The Intrinsic Dimensionality of Network Datasets and Its Applications

Matt Gorbett[a] Caspian Siebert[a] Hossein Shirazi[a] and Indrakshi Ray[a]
[a] *Colorado State University, Fort Collins, Colorado*

Abstract. Modern network infrastructures are in a constant state of transformation, in large part due to the exponential growth of Internet of Things (IoT) devices. The unique properties of IoT-connected networks, such as heterogeneity and non-standardized protocol, have created critical security holes and network mismanagement. In this paper we propose a new measurement tool, Intrinsic Dimensionality (ID), to aid in analyzing and classifying network traffic. A proxy for dataset complexity, ID can be used to understand the network as a whole, aiding in tasks such as network management and provisioning. We use ID to evaluate several modern network datasets empirically. Showing that, for network and device-level data, generated using IoT methodologies, the ID of the data fits into a low dimensional representation. Additionally we explore network data complexity at the sample level using Local Intrinsic Dimensionality (LID) and propose a novel unsupervised intrusion detection technique, the Weighted Hamming LID Estimator. We show that the algortihm performs better on IoT network datasets than the Autoencoder, KNN, and Isolation Forests. Finally, we propose the use of synthetic data as an additional tool for both network data measurement as well as intrusion detection. Synthetically generated data can aid in building a more robust network dataset, while also helping in downstream tasks such as machine learning based intrusion detection models. We explore the effects of synthetic data on ID measurements, as well as its role in intrusion detection systems.

Keywords. Intrusion Detection, IoT, Internet of Things, Intrinsic Dimensionality, Data Complexity, Anomaly detection

1. Introduction

With people, objects, sensors, and services all connected through devices ranging from household appliances to smartphones and PCs, the Internet of Things (IoT) network infrastructure faces the challenging task of managing heterogeneous devices and their communications in the absence of standardization. The proliferation of IoT systems has introduced new, and emerging security vulnerabilities [5, 53, 2, 13] which can be readily exploited to cause harm. Such vulnerabilities arise because of device manufacturers neglecting security for performance considerations [14], end-users not updating each device regularly [50], and a continually expanding marketplace of devices and manufacturers [13].

Further, the unique characteristics of IoT networks have introduced new complications. Notably, heterogeneity and non-standardized protocol of IoT networks have been posited as critical challenges for enhancing the security of IoT systems [40, 25, 11] –

networks with diverse devices ranging from single-purpose machines to robust servers, each with varied communication structures, are cumbersome to protect. Past work has proposed behavioral fingerprinting of devices [9], and further fine-tuning device-specific anomaly detection models depending on the complexity of devices [20]. Others propose supervised machine learning solutions [44, 28], utilizing modern network datasets such as Aposemat IoT-23 (IoT-23) [16].

In this work, we take a different approach by first examining the supposition that benign heterogeneous IoT networks have higher complexity than regular non-IoT datasets. We calculate ID, a property that has been proposed to measure the complexity of a data set as a whole [54] and evaluate it on four IoT datasets and two non-IoT datasets. We analyze the datasets from three perspectives: network level, device level, and through sampling, and show that, despite the variability of IoT devices, the complexity of benign network activity is low. Further, we show that at the sample-level, IoT traffic can be categorized as benign or malicious using a device-independent unsupervised model. We do this using LID, which estimates the intrinsic dimension around an individual data point, and show that malicious activity exhibits higher LID values than benign samples.

1.1. Problem Statement

We focus on the question of heterogeneity and complexity in IoT networks and their effects on detecting malicious activity via Network Intrusion Detection Systems (NIDS). Specifically, we ask the following questions:

1. Do the properties of multi-device heterogeneous IoT networks exhibit fundamentally more complex behavior?
2. What devices are harder to protect in machine learning-based NIDS frameworks?
3. Can we detect malicious activity at the IoT network level in an unsupervised manner, without the need to label each device and attack?
4. Can the use of synthetic datasets aid NIDS in building more robust ML-based detection models?

1.2. Limitations of Prior Attempts

Quantifying the complexity of IoT-based network traffic can aid in tasks such as network administration, provisioning, quality of service metrics, and security. However, current research has found that existing network complexity metrics are insufficient [30]. Notably, Liu et al. [30] examined several classical complexity metrics such as Normalized Trace Complexity, Multiscale Sample Entropy, and Plug-in Packet Timing Entropy, finding disagreement between the rank order results of each algorithm. In summary, they conclude that new quantitative measures are needed to incorporate the diversity of IoT traffic.

Additionally, various machine learning solutions have been proposed for NIDS in IoT networks. Such solutions incorporate deep learning such as autoencoders and classifiers [41, 47, 10, 48], as well as traditional machine learning algorithms [35]. While these solutions are each impactful in their own right, most are supervised learning solutions requiring a fully annotated dataset to train – a costly, time-intensive task. Moreover, with constantly evolving attack vectors (malicious actors acting in novel ways), supervised NIDS solutions trained on datasets with particular attack types become vulnerable

[51, 36]. Further, IoT networks are fundamentally different from standard networks – devices will be added to the network, and existing devices may have software/firmware updates with greater frequency. Supervised algorithms are unable to detect new devices or updates in these situations without expensive retraining or reconfiguration [9].

To mitigate the problems of supervised learning algorithms in IoT, Haefner and Ray [20] take the novel approach to intrusion detection in IoT from the perspective of device traffic complexity. The authors measure the complexity of network traffic on a per device basis to tune an (unsupervised) Isolation Forest algorithm. They find that several single-purpose IoT devices contain simple (non-malicious) network traffic, enabling us to assume trust of the device based on its low network packet variability. However, the tuning procedure used for the Isolation Forest assumes a particular contamination rate: for more complex devices, they assume a more significant and fixed percentage of network packets are anomalies, an assumption that could lead to false positives and not perform well in real-world scenarios.

1.3. Proposed Approach

In this work, we measure the complexity of IoT network traffic using the novel perspective of ID, testing the hypothesis that IoT networks have increased complexity as a result of their heterogeneous behavior. We first measure ID at the network dataset level, showing that, counter to intuition, several IoT datasets exhibit lower ID compared to non-IoT benchmarks. We expand this analysis to the IoT device level, confirming the work by Haefner and Ray [20] that single-purpose devices have low complexity measurements. Additionally, we show that the ID measurement used in our experiments exhibit similar *rank order* complexity as [20], *i.e.*, the complexity measurements of devices are arranged in a similar order in both works. Finally, we find that more complex devices still exhibit low ID, which can help in modeling and risk management of new devices in NIDS (lower ID indicates we can feasibly model these complex devices better than previously thought).

Second, we focus on the problem of detecting malicious actors in IoT networks. We measure the complexity of network packets using LID by formulating an entropy-weighted Hamming distance calculation on top of the LID measurement to construct a novel anomaly detection algorithm. The results of the algorithm show that benign network data in IoT datasets exhibit a lower LID measurement compared to malicious actors, which provides us the opportunity to threshold this measurement during test time. The unsupervised algorithm uses benign IoT network data as a training set and assumes any test sample under threshold τ is benign network behavior. If the LID estimate is above this threshold, we can flag the example as malicious.

Finally, we assess our proposed ID and LID approaches with synthesized network data. Specifically, we generate new synthesized benign and malicious datasets using a Variational Autoencoder (VAE). We hypothesize that by adding new data to our experimentation we can better assess the quality of the metrics. To begin, we assess the quality of our synthetically generated data by measuring its separability from real data. We achieve this with both the variational autoencoders reconstruction error of a sample and additionally using the receiver operating characteristic curve of the synthetic dataset compared to the real dataset. We then measure the ID of synthesized benign datasets to test whether the metric is consistent with the real samples. Next, we test the ability

of our Weighted Hamming LID Estimator against synthesized attack data. In our final experiment, we test whether the performance of unsupervised anomaly detection algorithms increases as a result of the newly added synthetic data. Our experiments show that we can build a robust dataset using the synthesized data while maintaining a similar ID measurement.

1.4. Key Contributions

Our contributions are as follows:

- We measure the complexity of several IoT and non-IoT datasets at both the network and device level using ID, testing the hypothesis that IoT networks contain complex interactions. Despite being heterogeneous in nature, we show that IoT network activity has low ID measurements, with ID values similar to device-level traffic. *Low ID measurements provide strong evidence that we can build robust and secure Machine Learning (ML) models to protect IoT networks.* To the best of our knowledge, this is the first work that analyzes IoT datasets using ID.
- We propose a novel algorithm for unsupervised anomaly detection using a combination of Hamming distance and the Hill Maximum Likelihood Estimator (MLE) LID, and show that the algorithm performs competitively with several state-of-the-art algorithms.
- We measure the complexity of network traffic using synthesized datasets, showing that ID metrics are similar across real and synthetic values. Additionally, we show the benefits of using synthesized data for enhancing unsupervised learning algorithms.
- We find that adding synthetic data to the benign and attack datasets provides new insights into the quality of different metrics and algorithms. In particular, adding synthetic data to the benign datasets provides enhanced generalization capabilities in IoT datasets. Additionally, we find that the anomaly detection algorithms studied in this paper are robust to synthetically generated attacks.

