Feature Selection for Network Intrusion Detection

Charles Westphal
University College London
London, United Kingdom
charles.westphal.21@ucl.ac.uk

Stephen Hailes
University College London
London, United Kingdom
s.hailes@ucl.ac.uk

Mirco Musolesi
University College London
London, United Kingdom
University of Bologna
Bologna, Italy
m.musolesi@ucl.ac.uk

Abstract

Network Intrusion Detection (NID) remains a key area of research within the information security community, while also being relevant to Machine Learning (ML) practitioners. The latter generally aim to detect attacks using network features, which have been extracted from raw network data typically using dimensionality reduction methods, such as principal component analysis (PCA). However, PCA is not able to assess the relevance of features for the task at hand. Consequently, the features available are of varying quality, with some being entirely non-informative. From this, two major drawbacks arise. Firstly, trained and deployed models have to process large amounts of unnecessary data, therefore draining potentially costly resources. Secondly, the noise caused by the presence of irrelevant features can, in some cases, impede a model's ability to detect an attack. In order to deal with these challenges, we present Feature Selection for Network Intrusion Detection (FSNID) a novel information-theoretic method that facilitates the exclusion of non-informative features when detecting network intrusions. The proposed method is based on function approximation using a neural network, which enables a version of our approach that incorporates a recurrent layer. Consequently, this version uniquely enables the integration of temporal dependencies. Through an extensive set of experiments, we demonstrate that the proposed method selects a significantly reduced feature set, while maintaining NID performance. Code available at https://github.com/c-s-westphal/FSNID.

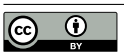
CCS Concepts

• **Security and privacy → Network security**; • **Mathematics of computing → Information theory**; • **Computing methodologies → Machine learning**.

Keywords

Feature selection, network intrusion detection, information theory, classification.

ACM Reference Format:
Charles Westphal, Stephen Hailes, and Mirco Musolesi. 2025. Feature Selection for Network Intrusion Detection. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '25), August 3–7, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3690624.3709339



KDD '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1245-6/25/08
https://doi.org/10.1145/3690624.3709339

1 Introduction

Network Intrusion Detection (NID) remains a key focus of the information security community given its substantial economic impact. For example, IBM estimates showed that, in 2023, the average cost of a data breach for the afflicted party was USD 4.45 million [24]. During standard network operations malicious attacks are typically absent. Therefore, the earliest Intrusion Prevention Systems (IPSs) were statistical methods developed to detect irregularities in network data [16]. These methods were among the early implementations of anomaly detectors, a broader category of techniques that continues to be effective in NID [25, 37]. Despite their success [1], traditional anomaly detection methods, primarily focused on single time series analysis, fall short in leveraging the interrelations among multiple data series. In contrast, anomaly detection methods designed to exploit these inter-series relationships often face significant computational complexity challenges [17, 22].

Starting from these considerations, NID has also been studied as a tabular data classification problem [28, 32, 58]. According to this paradigm, a machine learning (ML) model receives as input a vector that summarizes the network data at a given instance in time, and outputs a boolean or probability indicating whether or not the system is under attack, and, in some cases, what type of attack it is under. Usually, the vector of features is extracted from raw network data, which are collected and stored as packet capture (PCap) files. These can be represented as a highly-dimensional set of time-series. A dimensionality-reduction algorithm is then applied to extract individual features. Examples of techniques employed for this task include principal component analysis (PCA) [47, 50], Linear Discriminant Analysis (LDA) [10, 36] and, more recently, deep learning-based approaches [59]. Although these methods have the ability to cluster the data into features, they are unable to assess how relevant and informative they will be for classification. Consequently, some will be entirely non-informative and should be removed. Additionally, given the NID context, it is possible to outline three further reasons as to why one may wish to reduce the required feature data. Firstly, the acquisition of the network data may be costly since specific software (and, in some cases, hardware) has to be deployed. Secondly, measurements might introduce delays into the network operations. Finally, in the context of cybersecurity, minimizing measurements is of strategic importance. Since in certain situations, an advantage can be gained by giving the impression that the network is not being monitored at all. Furthermore, an understanding of the most important features helps penetration testers tailor their attack vectors to exploit areas where detection is least likely, thereby enhancing security.

This has led to a growing body of work in feature selection for NID [33, 34, 42, 52]. It is possible to identify two groups of

approaches. The first aims to maximize mutual information (MI), i.e., the correlation, between the chosen feature set and the attack vector labels [8, 11, 20, 48]. However, this requires the definition of the desired set size. In feature-rich datasets, including those reporting network traffic, the search space of this hyperparameter becomes impractically large. Moreover, the complexity of these algorithms scales polynomially in time with the number of features to be assessed. Again, this is non-desirable for highly dimensional data. The second, and most common, general paradigm is to rank features based on MI [38], or Shapley values [54], before selecting a predetermined number of the best performing features. However, these methods struggle to deal with highly correlated features [19, 35]. This issue is especially pertinent in network contexts, where homogeneous traffic patterns are repeated in individual features. Additionally, none of the methods discussed so far account for temporal dependencies, which are of key importance in NID. While the use of ML models with recurrent layers has been shown to enhance NID performance, the feature selection methods used to determine inputs for these models fail to leverage the same temporal relationships. Consequently, the set of selected features may lack crucial information that could otherwise be utilized effectively by the classification system. We identify four major challenges when selecting features for NID:

- **The complexity problem.** How does the complexity of the method scale with the number of selected features? (In NID where the number of features is large, complex algorithms quickly become impractical.)
- **The length k problem.** How does the feature selection method define the final number of selected features? (This is of particular relevance for datasets that have large numbers of features as the size of the resulting search space scales proportionally.)
- **The redundancy problem.** How does the feature selection method handle highly correlated variables? (This is key in NID as multiple features portrey similar traffic patterns.)
- **The temporal relationships problem.** Can the feature selection method incorporate temporal dependencies? (In NID incorporating temporal dependencies into classification tasks is known to improve accuracy, therefore, this should also inform the features selected.)

In response to these challenges, we present Feature Selection for Network Intrusion Detection (FSNID), a novel information-theoretic method that naturally overcomes such problems. FSNID relies upon the ability to neurally approximate the *entropy transferred* from the labels dataset to each feature. Once this quantity is obtained, we then exclude variables from our set of desired features if they are deemed to be uninformative. Furthermore, given it relies upon function approximation, we can use recurrent layers to incorporate time-based dependencies within our Transfer Entropy (TE) calculation. For example, using a simple DNN to estimate the entropy transferred will lead to a value that only reflects the relationships present in the data at each iteration of training. Meanwhile, if we were to use an Long-Short Term Memory (LSTM) network [23] to approximate this function we can also include relationships that span multiple iterations, overcoming the *temporal relationships problem* presented above. We show that incorporating such dependencies reduces the number of required features further. Consequently, our method reduces the feature space significantly, while not degrading classification performance. The contributions of the paper can be summarized as follows:

- We develop a method of feature selection for NID, based on excluding variables that *transfer* negligible entropy to the attack vector labels.
- The computation of the TE-based measure in FSNID depends on general function approximation methods. By incorporating recurrent layers into this estimation process, we leverage *temporal* dependencies.
- We show experimentally that using the described techniques leads to significant reductions in the size of the feature space without degrading classification performance.

2 Related Work

Feature selection can be defined as the process of reducing the dimensionality of the input to the ML model while maintaining the same classification or regression performance. Methods for feature selection can be divided into two conceptual groups, namely wrapper and filter methods, where the latter have been deployed more commonly in the field of feature selection for NID due to their performance for high-dimensional feature spaces.

Shapley value based filter methods. These involve selecting variables due to the underlying relationships within the data, independently from the model itself. A popular method relies on Shapley values as a score for 'feature importance' [13]. Shapley values, originated by Shapley as a means to efficiently allocate resources in cooperative game theory [54], were adapted for feature importance by having the input features 'cooperate' to predict the output of an ML model [39]. Despite Shapley value's widespread adoption, they fail to overcome the *redundancy problem*. Two perfectly correlated features, will both return Shapley values of zero, in spite of their importance for predicting the output of the model. Furthermore, [27] and [60] showed that, in some cases, non-negative Shapley values could be assigned to features that have no impact on the final outcome of an ML model. In response to this, literature has been developed which overcomes such issues. Namely, the authors of [26] and [9] developed methods to assign feature importance's that adhere to certain properties, which have been pre-specified in their axioms for feature importance. One such axiom ensures that when calculating the second feature importance of a pair of perfectly correlated features, it must receive the same score as the first. This leads to scores that reflect the importance of a feature in predicting the model output, therefore, negating the redundancy issue as described by [19, 35]. However, the use of these scores lead to the following problem: when selecting features for a model, either of the correlated features is just as likely to be selected. Consequently, both are selected. In practice, we require only one of these features. In computer networks, and networks more generally, we often see highly correlated features. Consequently, Shapley-based methods for feature selection are not well suited to network traffic data.

