Final Technical Report

Felix Childress
fdchild22@earlham.edu
Earlham College
Richmond, Indiana, USA

Abstract

Intrusion detection systems (IDS) play a critical role in identifying malicious activity within network traffic, yet traditional methods often rely on models that largely struggle to generalize novel threats. This project explores the application of natural language processing (NLP) techniques, particularly that of transformer-based models such as BERT (Bidirectional Encoder Representations from Transformers), to the domain of network packet analysis for intrusion detection. I propose a tokenization strategy that treats flow-level statistical features as structured sequences, enabling network behavior patterns to be analyzed similarly to natural language. Utilizing the CIC-IDS 2017 dataset, the performance of a fine-tuned BERT model is compared to that of a random forest baseline as a means of assessing the viability of NLP-driven approaches for cybersecurity applications as opposed to traditional methods. Core contributions include a novel tokenization pipeline for converting network flow features into BERT-compatible input, a systematic comparison of traditional and NLP-based detection techniques, and an evaluation of how feature representation and tokenization strategy influences detection performance.

CCS Concepts

• **Security and privacy → Intrusion detection systems**; • **Computing methodologies → Natural language processing**; *Classification and regression trees*; *Neural networks*; Supervised learning; • **Theory of computation →** *Network flows.*

Keywords

malware, networking, traffic, nlp, ids

1 Introduction

Recent years have witnessed an exponential rise in both the frequency and complexity of cyberattacks. As networked systems become more integrated into critical infrastructure, an increasing number of fields and disciplines have come to rely heavily on a continuous and secure exchange of data for operation. These systems depend on the smooth transmission of millions of network packets every second, each one carrying instructions, requests, or sensitive information. Malware targeting these systems attempt to disguise itself as legitimate traffic, making accurate detection a significant challenge.

Traditional intrusion detection systems face significant challenges as the thread landscape continues to evolve. Signature-based detection methods rely on predefined patterns of known attacks, which hinders their ability to identify new threats effectively. Anomaly-based detection methods, while better at handling these unknown attacks, suffer from high false positive rates and have a harder time adapting to changing network behaviors. Further, conventional machine learning methods applied to network security, such as support vector machines and decision trees, tend to treat network flows as independent feature vectors. This has the potential to miss the temporal and sequential patterns that characterize many sophisticated attacks, and is particularly problematic as modern attacks increasingly involve coordinated and multi-step processes.

To address these limitations, the application of various NLP techniques have begun to be explored for cybersecurity applications. As NLP models learn patterns in text by treating words as tokens in meaningful sequences, network traffic analysis could potentially benefit from treating network features as sequential elements with contextual relationships. However, little work exists focusing on adapting transformer-based architectures for network flow analysis. As such, this paper aims to address this gap by analyzing if transformer models like BERT can effectively detect network intrusions when applied to tokenized representations of network flow statistics. Data utilized originates from the CIC-IDS 2017 dataset, a widely-used benchmark for intrusion detection research.

This work tests the hypothesis that transformer-based NLP approaches can match or exceed the detection performance of traditional random forest classifiers when both are trained on appropriately tokenized flow-level network data.

2 Literature Review

2.1 Network Security

Network packets are small, individual units of data broken down from larger messages that are then transmitted across a network. A packet primarily consists of two components: the header and the payload. The header consists primarily of the metadata relevant to each individual packet, such as sequencing information or protocol types. The payload is the actual, intended message being transmitted. [15] In essence, the header provides context for the payload, acting as an "envelope" to send it on its way.

In contrast, network flow essentially represents a consecutive sequence of packets with common attributes that are transmitted between two endpoints during a single communication session. The most important and traditional of these attributes consist of:

- Source and destination IP address
- Source and destination port
- Protocol type of the sent packets

While individual packet headers provide all the granular details of each transmission, flows offer a higher- and session-level view of network activity. All of the packets within a given session that have the same attributes are aggregated into a single flow record to allow for multiple packets to be analyzed at once. This record, in essence, acts as a statistical summary of the entire session, not including any of the actual data transferred. [8]

It follows that flow-level analysis derives its statistical features from these aggregated packets, including fields such as:

- Packet size statistics
- Flag count
- Flow duration
- Forward and backward packet counts [11]

2.2 Intrusion Detection

2.2.1 Traditional Intrusion Detection. Traditional intrusion detection techniques for identifying malware in a system typically fall into one of two categories: signature-based and anomaly-based detection.