The rest of the paper is organized as follows. We first summarize related and prior work in Section 2. We then detail ID, and LID concepts in Section 3 as well as discuss our methodology. We summarize the datasets we used in experiments in Section 4. Next is our analysis of ID results in Section 5, followed by device level ID estimates in Section 6. In Section 7, we present results at the algorithm level and attack level. In Section 8 we examine the intrinsic dimensionality of synthetic datasets. Finally, we conclude this paper with a discussion and pointers to future directions in Section 9.

2. Related Work

In this section, we review Intrusion Detection Systems (IDS), IoT security research, current algorithms for IoT intrusion detection, and finally, open problems in deep learning anomaly detection.

2.1. Intrusion Detection Systems

IDS can broadly be classified based on 1) where the detection is placed (network or host) and 2) the detection method that is employed (ML anomaly-detection algorithms or traditional signature-based detection where attack patterns are defined in a database). In this work, we concentrate on network-based intrusion detection where anomalies are classified based on an ML algorithm. ML based NIDS are employed in production into a key point within a network to monitor traffic to and from all devices connected to the network. Network features are extracted from the entire subnet about the passing traffic and scored by the ML detection algorithm. NIDS can operate on-line (real-time detection) or off-line (batch detection). Real-time detection offers more robust security and beneficial results as long as it does not impair the overall speed of the network. Our study aims to characterize network data in real-time.

IoT is a rapidly evolving field, with research being done at dozens of institutions across industry and academia [11, 25, 40, 27]. It is postulated that IoT increases the vulnerability of networks because the *attack surface* has increased, with many new entry and exit points with new devices available on networks [42, 33]. A heterogeneous IoT network is typically made up of various sub-devices within a distributed network. It includes resource-constrained devices, such as a smart light bulb or garage door opener, and more powerful devices such as embedded and regular computers.

Existing research notes that IoT networks and devices have multiple intrusion sources: IoT backends, cloud services supporting an IoT device, and other hubs within the IoT system [1, 38], which makes it difficult to implement traditional intrusion detection approaches such as rule-based and signature-based methods.

2.2. State-of-the-Art Network Intrusion Models

Several works propose new and existing algorithms for intrusion detection on commonly used datasets. Moustafa [35] released dataset *TON_IoT* (*TON_IoT*), including baseline results that use many supervised learning algorithms. Sahu et al. [44] proposed a hybrid deep learning model which uses a CNN/LSTM framework to achieve 96% accuracy on the IoT-23 dataset generated by [16] and outperformed several proposed deep learning-based attack detection. Kozik et al. [28] use hybrid time window embeddings with a transformer neural network to classify IoT-23 data. This model achieves between 93% and 95% accuracy on attacks in IoT-23 and does better than three other proposed deep learning models: HaddadPajouh et al. [19] use an LSTM trained on IoT devices execution operation codes (OpCodes), Roy and Cheung [43] use a bi-directional LSTM for detecting attacks on UNSW-NB15, and Azmoodeh et al. [7] use OpCodes to train a deep Eigenspace model to detect attacks.

Moustafa and Slay released UNSW-NB15 dataset [36] that was generated using IXIA PerfectStorm. The dataset has been widely used as a benchmark for comparison. MStream [10] is an online neural network-based anomaly detection algorithm using both continuous and categorical features. This tool achieves 0.90 AUROC on UNSW-NB15 and is considered state-of-the-art according to PapersWithCode.com[1]. The Edge-detect model [48] is another neural network-based framework that proposes a lightweight model to detect anomalies on edge and was tested on UNSW-15 and is also considered

[1] https://paperswithcode.com/dataset/unsw-nb15

state-of-the-art. Meftah et al. [32] performed a similar approach to [23], using Recursive Feature Elimination and Random Forests to select features, achieving up to 86% F1 accuracy. It should be noted that UNSW-NB15 is only used in this paper to measure dataset complexity, not anomaly detection. We specifically concentrate our anomaly detection approach on IoT datasets, especially since the high complexity of UNSW-NB15 makes it a poor choice for our algorithm.

Several other papers are published using alternative datasets that propose different machine learning models for intrusion detection. Rezvy et al. [41] proposed a deep learning framework for intrusion classification and prediction in 5G and IoT networks. They propose an autoencoder neural network for detecting intrusion or attacks in 5G and IoT networks, evaluating the model on the Aegean Wi-Fi Intrusion dataset. Their results showed an overall detection accuracy of 99.9% for different types of attacks. Kasongo and Sun [23] argue that feature selection is essential for the performance of ML models in intrusion detection since model accuracy decreases with more high-dimensional datasets. They apply a filtering technique on features and train several ML models using this technique, showing strong performance. They relate the feature selection to IoT devices with limited capacity, showing that less robust modeling techniques are favorable in limited-capacity systems such as small IoT devices.

2.3. Connections to Deep Learning

Neural networks trained with back-propagation provide diverse structures and objectives to learn from high-dimensional data. Despite their incredible power, anomaly detection remains an open research problem, even in state-of-the-art models. Notably, several works in computer vision have shown that classification, generative, and unsupervised deep neural networks are all susceptible to anomalous data [21, 17, 37, 46, 34]. For example, one common computer vision experiment involves training a deep learning model on the CIFAR-10, a dataset with 60,000 images labeled into ten classes. It is expected that the likelihood of a CIFAR-10 test image will be higher than images from other datasets during test time. However, several papers have shown that the examples from the dataset SVHN produces a higher likelihood when passed through the model trained on CIFAR-10 [37, 46, 34]. Recently, Serra et al. showed that anomalous high-likelihood data could be linked to complexity [46]. They find that the simplicity of SVHN data compared to CIFAR-10 data causes the deep learning model to exercise a higher likelihood on SVHN examples than the complex CIFAR-10 data. They use image compression scores as a complexity metric (likelihood ratio) to determine whether the high likelihood can be attributed to lower complexity.

Interestingly, a similar complexity finding was found in a recent security paper published by Haefner and Ray [20]. Using data from various IoT devices, they find that each device has varying complexity. They formalize a complexity measure (IP Spread/IP Depth) per device in order to fine-tune an Isolation Forest anomaly detection algorithm. Their architecture, ComplexIoT, measures network traffic on a device level, which can be used in Host Intrusion Detection Systems.

This work is similar to ComplexIoT [20] in that we propose a complexity measurement; however, there are several key differences:

- We analyze IoT datasets both from the point of view of network-level and the device level, while ComplexIoT only looks at device level complexity.

- ComplexIoT proposes a device complexity score to moderate the contamination rate of an Isolation Forest. This is problematic as it assumes $x\%$ of a device's traffic will be malicious given a complexity score and may lead to false positives.
- The ComplexIoT complexity score is based on IP spread and IP depth and does not consider other network features to compute its complexity estimate.
- The efficacy of the ComplexIoT approach has not been measured via binary classification metrics on benign and malicious examples. In this paper, we measure the results of the weighted Hamming LID estimator on common IoT network intrusion datasets.

3. Methodology

In this section we first briefly explain the concepts and the mathematics behind ID and LID. We will later use ID to measure both network and device level IoT and non-IoT datasets, showing how this complexity measurement is a strong tool of our ability to assess network data at multiple levels. Next, we propose our algorithm, the Weighted Hamming Distance LID Estimator, include the algorithm details, baseline models to compare against, and our experimental protocol.

3.1. Intrinsic Dimensionality

The ID of a dataset is the minimum number of variables needed to retain a full approximation of the data [8]. It is based on the observation that high-dimensional data can often be described by a smaller number of variables. The utility of lower dimensional representations is apparent throughout ML research, from data compression (such as autoencoders [22]) to dimensionality reduction (PCA). ID is akin to autoencoders and PCA, however quite distinct in that its an estimate of the lowest possible dimension of a dataset (e.g. the lowest possible bottleneck size in an autoencoder), and not a reduction technique in itself. ID can be thought of as a geometric property to measure complexity of a dataset as a whole [54].

Formally, the ID of dataset $\mathscr{X} \in \mathbb{R}^{mxn}$, with m samples and n features, lies on a lower dimensional manifold $\mathscr{M}$, where $ID = dim(\mathscr{M})$, i.e. ID is the dimension of the manifold $\mathscr{M}$ of the data. Usually, the ID measurement is significantly less than extrinsic dimension n, which corresponds to the number of features.

As an intuitive example, points $x_1 ... x_m$ exist on a piece of paper in three dimensional space. We can describe the points relative to the three dimensional space, (d_1, d_2, d_3), or we can describe them relative to their position on the piece of paper, where only two variables are needed. Here, the representation of points $x_1 ... x_m$ in 3D space is the *extrinsic* dimension, whereas their points relative to the piece of paper are their ID.

The main approach to estimate ID involves examining the neighborhood around a reference point x_i for each x in $\mathscr{X}$. A common equation used in existing research was proposed by Levina and Bickel [29]:

$$ID(\mathscr{X}) = \left(\frac{1}{m(k-1)} \sum_{i=1}^{m} \sum_{j=1}^{k-1} log \frac{T_k(x_i)}{T_j(x_i)} \right)^{-1} \tag{1}$$

where m is the number of samples, x_i is a sample in the dataset, k and j are the k^{th} and j^{th} nearest neighbors. $T_k(x_i)$ is the distance between x_i and x_k, similarly, $T_j(x_i)$ is the distance between x_i and x_j. Intuitively, Equation 1 measures the rate that new neighbors are encountered as we move out from the reference point x_i. We use this equation for all ID estimates in Figure 2.