Information-theoretic filter methods. As a result, information-theoretic filter methods are more commonly used for NID. These methods aim to exploit underlying relationships in the data to select the features without knowledge of the model in use. A standard

method [4] (see also the variations presented in [8, 48]) involves augmenting the feature that maximizes the MI between the chosen subset and the ground truth. However, such methods fail to overcome the *length k problem* and the *complexity problem*. Another related method involves exploring all the permutations of the entire feature set to optimize the MI between the subset and the target variable [11, 20, 68]. To manage the computationally intensive task of searching through permutations, such techniques limit the size of the search space by fixing the cardinality of these subsets as k. However, the selected features cannot guarantee optimality, unless all possible values of k are compared. Therefore, these methods also suffer from *the length k problem*. Despite these drawbacks, both techniques represent the state of the art and they are widely adopted by the NID community [3, 6, 52].

Feature selection methods for network intrusion detection. Many specialized feature selection methods for NID have been developed [3, 6, 33, 34, 42, 52]. Often, they ensure that the complexity of the algorithm scales non-polynomially in time with respect to the number of features, but this comes at the cost of robustness. For example, in [12] 'irrelevant' features are removed on a case-by-case basis if they are uncorrelated with the target. This is done without considering if these features provide *synergistic* information when considered in groups, whereas in [52], instead of searching the full space of potential feature subsets, the authors only consider the subsets of size k. Therefore, such methods suffer from the length k problem.

3 Preliminaries

3.1 Notation and Terminology

We denote sets of random variables using calligraphic symbols (e.g., $\mathcal{X}$), single random variables using capital letters (e.g., X), and their realizations with lowercase letters (e.g., x). To label a selected feature set we use $\mathcal{X}_*$. The function $A(\cdot)$ operates on random variables, yielding the set of all their possible realizations. Meanwhile, $\mathcal{P}(\cdot)$ produces the powerset of its argument. We use Shannon's entropy (e.g., $H(X)$) to represent uncertainties. Let the following list denote the features of a computer network at time t, $\xi^t = (x_1^t, x_2^t ... x_N^t)$, where ξ^t will act as the input to our classification model. By sampling jointly from the elements of our input list $(x_1^t, x_2^t \dots x_N^t)$, and from the realizations of the ground truth y^t, we realize not only the set of all possible features as random variables (written as $\mathcal{X} = \{X_1, X_2 \dots X_N\}$), but also the ground truth (written as Y).

3.2 Information-theoretic Concepts

We now briefly review the key concepts from Information Theory at the basis of the proposed method.

Mutual information (MI). The MI is a non-linear extension of the Pearson correlation. Formally it is written as the difference between a conditional and non-condition Shannon's entropy [53]. For example:

$$I(X;Y) = H(X) - H(X|Y). \tag{1}$$

Equation 1 describes the reduction in uncertainty of the observation of X caused by observing Y. MI is widely used for feature selection, both when applied to NID [3, 6, 52] and more generally [8, 11, 14, 20, 48].

Transfer entropy (TE). TE measures the directed transfer of information between two random processes [51]. More formally, we write this quantity as:

$$TE_{X\to Y} = H(X^{t+1}|X^t) - H(X^{t+1}|X^t, Y^t). \tag{2}$$

Equation 2 measures the *extra* reduction in uncertainty of the *next* realization of X by considering the current observation of Y.

Redundancy. Redundancy measures the difference between the maximum and observed uncertainty of an ensemble of random variables. If this difference is large, it implies that many of the variables provide indistinguishable information [45]. To explain why this is a key concept in feature selection, we present the following example. Let there exist two variables (X_i and X_j) that provide identical information regarding the target (Y), more formally $I(X_i;Y) = I(X_j;Y) = I(X_j, X_i;Y)$. An optimal feature selection algorithm will include only one, to reduce the input dimensionality, while maintaining the total information. However, this has been proven difficult to rigorously implement [19, 35]. Some methods [7, 15], assign a feature importance score of zero to both variables, resulting in neither being selected. Conversely, other techniques identify them as equally important and select both [9, 26].

Synergy. Synergy describes the extra information provided due to variables being considered in combination as opposed to individually. The classic example given of such a relationship is implemented using an XOR function. Let there exist two binary string variables X_i and X_j. Furthermore, let us suppose that a third variable Y is calculated as the XOR of these two variables realizations. It follows that X_i and Y and X_j and Y are completely uncorrelated ($I(X_i;Y) = I(X_j;Y) = 0$), but when considered together they are fully informative ($I(X_i, X_j;Y) = H(X)$) [65]. Features that are characterized by these relationships are assigned negligible feature importance scores by correlation-based measures (such as those in [8, 12, 14]), despite the information they provide in combination.

4 Method

4.1 Overview

We now present our information-theoretic filter method of feature selection for NID. Herein, we consider only filter-based approaches, whether that be FSNID or baselines, due to the advantages they possess for highly dimensional feature spaces (see section 2). Initially, we discuss the task of detecting network intrusions using supervised learning. The subsequent part of this section is dedicated to elucidating our feature selection technique.

We start by presenting the mathematical details of the TE based measure that signifies feature importance. Following that, we outline the algorithm we develop to leverage this measure, discussing how it addresses the first three problems highlighted in Section 1. We also present a procedure for estimating the value of the measure. We then describe the method through which we integrate temporal dependencies into the estimation process, thus resolving the issue of temporal relationships.

4.2 Network Intrusion Detection as Supervised Classification

In this paper, analogously to [2, 29, 31, 57, 64, 70], NID is viewed as a supervised classification task. Let us start by considering the

case in which all the N features act as inputs to the classifier. They describe the properties of a network at an instance in time, and are characterized by the realizations of $\mathcal{X}$ at time t, which we indicate with $\xi^t = (x_1^t, x_2^t, \ldots, x_N^t)$. Meanwhile, y^t describes whether or not these properties correspond to benign or malicious traffic. The primary objective of an NID classifier is to infer the value of y^t as output, given ξ^t as input. This process can be conceptualized as a mapping function f, which translates the space of all possible combinations of network properties into the space of all possible attack types, with one such type being the possibility of no attack at all. Formally, this is represented as $f : A(\mathcal{X}) \mapsto A(Y)$. In this paper, our goal is not only to detect attacks, but also to classify their type. Consequently, we model f using either a multi-layer-perceptron or a Long-Short Term Memory (LSTM) network with a negative log likelihood loss, where the number of potential outputs is equal to the total number of attack types plus one (the one characterizes benign traffic). An optimal function f^* outputs the correct attack label for each iteration (more formally $f^*(\xi^t) = y^t$). The goal of feature selection is to reduce the dimensionality of the input to our classifier, without affecting its performance.

4.3 Measuring Transfer Entropy

We now define the measure $\Phi_{X_i;\mathcal{X}\to Y}$, which quantifies the difference in the uncertainty of the target variable, Y, when the set $\mathcal{X}$, does and does not include some feature of interest $X_i \in \mathcal{X}$:

$$\Phi_{X_i;\mathcal{X}\to Y} = H(Y|\mathcal{X}_{\setminus X_i}) - H(Y|\mathcal{X}). \tag{3}$$

This quantity is an adaptation of Schreiber's TE, and describes how features transfer information to the ground truth variable [51]. Equation 3 measures *the reduction in uncertainty of Y's observations given X_i is added back to $\mathcal{X}$*. Therefore, if $\Phi_{X_i;\mathcal{X}\to Y} = 0$, adding the variable X_i back to the set $\mathcal{X}$ does not reveal further information about the target variable. The core principle of the methodology is to exclude such variables from the set of features, as they are said to be uninformative regarding the target. This measure has the desirable quality that by considering the effect of *combining* the feature of interest with the remaining variables in $\mathcal{X}$, it natively considers high-order synergistic relationships. However, in the following section, we will illustrate that evaluating this measure for multiple features simultaneously does not overcome the *redundancy problem*, for analogous reasons to the methods developed in [7].

4.4 Description of the FSNID Algorithm

In this section, we discuss the affects of the *redundancy problem*, before describing the design of the overall FSNID algorithm. We then analyze the performance of FSNID in handling *the complexity* and *length k problems*.

Dealing with redundancy. In practice, a simple simultaneous application of Equation 3 to all variables fails to overcome the *redundancy problem* [19, 35]. To explain this, let us suppose there are two perfectly correlated features (X_i, X_j), and we remove either one from the set $\mathcal{X}$ to calculate $\Phi_{X_i;\mathcal{X}\to Y}$. All the information that would be lost by removing X_i is replicated in X_j and vice versa. As a result, we calculate $\Phi_{X_i;\mathcal{X}\to Y} = 0$ and $\Phi_{X_j;\mathcal{X}\to Y} = 0$, and both features are excluded from $\mathcal{X}$, in spite of their ability to reduce the uncertainty of the target. To mitigate the *redundancy problem*, we apply the measure defined in Equation 3 in a sequential manner, as detailed in Algorithm 1. Under such circumstances, permanently eliminating variables from the set $\mathcal{X}$ (line 4 of Algorithm 1), if they satisfy $\Phi_{X_i;\mathcal{X}\to Y} = 0$ ensures that, upon encountering the first of the redundant variables, it is removed. As a result, there is no longer redundant information in the set $\mathcal{X}$, and the calculation of the remaining variables' contribution is heavily simplified [19, 21, 35]. Therefore, this simple method overcomes the *redundancy problem*. A schematic illustration of this process can be seen in Figure 1.