Signature-based detection relies primarily on signatures, which are recognizable patterns or characteristics that are associated with malicious activity in some way. These are extracted from a packet's payload, and from there they're compared to what is known as a signature library—essentially just a database for known or common signatures. An alert is sent forward if a match is found, and the packet is then typically redirected to a separate application that filters or disposes of the packet. [2] This proves effective in the case of malware with payloads that are both consistent and unencrypted; however, if a threat is new and lacking a defined signature, or the payload may be encrypted or otherwise obfuscated, then it is possible that it will slip under the radar. In addition, the packet header is rarely considered when it comes to signature-based detection, and thus can ignore the "red flags" manifesting in metadata.

There is also anomaly-based detection, a machine learning (ML) approach to intrusion detection that flags deviations from expected, normal activity occurring within a system. What is considered "normal activity" for a system can be identified in multiple different ways, but typically always involves analyzing the behavior of a user profile in some way over time and creating a rule-based model with this data that can be later used as a baseline comparison. [3] As opposed to signature-based detection, anomaly-based systems are much better at identifying novel threats, and as many of them are based in ML, accuracy when identifying what is or is not a threat can be improved over time. However, there are still a host of issues. Most of these arise as it is difficult to find attack-free data to train with. If this data includes attacks, any behaviors associated with them that affect the system are often mistakenly trained to be seen as normal, meaning that similar behavior in other attacks might get overlooked. However, if the data is completely attack free, this can lead to the model having an increased sensitivity to any slight change, resulting in a higher rate of false positives for malicious behavior. [16]

2.2.2 Supervised Machine Learning. Beyond these, a third approach is that of supervised machine learning, a type of ML where an algorithm learns from labeled data to identify underlying patterns and make predictions in order to "forecast" a particular outcome. [10] Supervised machine learning methods devise patterns from both benign and malicious traffic directly. This allows for them to classify new or incoming flows without relying on predefined rules (as with signature-based detection) or statistical baselines (as with anomaly-based detection), and for greater flexibility in identifying both novel and known attacks.

One of the most widely adopted of these methods for network intrusion detection is random forests. With a random forest, multiple individual trees are built on a random subset of training data and, at each split, a random subset of statistical features. The predictions of each individual tree is then aggregated and averaged together into a single forest that aim to show which flow-level characteristics most strongly resonate with or indicate malicious activity. [10] Random forests help to reduce overfitting, or a model learning training data too well, as well as improving generalization and helping to keep flexibility in adapting to new data. [6]

2.3 Natural Language Processing

Natural language processing is a subfield of computer science that aims to train computers to process, generate, and manipulate human language. [19] This includes utilizing extensive linguistic knowledge to analyze the overall structure of a language from the ground up, starting at the word level and gradually moving up to the sentence at large and the overall context of a piece of text. [5] This data is typically fed to different ML algorithms that can then create a conceptual model of how it believes a language operates, which can then be used for a variety of tasks, such as generating predictions or classifying content according to their linguistic features.

The leading architecture for large-language models and greater NLP applications at large are that of transformers. These are essentially models that feed input through a variety of neural networks in order to calculate the importance of different parts of input. [14] This allows for the model to learn the relationship that these parts have to one another, and as such create more accurate and realistic output. Some common transformer-based models include Google's BERT, which had been pre-trained using a large corpora of English text for ease of use [9], and the earlier Word2vec, which aims to create vector representations of individual words.

This approach to sequencing and processing data offers a unique advantage when it comes to intrusion detection. Packet headers and flow-level features can be "tokenized," or split into individual, manipulatable units, and then fed into ML models. [12] These models can then analyze and identify further patterns in how the different components interact both within individual packets and across entire sessions. In addition, packets and their subsequent flows are inherently sequential as a means of ensuring data is delivered and assembled in the correct order. [7] NLP models tend to be well-equipped for this type of data as they are designed to understand and model the relationships between tokenized elements. Just as words in a sentence follow grammatical and syntactical rules that determine their overall order and relationship, the fields of a packet header follow similar structural conventions. Once treated as language-like input, they can be contextually analyzed. This allows for the model to detect subtle deviations in both typical traffic patterns [20] and expected context for the elements surrounding a token [9], which can then be used to indicate malicious or otherwise abnormal activity.