Recently, ID has been gaining relevance in the machine learning community [39, 6, 4]. Pope et al. [39] showed that common computer vision datasets exhibit very low intrinsic dimension relative to their number of pixels. They also showed that the intrinsic dimension greatly impacts learning: the higher the intrinsic dimension of a dataset, the harder it is to learn from it. In addition, they showed that the extrinsic dimension of the dataset, *i.e.*the total number of pixels per image in a dataset, did not effect learning and generalization, indicating that sample complexity only depends on the intrinsic dimension rather than the total dimension of the dataset. Ansuini et al. [6] showed that neural networks exhibit low intrinsic dimensionality at deep layers of neural networks. Outside of deep learning, intrinsic dimensionality has been used in applications such as anomaly detection[49], clustering, similarity search, and deformation in complex materials.

3.2. Local Intrinsic Dimensionality

In contrast to ID, LID estimates individual data samples, rather than the full dataset. It is based on the observation that individual data points in a dataset often fit within a specific lower-dimensional structure when only considering a subset of the nearby data. As a result, these values can vary greatly within a dataset. Intuitively, the LID measurement can be interpreted as the dimension immediately surrounding a data point.

LID has been proposed for anomaly/out-of-distribution detection [52] as well as detection of adversarial examples in deep neural nets [31]. Theoretically, examples within a dataset should have lower LID values than anomalous examples generated from an alternative source.

Amsaleg et al. [4] propose several estimators for the LID, though they note that these are theoretical quantities and only estimates of the true local dimension. We use their Maximum Likelihood Estimator in Section 3.3. Their equation provides a strong balance between efficiency and complexity:

$$\widehat{LID}(x) = -\left(\frac{1}{k}\sum_{i=1}^{k} log\frac{r_i(x)}{r_k(x)}\right)^{-1} \tag{2}$$

where r_i is the distance of data point x to the i^{th} closest neighbor and r_k is the distance to the k^{th} neighbor. Additionally, it has been shown that hyperparameter k is sensitive and must be experimentally tuned. Equation 2 is a theoretical quantity, and it should be noted that $\widehat{LID}(x)$ is an *estimation*. Further, the dimension estimate is usually not an integer value, except in idealized distributions and datasets.

Ma et al. [31] used LID estimates to characterize adversarial subspaces in deep learning. They showed how traditional density measures can fail to detect adversarial examples in the final layers of deep learning models, while LID measurements can better characterize these subspaces. This is because traditional measures only measure the density

of neighboring points surrounding an example, whereas LID measures the rate at which new neighbors occur.

3.3. Weighted Hamming Distance LID Estimator

LID is typically measured on a sample using distances from its neighbors in a dataset, and can be thought of as the *rate of growth* between a point and its neighbors. In this work, we use Equation 2 for LID estimation, using the training data $\mathscr{X}_{train}$ as neighboring points.

Distance Metric. For the distance metric required in Equation 2, we use Hamming Distance to compute similarity between both categorical and continuous feature points. While Euclidean Distance is typically used in Equation 2 to measure LID, Ma et al. [31] suggested not using Euclidean Distance as the underlying distance metric. Choosing the Hamming Distance metric over Euclidean Distance for continuous variables showed better experimental results. Effectively, this turns each pairwise feature distance into a binary metric: 0 for same, 1 for different. We compute Hamming Distance using the Python SciPy library as:

$$\mathscr{H}(x_i, x_j) = \frac{\text{Number of mismatching features}}{\text{Total Features}} \tag{3}$$

Entropy. We calculate entropy of each feature and set it as the weight. In a dataset with n features, we set weight w_i for feature i to $n/Entropy(i)$, where the entropy of a feature i is:

$$-\sum_{j=1}^{m} p_j log_2(p_j) \tag{4}$$

and n is the total number of features and j are specific classes in the feature. p_j is calculated by getting the counts of each class within feature i. Equation 3.3 denotes the explicit form of entropy [18]. For example, the protocol feature may have TCP and UDP classes, we compute the counts for each to calculate entropy. We find that features with low entropy should be weighted more since they are stable properties of benign samples. For example, if benign samples come from TCP protocol 99% of the time, we can theorize that new samples matching the TCP protocol may be similar to a benign example.

After computing Hamming distance between x_i and the set of $\mathscr{X}_{train}$, we have a set of $\mathscr{H}_1 ... \mathscr{H}_m$ Hamming distances the size of $\mathscr{X}_{train}$. We filter all duplicate distances where $\mathscr{H}_i = \mathscr{H}_j$, to include only a single distance value for each cluster of distances, leaving a set of $\mathscr{H}_1 ... \mathscr{H}_n$ distances where $n \leq m$. In other words, when examples have the same distance $\mathscr{H}$ to a reference point x_i, we filter them into a single distance in order to gather a unique set of distances. From here, we are able to compute the final LID score using Equation 2. Algorithm 1 details the Weighted Hamming LID estimator for anomaly detection.

Figure 1 depicts a visual example of how the Weighted Hamming LID Estimator can classify an anomalous example where the KNN algorithm will fail. For example,

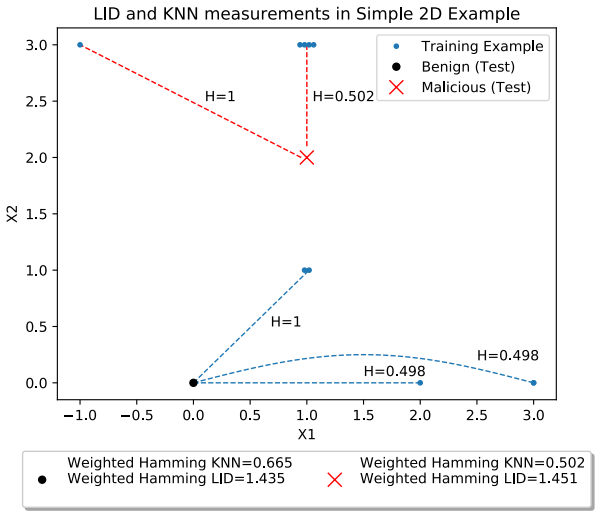

Figure 1. A visual explanation of how the Weighted Hamming Distance LID estimator can detect anomalous examples where the traditional KNN algorithm will fail. See the end of Section 3.3 for more details.

Algorithm 1 Weighted Hamming Distance LID Estimator

Require: $\mathscr{X}_{train}, \mathscr{X}_{test}$, nearest neighbors k, threshold τ
Require: $\mathscr{X}_{train}$ contains only benign examples
  for x_i in $\mathscr{X}_{test}$ **do**
    $\{\mathscr{H}_1 ... \mathscr{H}_m\} \leftarrow \mathscr{H}(x_i, \{\mathscr{X}_{train}\})$
    $\{\mathscr{H}_1 ... \mathscr{H}_n\} \leftarrow$for all distinct $\mathscr{H}_i \in \{\mathscr{H}_1 ... \mathscr{H}_m\}$
    if 0 is in $\{\mathscr{H}_1 ... \mathscr{H}_n\}$ **then**
      x_i is benign (exact match)
    else
      $LID(x_i) \leftarrow LID(\{\mathscr{H}_1 ... \mathscr{H}_m\}, k)(Eq.2)$
      if $LID(x_i) \leq \tau$ **then**
        x_i is benign
      else
        x_i is malicious
      end if
    end if
  end for

the traditional KNN algorithm can discriminate test examples as benign or malicious by measuring a samples average distance to K training samples, where lower average distances indicate a benign example and higher distances indicate a malicious example. Figure 1 shows how KNN can fail on a simple two dimensional problem: the red X is closer to several training examples, however, the black dot matches the more relevant feature, X2. We show how the LID estimate corrects this issue, yielding a lower value for the benign example compared to the malicious example. We weight the Hamming distances in the KNN and LID estimates with 2/Entropy of each feature. Entropy for X1 is 1.99 and 2.01 for X2. In the image, results are presented with a weighted Hamming

distance, however Euclidean distance yields the same results on KNN. K=3. Here, the LID is calculated as -1/$ln(0.502)$ and -1/$ln(0.498)$.

3.4. Baseline models

In the previous section, we explain our proposed Weighted Hamming Distance LID Estimator model. We compare our proposed model with several models as baselines in the tasks of detecting attacks in the IoT networks. These models include several modern and classic algorithms: KNN, Isolation Forest, and Autoencoder. In the following, we briefly explain each of these models.

3.4.1. KNN Algorithm

K-Nearest Neighbors (KNN) classification is an unsupervised machine learning model that measures the distance between a sample point and its neighbors. It takes an arbitrary distance measurement and measures the average distance between a reference point and its neighbors using this distance. In our experiments, we take the average of these nearest neighbor distances and use them to threshold scores. Theoretically, reference points with lower KNN averages should belong to the normal, non-malicious examples in the dataset, and malicious examples should have higher KNN averages.