Algorithm 1 FSNID.

Input: Network features $\mathcal{X}$ and ground truth labels Y, respectively (see Section 3.1). **Output**: $\mathcal{X}_*$ (a desirable subset of features).

1: Initialize $\mathcal{X}_* = \{\}$
2: **for** $i = 1$ to N **do**
3: **if** $\Phi_{X_i;\mathcal{X}\to Y} = 0$ **then**
4: $\mathcal{X} = \mathcal{X}\setminus\{X_i\}$
5: **else**
6: $\mathcal{X}_* = \mathcal{X}_* \cup \{X_i\}$
7: **end if**
8: **end for**
9: **return** $\mathcal{X}_*$

Dealing with the length k problem and scalability. It is trivial to see that Algorithm 1 only requires the calculation of $\Phi_{X_i;\mathcal{X}\to Y}$ once per feature. Consequently, its complexity is linear in time with respect to the number of features. Otherwise, the method becomes non-scalable and inapplicable to highly-dimensional spaces. Furthermore, we note that this Algorithm derives the size of $\mathcal{X}_*$, overcoming the length k problem. Again, this is highly desirable for large feature spaces.

4.5 Transfer Entropy Neural Estimation

In this section, we explain how we neurally estimate $\Phi_{X_i;\mathcal{X}\to Y}$. Unless we restrict our domain of interest to scenarios not applicable to the real world, the calculation of $\Phi_{X_i;\mathcal{X}\to Y}$ will be intractable without infinite samples. For this reason, we use a neural function approximator to estimate this value. One of the most commonly estimated information theoretic quantities is MI [5, 40, 46, 49, 63]. In particular, we estimate $\Phi_{X_i;\mathcal{X}\to Y}$ as the difference between two I's. More formally, we have:

$$\Phi_{X_i;\mathcal{X}\to Y} = I(\mathcal{X}; Y) - I(\mathcal{X}_{\setminus X_i}; Y). \tag{4}$$

The method we use for estimating MI is based on the work by Belghazi et al. [5], as it is applicable to both continuous and discrete variables. In particular, in [5] the authors prove that the MI can be re-written via the Donsker Vardhan representation, such that:

$$I(\mathcal{X}; Y) = \sup_{T:A(\mathcal{X})\times A(Y)\mapsto\mathbb{R}} \mathbb{E}_{P_{\mathcal{X},Y}}[T(\mathcal{X}, Y)] - \log \mathbb{E}_{P_{\mathcal{X}},P_Y}[e^{T(\mathcal{X},Y)}] \tag{5}$$

where $\mathbb{E}_{P_{\mathcal{X},Y}}$ is the expectation under the joint distribution $P_{\mathcal{X},Y}$ and $\mathbb{E}_{P_{\mathcal{X}},P_Y}$ is the expectation under the marginal distribution $P_{\mathcal{X}} \otimes P_Y$. The form of Equation 5 is such that it can be used for gradient ascent, where $T(\mathcal{X}, Y)$ is the output of a neural estimator, which takes as its input the full set of variables $\mathcal{X}$ and the target Y sampled according to the expectation to which $T(\mathcal{X}, Y)$ is subjected. For

instance, $\mathbb{E}_{P_{\mathcal{X},Y}}[T(\mathcal{X},Y)]$ (the left term) is the average output of the neural function approximator when presented with $\mathcal{X}$ and Y sampled jointly. Meanwhile, $\log \mathbb{E}_{P_{\mathcal{X}},P_Y}[e^{T(\mathcal{X},Y)}]$ (the right term) is the log of the average exponential output when $\mathcal{X}$ and Y are sampled marginally. The equivalence in Equation 5 ensures we can use the relationship in Equation 4 to calculate $\Phi_{X_i;\mathcal{X}\to Y}$, and apply Algorithm 1 to identify our subset of features. More formally, $\Phi_{X_i;\mathcal{X}\to Y}$ can be written as:

$$\begin{aligned}\Phi_{X_i;\mathcal{X}\to Y} = &\sup_{T:A(\mathcal{X})\times A(Y)\mapsto\mathbb{R}} (\mathbb{E}_{P_{\mathcal{X},Y}}[T(\mathcal{X},Y)] - \log \mathbb{E}_{P_{\mathcal{X}},P_Y}[e^{T(\mathcal{X},Y)}]) \\ &- \sup_{U:A(\mathcal{X}_{\setminus X_i})\times A(Y)\mapsto\mathbb{R}} (\mathbb{E}_{P_{\mathcal{X}_{\setminus X_i},Y}}[U(\mathcal{X}_{\setminus X_i},Y)] \\ &- \log \mathbb{E}_{P_{\mathcal{X}_{\setminus X_i}},P_Y}[e^{U(\mathcal{X}_{\setminus X_i},Y)}]).\end{aligned} \tag{6}$$

Calculating our measure using Equation 6 yields a positive real number, expressed in nats, which quantifies the reduction in uncertainty of the target variable due to the inclusion of feature X_i. This implies that our measure - and the method employed to calculate it - offers an intuitive interpretation, aiding users not only in terms of the feature selection task but also in understanding why some features are selected over others.

4.6 Incorporating Time Dependencies

We now describe the motivation of including a recurrent layer in the classification and feature selection tasks. We select an LSTM-based solution in our implementation after comparing different architectures that integrate temporal dependencies. In Section 5.4, we report the experimental results supporting this choice.

An LSTM-based NID system receives network data of sequence size s, before processing it for the purpose of classification. Therefore, our NID mapping function no longer takes as its input ξ^t, but rather it receives a list of time ordered network data that it uses to infer the attack status. More formally $f^{LSTM}(\xi^{t-s}, \xi^{t-s+1} \dots \xi^t) \in A(Y)$, where f^{LSTM} is an updated mapping function such that $f^{LSTM} : A(\mathcal{X}^{t-s:t}) \mapsto A(Y)$. The set $\mathcal{X}^{t-s:t}$ in its full form is $\mathcal{X}^{t-s:t} = \{X_1^{t-s:t}, X_2^{t-s:t} \dots X_N^{t-s:t}\}$, and the superscript $^{t-s:t}$, as applied to variables, or sets of variables, indicates that sampling is conducted such that realizations are accompanied by the s values that came before it in time, where s indicates the sequence size. The recurrent layer in an LSTM has the ability to incorporate all the information within the sequence, despite the extended time horizon [23]. This differs from standard methods for function approximation that only consider a single instance in time. Detecting dependencies in time is of importance when classifying cyberattacks. To explain why, we present the following simple example. Firstly, let us suppose a system is under attack at time t, it is obviously more likely to also be under attack at time $t+1$. Additionally, non-automated attacks are likely to occur at times when people are awake. Methods that account for time dependencies can exploit such relationships, boosting their detection accuracy [30, 43, 67, 69].

However, in this paper, we intend to perform the classification task on a reduced subset of selected features. Therefore, the method used to select these features should also incorporate dependencies in time (introduced before as the *temporal relationship problem*). Otherwise, our feature set may omit information our classification system would typically utilize. To do so, we rewrite Equation 5 such that our estimation of $\Phi_{X_i;\mathcal{X}\to Y}$ now incorporates temporal relationships:

$$\begin{aligned}&I(\mathcal{X}^{t-s:t};Y^{t-s:t}) \\ &= \sup_{L:A(\mathcal{X}^{t-s:t})\times A(Y^{t-s:t})\mapsto\mathbb{R}} \mathbb{E}_{P_{\mathcal{X}^{t-s:t},Y^{t-s:t}}}[L(\mathcal{X}^{t-s:t},Y^{t-s:t})] \\ &- \log \mathbb{E}_{P_{\mathcal{X}^{t-s:t}},P_{Y^{t-s:t}}}[e^{L(\mathcal{X}^{t-s:t},Y^{t-s:t})}].\end{aligned} \tag{7}$$

The function $L(\cdot)$ is again the output of a neural function approximator, where now the input is the variables $\mathcal{X}^{t-s:t}$ and $Y^{t-s:t}$. Each variable is sampled as chronologically ordered sequences of size s. These are then combined either jointly or marginally according to the expectations to which they are subjected. Furthermore, a similar update can be made to Equation 6; although, we omit it here for space and clarity. Throughout the remainder of the paper, we will distinguish between our methodology that incorporates temporal dependencies from that which does not by using the terms, LSTM-based FSNID and FSNID, respectively.

