for NLP tasks. For our future work, we will investigate the merits of gMLP for cybersecurity-specific NLP applications. Additionally, to further strengthen the exploration of feature mapping approaches within our overall architecture, we also plan to investigate a transformer two-stack architecture, which concatenates a transformer architecture on top of our existing BERT model, in order to explore the feasibility of using transformers to extract features from the BERT model's outputs.

Author Contributions: Investigation, K.A., M.H. and H.S.; writing—original draft preparation, K.A. and M.H.; writing—review and editing, K.A., M.H., H.S., K.P. and J.L.J.; supervision, H.S., M.H., K.P. and J.L.J.; project administration, K.P. and J.L.J.; funding acquisition, H.S., M.H., K.P. and J.L.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the US. Dept of Energy through a subcontract from Oak Ridge National Laboratory, project No. 4000175929 (project CYVET).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This research has been supported in part by the Department of Energy Cybersecurity for Energy Delivery Systems program, and the Oak Ridge National Laboratory.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ASGD	Averaged stochastic gradient
AUC	Area Under the ROC Curve
BERT	Bidirectional Encoder Representations from Transformers
BiLSTM	Bidirectional LSTM
CI	confidence interval
CNN	Convolutional Neural Network
CR	Cybersecurity requirements
CKG	Cybersecurity Knowledge Graph
CyBERT	Cybersecurity BERT
CYVET	Cyber-physical security assurance
ELMo	Embeddings from Language Models
FFN	Feed Forward Neural Network
GPT	Generative Pre-Training
GCN	Graph Convolutional Network
ICS	Industrial Control Systems
LR	Learning Rate
LSTM	Long Short-Term Memory
NER	Name Entity Recognition
NLP	Natural Language Processing
NN	Neural Network
NSP	Next Sentence Prediction

Multi-Layer Perceptron	MLPs
Multiscale Vision Transformers	MViT
OCR	Optical Character Recognition
OT	Operational Technology
ROC	Receiver Operating Characteristic
ULMFiT	Universal Language Model with Fine-Tuning
VSF	Vendor-supplied features

References

1. Perumalla, K.; Lopez, J.; Alam, M.; Kotevska, O.; Hempel, M.; Sharif, H. A Novel Vetting Approach to Cybersecurity Verification in Energy Grid Systems. In Proceedings of the 2020 IEEE Kansas Power and Energy Conference (KPEC), Manhattan, KS, USA, 13–14 July 2020; pp. 1–6.
2. Ameri, K.; Hempel, M.; Sharif, H.; Lopez Jr, J.; Perumalla, K. Smart Semi-Supervised Accumulation of Large Repositories for Industrial Control Systems Device Information. In Proceedings of the ICCWS 2021 16th International Conference on Cyber Warfare and Security, Cookeville, TN, USA, 25–26 February 2021; pp. 1–11.
3. Zheng, X.; Burdick, D.; Popa, L.; Zhong, X.; Wang, N.X.R. Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 3–8 January 2021; pp. 697–706.
4. Peters, M.E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. *arXiv* **2018**, arXiv:1802.05365.
5. Howard, J.; Ruder, S. Universal language model fine-tuning for text classification. *arXiv* **2018**, arXiv:1801.06146.
6. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
7. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog* **2019**, *1*, 9.
8. Ameri, K.; Hempel, M.; Sharif, H.; Lopez Jr, J.; Perumalla, K. CyBERT: Cybersecurity Claim Classification by Fine-Tuning the BERT Language Model. *J. Cybersecur. Priv.* **2021**, *1*, 615–637. [CrossRef]
9. Akbik, A.; Blythe, D.; Vollgraf, R. Contextual string embeddings for sequence labeling. In Proceedings of the 27th International Conference on Computational Linguistics, Santa Fe, NM, USA, 20–26 August 2018; pp. 1638–1649.
10. Wu, Y.; Schuster, M.; Chen, Z.; Le, Q.V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv* **2016**, arXiv:1609.08144.
11. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017; Volume 30, pp. 5998–6008.
12. Wang, Y.; Huang, M.; Zhu, X.; Zhao, L. Attention-based LSTM for aspect-level sentiment classification. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Austin, TX, USA, 1–5 November 2016; pp. 606–615.
13. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
14. Chen, M.; Radford, A.; Child, R.; Wu, J.; Jun, H.; Luan, D.; Sutskever, I. Generative pretraining from pixels. In Proceedings of the International Conference on Machine Learning, Virtual Event, 13–18 July 2020; pp. 1691–1703.
15. Fan, H.; Xiong, B.; Mangalam, K.; Li, Y.; Yan, Z.; Malik, J.; Feichtenhofer, C. Multiscale vision transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Montreal, QC, Canada, 1–17 October 2021; pp. 6824–6835.
16. Atienza, R. Vision transformer for fast and efficient scene text recognition. In Proceedings of the International Conference on Document Analysis and Recognition, Lausanne, Switzerland, 5–10 September 2021; pp. 319–334.
17. Hong, Y.; Wu, Q.; Qi, Y.; Rodriguez-Opazo, C.; Gould, S. Vln bert: A recurrent vision-and-language bert for navigation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021; pp. 1643–1653.
18. Chen, Y.C.; Li, L.; Yu, L.; El Kholy, A.; Ahmed, F.; Gan, Z.; Cheng, Y.; Liu, J. Uniter: Learning universal image-text representations. *arXiv* **2019**, arXiv:1909.11740.
19. Liu, H.; Xu, S.; Fu, J.; Liu, Y.; Xie, N.; Wang, C.c.; Wang, B.; Sun, Y. CMA-CLIP: Cross-Modality Attention CLIP for Image-Text Classification. *arXiv* **2021**, arXiv:2112.03562.
20. Li, G.; Duan, N.; Fang, Y.; Gong, M.; Jiang, D. Unicoder-vl: A universal encoder for vision and language by cross-modal pre-training. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 11336–11344.
21. Chen, S.; Guhur, P.L.; Schmid, C.; Laptev, I. History aware multimodal transformer for vision-and-language navigation. In Proceedings of the Advances in Neural Information Processing Systems, NeurIPS, Virtual, 6–14 December 2021; Volume 34, pp. 5834–5847.
22. Dou, Z.Y.; Xu, Y.; Gan, Z.; Wang, J.; Wang, S.; Wang, L.; Zhu, C.; Zhang, P.; Yuan, L.; Peng, N.; et al. An Empirical Study of Training End-to-End Vision-and-Language Transformers. *arXiv* **2021**, arXiv:2111.02387.

23. Zhai, X.; Wang, X.; Mustafa, B.; Steiner, A.; Keysers, D.; Kolesnikov, A.; Beyer, L. LiT: Zero-Shot Transfer with Locked-image Text Tuning. *arXiv* **2021**, arXiv:2111.07991.
24. Wang, Z.; Shan, X.; Yang, J. N15News: A New Dataset for Multimodal News Classification. *arXiv* **2021**, arXiv:2108.13327.
25. Oyegoke, T.O.; Akomolede, K.K.; Aderounmu, A.G.; Adagunodo, E.R. A Multi-Layer Perceptron Model for Classification of E-mail Fraud. *Eur. J. Inf. Technol. Comput. Sci.* **2021**, *1*, 16–22. [CrossRef]
26. Su, X.; You, S.; Xie, J.; Zheng, M.; Wang, F.; Qian, C.; Zhang, C.; Wang, X.; Xu, C. Vision transformer architecture search. *arXiv* **2021**, arXiv:2106.13700.
27. Parisotto, E.; Song, F.; Rae, J.; Pascanu, R.; Gulcehre, C.; Jayakumar, S.; Jaderberg, M.; Kaufman, R.L.; Clark, A.; Noury, S.; et al. Stabilizing transformers for reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual Event, 18–24 July 2020; pp. 7487–7498.
28. Tolstikhin, I.O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. Mlp-mixer: An all-mlp architecture for vision. In Proceedings of the Advances in Neural Information Processing Systems, NeurIPS, Virtual, 6–14 December 2021; Volume 34, pp. 24261–24272.
29. Liu, H.; Dai, Z.; So, D.; Le, Q. Pay attention to MLPs. In Proceedings of the Advances in Neural Information Processing Systems, NeurIPS, Virtual, 6–14 December 2021; Volume 34, pp. 9204–9215.