3.4.2. Isolation Forest

Isolation Forest is an efficient algorithm to determine anomalies in an unsupervised manner. It does not need a profile of what is normal and not normal and identifies anomalies independent of labels. The algorithm relies on the tendency of anomalies to be easier to separate from the rest of the sample compared to normal points. It recursively generates partitions on the sample by randomly selecting an attribute and then randomly selecting a split value for the attribute between the minimum and maximum values allowed for that attribute.

Notably, an Isolation Forest was used in [20], where they tune the contamination parameter based on the complexity of an IoT device. They argue that devices with low complexity should have contamination values close to zero because their expected network traffic should fit certain patterns, hence the device should never receive anomalous traffic. This means that, should the device be compromised, the algorithm would likely not classify the attack as anomalous because of the low expected contamination. As a result, the algorithms assumption that a certain percentage of examples are contaminated makes it vulnerable to changes in the number of contaminated records.

3.4.3. Autoencoder

Autoencoder is a neural network based model commonly used for unsupervised anomaly detection, such as in [41]. The model compresses the training set into a bottleneck representation before reconstructing it. We train the autoencoder on benign examples, and in theory, the reconstruction loss should be smaller for all benign examples compared to malicious examples.

The objective function is the reconstruction loss: given example x and continuous feature x_c we use mean-squared error:

$$\mathscr{L}(x_c, x'_c) = ||x_c - x'_c||^2 \tag{5}$$

where x' is the reconstructed output of the neural network. For discrete categorical features, we encode the categories into embedding layers to input them into the model. The outputs of the autoencoder for categorical variables are one-hot vectors, denoted x_d, and we use cross entropy for the objective function:.

$$\mathscr{L}(x_d, x'_d) = x_d log(x'_d) + (1 - x_d) log(1 - x'_d) \tag{6}$$

We sum the loss of the continuous and categorical variables to obtain the full reconstruction loss. We use Adam optimization with the default learning rate.

3.5. Experimental Setup

To train each model in an unsupervised manner, we first take all clean examples ($\mathscr{X}_{benign}$). For each experiment, we run the algorithm on $\mathscr{X}_{benign}$ using leave-one-out cross validation, *i.e.*, calculate distances $\mathscr{H}$ from x_i on each member of the training set but x_i. We also pass it the entire set of malicious data, $\mathscr{X}_{malicious}$. The result of each sample is either 0 distance, exact match, or a weighted Hamming LID estimate. Zero's are automatically classified as benign, while lower LID estimates are also classified as benign.

A proper threshold τ can be determined based on the desired accuracy rate. In other words, if it is important to classify all malicious samples, we can set a lower τ for higher detection, though this may lead to some benign examples being classified as malicious (false positive).

To measure results, we use Area Under the Receiver Operating Characteristic curve (AUROC) and Area Under the Precision-Recall curve (AUPR). AUROC plots the True Positive Rate (TPR) against False Positive Rate (FPR). AUPR plots the precision versus the recall calculated by following formulas:

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{7}$$

$$recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{8}$$

4. Datasets

This section summarizes the datasets we use in our experiments. We use two common non-IoT network intrusion datasets, UNSW-NB15 and KDD Cup, and four IoT related datasets (*TON_IoT*, NetFlow Bot-IoT (NF Bot-IoT), IoT-23, IoT Sense).

4.1. Non-IoT: UNSW-NB15

UNSW-NB15 [36] (2015) is a standard and commonly used network intrusion dataset from the USNW at Canberra Cyber Range lab. The dataset provides modern network traffic scenarios compared to the KDD datasets, which are more than a decade old. There are 47 features (of which we use 42), ranging from basic features to content and time-related features. Nine types of attacks are included in the dataset.

4.2. Non-IoT: KDD Cup 1999

We use a variation of the KDD Cup 1999 dataset [15] located on Kaggle. The dataset consists of 13,449 benign instances and 41 features, which we use to measure ID.

4.3. TON IoT

TON_IoT [35, 3], published in 2020, comprises heterogeneous IoT data across several devices. The work uses several data source types, including sensor, raw, and log data. Additionally, it includes several infrastructure layers in the testbed architecture, such as the edge, fog, and cloud layers with nine types of generated attacks: Distributed Denial-of-Service (DDoS), Scanning, Ransomware, Backdoor, and Injection attacks. The dataset has 41 total features; however, the authors recommend not to use source IP/port and destination IP/port. The dataset simulates traffic from seven IoT sensors: weather, smart garage door, smart fridge, smart TCP/IP Modbus, GPS tracker, motion-enabled light, and a smart thermostat.

For measuring ID, we deduplicated data instances, and as a result, 61.8% of instances have been removed. However, for the anomaly detection task in Section 7 and Table 3, we consider packets that are duplicate of benign data as being benign.

4.4. NF Bot-IoT

NF Bot-IoT [45], published in 2020, is a dataset based on the BotNet IoT dataset [27, 26]. Botnets are an important attack vector to protect against as they have been the source of several breaches over the past few years [26]. NF Bot-IoT converts four common network NIDS datasets into network flow datasets using the commonly deployed NetFlow [12] protocol for network traffic collection. Authors argue NetFlow's features are easier to extract compared to the complex features used in the original NIDS datasets since NetFlow's features are usually extracted from packet headers. The dataset includes several attacks, including DDoS, Denial-of-Service (DoS), OS and Service Scan, Keylogging, and Data exfiltration attacks.

4.5. IoT-Sense

IoT Sense [9], published in 2018, is a dataset of benign examples generated based on 14 real IoT devices. Authors activated different functionalities of each device using controller apps and captured packets. There are 21 features captured in the dataset, with labeled devices for each sample. We categorize devices in this dataset into four categories (Light, Appliance, Hub/Outlet, Smart Controller).

Table 1. Dataset summaries including total number of samples, percentage of benign samples, percentage of malicious samples, percent duplicates, number of features, and number of attacks.

Name	#To.	%Ben.	%Mal.	%Dup.	#Fea.	#Att
UNSW-NB15	82K	45%	55%	12.5%	42	-
KDD Cup	24K	45.8%	54.2%	0%	41	-
TON_IoT	461K	65%	35%	62%	38	9
IoT-23	1M	50%	50%	2%	19	7
IoT Sense	54K	100%	0	63%	21	-
NF Bot-IoT	599K	21.7%	78.3	0%	12	4

Devices include TCP Light, Avox Light, Musiac Music Player, DLink Camera, iDeviceSocket, iView Light, LutronHub, Netamo Climate, Omna camera, Phillips HUE, Tplink Light, Wemo Outlet, Wink Hub, and Smart Things Hub. We use this dataset for both ID measurements as well as device-specific ID measurements in Sections 5 and 6.

4.6. IoT-23

IoT-23 [16] was released in 2020. The dataset has 23 different scenarios, of which three are benign traffic scenarios captured on real IoT devices. The dataset contains almost 11 million total records; however, with the difficulty of modeling this much data, we sample a million records with the following logic: From the entire dataset, we sample 500K malicious records and 500K benign examples from simulated files that contain a source IP or destination IP with an internal IP address and have at least 50 samples belonging to that specific IP address. We find that 99.99% of internal IPs have at least 50 samples. We also included all samples from three real devices (Philips HUE smart LED lamp, Amazon Echo, and a Somfy smart door lock) with 1,634 total packets.

We categorize these devices similar to IoT-Sense as a light (Philips HUE smart LED lamp), Smart Controller (Amazon Echo), and Appliance (Somfy smart door lock). Philips HUE light is in both datasets so that it can be used for comparison device-specific ID measurements.

This dataset also captures 20 simulated scenarios of both benign and malicious traffic. It offers several attack examples: DDoS, FileDownload (to infected device), HeartBeat (indicates packets sent on the connection are used to keep track of infected host by CC server), Mirai, Torii, and Okiru BotNets (new common attacks), and HorizontalPortScan (used to gather information for further attacks).

4.7. IoT Dataset Features

We use the available features for each dataset, except we exclude source and destination IP and port as well as any ID or timestamp columns for *TON_IoT* and IoT-23. We include IPs and ports in NF Bot-IoT because of its small number of available features to provide more discriminability. Features among the datasets include protocol, source, and destination bytes, connection state, service, duration, missed bytes, number of packets, window size, payload, entropy, DNS, SSL, and HTTP properties.

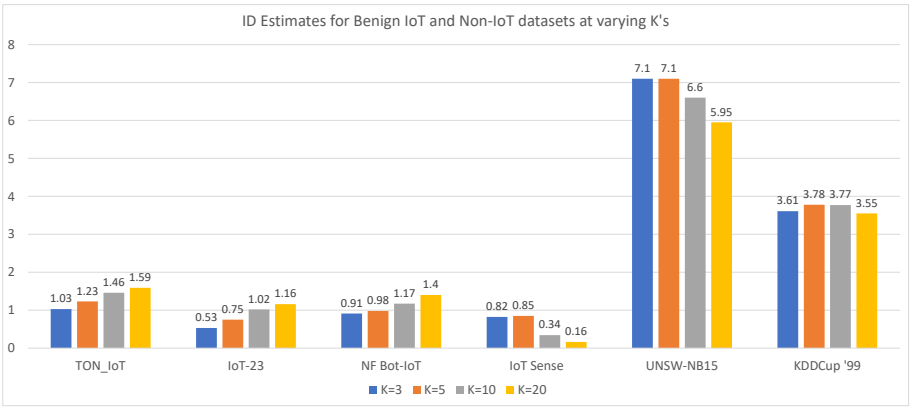

Figure 2. ID measurement for six different datasets. The four left datasets are IoT datasets and the two right datasets are non-IoT datasets. This figure also depicts different K values for ID.