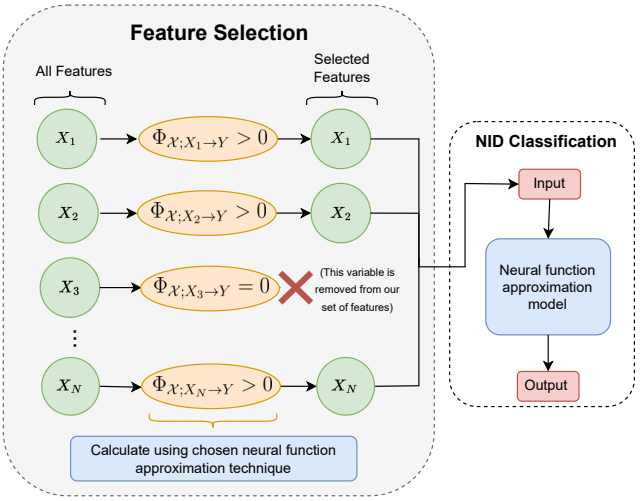


Figure 1: Diagrammatic representation of FSNID.

4.7 Methods for Approximating State Variable Inclusion Conditions

Since we estimate $\Phi_{X_i;\mathcal{X}\to Y}$ neurally rather than calculating it analytically, the values of $\Phi_{X_i;\mathcal{X}\to Y}$ randomly fluctuate. Therefore, the condition $\Phi_{X_i;\mathcal{X}\to Y} > 0$ in Algorithm 1 is susceptible to the inclusion of uninformative variables in our target set. In this section, we explain how we approximate the condition $\Phi_{X_i;\mathcal{X}\to Y} = 0$ for varying estimates.

Our approach mirrors that introduced in [66]; specifically, it involves implementing a null model for comparative analysis. In this paper, we introduce a random variable NM into $\mathcal{X}$, whereas [66] remove dependencies by shuffling one of its pre-existing members. The expectation is that the ground truth should not be dependent on NM, implying that $\Phi_{NM;\mathcal{X}\to Y} = 0$. Therefore, variables that demonstrate a significant transfer of entropy to Y are expected to

deviate from this null model. We assume that both the null model and the features in $\mathcal{X}$ follow normal distributions. By adopting the Neyman-Pearson's methodology [44], we identify variables X_i within $\mathcal{X}$ where $\Phi_{X_i;\mathcal{X}\to Y}$ has a 95% likelihood of not conforming to the range specified by the null model. These identified variables are considered to exhibit a statistically significant divergence from the null model, thereby satisfying the condition $\Phi_{X_i;\mathcal{X}\to Y} > 0$.

5 Experimental Evaluation

In this section, we outline the motivation and results of our experiments. Initially, we specify the research questions we aim to address in the evaluation of the method and we detail how these informed the selection of our comparative baselines. Our evaluation will first focus on assessing FSNID's ability to parsimoniously select features for NID. Finally, we will analyze the temporal complexity of the proposed method.

5.1 Research Questions

We now outline the specific research questions that are of importance when aiming to select optimal features for NID.

- **RQ1.** *To what extent does FSNID reduce the feature set compared to the baselines, and how does this affect classification performance?*
- **RQ2.** *How does FSNID generalize across a variety of datasets with different characteristics?*
- **RQ3.** *Is FSNID scalable in the presence of large feature spaces?*

5.2 Baselines

In this section, we introduce five methods for feature selection that we will use as comparative baselines.

As examples of classic lightweight and linear in time methods with respect to the number of features, we selected Permutation Importance (PI) and least absolute shrinkage and selection operator (LASSO) [7, 62]. They are both highly scalable, and appropriate for the large datasets of interest. We also adopted the ultra marginal feature importance (UMFI) algorithm with optimal transport [26]. We chose this technique as it is a state-of-the-art method for assigning feature importance. To the best of our knowledge, this is the only technique that attempts to natively deal with highly correlated variables, in a manner that is linear in time with respect to the number of features. This is essential in the presence of highly dimensional feature spaces. We also compare to the mutual information firefly algorithm (MIFA) [52], as a representative and popular example of metaheuristics. Additionally, we include the classic and widely adopted Conditional Likelihood Maximization (CLM) framework by Brown et al. [8]. Unlike the first three baselines, the temporal complexity of MIFA and CLM is non-linear in time with respect to the number of features. Where possible, we utilize null models to determine whether or not a feature is informative. However, in the MIFA case we must derive the number of features k using other methods. We explain in detail how we handle the length-k problem for each baseline in Appendix B.

5.3 Feature Selection Performance Evaluation

We first address **RQ1** and **RQ2** by presenting the results of an extensive evaluation, comparing the feature selection performance of FSNID and the baselines.

5.3.1 Experimental Settings. First, we introduce the datasets used for the evaluation. We use the train-test split datasets if provided. Otherwise, we apply a standard $80:20$ split.

TON-IoT. This dataset provides a comprehensive view of network traffic from diverse IoT and IIoT sensors, combined with system traces from both Linux and Windows hosts, improving the generality and diversity of the datastreams available from the Bot-IoT dataset [41]. For this study, the dataset from a Windows 10 network was employed, capturing various system activities across 124 attributes. The attack vector in this dataset is comprised of seven unique attack types, accounting for 53.0% of the target data, while the remaining 47.0% is benign.

NSL-KDD. Designed to address the shortcomings of the KDD'99 dataset [61], NSL-KDD encompasses 41 features, split into 34 numeric, 4 binary, and 3 nominal attributes. Categorized based on packet, content, traffic, and host characteristics, it classifies attacks into four primary categories: DoS, R2L, U2R, and Probe. These main categories are further subdivided into forty specific attack classes [52], which in total characterize 75.6% of the total traffic, while the remainder is benign. Unlike the TON-IoT dataset, NSL-KDD contains a large variety of attack styles that are applicable to non-IoT computer networks.

CIC-DDoS2019. This dataset focuses on a variety of DDoS attacks, mirroring genuine real-world traffic conditions. Notably, it was generated using data describing the behavior of 25 users across a variety of protocols. For the purpose of our research, we focus on the UDPLag dataset, as this attack type is uncommon, and unique to this dataset [56]. For this dataset, 98.9% of the data was benign while the rest were attacks.

UNSW-NB 15. The UNSW-NB 15 dataset consists of 49 features and nine attack groups, which are associated with 68.1% of the traffic, while 31.9% is benign. The classification task is particularly challenging for this dataset as it requires the consideration of time dependencies in order to achieve high classification accuracies.

CICIDS17. This dataset offers a snapshot of benign traffic with a sparse presence of state-of-the-art attacks, ensuring the data mirrors real-world conditions [55]. 88 network flow features were extracted from the activities of 25 users considering different protocols. Attacks such as Brute Force FTP and DDoS were incorporated during different times of the day and week, with 63.6% benign and 36.4% malicious behaviors.

These datasets essentially encompass a broad spectrum of attacks, each varying in frequency and occurring across different applications, systems, or devices. As a result, by evaluating this combination of datasets we aim to answer **RQ2**. In Figure 2, we present the classification accuracy achieved when using all features compared to the case in which only subsets (selected according to the methods taken into consideration) are used. Furthermore, we also present false positives and F1 scores. These are of key importance when assessing NID due to the presence of imbalanced datasets (e.g., CIC-DDoS2019). Therefore Figure 2, allows direct comparison of each methods ability to select a useful feature set. To