Unlike traditional signature-based detection, which rely on predefined rules and analyze data in isolation, NLP models, especially those based in ML, can infer new patterns from the given data itself, which makes it both adaptable and naturally well-equipped for detecting previously unseen threats. [17] In regards to anomaly-based detection, which often struggle adapting to changing traffic patterns and distinguishing between malicious and benign activity, NLP models allow for characterization of normal behavior that takes into consideration latent patterns, contextual dependencies, and abstract relationships between elements that can be overlooked otherwise. This in turn allows for a more nuanced understanding of traffic behavior that avoids tripping up the system in the same way. [17]

3 Dataset

The dataset being utilized is the CIC-IDS 2017 dataset created by the Canadian Institute for Cybersecurity from the University of New Brunswick. A variety of attack types are simulated over the course of five work days, with each day including new attacks:

- **Monday:** fully benign traffic
- **Tuesday:** brute force attacks via FTP and SSH
- **Wednesday:** DoS and Heartbleed
- **Thursday:** web attacks via brute force, CSS, and SQL injection; Metasploit and infiltration over Dropbox
- **Friday:** ARES botnet; registry modification; DDoS

The data is stored in two forms: PCAP, which simulates realistic, raw traffic data; and CSV, which consist of pre-analyzed, labeled network flows representing traffic characteristics. It is available for free use with proper credit. [13] This project utilizes the pre-aggregated CSVs.

4 Methods

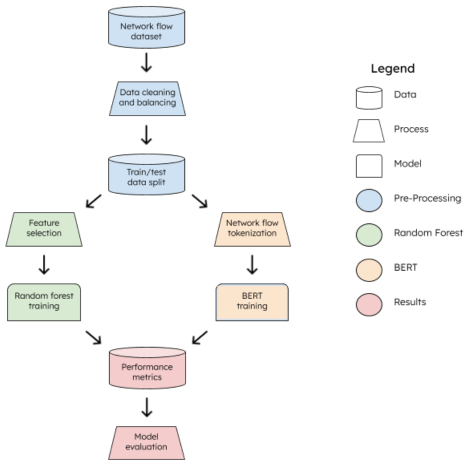

Figure 1: Data architecture diagram

4.1 Preprocessing

Data preprocessing involved several key steps:

(1) Importing each dataset using `pandas` and examining for inconsistent, unused, or missing values
(2) Removing features irrelevant to flow-level behavior, such as timestamps or specific IP addresses
(3) Balancing the dataset to get a roughly equal class representation due to high imbalance between benign and attack flows
(4) Splitting data into a training set, which includes the class label, and a testing set, which doesn't, using an 75/25 split

This is all to ensure that each model would only be trained and tested using data that is fully relevant to their specific task, improving accuracy and efficiency.[4]

4.2 Model Design and Training

4.2.1 BERT. The transformer-based model was implemented using Hugging Face's BERT, which primarily utilizes Pytorch.

4.3 Tokenization Process

For the BERT model, feature selection was performed using mutual information scoring to identify the most relevant flow statistics for each attack type. This essentially measures the statistical dependence each feature has on the class label and provide an estimation as to how much information a feature gives on whether a flow is benign or malicious. [1] The top ten features were selected based on this score, along with their class labels, and tokenized into four main semantic groups:

- Categorical features, such as ports and protocols, preserved as discrete identifiers
- Conditional features, such as flags, encoded as binary states
- Numerical features, which are divided into one of three subgroups (`LOW`, `MED`, or `HIGH`) based on percentile
- Derived features, which consists of duplicate information from other features (**e.g.** `Subflow Fwd Packets`)

This tokenization strategy converts each network flow into a sequence of tokens that can be processed by BERT's Transformer architecture. [9]