5. Intrinsic Dimensionality of Network Datasets

Our first ID analysis is measuring benign networks at the full dataset level, using the approach explained in Section 3.1. We measure the ID over several K estimates (K=3, K=5, K=10, and K=20). Results of this experiment are depicted in Figure 2.

IoT datasets vs. Non-IoT Datasets. First, we look at the benign subsets of four IoT datasets (*TON_IoT*, IoT-23, NF Bot-IoT, IoT-Sense). Each dataset contains an ID estimate under 2. Comparatively, the non-IoT network datasets of UNSW-NB15 and KDD Cup '99 have ID estimates between 3.61 and 7.1, substanially higher than the IoT network data. The relative simplicity of IoT network data indicates it will be easier to estimate its behavior, leading to better NID models and more robust detection of attacks.

Effects of K Value. One other observation is related to K values. Several works note that the ID estimate is sensitive to K [4, 39], so estimating ID values over several K's gives us a robust picture. As Figure 2 shows, rank order of each dataset does not change substantially given the choice of K. Hamming Distance is used for all distance computations as described in Section 3.3.

Effects of Number of Features. Another notable finding across all datasets is that the extrinsic dimensionality, or number of features in the datasets, does not appear to be correlated with its intrinsic dimensionality. For example, *TON_IoT* contains almost as many features as UNSW-NB15 and KDD Cup 1999, however, its ID estimates are substantially lower than each. This indicates that the features of IoT network data are more simplistic in nature than both non-IoT datasets.

IoT vs. Computer Vision Datasets. Finally, we can compare these values to more difficult modeling on common computer vision datasets. Pope et al. [39] show that MNIST has the lowest ID, estimated between 7 and 13, with the state-of-the-art accuracy of 99.84%. In contrast, ImageNet has an ID between 26 and 43 with a state-of-the-art accuracy of 88.5%, indicating that datasets with a higher ID may be difficult to model. This is further examined in [39]. Relating these values to ID estimates on network data, we see that UNSW-NB15 has a similar ID to MNIST. While this indicates that UNSW-NB15 can still be modeled with very high accuracy, its ID is substantially higher than IoT network datasets, indicating IoT networks may be easier to model.

Potential Applications As noted by Liu et al. [30], a common complexity metric has use in several network-related tasks. For example, network management and provisioning, quality of service metrics, and security can each benefit from a metric to understand the complexity of the network.

In our results above, we show that non-IoT datasets have low ID values. Evidence suggests that such observations can lead to more robust and reliable modeling and quantification [39]. We explore this observation more in the following sections.

6. Intrinsic Dimensionality of IoT Devices

In this section, we analyze benign IoT traffic for specific devices. The purpose of this analysis is to compare device-specific complexity via ID in order (i) verify the behavior of various devices as described in [20] and (ii) reason about the ID values across different IoT devices.

We hypothesize that IoT datasets have lower complexity because the networks contain devices with simplistic behavior compared to non-IoT datasets. For example, devices such as lights and appliances are likely to have more straightforward network interactions compared to devices such as laptops and televisions.

6.1. Category-Based Results

For this task, we need to have the labeled devices in each dataset; however, not all IoT datasets labeled their devices or used simulated devices like IoT-23 for attack data. Consequently, we excluded those from this experiment. We were able to find three real IoT devices in IoT-23 and 14 other real devices from IoT-Sense dataset. We categorized devices into four categories of Lights, Appliances, Hubs/Outlets, and Smart Controller. In total, we have 17 labeled devices with benign traffic, with one device (Phillips HUE Light) in both IoT-23 and IoT Sense. Table 2 reports results of this experiment.

6.2. Complexity-Based Results

Haefner and Ray [20] defined a spectrum for the complexity for IoT devices, starting with simple devices such as single-purpose machines with low variability in their network interactions to complex devices (like Amazon Echo) with high variability in their network interactions. Similarly, and to simplify understanding of our results, we split ID measures into three categories: Low (0 to 0.5), Medium (0.5 to 0.7), and High ID (more than 0.7+).

In our datasets, 6 of these devices (Omna Camera, Smart Things huB, Netmao Climate, Lutron Hub, Wemo Outlet, and iDevice Socket) are labeled as low complexity devices. Three devices of iView Light, TP-Link Light, and D-Link Camera are labeled as medium complexity. Seven devices are classified with high complexity measures (Wink Hub, Philips HUE light, Door Lock, Musiac Music, AWOX light, and Amazon Echo). Multiple observations can be made here.

First, Amazon Echo has the highest ID value among all devices, which makes sense. In addition, while Haefner and Ray [20] do not measure Amazon Echo directly, they measured a similar device in Alexa. Their results confirmed that Amazon Alexa had a high complexity measure, which matches our findings. Further, the rank order of Com-

Table 2. Results for LID experiments for 17 different devices on IoT-23 and IoT-Sense.

Category	Device	Dataset	ID
Light	TCP light	IoTSense	0.787
Light	iView Light	IoTSense	0.543
Light	AWOX Light	IoTSense	0.845
Light	Phillips Hue	IoTSense	0.796
Light	TP-Link Light	IoTSense	0.55
Light	Phillips Hue	IoT-23	0.73
Appliance	Musiac Music player	IoTSense	0.866
Appliance	D-Link Camera	IoTSense	0.595
Appliance	Omna Camera	IoTSense	0.323
Appliance	Netamo Climate	IoTSense	0.439
Appliance	Somfy Door Lock	IoT-23	0.81
Hub/Outlet	iDevice Socket	IoTSense	0.474
Hub/Outlet	WEMO Outlet	IoTSense	0.483
Hub/Outlet	Lutron Hub	IoTSense	0.447
Hub/Outlet	Wink Hub	IoTSense	0.881
Hub/Outlet	Smart Things Hub	IoTSense	0.353
Smart Controller	Amazon Echo	IoT-23	1.14

plexIoT is similar to ours: both sets had TP-Link Light, Philips HUE, Smart Hub, and Alexa/Echo devices. ComplexIoT measured TP-Link as the lowest complexity, followed by the HUE, Smart Things Hub, and Alexa. Our measurements indicated that the Smart Things Hub had the lowest complexity, followed by the TP-Link Light, HUE, and Echo. The only inconsistency in the ranks was the Smart Things Hub, where Complex IoT measured a higher complexity value. Otherwise, the order of complexities of each device was the same. The Smart Things Hub could have yielded different measurements between the two results because of varying network captures between the two datasets.

All devices in the low-complexity range are aligned with our understanding of the simple-functionalities of these devices, except the Omna Camera. Like a camera with a high volume of sending and receiving data, we expect it will fall in the medium complexity range, similar to the D-Link Camera. One reason could be that the camera uses a specific protocol for sending images/videos (like UDP) that are not captured in the dataset we had, and only command packets have been captured. Consequently, the LID model does not give it a high value.

In our experiment, we had six lights, and all of them fell into medium and high complexity, a consistent result among them. We had two instances of the same device (Philips Hue light) in two datasets with different sets of features. The results of these two devices are very close to each other, 0.796 and 0.73 for IoT-Sense and IoT-23 datasets, respectively. This shows that regardless of the different features in the two datasets, our proposed approach consistently evaluated the same device, which is promising.

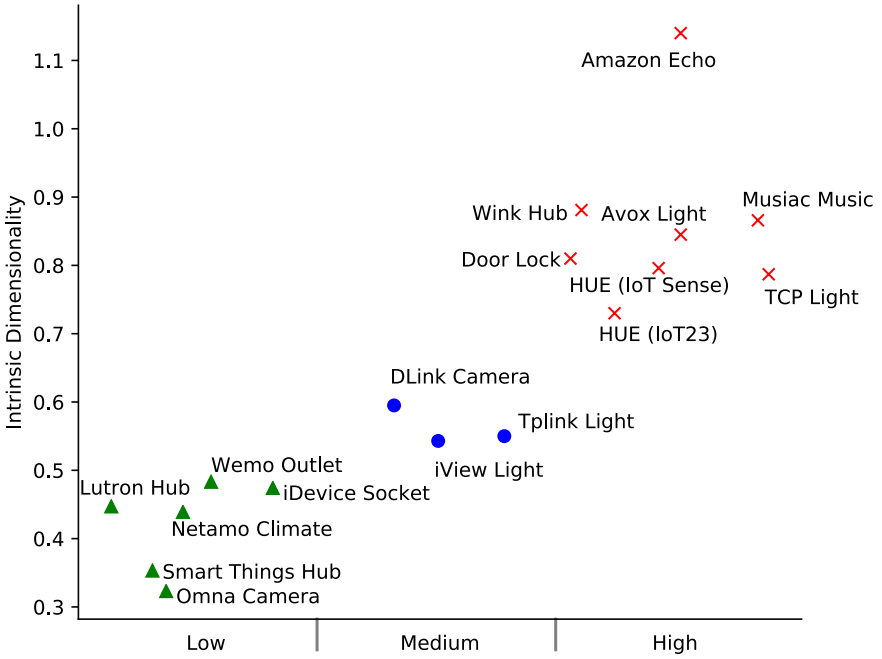

Figure 3. Intrinsic Dimensionality of specific devices for each dataset. We use 14 devices from IoT Sense dataset plus the 3 real devices from IoT-23 to measure ID. We categorize each value into Low (up to 0.5), Medium (0.5 to 0.7), and High ID (0.7+). The x-axis represents the three categories (low, medium, high) while the y-axis denotes the exact numbers for each device.