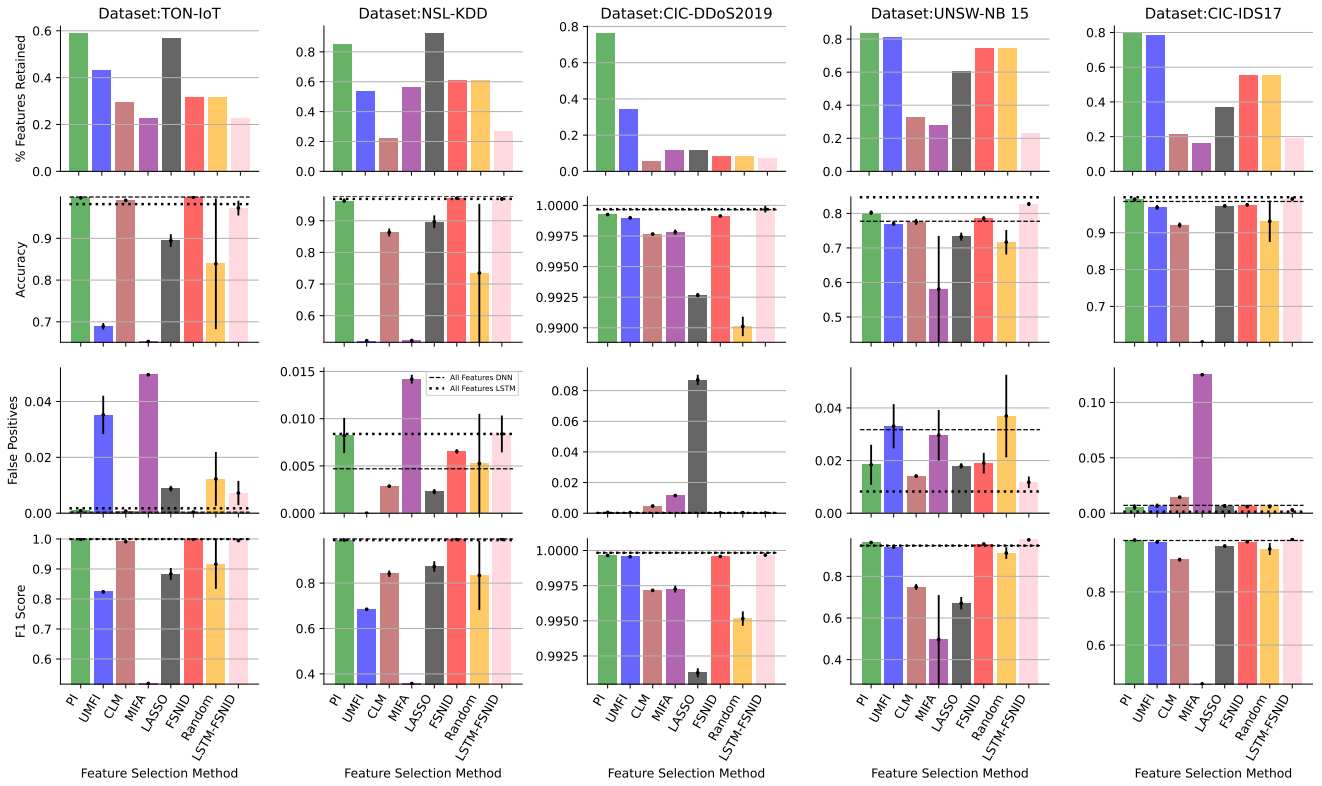


Figure 2: Comparison of the vanilla (red bars) and LSTM-based (pink bars) versions of FSNID to PI (green bars), UMFI (blue bars), CLM (brown bars), MIFA (purple bars) and LASSO (black bars). The yellow bar corresponds to a randomly selected set of features equal in size to the set of features selected using our vanilla method. From top to Bottom we present the proportion of features that were retained using each method, the accuracy achieved during the classification task using that set, the false positive rate and F1 score.

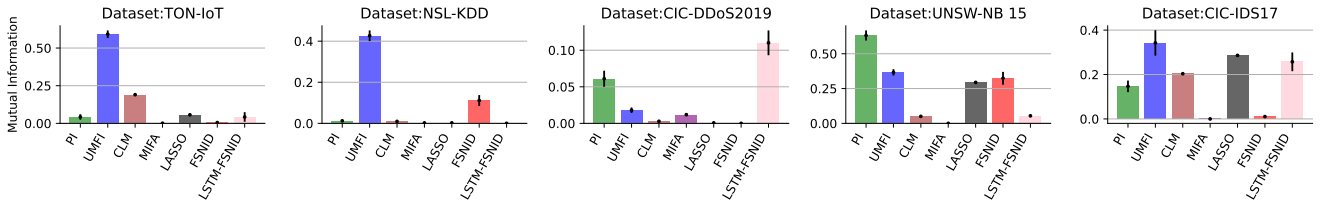


Figure 3: Performance comparison of FSNID against the chosen baselines in terms of their ability to deal with highly correlated features. Specifically, we plot the average MI shared between the top three features for each method with respect to each dataset.

verify these methods outperform a stochastic baseline, we compare the results to a set of randomly chosen features, equal in size to the set selected by our vanilla method. Additionally, in Figure 3, we examine the MI between the top three features. This analysis shows how each method addresses the *redundancy problem*. The calculation of $\Phi_{X_i;\mathcal{X}\to Y}$ and the classification results are obtained over 5 runs. We adopt ±95% confidence intervals for the classification task. Hyperparameter tuning is conducted via a simple grid search (for a full discussion, please refer to Appendix C).

5.3.2 Experimental Results. **TON-IoT.** In this dataset, FSNID achieves near-optimal performance despite a large reduction in the feature space; it is apparent that UMFI's performance does not surpass that of a randomly chosen set of variables equivalent in size to our DNN-based approach. Notably, this occurs even though UMFI selects a larger number of features than FSNID. This can be explained with reference to Figure 3, in which it is possible to observe higher correlation among the features chosen by UMFI. This suggests that many of the attributes selected by UMFI might be superfluous for

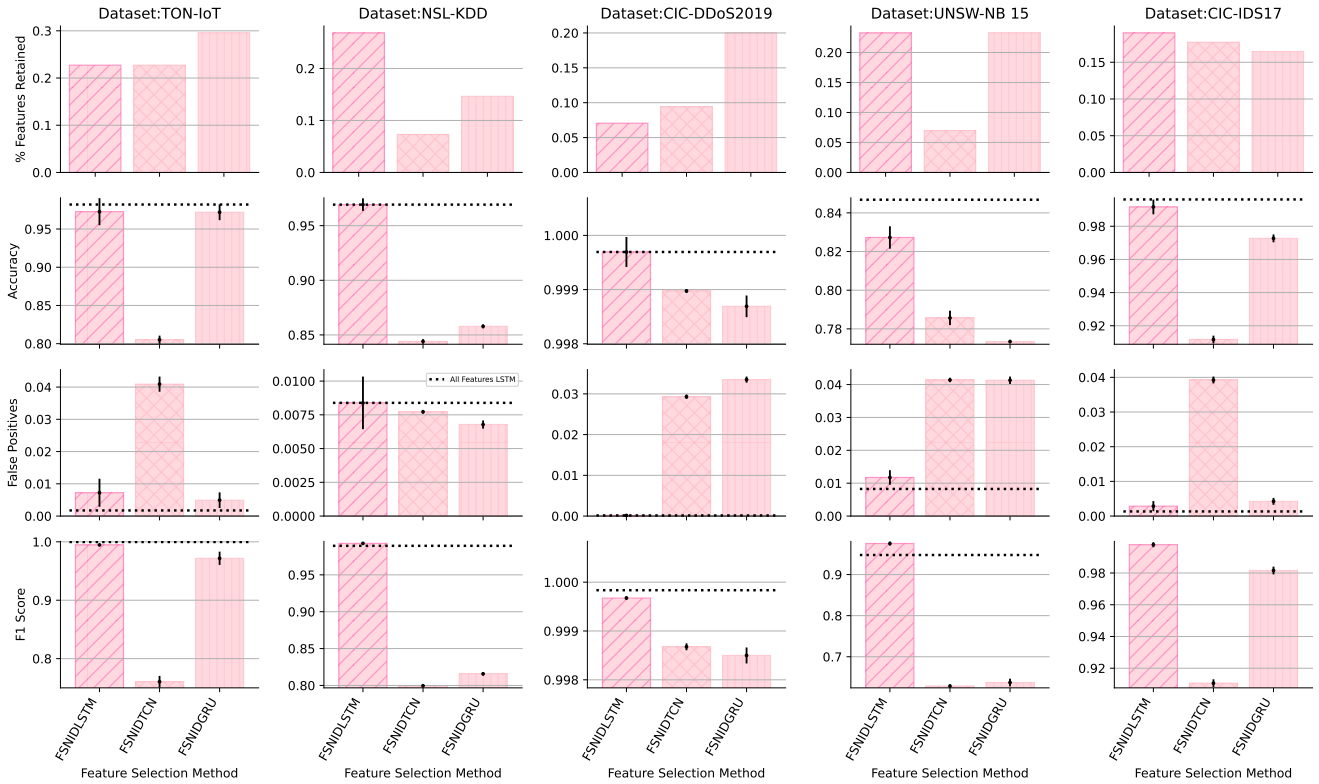


Figure 4: Performance comparison of different neural architectures to evaluate their ability to incorporate temporal dependencies into feature selection and subsequent classification tasks.

effective classification. Such redundant or highly correlated features can impede a model's ability to detect attacks. The MIFA method correctly evaluates the formed feature subsets (using MI). However, the heuristic employed to update these subsets is inefficient. This approach assigns uniform probabilities across all features with the potential to be switched on or off, without considering their individual contributions to overall performance. Consequently, to discover a viable feature set using MIFA, one must either evaluate a very large number of feature subsets (fireflies) or allow a significant number of feature-flipping events. Both options increase the computational requirements beyond practical limits for large datasets. The LASSO algorithm selects a large number of features without achieving competitive performance. LASSO exclusively considers linear relationships; for this reason, it is unable to incorporate synergistic interactions or resolve complex redundancies. The inability to detect synergistic relationships can lead to too few features being detected, while the inability to handle highly correlated features can lead to over-selection [18]. In this case, the latter occurs. While CLM selects fewer features, it is marginally outperformed by FSNID.