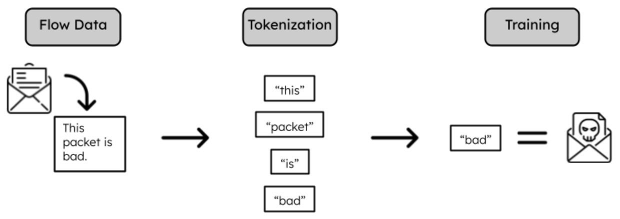

Figure 2: Graphical abstract of BERT-tokenization process

4.3.1 Random Forest. The RF classifier was implemented using the `scikit-learn` library [18], and in particular the `RandomForestClassifier` module. Two variants were trained to allow for a more fair comparison with BERT:

- `random_forest`, which was trained on all numeric flow-level features present in each dataset. This represents the traditional approach, where RF operates at full-capacity
- `random_forest_matched`, which was restricted to only train with the same ten features selected for BERT using mutual information scoring to allow for a more direct comparison between the two models.

Both models were initialized with:

- 200 estimators, or trees
- No maximum depth constraints; while in a single decision tree this may lead to overfitting, due to the feature subsampling done, it's actually the optimal configuration considering the size of the dataset
- `min_samples_split = 5`, which establishes a minimum number of features needed to split a node in a tree
- `min_samples_leaf = 3`, which establishes a minimum number of features needed for each leaf of the tree
- `max_features="sqrt"`, where each split considers $\sqrt{n}$ features, with n being the total number of features
- `bootstrap = True`, which allows for each tree to repeatedly retrain and resample data as it goes deeper
- `random_state = 42` for reproducibility
- `n_jobs=-1` to allow for parallel processing across all available CPU cores for training acceleration

Training was performed on a normalized feature set excluding class label to improve speed and performance.[4] Non-numerical columns and metadata columns, such as `Label` and `Label_binary`, were also excluded in training. Both models also utilized 75/25 train-test splits. Feature importance scores were extracted post-training to analyze the relative contribution of each flow statistic to model decisions.

5 Results

Model performance was evaluated using standard classification metrics:

- F1 score, which measures overall model performance based on its precision and recall scores [20]
- Accuracy
- Precision
- Recall
- ROC-AUC curve
- Training time
- Confusion matrices for each model to visualize class-level performance

5.1 Tuesday

After pre-processing, the Tuesday dataset contained 445,645 network flows, of which:

- 432,074 were benign (96.9%)
- 7,938 were patator over FTP (1.7%)
- 5,897 were patator over SSH (1.3%)

To address this class imbalance, a balanced subset of 27,664 flows was created by sampling 13,832 benign flows to match the combined attack count. This was then split 75/25 for training and testing.

Table 1: Tuesday Results (SSH and FTP Brute Force)

Metric	RF (All)	RF (Matched)	BERT
F1-Score	0.999855	0.999566	0.996542
Accuracy	0.999855	0.999566	0.996530
Precision	0.999711	0.999711	0.993107
Recall	1.000000	0.999422	1.000000
ROC-AUC	1.000000	0.999999	0.998648
Training Time (s)	0.418238	0.288319	207.857667

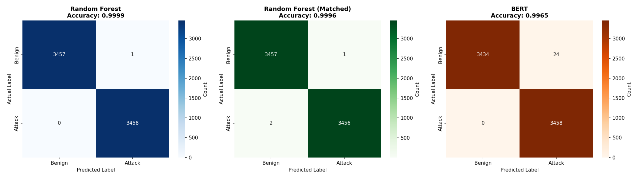

Figure 3: Model Confusion Matrices (Tuesday)

The models exhibited slightly different error profiles. The feature-matched RF model was the only one of the three with instances of false negatives, or malicious flows being misclassified as benign. This isn't an issue with the BERT model or the full feature RF, as neither misclassify any flows. However, BERT noticeably has more of an issue with false positives; the full feature RF model only flags one false positive, while BERT counts 24. Curiously, BERT and the full feature RF model both have perfect recall scores.