7. Anomaly Detection Results

In this section, we extend our findings of ID measurements on IoT networks to the *sample* level, showing that we can use the localized complexity measurement of LID to detect anomalous behavior.

7.1. Algorithm Comparison Results

We show that all IoT network datasets exhibit low complexity in Section 5. However, benign samples individually fit this same pattern, while malicious activity exhibit's higher LID measurements when measuring it against a benign baseline. We use this finding to the task of anomaly detection, using three public IoT datasets with benign and malicious examples. Results are compared against each other in Table 3.

Table 3 reports results of our proposed Weighted Hamming LID versus other baseline models of KNN and Weighted Hamming KNN, Isolation Forest, and Autoencoder. We run experiments with four different K values of 3,5,10, and 20. Our algorithm outperforms the Isolation Forest in every dataset, the weighted Hamming KNN and Autoencoder in four out of five datasets, and the standard KNN in three out of five datasets. Overall, the algorithm showed the best results in three out of five datasets, which is a significant result.

Table 3. Results for our model and 4 baseline models. We experiment with K values of 3,5,10, and 20 for the KNN, KNN (Weighted Hamming), and Weighted Hamming LID estimators, but report best results for each K in each dataset. We choose the K with the best results in the results. K = 5, 10, and 3 give best results for three datasets of NF Bot-IoT, *TON_IoT*, and IoT-23 respectively. The best ROC and PR for each dataset is in bold.

	NF Bot-IoT		***Ton-IoT***		**IoT-23**	
Test Type	ROC	PR	ROC	PR	ROC	PR
Isolation Forest	0.957	0.999	0.574	0.442	0.492	0.594
KNN	0.961	0.999	0.834	0.716	0.990	0.970
Weighted Hamming KNN	0.955	0.998	0.804	0.612	0.990	0.918
Autoencoder	0.919	0.998	0.622	0.789	0.572	.860
Weighted Hamming LID (Ours)	**0.970**	**0.999**	**0.917**	**0.831**	**0.998**	**0.994**

The most notable result was with the *TON_IoT* dataset, where the algorithm outperformed other unsupervised learning algorithms by a large margin. Figure 4 shows the distinct advantage of using the weighted Hamming algorithm over both the KNN and weighted KNN algorithms, with an ROC score more than 0.08 higher than the KNN algorithm and a PR score almost 0.12 above the KNN.

Algorithms that performed distance computations to their closest neighbors exhibited the best results (KNN, weighted Hamming KNN, and LID algorithms). Isolation Forest had low results for both *TON_IoT* and IoT-23. The Autoencoder did well on four out of five datasets, but had nearly random results on *TON_IoT*.

Overall, results indicate the Weighted Hamming LID estimator is a strong alternative to classic anomaly detection algorithms such as the Autoencoder, Isolation Forest, and K-nearest neighbors for IoT datasets. The algorithm shows strong results across three different IoT datasets, indicating it generalizes well to several types of attacks and network datasets.

Table 4 shows the poor results of the Weighted Hamming LID algorithm on the non-IoT datasets. Both non-IoT datasets have a higher complexity, which is reflected in their individual samples being closer to zero. As a result, the Weighted Hamming LID algorithm has a harder time distinguishing benign examples from malicious examples. For example, it is harder to distinguish between benign and malicious examples if a device is performing many complex interactions on a network compared to a network full of devices which only do simple things.

7.2. Attacks Specific Results

In this experiment, we break down our results by attack in each IoT dataset using the Weighted Hamming Distance LID estimator. We group some IoT-23 attacks based on their broad category; for example, we group File Download and Heartbeat attacks as C&C, because these attacks both come from a known C&C server and they have a small overall sample size. The results of this experiment is reported in Table 5.

Results are varied for IoT-23 and *TON_IoT*; stronger results for NF Bot-IoT. While some attacks are easily recognizable by our algorithm, such as Distributed Denial-of-Service (DDoS) and DoS attacks, others do very poorly, such as *TON_IoT*'s Ransomware, Backdoor, and Man-in-the-middle attack (MITM) attacks, with Precision-Recall (PR) scores under 0.2 in each case.

	UNSW-NB15		Kaggle NID	
Test Type	ROC	PR	ROC	PR
Isolation Forest	0.779	0.896	0.2288	0.324
KNN	0.206	0.514	0.956	0.936
Weighted Hamming KNN	0.279	0.542	0.951	0.907
Autoencoder	**0.829**	**0.902**	**0.974**	**0.981**
Weighted Hamming Lid	0.4	0.615	0.86	0.784

Table 4. Results of unsupervised anomaly detection algorithms on Non-IoT datasets. Results indicate that the Weighted Hamming LID estimator is correlated with the ID estimate of the dataset: for a higher ID, such as the UNSW-NB15, the algorithm performs poorly, while the more moderate ID of Kaggle NID performs better. These values indicate we can use ID as a gauge for how well the Weighted Hamming LID anomaly detection will perform. These results also indicate that more complex datasets are better handled by deep learning methods (Autoencoder).

The first two datasets in Table 5, IoT-23 and NF Bot-IoT, performed well on all ROC metrics for each attack. Precision-Recall metrics performed slightly worse than ROC in IoT-23, with lower values for Okiru and C&C attacks. We found that both attacks had network packets that were largely indistinguishable from benign network. For example, Okiru attacks were generally TCP protocol with packet count of 1 and low byte count, similar to many benign network packets.

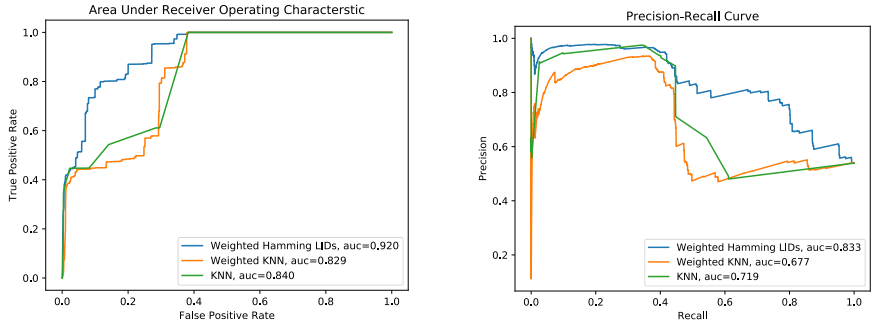

Figure 4. The ROC and PR curves for KNN, Weighted Hamming KNN, and Weighted Hamming LID on *TON_IoT* dataset.

While the results of attack detection using the Weighted Hamming Distance LID estimator are stronger than other algorithms, results on specific attack types show that IoT networks may still be vulnerable in certain cases. In particular, *TON_IoT* performed poorly on most attacks, leading us to further analyze the data. We found that of 300K benign examples, 61.8% were packets with an exact match with another benign packet, *i.e.*, all 38 features had the same value. Further, more than 26K malicious examples had an exact match with at least one benign sample. These factors indicate that the data generated for *TON_IoT* has low discriminability when excluding source and destination IPs and ports, as we did. We note in Section 4.7 that the authors of the original papers [35, 3] performed supervised classification on *TON_IoT*, yielding high accuracy using both source and destination IPs and ports. They recommended removing source and des-

Table 5. Anomaly detection results for specific attack types available in each dataset. Varied results in individual datasets indicate networks may be more prone to specific attacks in IoT networks.

Dataset	Attack	Sample Size	ROC	PR
IoT-23	Horiz.PortScan	199,283	0.998	0.989
	DDoS	54,750	0.999	0.892
	Okiru	53,959	0.999	0.701
	C&C	271	0.999	0.632
NF Bot-IoT	Theft	1,909	0.990	0.902
	DDoS	56,844	0.999	0.999
	DoS	56,833	0.998	0.999
	Recon.	470,655	0.963	0.999
TON_IoT	Scanning	19,995	0.946	0.343
	DoS	19,994	0.900	0.374
	Injection	19,930	0.991	0.895
	DDoS	19,790	0.934	0.612
	Password	17,428	0.970	0.794
	XSS	8,914	0.972	0.670
	Ransomware	7,221	0.886	0.088
	Backdoor	19,908	0.756	0.110
	MITM	1,041	0.971	0.182

tination IPs and ports for further experimentation, however, the remaining 38 features contained identical or close to identical values for benign and malicious examples.