30. Jwa, H.; Oh, D.; Park, K.; Kang, J.M.; Lim, H. exbake: Automatic fake news detection model based on bidirectional encoder representations from transformers (bert). *Appl. Sci.* **2019**, *9*, 4062. [CrossRef]
31. Vogel, I.; Meghana, M. Detecting Fake News Spreaders on Twitter from a Multilingual Perspective. In Proceedings of the 2020 IEEE 7th International Conference on Data Science and Advanced Analytics (DSAA), Sydney, Australia, 6–9 October 2020; pp. 599–606.
32. Liu, C.; Wu, X.; Yu, M.; Li, G.; Jiang, J.; Huang, W.; Lu, X. A two-stage model based on BERT for short fake news detection. In Proceedings of the International Conference on Knowledge Science, Engineering and Management, Athens, Greece, 28–30 August 2019; pp. 172–183.
33. Sun, C.; Qiu, X.; Xu, Y.; Huang, X. How to fine-tune bert for text classification? In Proceedings of the China National Conference on Chinese Computational Linguistics, Kunming, China, 18–20 October 2019; pp. 194–206.
34. Khetan, V.; Ramnani, R.; Anand, M.; Sengupta, S.; Fano, A.E. Causal BERT: Language models for causality detection between events expressed in text. *arXiv* **2020**, arXiv:2012.05453.
35. Lee, J.; Yoon, W.; Kim, S.; Kim, D.; Kim, S.; So, C.H.; Kang, J. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* **2020**, *36*, 1234–1240. [CrossRef]
36. Beltagy, I.; Lo, K.; Cohan, A. SciBERT: A pretrained language model for scientific text. *arXiv* **2019**, arXiv:1903.10676.
37. Edwards, A.; Camacho-Collados, J.; De Ribaupierre, H.; Preece, A. Go simple and pre-train on domain-specific corpora: On the role of training data for text classification. In Proceedings of the 28th International Conference on Computational Linguistics, Barcelona, Spain, 13–18 September 2020; pp. 5522–5529.
38. Safaya, A.; Abdullatif, M.; Yuret, D. Kuisail at semeval-2020 task 12: Bert-cnn for offensive speech identification in social media. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, Barcelona, Spain, 12–13 December 2020; pp. 2054–2059.
39. Rodrigues Makiuchi, M.; Warnita, T.; Uto, K.; Shinoda, K. Multimodal fusion of bert-cnn and gated cnn representations for depression detection. In Proceedings of the 9th International on Audio/Visual Emotion Challenge and Workshop, Nice, France, 21 October 2019; pp. 55–63.
40. He, C.; Chen, S.; Huang, S.; Zhang, J.; Song, X. Using convolutional neural network with BERT for intent determination. In Proceedings of the 2019 International Conference on Asian Language Processing (IALP), Shanghai, China, 15–17 November 2019; pp. 65–70.
41. Rahali, A.; Akhloufi, M.A. Malbert: Using transformers for cybersecurity and malicious software detection. *arXiv* **2021**, arXiv:2103.03806.
42. Zhou, S.; Liu, J.; Zhong, X.; Zhao, W. Named Entity Recognition Using BERT with Whole World Masking in Cybersecurity Domain. In Proceedings of the 2021 IEEE 6th International Conference on Big Data Analytics (ICBDA), Xiamen, China, 5–8 March 2021; pp. 316–320.
43. Chen, Y.; Ding, J.; Li, D.; Chen, Z. Joint BERT Model based Cybersecurity Named Entity Recognition. In Proceedings of the 2021 The 4th International Conference on Software Engineering and Information Management, Yokohama, Japan, 16–18 January 2021; pp. 236–242.
44. Gao, C.; Zhang, X.; Liu, H. Data and knowledge-driven named entity recognition for cyber security. *Cybersecurity* **2021**, *4*, 1–13. [CrossRef]
45. Ranade, P.; Piplai, A.; Mittal, S.; Joshi, A.; Finin, T. Generating fake cyber threat intelligence using transformer-based models. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; pp. 1–9.