5.2 Wednesday

After pre-processing, the Wednesday dataset contained 691,406 network flows, of which:

- 440,031 were benign (63.6%)
- 231,073 total DoS attacks (36.5%) using HULK (231,073 flows), GoldenEye (10,293 flows), slowloris (5,796 flows), and SlowHTTPTest (5,499 flows)
- 11 were Heartbleed (0.001%)

Balancing this proved to be trickier, both due to the sheer size of the data and the overrepresentation of DoS attacks as opposed to Heartbleed. Ultimately, to keep consistency, the same process was kept as with Tuesday, with 251,723 benign flows were sampled to match the total account count. This was done as, in a real attack scenario, it's unlikely that only one type of attack will be used; as such, all attack types were considered. Further, the 75/25 split was maintained for training (377,584 flows) and testing (124,852 flows).

The differences in each models from Tuesday to Wednesday is stark, but each points towards their own limitations. At 13 false positives and 38 false negatives, the high accuracy of the full feature random forest is very much indicative either of data leakage somewhere in the pre-processing pipeline or the possibility that DoS attacks produce traffic that is so distinctive that they end up

Table 2: Wednesday Results (DoS and Heartbleed)

Metric	RF (All)	RF (Matched)	BERT
F1-Score	0.999595	0.994645	0.970563
Accuracy	0.999595	0.989542	0.974203
Precision	0.999396	0.999857	0.966725
Recall	0.999793	0.994673	0.970450
ROC-AUC	0.999996	0.999776	0.994092
Training Time (s)	14.257869	13.463640	4984.638574

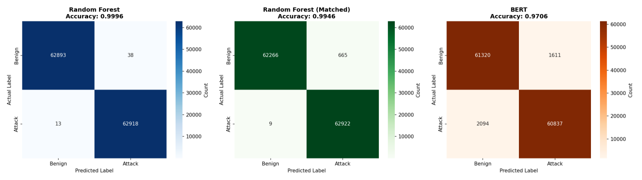

Figure 4: Model Confusion Matrices (Wednesday)

looking trivially separable when the full list of features is available. The model's reliance on these features may allow it, for example, to exploit metadata patterns or dataset artifacts that would not otherwise generalize attack variants. In contrast, for the feature-matched RF model, since samples are forceably matched, the model cannot rely on metadeta patterns in the same way the core RF model can. This is why the results are different between the two, with 9 false positives and 665 false negatives. BERT, however, suffered: 2,094 false positives and 1,611 false negatives. Interestingly, this suggests that the model was likely more conservative in its decision-making when identifying malicious traffic, and with fewer "references" to go off, more end up slipping through unnoticed.

In addition, BERT's performance significantly degraded from Tuesday to Wednesday, having nearly 2,100 false positives and over 1,600 false negatives. This suggests that the model struggled with the decision boundary itself as opposed to being systematically biased towards one class. The most likely reason for this seems to be that the discretization of various features while during tokenization, particularly for numerical features such as packet timing or flow duration, may have accidentally destroyed information that could have been necessary to distinguish between normal traffic bursts and a coordinated DoS attack.

From a practical perspective, false negatives are almost universally considered more costly than false positives, which make BERT and its feature-matched RF more problematic to actually use despite relatively high metrics.

5.3 Thursday

After pre-processing, the Tuesday dataset contained 458,626 network flows, of which:

- 456,752 were benign (99.5%)
- 2,180 were web attacks (0.4%) using brute force methods (1,507 flows), XSS (652), and SQL injection (21)
- 36 were instances of infiltration (0.007%)

The Thursday datasets contain an extreme class imbalance. Actual attack instances making up less than a percent of the given data, which creates a fundamentally different challenge compared to the rest of the days in the dataset. To address this class imbalance, a much smaller subset had to be created, consisting of 2,216 sampled benign flows to match the combined attack count. This was then split 75/25 for training and testing.