8. Intrinsic Dimensionality of Sythesized Datasets

In this section, we provide additional analysis of ID and the Weighted Hamming LID estimator using synthesized data. We hypothesize that adding synthetic data to the benign and attack datasets can provide new insights into the quality of different metrics and algorithms used in this paper.

We use a VAE [24] to generate new synthesized data for both benign and attack sets. VAE's are generative models with a similar structure to the autoencoder, which additionally have the ability to regularize the latent space (bottleneck) of the autoencoder in order to generate new samples. In particular, the VAE encodes a *distribution* of a point over the latent space, with the goal of maximizing the likelihood that data point x belongs to a Gaussian distribution – $N(x|\mu,\sigma)$. We denote this $p(x|\theta)$. In addition to Equations 3.4.3 and 3.4.3, the VAE subtracts an additional term (the Kullback-Leibler divergence) to the autoencoder loss function, which minimizes the distance between the encoder q_ϕ and the Gaussian distribution for latent variable z:

$$D_{KL} = q_\phi(z|x)||p_\theta(z|x) \tag{9}$$

The resulting model enables us to sample from a Gaussian distribution, which the encoder has fit our dataset to, and produce new synthetic examples produced from the decoder part of the model.

We produce synthesized data for both benign and malicious sets of each dataset independently. To ensure the synthetic data is in alignment with real data, for continuously valued features we map the synthetic feature values to the closest point-wise features in a randomly 2,500 samples from the original dataset. This additional step ensures our model is producing features which have been observed by the network.

To assess the ability of anomaly detection algorithms on synthetic data, we ask the following research questions:

Q1. How robust are the anomaly detection algorithms studied in this paper to synthetically generated attacks?

Q2. Can we create a more robust unsupervised learning algorithm by including additional synthetically-generated benign samples?

To address Q1, we assess each anomaly detection algorithm against a new synthetically generated attack dataset. Each algorithm uses the benign example set as the training set plus a new synthetically generated benign set, and we measure the ability of the anomaly detection algorithm to detect new synthetically generated attacks. To be consistent with previous sections, our synthetic datasets are the same size as the real attack data.

To address Q2, we create new benign examples using a VAE. We add these new samples to the training dataset along with the real samples, and test the models' ability to detect attacks given the enhanced training set. We hypothesize that each algorithm will be more robust to attacks with the enhanced dataset.

8.1. Quality of Synthetic Data

Before addressing Q1 and Q2, we first measure the quality of the synthesized datasets. Ensuring our data fits the underlying properties of the original dataset is crucial for success in downstream tasks.

To ensure quality, we analyze two properties of the new synthetic dataset, which we denote X'. First, we measure the X'_{benign} ID for each dataset to ensure it is similar to the original values. Since our synthetic datasets produce the same number samples and features as the original datasets, we expect the ID to be similar.

Additionally, we measure each synthetic datasets ROC compared to its real counterpart using the evidence lower bound (ELBO) from the VAE. For example, when generating synthetic samples from the VAE, we pass the examples through the full model to obtain the ELBO. The ELBO can be obtained through the full error function of the model:

$$\mathscr{L}(x_c, x'_c) + \mathscr{L}(x_d, x'_d) - D_{KL} \tag{10}$$

After obtaining the ELBO for both real samples and synthetic samples from the benign dataset passed through the VAE, we pass the values through the ROC calculation, with 0's being real values and 1's being synthetic values. If the ROC diverges signifi-

Table 6. Data Quality Tests. We test our synthetic datasets by comparing ID between real data (first row) and synthetic data (second row), finding that ID estimates are similar in IoT datasets and slightly higher in non-IoT datasets. Our second test, X vs. X' ROC, yields almost random results, indicating the synthetic datasets are indistinguishable from real datasets.

Test Type	Dataset	NF Bot-IoT	*TON_IoT*	IoT-23	UNSW-NB15	Kaggle NID
ID (K=10)	X	1.17	1.46	1.02	6.6	3.77
ID (K=10)	X'	2.04	1.62	0.31	10.6	8.71
ROC	X, X'	0.509	0.499	0.541	0.523	0.507

cantly from 0.5, we can consider synthesized dataset as considerably different from the real dataset.

Results for both tests are depicted in Table 6. In the first two rows, we show the ID for the real dataset (row 1) and the synthetic dataset (row 2). We find similar results between real and synthetic datasets for IoT datasets, and higher results for synthetic ID on non-IoT datasets. We conjecture that higher results on non-IoT datasets are a result of higher ID on the original data, leading to the VAE having a more difficult time learning the complexities of the data. For example, VAE's do very well with simple image datasets such as MNIST, while struggling more on complex colored images such as ImageNet.

Our second data quality test yields affirmative results as to the quality of our synthetic data. ROC scores between X and X' are between 0.499 and 0.541, which provide confidence that our synthetic examples are indistinguishable from real samples.

8.2. Results: Synthetic Attacks

Our first experiment involves testing various unsupervised anomaly detection algorithms on synthetically generated attacks. Our goal is to test whether new attacks can bypass the various algorithms, in particular the Weighted Hamming LID. To generate new attacks, we train malicious examples on a VAE model, sampling from the latent Gaussian distribution upon completion of training.

In Table 7, we find that the synthetic attacks are more easily detectable than the real attacks across all datasets. For the Isolation Forest, performance improvements are marginal, however for all other anomaly detection algorithms we experience a substantial improvement. In a real world context this has two implications: 1) Attacks similar to those provided in each of the five datasets (i.e. from the same distribution) will be reasonable to detect at the rates observed by each algorithm, and 2) Attackers attempting to maneuver around NIDS systems via synthetically generated attacks may have less success than by using their original approach.

8.3. Results: Synthetically-Generated Benign Examples

Our second experiment involves testing whether unsupervised anomaly detection algorithms can be *more robust* given new synthetically generated benign examples. We enhance each unsupervised algorithm by appending synthetically generated benign samples onto the real benign sample set. We hypothesize that: 1) A more robust training set will provide better coverage of benign samples compared to the baseline training set, and 2) The enhanced training set will better distinguish benign samples from malicious

Table 7. Synthetic Attacks We find that each unsupervised anomaly detection algorithm is robust to new synthetically generated attacks. Performance improvements are substantial for all algorithms except for the Isolation Forest.

Dataset	NF Bot-IoT		*TON_IoT*		IoT-23		UNSW-NB15		Kaggle NID	
Test Type	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR
Iso.Forest	0.958	0.998	0.584	0.453	0.493	0.594	0.830	0.946	0.229	0.324
KNN	0.997	0.999	0.943	0.889	0.988	0.959	0.594	0.703	**0.993**	**0.989**
W.H. KNN	0.996	0.99	0.914	0.792	0.994	0.946	**0.902**	**0.935**	0.966	0.922
Autoencoder	0.995	0.999	0.711	0.528	0.604	0.682	0.718	0.838	0.975	0.987
W.H. LID	**0.998**	**0.999**	**0.953**	**0.892**	**0.999**	**0.993**	0.865	0.877	0.845	0.784

Table 8. IoT Datasets Enhanced with Synthesized Samples: We highlight the performance improvements achieved by using synthetically generated benign samples to enhance the unsupervised anomaly detection algorithms. **Top:** The new ROC/PR scores for each algorithm. **Bottom:** The performance improvement achieved by using real samples plus synthetically generated samples against using only real samples.

Dataset	NF Bot-IoT		*TON_IoT*		IoT-23	
Test Type	ROC	PR	ROC	PR	ROC	PR
Isolation Forest	0.958	0.999	0.542	0.35	0.431	0.565
KNN	0.974	0.999	**0.945**	**0.901**	0.988	0.949
Weighted Hamming KNN	0.971	0.999	0.908	0.796	0.987	0.916
Autoencoder	0.908	0.998	0.884	0.844	0.715	0.828
Weighted Hamming LID	**0.975**	**0.999**	0.943	0.865	**0.998**	**0.988**
Improvement Against Baseline (only real samples)						
	ROC	PR	ROC	PR	ROC	PR
Isolation Forest	+0.001	0	-0.032	-0.092	-0.061	-0.029
KNN	+0.013	0	+0.111	+0.185	-0.002	-0.021
Weighted Hamming KNN	+0.016	+0.001	+0.104	+0.184	-0.003	-0.002
Autoencoder	-0.011	0	+0.262	+0.055	+0.144	-0.032
Weighted Hamming LID	+0.005	0	+0.026	+0.034	0	-0.006

samples because of this enhanced coverage. For this experiment, our test set consists of the real set of benign samples and the real set of malicious samples.

Results in Tables 8 and 9 show the increased performance of using the enhanced training set across each algorithm. For example, in Table 8, the Weighted Hamming LID estimator improves by 0.026 ROC and 0.034 PR for the *TON_IoT* dataset. Despite this, the KNN improves even more, with performance improvements of 0.111 ROC and 0.185 PR. In Table 9, we find that the non-IoT datasets achieve even greater performance improvements using the enhanced datasets, with improvements up to 0.469 (Weighted Hamming LID, UNSW NB15) ROC and 0.420 (KNN, UNSW NB15) PR, respectively.