NSL-KDD. We observe that the performance of UMFI and LASSO is poor, given the amount of features they select. This is likely due to over-representation of correlated features, which is clearly evidenced for UMFI in Figure 3. In this case, we observe that CLM selects the fewest number of features, although this does prevent the model learning an optimal detection strategy. This underperformance is primarily due to an inability to recognize synergistic features as important. To clarify, CLM focuses on adding individual features based on their ability to increase MI, neglecting the potential benefits of their combinations: as a result, synergistic combinations remain undiscovered. Overlooking such interactions consistently leads to too few features being selected. This has an impact on performance. On the other hand, FSNID natively considers synergistic interactions, leading to the selection of an appropriate feature set which in turn leads to good performance.

CIC-DDoS2019. We observe that FSNID achieves comparable performance with both PI and UMFI despite these methods selecting much larger feature sets. CLM's inability to consider purely synergistic interactions ensures that it again selects the fewest number of features. That said, it outperforms LASSO and MIFA.

UNSW-NB 15. It is observed that although the LSTMFSNID method selects a limited number of features, it performs comparably with a full feature subset. Given the UNSW-NB 15 dataset is characterized by attacks that are time-dependent, the use of an LSTM not only reduces the required number of features but also leads to improved performance. CLM here selects the fewest features and performs well. However, this methods inability to solve the complexity problem means the selection of this set took 2.5x the amount of time needed for FSNID, as presented in Appendix A. Meanwhile, the

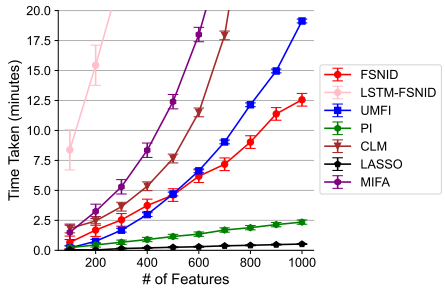


Figure 5: Temporal complexity of FSNID and comparators with respect to the number of features.

MIFA algorithm performs worse than random selection, although this is likely due to the low number of features MIFA selected.
CIC-IDS 17. Also in this case, we observe that FSNID drastically reduces the size of the set of features, while not significantly affecting the predictive power of the model. LSTM-based FSNID is able to achieve even better performance, for similar reasons to that described for UNSW-NB 15.

To summarize, in these experiments, by means of a variety of datasets, we have demonstrated our method's ability to select a reduced set of highly informative features that result in near-optimal classification accuracy. In other words, we have provided empirical evidence to address both **RQ1** and **RQ2**. Furthermore, we have demonstrated that by exploiting the temporal relationships present in the data, we can enhance classification performance while using even smaller sets of features.

5.4 Recurrent Layer Ablation Studies

In the previous section, we demonstrated that using an LSTM when estimating Φ can improve the performance of FSNID. In this section, we motivate the choice of an LSTM by comparing three architectures that can incorporate temporal dependencies, for hyper-parameters refer to Appendix C. Specifically, we compare the abilities of Temporal Convolutional Networks (TCN), Gated Recurrent Units (GRU), and LSTM during the NID feature selection and classification tasks. As shown in Figure 4, the LSTM model consistently outperforms TCN and GRU models across the metrics investigated. However, the LSTM is the most complex architecture of the three. This complexity results in a slight increase in runtime, as detailed in Appendix A. Although, this extended runtime is not significant when considering the superior performance indicated by the NID results.

5.5 Analysis of the Computational Cost

In this section, we present a set of results that provides evidence for addressing **RQ3** following an experimental methodology similar to that presented in [26].

5.5.1 Experimental Settings. In the following experiments, we intend to demonstrate that the temporal complexity of our method scales linearly with respect to the number of features. We are not interested, at this stage, in each methods ability to select features. Consequently, to reduce the computation associated with this evaluation, we use simple synthetic data. The synthetic features ($X_i \in \mathcal{X}$) and the target (Y) are independent stochastic binary arrays of length 500, for a discussion of the hyperparameter see Appendix C.

5.5.2 Experimental Results. The results indicate that the complexity of our method scales approximately linearly in time with respect to the number of features. Although UMFI is theoretically linear in time, its practical implementation involves a pre-processing step that leads to the curve observed in Figure 5. CLM exhibits a non-linear time complexity with respect to the total number of features N and the selected number k, scaling proportionally to $O(Nk - \frac{k(k-1)}{2})$. This complexity arises as each selected feature must maximally increase the correlation with the target, making this a greedy maximization algorithm, where this complexity is a common result. On the other hand, the firefly algorithm scales proportionally with respect to the number of features, but is quadratic in time with respect to the number of fireflies (potential feature subsets). In practice, it is recommended to increase the number of fireflies as the number of features increases. For our experiments, we chose the number of fireflies to be equal to half the number of features, yielding the results presented in Figure 5. Overall, this dependence on both the features and fireflies makes the algorithm exceedingly complex. These aspects render our method more suitable for high-dimensional feature spaces than any of the baselines discussed thus far. However, the scalability of both LASSO and PI exceeds that of any other baselines, despite their troubles selecting features. Our results also highlight the additional computational load required to integrate temporal dependencies. In any case, the LSTM-based version of FSNID still shows a linear time complexity with respect to to the number of features.

6 Conclusion

Feature selection is a fundamental problem in network intrusion detection. This is due to the computational cost associated to large input spaces and the requirement of minimizing monitoring for both economic and strategic reasons. In this paper, we have introduced FSNID, a new information-theoretic framework designed to select features for detecting network intrusions. Our approach estimates a TE-based 'feature importance' using a neural function approximator. FSNID effectively identifies the minimal number of features even if they are highly correlated, while its computation time scales linearly relative to the number of features. All of these are desirable attributes when applied to highly-dimensional network data. The estimation of the measure central to FSNID is agnostic to the function approximation technique used. By integrating a recurrent layer into the neural architecture, it is possible to incorporate time-dependent information into our feature importance calculation. This enables a further reduction in the number of features selected when compared to our vanilla method. In our experimental evaluation, we have shown that FSNID can consistently identify a lightweight set of network features, without notably compromising classification performance.

Acknowledgments

Charles Westphal's PhD studies are supported by the UK Engineering and Physical Sciences Research Council Grant EP/S022503/1.

References

[1] Mohiuddin Ahmed, Abdun Naser Mahmood, and Jiankun Hu. 2016. A survey of network anomaly detection techniques. *Journal of Network and Computer Applications* 60, 3 (2016), 19–31.

[2] Ahmed Ahmim, Leandros Maglaras, Mohamed Amine Ferrag, Makhlouf Derdour, and Helge Janicke. 2019. A novel hierarchical intrusion detection system based on decision tree and rules-based models. In *19th International Conference on Distributed Computing in Sensor Systems (DCOSS'19)*. IEEE, 228–233.

[3] 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.

[4] R. Battiti. 1994. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on Neural Networks* 5, 4 (1994), 537–550.

[5] Ishmael Belghazi, Sai Rajeswar, Aristide Baratin, R Devon Hjelm, and Aaron Courville. 2018. MINE: Mutual Information Neural Estimation. In *19th International Conference on Machine Learning (ICML'18)*. PMLR, 531–540.

[6] Hamid Bostani and Mansour Sheikhan. 2017. Hybrid of binary gravitational search algorithm and mutual information for feature selection in intrusion detection systems. *Soft Computing* 21 (2017), 2307–2324.

[7] Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1 (2001), 5–32.

[8] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. 2012. Conditional Likelihood Maximisation: A Unifying Framework for Information Theoretic Feature Selection. *Journal of Machine Learning Research* 13, 2 (2012), 27–66.

[9] Amnon Catav, Boyang Fu, Yazeed Zoabi, Ahuva Libi Weiss Meilik, Noam Shomron, Jason Ernst, Sriram Sankararaman, and Ran Gilad-Bachrach. 2021. Marginal Contribution Feature Importance - an Axiomatic Approach for Explaining Data. In *19th International Conference on Machine Learning (ICML'21)*. PMLR, 1324–1335.

[10] Jinfu Chen, Yuhao Chen, Saihua Cai, Shang Yin, Lingling Zhao, and Zikang Zhang. 2023. An optimized feature extraction algorithm for abnormal network traffic detection. *Future Generation Computer Systems* 149, 1 (2023), 330–342.

[11] Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. 2018. Learning to explain: An information-theoretic perspective on model interpretation. In *19th International Conference on Machine Learning (ICML'18)*. PMLR, 883–892.

[12] Te-Shun Chou, Kang K Yen, and Jun Luo. 2008. Network intrusion detection design using feature selection of soft computing paradigms. *International Journal of Computer and Information Engineering* 2, 11 (2008), 3722–3734.