Table 3: Thursday Results (Web Attacks and Infiltration)

Metric	RF (All)	RF (Matched)	BERT
F1-Score	0.992767	0.994645	0.951712
Accuracy	0.992780	0.993682	0.950361
Precision	0.994565	0.996370	0.926496
Recall	0.990975	0.994673	0.978339
ROC-AUC	0.999270	0.999776	0.984198
Training Time (s)	0.430922	0.248987	90.301319

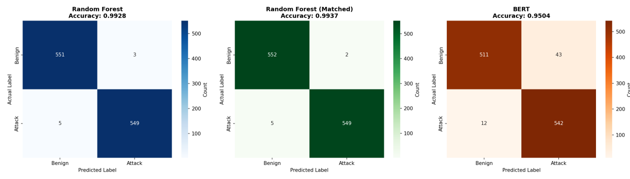

Figure 5: Model Confusion Matrices (Thursday)

Interestingly, for the first time, the feature-matched RF model outperformed that of the full feature RF. This suggests that the full feature model may contain a significant amount of noise or collinear features that hindered the model when training data was, for the first time, severely limited, prohibiting learning just the most informative features. Both models overall performed well, with only five false positives each

However, a similar question to Wednesday arises whether web attacks and infiltration attempts produce sufficiently distinctive patterns to be trivially detectable, or whether the aggressive undersampling done created an artificially easy task.

BERT's performance lagged once again behind both models, though this largely has to due to BERT's architecture. Transformers, being language models, require a significant amount of data in order to effectively learn meaningful patterns. With such a small pool of data to work with, the model likely failed to adequately fine-tune its pre-trained weights in a way that would work with detecting malicious activity. Further, BERT's token discretization continues to cause issues, as web attacks notably result in subtle anomalies rather than the coordinated statistical patterns of a DoS. The difference between a legitimate web request and an SQL injection attempt, for example, may have hinged on precise numeric values that cease to exist once data was sorted into the `MED` or `HIGH` bins.

The Thursday results therefore reveal a critical limitation in applying transformer-based models to network intrusion detection. BERT's parameter-heavy architecture has the potential to become more of a liability than an asset as its need for data contradicts the reality of new attacks popping up with only a few initial examples.

5.4 Friday

After pre-processing, the Friday dataset contained 702,718 network flows, of which:

- 414,322 were benign (58.9%)
- 158,930 were port scans (22.6%)
- 128,027 were DDoS LOIT (18.2%)
- 1,966 were botnets ()

The combination of reconnaissance activity, volume-based attack, and coordinated malware presented another unique balancing challenge, as each model would have to get properly acquainted with identifying all three within a flow rather than honing in on the behavior of any one attack type. The class distribution was much more balanced, however, with 41.1% of flows being malicious, offering a naturally more even dataset. To ensure fully equal representation, a balanced subset of 577,570 flows was created by sampling 288,785 benign flows to match the combined attack count. This was then split 75/25 for training and testing, which resulted in the largest evaluation set out of all four days analyzed at 144,393 samples.

Table 4: Friday Results (Port Scans, DDoS, and Botnets)

Metric	RF (All)	RF (Matched)	BERT
F1-Score	0.999571	0.996684	0.986251
Accuracy	0.999571	0.996690	0.986156
Precision	0.999764	0.998512	0.979520
Recall	0.999377	0.994861	0.993074
ROC-AUC	0.999999	0.999935	0.998053
Training Time (s)	14.363404	6.193152	2967.522249

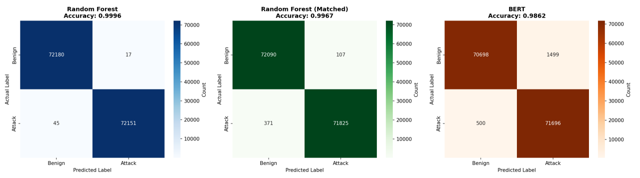

Figure 6: Model Confusion Matrices (Friday)

The results from analyzing the Friday dataset reaffirmed the patterns observed throughout the week, with the full feature RF model having a near-perfect score in all metrics. The model held an error rate of 0.043% as well, with only 17 false positives and 45 false negatives out of the 144,393 test samples. This represents the strongest performance of any model of any day, and suggests that RF's ensemble approach effectively captures the statistical signatures of malicious behavior when provided with comprehensive flow-level features. However, the persistent near-perfect metrics continue to raise concerns about potential leakage or artifacts within the data itself. The feature-matched RF model also performed extremely well, missing less than 0.51% of attacks as evident by its recall value while maintaining 99.6% precision. This performance continues to suggest that mutual information-based feature selection successfully identified the features most indicative of malicious behavior in each flow, and that these can be effectively leveraged by RF without requiring the full feature space.