46. Tikhomirov, M.; Loukachevitch, N.; Sirotina, A.; Dobrov, B. Using bert and augmentation in named entity recognition for cybersecurity domain. In Proceedings of the International Conference on Applications of Natural Language to Information Systems, Saarbrücken, Germany, 24–26 June 2020; pp. 16–24.
47. Oliveira, N.; Sousa, N.; Praça, I. A Search Engine for Scientific Publications: A Cybersecurity Case Study. In Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence, Salamanca, Spain, 6–8 October 2021; pp. 108–118.

48. Ranade, P.; Piplai, A.; Joshi, A.; Finin, T. CyBERT: Contextualized Embeddings for the Cybersecurity Domain. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021; pp. 3334–3342.
49. Nguyen, C.M. A Study on Graph Neural Networks and Pretrained Models for Analyzing Cybersecurity Texts. Master's Thesis, Japan Advanced Institute of Science and Technology, Nomi, Japan, 2021.
50. Xie, B.; Shen, G.; Guo, C.; Cui, Y. The Named Entity Recognition of Chinese Cybersecurity Using an Active Learning Strategy. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 6629591. [CrossRef]
51. Pal, K.K.; Kashihara, K.; Banerjee, P.; Mishra, S.; Wang, R.; Baral, C. Constructing Flow Graphs from Procedural Cybersecurity Texts. *arXiv* **2021**, arXiv:2105.14357.
52. Yin, J.; Tang, M.; Cao, J.; Wang, H. Apply transfer learning to cybersecurity: Predicting exploitability of vulnerabilities by description. *Knowl.-Based Syst.* **2020**, *210*, 106529. [CrossRef]
53. Smith, L.N. Cyclical learning rates for training neural networks. In Proceedings of the 2017 IEEE winter conference on applications of computer vision (WACV), Santa Rosa, CA, USA, 24–31 March 2017; pp. 464–472.
54. Shrestha, A.; Mahmood, A. Review of deep learning algorithms and architectures. *IEEE Access* **2019**, *7*, 53040–53065. [CrossRef]
55. Fahad, S.A.; Yahya, A.E. Inflectional review of deep learning on natural language processing. In Proceedings of the 2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE), Shah Alam, Malaysia, 11–12 July 2018; pp. 1–4.
56. Yin, W.; Kann, K.; Yu, M.; Schütze, H. Comparative study of CNN and RNN for natural language processing. *arXiv* **2017**, arXiv:1702.01923.
57. Batbaatar, E.; Li, M.; Ryu, K.H. Semantic-emotion neural network for emotion recognition from text. *IEEE Access* **2019**, *7*, 111866–111878. [CrossRef]
58. Holland Computing Center (HCC) at University of Nebraska-Lincoln. Available online: https://hcc.unl.edu/ (accessed on 1 February 2022).
59. Zhou, C.; Sun, C.; Liu, Z.; Lau, F. A C-LSTM neural network for text classification. *arXiv* **2015**, arXiv:1511.08630.
60. Liu, G.; Guo, J. Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing* **2019**, *337*, 325–338. [CrossRef]
61. Liu, P.; Qiu, X.; Huang, X. Recurrent neural network for text classification with multi-task learning. *arXiv* **2016**, arXiv:1605.05101.
62. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
63. Yu, F.; Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv* **2015**, arXiv:1511.07122.
64. Cui, Y.; Zhou, F.; Wang, J.; Liu, X.; Lin, Y.; Belongie, S. Kernel pooling for convolutional neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2921–2930.
65. Wang, Y.; Li, Y.; Song, Y.; Rong, X. The influence of the activation function in a convolution neural network model of facial expression recognition. *Appl. Sci.* **2020**, *10*, 1897. [CrossRef]
66. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 1097–1105. [CrossRef]
67. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; Adaptive Computation and Machine Learning Series; MIT Press: Cambridge, MA, USA, 2017; pp. 321–359.
68. Dodge, J.; Ilharco, G.; Schwartz, R.; Farhadi, A.; Hajishirzi, H.; Smith, N. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv* **2020**, arXiv:2002.06305.
69. Fawcett, T. An introduction to ROC analysis. *Pattern Recognit. Lett.* **2006**, *27*, 861–874. [CrossRef]