[13] Ian C. Covert, Scott Lundberg, and Su-In Lee. 2020. Understanding Global Feature Contributions with Additive Importance Measures. In *34th Annual Conference on Neural Information Processing Systems (NeurIPS'20)*. 17212–17223.

[14] Ian Connick Covert, Wei Qiu, Mingyu Lu, Na Yoon Kim, Nathan J White, and Su-In Lee. 2023. Learning to Maximize Mutual Information for Dynamic Feature Selection. In *40th International Conference on Machine Learning (ICML'23)*. PMLR, 6424–6447.

[15] Dries Debeer and Carolin Strobl. 2020. Conditional permutation importance revisited. *BMC Bioinformatics* 21, 1 (07 2020), 1–30.

[16] Dorothy Denning. 1987. Algorithmic Enumeration of Ideal Classes for Quaternion Orders. *IEEE Transactions on Software Engineering* 12, 2 (1987), 222–232.

[17] Nicolas Dobigeon, Jean-Yves Tourneret, and Manuel Davy. 2007. Joint segmentation of piecewise constant autoregressive processes by using a hierarchical model and a Bayesian sampling approach. *IEEE Transactions on Signal Processing* 55, 4 (2007), 1251–1263.

[18] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software* 33, 1 (2010), 1.

[19] Christopher Frye, Colin Rowat, and Ilya Feige. 2020. Asymmetric Shapley Values: Incorporating Causal Knowledge into Model-Agnostic Explainability. In *34th Annual Conference on Neural Information Processing Systems (NeurIPS'20)*. 1229–1239.

[20] Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. 2016. Variational Information Maximization for Feature Selection. In *30th Annual Conference on Neural Information Processing Sytems (NeurIPS'16)*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Eds.). 1285–1312.

[21] Baptiste Gregorutti, Bertrand Michel, and Philippe Saint-Pierre. 2017. Correlation and variable importance in random forests. *Statistics and Computing* 27 (2017), 659–678.

[22] Flore Harle, Florent Chatelain, Cedric Gouy-Pailler, and Sophie Achard. 2014. Rank-based multiple change-point detection in multivariate time series. In *2014 22nd European Signal Processing Conference (EUSIPCO'14)*. IEEE, 1337–1341.

[23] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[24] IBM. 2023. Cost of a Data Breach Report 2023. https://www.ibm.com/reports/data-breach

[25] Ja Jabez and B Muthukumar. 2015. Intrusion Detection System (IDS): Anomaly detection using outlier detection approach. *Procedia Computer Science* 48 (2015), 338–346.

[26] Joseph Janssen, Vincent Guan, and Elina Robeva. 2023. Ultra-marginal Feature Importance: Learning from Data with Causal Guarantees. In *26th International Conference on Artificial Intelligence and Statistics (AISTATS'23)*. PMLR, 10782–10814.

[27] Dominik Janzing, Lenon Minorics, and Patrick Bloebaum. 2020. Feature relevance quantification in explainable AI: A causal problem. In *23rd International Conference of Artificial Intelligence and Statistics (AISTATS'20)*. PMLR, 2907–2916.

[28] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. 2016. A deep learning approach for network intrusion detection system. EAI, 21–26.

[29] Yang Jia, Meng Wang, and Yagang Wang. 2019. Network intrusion detection algorithm based on deep neural network. *IET Information Security* 13, 1 (2019), 48–53.

[30] Kaiyuan Jiang, Wenya Wang, Aili Wang, and Haibin Wu. 2020. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE Access* 8 (2020), 32464–32476.

[31] Gozde Karatas, Onder Demir, and Ozgur Koray Sahingoz. 2020. Increasing the Performance of Machine Learning-Based IDSs on an Imbalanced and Up-to-Date Dataset. *IEEE Access* 8 (2020), 32150–32162.

[32] Jasmin Kevric, Samed Jukic, and Abdulhamit Subasi. 2017. An effective combining classifier approach using tree algorithms for network intrusion detection. *Neural Computing and Applications* 28, 1 (2017), 1051–1058.

[33] Chaouki Khammassi and Saoussen Krichen. 2017. A GA-LR wrapper approach for feature selection in network intrusion detection. *Computers & Security* 70 (2017), 255–277.

[34] Chaouki Khammassi and Saoussen Krichen. 2020. A NSGA2-LR wrapper approach for feature selection in network intrusion detection. *Computer Networks* 172 (2020), 107183.

[35] Elizabeth Kumar, Suresh Venkatasubramanian, Carlos Scheidegger, and Sorelle A. Friedler. 2020. Problems with Shapley-Value-Based Explanations as Feature Importance Measures. In *37th International Conference on Machine Learning (ICML'20)*. PMLR, 5491–5500.

[36] Peter A Lachenbruch and Matthew Goldstein. 1979. Discriminant analysis. *Biometrics* 2, 11 (1979), 69–85.

[37] Aleksandar Lazarevic, Levent Ertoz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. 2003. A comparative study of anomaly detection schemes in network intrusion detection. In *3rd IEEE International Conference on Data Mining (ICDM'03)*. IEE, 25–36.

[38] Huawen Liu, Jigui Sun, Lei Liu, and Huijie Zhang. 2009. Feature selection with dynamic mutual information. *Pattern Recognition* 42, 7 (2009), 1330–1339.

[39] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions (NeurIPS'17). In *Neural Information Processing Systems (NeurIPS'17)*. 17212–17223.

[40] Young-Il Moon, Balaji Rajagopalan, and Upmanu Lall. 1995. Estimation of mutual information using kernel density estimators. *Physical Review E* 52, 3 (1995), 2318–2321.

[41] Nour Moustafa. 2021. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society* 72 (2021), 102994.

[42] Nour Moustafa and Jill Slay. 2015. A hybrid feature selection for network intrusion detection systems: Central points. In *Australian Information Warfare Conference (AIWC'15)*, Vol. 16. Security Research Institute (SRI), Brisbane, 5–13.

[43] Sheraz Naseer, Yasir Saleem, Shehzad Khalid, Muhammad Khawar Bashir, Jihun Han, Muhammad Munwar Iqbal, and Kijun Han. 2018. Enhanced network anomaly detection based on deep neural networks. *IEEE Access* 6 (2018), 48231–48246.

[44] Jerzy Neyman and Egon Pearson. 1933. The testing of statistical hypotheses in relation to probabilities a priori. *Mathematical Proceedings of the Cambridge Philosophical Society* 29, 4 (1933), 492–510.

[45] Milan Paluš. 1996. Detecting nonlinearity in multivariate time series. *Physics Letters A* 213, 3 (1996), 138–147.

[46] Liam Paninski. 2003. Estimation of Entropy and Mutual Information. *Neural Computation* 15, 6 (2003), 1191–1253.

[47] Karl Pearson. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2, 11 (1901), 559–572.

[48] Hanchuan Peng, Fuhui Long, and Chris Ding. 2005. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27, 8 (2005), 1226–1238.

[49] Ben Poole, Sherjil Ozair, Aaron Oord, Alexander Alemi, and George Tucker. 2019. On Variational Bounds of Mutual Information. In *37th International Conference of Machine Learning (ICML'19)*. PMLR, 5171–5180.

[50] Benni Purnama, Eko Arip Winanto, Deris Stiawan, Darmawiiovo Hanapi, Mohd Yazid bin Idris, Rahmat Budiarto, et al. 2020. Features extraction on IoT intrusion detection system using principal components analysis (PCA). In *7th International Conference on Electrical Engineering, Computer Sciences and Informatics (EECSI'20)*. IEEE, 114–118.

[51] Thomas Schreiber. 2000. Measuring information transfer. *Physical Review Letters* 85, 2 (2000), 461–464.