BERT's performance improved drastically relative to Thursday, but continued to remain below both RF variants. Friday's data granted BERT the highest recall it's displayed on all four days, suggesting that it learned to identify most attack samples with sufficient training data, but at the cost of a lower precision. The model showed an error rate of 1.03%, producing 1,499 false positives and 500 false negatives. This is sixteen times higher than the full feature RF model and four times that of the feature-matched. This trade-off is problematic from a practical point, as such a high error rate would generate thousands of false alarms that could potentially mask legitimate threats in its noise.

There are several factors that may explain the performance gap in BERT and RF. For one, the diversity of Friday's attack types may have exceeded BERT's capacity to learn from tokenized sequences. The variation in patterns would require the model to learn fundamentally and drastically different "grammars" that lack a unified syntax in the way of linguistic structure within the same classification task. In addition, despite the larger dataset, BERT evidently continued to struggle with the lack of information afforded to the discretized features. As such, the model seemed to have picked up overly broad generalizations in malicious activity that resulted in a bias towards false positives. FEffective port scan detection, for example, would rely on precise port numbers and connection timing information that would have been essentially discarded as the sequences were tokenized. A scan targeting ports 20 to 25 would, with this pipeline, produce identical tokens as ports 8080 to 8085 if both scans happened to be below the 33rd percentile, which would obscure important characteristics that the model would be able to use to otherwise distinguish between them.

6 Conclusion

Contrary to the initial hypothesis that a pretrained language model would outperform traditional methods of intrusion detection, the results consistently show that the more traditional random forest models achieved equal or superior performance across all evaluated datasets. This was the case even when the random forest classifier was constrained to the same limited feature subset as BERT, maintaining performance levels largely comparable to that of the full model. While performance was competitive and despite BERT successfully learning patterns, BERT gains no measurable advantage over random forest in most cases. In instances where BERT does appear to match, it is marred by the substantial amount of computational resources needed for proper training.

Random forest's ensemble approach, in comparison, effectively captures the statistical signatures of malicious behavior when provided with comprehensive flow-level features. Further, the mutual information-based feature selection done for BERT training is shown to be able to effectively identify features most indicative of malicious behavior in each flow. These features can evidently be effectively leverage by random forest without requiring the full feature space, in most cases having comparable or superior metrics, and as such shows that BERT should have been able to mark out behaviors based on most of these features alone.

A key factor to these findings lies in the nature of the input data, which are engineered, low-level numerical descriptors. These features are well-suited to tree-based models, which excel at handling heterogeneous input, capturing non-linear interactions, and performing feature selection. In contrast, when these same values are discretized and reduced to categorical labels, while this allowed for BERT to successfully use the data to make predictions, it significantly limited any semantic richness or continuous information present in the original sequence. As a result, BERT was unable to leverage its pretrained linguistic knowledge or contextual modeling capabilities. Instead, the model functioned more as a high-capacity sequence classifier over symbolic tokens that were ultimately shallow.

This highlights limitations when applying language models to structured, tabular data. The findings also suggest that the underlying structure of the CIC-IDS2017 flow features is highly separable using tree-based decision boundaries, and as such, the additional representational capacity of transformer models is largely unnecessary. The performance gap therefore is less indicative of an issue with transformer architectures, but rather a reflection of an inherent mismatch between data modality and the inductive biases of each model.

7 Future Work

As this project utilized the pre-analyzed network traffic flows available in the dataset's CSV files, there was no testing done on raw packet data by way of the PCAP files. As such, testing should be done to see if results change significantly when raw numeric features are combined with contextual token embeddings.

In addition, due to the limitations of BERT's training on an English corpus, it may be useful to train a transformers model from scratch on large traffic datasets instead. This would allow for the model to learn network-specific "language" patterns from the data itself as opposed to operating under the assumption that the data is structured in the same way as a natural language. Further, continuous token embeddings may be better suited to retaining any semantics present in the original sequences as opposed to the strict discrete categorization done.

References

[1] Fatemeh Amiri, MohammadMahdi Rezaei Yousefi, Caro Lucas, Azadeh Shakery, and Nasser Yazdani. 2011. Mutual information-based feature selection for intrusion detection systems. *Journal of network and computer applications* 34, 4 (2011), 1184–1199.

[2] Ahmad Azab, Mahmoud Khasawneh, Saed Alrabaee, Kim-Kwang Raymond Choo, and Maysa Sarsour. 2024. Network traffic classification: Techniques, datasets, and challenges. *Digital Communications and Networks* 10, 3 (2024), 676–692.

[3] Monowar H Bhuyan, Dhruba Kumar Bhattacharyya, and Jugal K Kalita. 2013. Network anomaly detection: methods, systems and tools. *Ieee communications surveys & tutorials* 16, 1 (2013), 303–336.

[4] Volkan Çetin and Oktay Yıldız. 2022. A comprehensive review on data preprocessing techniques in data analysis. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi* 28, 2 (2022), 299–312.

[5] KR1442 Chowdhary and KR Chowdhary. 2020. Natural language processing. *Fundamentals of artificial intelligence* (2020), 603–649.

[6] Nabila Farnaaz and MA Jabbar. 2016. Random forest modeling for network intrusion detection system. *Procedia Computer Science* 89 (2016), 213–217.

[7] Glen Gibb, George Varghese, Mark Horowitz, and Nick McKeown. 2013. Design principles for packet parsers. In *Architectures for Networking and Communications Systems*. IEEE, 13–24.

[8] Rick Hofstede, Pavel Čeleda, Brian Trammell, Idilio Drago, Ramin Sadre, Anna Sperotto, and Aiko Pras. 2014. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys & Tutorials* 16, 4 (2014), 2037–2064.

[9] Md Saiful Islam and Long Zhang. 2024. A Review on BERT: Language Understanding for Different Types of NLP Task. *Preprints. org* (2024).

[10] Tammy Jiang, Jaimie L Gradus, and Anthony J Rosellini. 2020. Supervised machine learning: a brief primer. *Behavior therapy* 51, 5 (2020), 675–687.

[11] Bingdong Li, Jeff Springer, George Bebis, and Mehmet Hadi Gunes. 2013. A survey of network flow applications. *Journal of Network and Computer Applications* 36, 2 (2013), 567–581.

[12] Sabrina J Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot, et al. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *arXiv preprint arXiv:2112.10508* (2021).

[13] Ranjit Panigrahi and Samarjeet Borah. 2018. A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. *International Journal of Engineering & Technology* 7, 3.24 (2018), 479–482.

[14] Telmo Pires, António Vilarinho Lopes, Yannick Assogba, and Hendra Setiawan. 2023. One Wide Feedforward Is All You Need. In *Proceedings of the Eighth Conference on Machine Translation*, Philipp Koehn, Barry Haddow, Tom Kocmi, and Christof Monz (Eds.). Association for Computational Linguistics, Singapore, 1031–1044. doi:10.18653/v1/2023.wmt-1.98

[15] Leslie F Sikos. 2020. Packet analysis for network forensics: A comprehensive survey. *Forensic Science International: Digital Investigation* 32 (2020), 200892.

[16] Robin Sommer and Vern Paxson. 2010. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*. IEEE, 305–316.

[17] Zarrin Tasnim Sworna, Zahra Mousavi, and Muhammad Ali Babar. 2023. NLP methods in host-based intrusion detection Systems: A systematic review and future directions. *Journal of Network and Computer Applications* 220 (2023), 103761.

[18] Abdelaziz Testas. 2023. Random forest classification with scikit-learn and pyspark. In *Distributed Machine Learning with PySpark: Migrating Effortlessly from Pandas and Scikit-Learn*. Springer, 243–258.

[19] David Okore Ukwen and Murat Karabatak. 2021. Review of NLP-based systems in digital forensics and cybersecurity. In *2021 9th International symposium on digital forensics and security (ISDFS)*. IEEE, 1–9.

[20] Yong Yang and Xing Peng. 2025. BERT-based network for intrusion detection system. *EURASIP Journal on Information Security* 2025, 1 (2025), 11.