[52] B Selvakumar and Karuppiah Muneeswaran. 2019. Firefly algorithm based feature selection for network intrusion detection. *Computers & Security* 81 (2019), 148–155.
[53] Claude E Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27, 3 (1948), 379–423.
[54] Lloyd Shapley. 1953. A Value for n-Person Games. *Contributions to the Theory of Games* 2, 28 (1953), 307–318.
[55] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization.. In *4th International Conference on Information Systems Security and Privacy (ICISSP'18)*. 108–116.
[56] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak, and Ali A Ghorbani. 2019. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In *2019 International Carnahan Conference on Security Technology (ICCST'19)*. IEEE, 1–8.
[57] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. 2018. A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence* 2, 1 (2018), 41–50.
[58] Basant Subba, Santosh Biswas, and Sushanta Karmakar. 2016. A neural network based system for intrusion detection and attack classification. In *National Conference on Communication (NCC'16)*. 1–6.
[59] Pengfei Sun, Pengju Liu, Qi Li, Chenxi Liu, Xiangling Lu, Ruochen Hao, and Jinpeng Chen. 2020. DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system. *Security and Communication Networks* 20 (2020), 1–11.
[60] Mukund Sundararajan and Amir Najmi. 2020. The Many Shapley Values for Model Explanation. In *37th International Conference on Machine Learning (ICML'20)*. PMLR, 9269–9278.
[61] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. 2009. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE, 1–6.
[62] Robert Tibshirani. 1996. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58, 1 (1996), 267–288.
[63] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2019. Representation Learning with Contrastive Predictive Coding. arXiv:1807.03748 [cs.LG]
[64] Abhishek Verma and Virender Ranga. 2019. ELNIDS: Ensemble learning based network intrusion detection system for RPL based Internet of Things. In *4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU'19)*. IEEE.
[65] Paul L. Williams and Randall D. Beer. 2010. Nonnegative Decomposition of Multivariate Information. arXiv:1004.2515 [cs.IT]
[66] Patricia Wollstadt, Sebastian Schmitt, and Michael Wibral. 2023. A rigorous information-theoretic definition of redundancy and relevancy in feature selection based on (partial) information decomposition. *Journal of Machine Learning Research* 24, 131 (2023), 1–44.
[67] Congyuan Xu, Jizhong Shen, Xin Du, and Fan Zhang. 2018. An intrusion detection system using a deep neural network with gated recurrent units. *IEEE Access* 6 (2018), 48697–48707.
[68] Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. 2020. Feature Selection using Stochastic Gates. In *37th International Conference on Machine Learning (ICML'20)*. PMLR, 10648–10659.
[69] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. 2017. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* 5 (2017), 21954–21961.
[70] Yi Zeng, Huaxi Gu, Wenting Wei, and Yantao Guo. 2019. $Deep - Full - Range$: a deep learning based network encrypted traffic classification and intrusion detection framework. *IEEE Access* 7 (2019), 45182–45190.

A Wall Clock Runtimes

In Table 1, we present the wall clock runtimes for each feature set selection method in seconds.

B Handling the Length-k Problem for Baselines

In this section, we explain how we systematically place an upper bound on the number of features selected using each baseline, thus avoiding the length-k problem. This step is unnecessary for the LASSO algorithm since it is inherently able to deal with it. For the FSNID method, the procedure is detailed in Section 4.7 and will not be repeated here.

PI. For the permutation importance, we again utilize a null model. After five experiments, a feature is not selected if its permutation importance does not have a 95% probability of being higher than that of a random feature.

UMFI. We adopt a null model as in the PI case.

CLM. This technique selects features that increase the MI between the target and feature set. Feature addition typically stops when the MI stops rising. To identify this point, we use a null model, adding features as long as the chosen feature increases the MI more than a random feature would.

MIFA. This soft computing technique involves using a metaheuristic to select features that improve MI with the target. Without restricting the number of selected features, the method tends to select all possible features, an undesirable result. To address this, we increase k (the number of possible features) starting from 5 in increments of 5. We then select the smallest feature set for which the next set of features up did not have a statistically significant improvement in MI. This process underscores the challenges of methods that cannot solve the length-k problem, repeating the experiments for multiple values of k exponentially increases temporal complexity.

C Selection of the Hyperparameters

C.1 Feature Selection Experiments

When estimating Φ we used $b = 100$ and $N = 10000$ (as defined in Algorithm 2 in Appendix D) for the vanilla version of FSNID, MIFA, and CLM. Meanwhile, for experiments considering temporal relationships we used, $b = 100$ and $N = 20,000$ and sequence length s (the range over which we incorporate temporal dependencies) was set to 10. The differences are due to the more complex temporal relationships requiring extended training to be fully captured. In the vanilla implementation of our method, the function approximator is a standard feedforward multi-layer perceptron, featuring one hidden layer consisting of 50 nodes. For the LSTM-based approach, the architecture remained the same, but it included an additional LSTM layer where the hidden layer also included 50 nodes. This was repeated for GRU and TCN modules. Furthermore, we use a learning rate of 0.0001 for the calculation of $\Phi_{X_i;\mathcal{X}\to Y}$ and a learning rate of 0.01 for the classification task. The networks used for both tasks in both cases where identical, except for MINE (MI neural estimation), we are undertaking a regression task. Therefore, we use a mean squared error loss and an Adam optimizer. Meanwhile, for the classification task we use a Negative Log Likelihood Loss and a standard stochastic gradient descent optimizer. For both PI and UMFI we use 100 random trees and for PI we ran 10 repeats. For LASSO we employed a regularization strength of $\alpha = 0.001$, in order to select enough features. Finally, for the MIFA algorithm we set the number of fireflies to $0.5 * N$, where N is the max number of features, and allowed 10 iterations of brightness reconfiguration.

C.2 Complexity Analysis Experiments

For these experiments we use the same hyperparameters as for the feature selection experiments, except in this case we use $b = 10$ and $N = 100$ (as defined in Algorithm 1) for both our vanilla and LSTM-based methods. While the number of random trees in use for our baselines is 10. These values were chosen, not due to their feature selecting abilities but due to their speed.

Table 1: Wall clock runtimes for each feature set selection method in seconds.

	ToN-IoT	NSL-KDD	CIC-DDoS2019	UNSW-NB15	CIC-IDS17
UMFI	1167	312	1018	765	8883
MIFA	5402	1933	56389	10170	189101
CLM	3900	551	13713	1888	20001
PI	198	167	304	389	2861
LASSO	4.5	0.42	6.71	19.7	34
FSNID	1723	432	3189	748	13565
FSNIDLSTM	185477	37258	203904	71501	508093
FSNIDTCN	116421	31928	171970	64801	499390
FSNIDGRU	121651	38314	192191	67647	507430

Algorithm 2 Estimation of $\Phi_{X_i;\mathcal{X}\to Y}$.

Input: Training dataset $(i^1, y^1, i^2, y^2 ... i^T, y^T)$, and feature of interest X_i.
Output: $H(Y|\mathcal{X}_{\setminus X_i}) - H(Y|\mathcal{X})$

Initialize weights for θ and $\theta_{\setminus X_i}$
for 1 to N **do**
  Draw mini batch samples of length b from the joint distribution of the target and the network features with all possible variables included $p_{Y,\mathcal{X}} \sim (y^{t_1}, x_1^{t_1}, x_i^{t_1} \dots x_N^{t_1}), \dots, (y^{t_b}, x_1^{t_b}, x_i^{t_b} \dots x_N^{t_b})$, and repeat for the marginal distribution $p_Y \otimes p_{\mathcal{X}} \sim (y^{t'_1}, x_1^{t_1}, x_i^{t_1} \dots x_N^{t_1}), \dots, (y^{t'_b}, x_1^{t_b}, x_i^{t_b} \dots x_N^{t_b})$, where $t'_i \neq t_i$.
  Draw mini batch samples of length b from the joint distribution of the target and the network features with variable X_i missing. $p_{Y,\mathcal{X}_{\setminus X_i}} \sim (y^{t_1}, x_1^{t_1} \dots x_N^{t_1}), \dots, (y^{t_b}, x_1^{t_b} \dots x_N^{t_b})$, and repeat for the marginal distribution $p_Y \otimes p_{\mathcal{X}_{\setminus X_i}} \sim (y^{t'_1}, x_1^{t_1}, \dots x_N^{t_1}), \dots, (y^{t'_b}, x_1^{t_b} \dots x_N^{t_b})$, where $t'_i \neq t_i$.
  $I(Y;\mathcal{X}) \geq \frac{1}{b}\sum_{j=1}^{b} F_\theta(y^{t_j}, x_1^{t_j}, x_i^{t_j} \dots x_N^{t_j})) - \frac{1}{b}\sum_{j=1}^{b} \log e^{F_\theta((y_1^{t'_j}, x_1^{t_j}, x_i^{t_j} \dots x_N^{t_j})}$
  $I(Y;\mathcal{X}_{\setminus X_i}) \geq \frac{1}{b}\sum_{j=1}^{b} F_{\theta_{\setminus X_i}}((y^{t_j}, x_1^{t_j} \dots x_N^{t_j})) - \frac{1}{b}\sum_{j=1}^{b} \log e^{F_{\theta_{\setminus X_i}}(y_1^{t'_j}, x_1^{t_j} \dots x_N^{t_j})}$
  $\theta \leftarrow \tilde{\nabla}_\theta I(Y;\mathcal{X})$
  $\theta_{\setminus X_i} \leftarrow \tilde{\nabla}_{\theta_{\setminus X_i}} I(Y;\mathcal{X}_{\setminus X_i})$
end for
return $I(Y;\mathcal{X}) - I(Y;\mathcal{X}_{\setminus X_i})$

D Estimation of Φ

Φ is estimated by means of Algorithm 2.

E Hardware and Computational Resources

The classification and feature selection experiments presented in Section 5.3.2 were performed on a P100 Nvidia GPU cluster with 68GB of RAM. Meanwhile, the complexity analysis experiments in Section 5.5 were completed using an Apple Macbook Pro with a 2021 M1 processor, and 16GB of RAM.