The second part of each Table denotes the performance improvements for each algorithm compared to only using the real data as the training set. While some datasets and algorithms performance dropped marginally, we found that most algorithms achieved improved performance with the synthetic additions to the training set.

Table 9. Non-IoT Datasets Enhanced with Synthesized Samples: Non-IoT datasets see a more substantial improvement compared to IoT datasets. We conjecture that this is because the datasets have more potential for improvement because of increased ID values

Dataset	UNSW NB15		Kaggle NID	
Test Type	ROC	PR	ROC	PR
Isolation Forest	0.808	0.939	0.188	0.314
KNN	0.63	0.934	0.986	0.981
Weighted Hamming KNN	0.921	0.947	0.958	0.918
Autoencoder	0.938	0.954	0.995	0.997
Weighted Hamming LID	0.869	0.863	0.922	0.869
Improvement Against Baseline (only real samples)				
Isolation Forest	+0.029	+0.043	-0.0408	-0.01
KNN	+0.424	+0.420	+0.030	+0.045
Weighted Hamming KNN	+0.642	+0.405	+0.007	+0.011
Autoencoder	+0.109	+0.052	+0.021	+0.016
Weighted Hamming LID	+0.469	+0.248	+0.062	+0.085

9. Conclusion

In this work, we view several network datasets through the lens of complexity and show that IoT datasets exhibit a lower ID complexity estimate than standard network collections. This finding extends to the point-wise estimation of complexity, where individual samples in (benign) IoT datasets contain low LID measures. We show that benign examples can be identified by either 1) exactly matching the features of a training set sample or 2) by a low LID estimate. We propose a novel algorithm for detecting malicious actors in an unsupervised manner, providing the ability to deploy a model into production with only two hyperparameters needed (k value for distance measurements, and threshold value t). The algorithm is based on the theoretical LID estimation using the Hill MLE estimator, using an entropy weighted Hamming distance for measuring distances between points and features. In addition, we show the benefits of using synthetically generated data to enhance our training set, reporting increased performance across by IoT and non-IoT datasets. Finally, we find that synthetically-generated attacks are easily detectable by various unsupervised learning algorithms.

One potential pitfall of the Weighted Hamming LID algorithm is that it is an expensive computation, requiring distance calculations between a reference point and each point of the training set. Similar to KNN, this calculation can be computationally expensive, and further experimentation is required to measure the performance of such a system in practice. However, state-of-the-art models such as deep networks share this same deficiency: large and typically expensive to run. Further experimentation could provide a thorough comparison of the two.

Another caveat to the algorithm is obtaining a clean dataset. While the unsupervised learning assumption holds when the model is trained with a training set of benign examples, obtaining a reliable benign dataset in an arbitrary IoT network may be problematic in practice since it can be hard to ascertain whether a collection of network packets contains 100% clean data. However, we show the potential to circumvent this issue by creating synthetic examples.

Despite these potential downfalls, our complexity analysis and algorithmic approach provide a novel mathematical look into the details surrounding several IoT network datasets. We show the relative simplicity of these network collections through ID estimates. Additionally, we make connections between complexity in IoT security and open problems in deep learning, such as the difficulty in modeling increasingly complex data such as large images. Connecting the dots between security and anomaly detection in machine learning remains an essential facet of developing secure systems, and we hope this paper can provide researchers with a unique perspective towards building more robust and secure frameworks.

References

[1] Rasheed Ahmad and Izzat Alsmadi. Machine learning approaches to IoT security: A systematic literature review. *Internet of Things*, 14:100365, 2021. ISSN 2542-6605. doi: 10.1016/j.iot.2021.100365. URL https://www.sciencedirect.com/science/article/pii/S2542660521000093.

[2] Fadele Ayotunde Alaba, Mazliza Othman, Ibrahim Abaker Targio Hashem, and Faiz Alotaibi. Internet of Things security: A survey. *Journal of Network and Computer Applications*, 88:10–28, 2017. ISSN 1084-8045. doi: 10.1016/j.jnca.2017.04.002. URL https://www.sciencedirect.com/science/article/pii/S1084804517301455.

[3] Abdullah Alsaedi, Nour Moustafa, Zahir Tari, Abdun Mahmood, and Adnan Anwar. TON_iot Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems. *IEEE Access*, 8:165130–165150, 2020. ISSN 2169-3536. doi: 10.1109/ACCESS.2020.3022862.

[4] Laurent Amsaleg, Oussama Chelly, Teddy Furon, Stéphane Girard, Michael E. Houle, Ken-ichi Kawarabayashi, and Michael Nett. Estimating Local Intrinsic Dimensionality. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 29–38. Association for Computing Machinery, 2015. ISBN 978-1-4503-3664-2. doi: 10.1145/2783258.2783405. URL https://doi.org/10.1145/2783258.2783405.

[5] Ioannis Andrea, Chrysostomos Chrysostomou, and George Hadjichristofi. Internet of Things: Security vulnerabilities and challenges. In *IEEE Symposium on Computers and Communication (ISCC)*, pages 180–187, 2015. doi: 10.1109/ISCC.2015.7405513.

[6] Alessio Ansuini, Alessandro Laio, Jakob H. Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *arXiv:1905.12784 [cs, stat]*, 2019. URL http://arxiv.org/abs/1905.12784.

[7] Amin Azmoodeh, Ali Dehghantanha, and Kim-Kwang Raymond Choo. Robust Malware Detection for Internet of (Battlefield) Things Devices Using Deep Eigenspace Learning. *IEEE Transactions on Sustainable Computing*, 4(1):88–95, 2019. ISSN 2377-3782. doi: 10.1109/TSUSC.2018.2809665.

[8] D. Bernal. 3 - Analytical techniques for damage detection and localization for assessing and monitoring civil infrastructures. In M. L. Wang, J. P. Lynch, and H. Sohn, editors, *Sensor Technologies for Civil Infrastructures*, volume 56, pages 67–92. Woodhead Publishing, 2014. ISBN 978-1-78242-242-6.

doi: 10.1533/9781782422433.1.67. URL https://www.sciencedirect.com/science/article/pii/B978178242242650003X.

[9] Bruhadeshwar Bezawada, Maalvika Bachani, Jordan Peterson, Hossein Shirazi, Indrakshi Ray, and Indrajit Ray. IoTSense: Behavioral Fingerprinting of IoT Devices. *arXiv:1804.03852 [cs]*, 2018. URL http://arxiv.org/abs/1804.03852.

[10] Siddharth Bhatia, Arjit Jain, Pan Li, Ritesh Kumar, and Bryan Hooi. MSTREAM: Fast Anomaly Detection in Multi-Aspect Streams. *Proceedings of the Web Conference 2021*, pages 3371–3382, 2021. doi: 10.1145/3442381.3450023. URL http://arxiv.org/abs/2009.08451.

[11] Deepak Choudhary. Security Challenges and Countermeasures for the Heterogeneity of IoT Applications. *Journal of Autonomous Intelligence*, 1:16, 2019. ISSN 2630-5046. doi: 10.32629/jai.v1i2.25. URL http://en.front-sci.com/index.php/JAI/article/view/25.

[12] Benoît Claise. Cisco Systems NetFlow Services Export Version 9. Request for Comments RFC 3954, Internet Engineering Task Force, 2004. URL https://datatracker.ietf.org/doc/rfc3954.

[13] Mauro Conti, Ali Dehghantanha, Katrin Franke, and Steve Watson. Internet of Things Security and Forensics: Challenges and Opportunities. *Future Generation Computer Systems*, 78:544–546, 2018. ISSN 0167739X. doi: 10.1016/j.future.2017.07.060. URL http://arxiv.org/abs/1807.10438.

[14] Ensieh Modiri Dovom, Amin Azmoodeh, Ali Dehghantanha, David Ellis Newton, Reza M. Parizi, and Hadis Karimipour. Fuzzy pattern tree for edge malware detection and categorization in IoT. *Journal of Systems Architecture*, 97: 1–7, 2019. ISSN 1383-7621. doi: 10.1016/j.sysarc.2019.01.017. URL https://www.sciencedirect.com/science/article/pii/S1383762118305265.

[15] D. Dua and C. Graff. Uci machine learning repository. URL http://archive.ics.uci.edu/ml.

[16] Sebastian Garcia, Agustin Parmisano, and Maria Jose Erquiaga. IoT-23 Dataset: A labeled dataset of Malware and Benign IoT Traffic.(version 1.0.0) [data set]. zenodo. URL https://www.stratosphereips.org/datasets-iot23.

[17] Matt Gorbett and Nathaniel Blanchard. Utilizing Network Properties to Detect Erroneous Inputs. *arXiv:2002.12520 [cs]*, 2020. URL http://arxiv.org/abs/2002.12520.

[18] Robert M. Gray. *Entropy and Information Theory*. Springer Publishing Company, Incorporated, 2nd edition, 2011. ISBN 9781441979698.

[19] Hamed HaddadPajouh, Ali Dehghantanha, Raouf Khayami, and Kim-Kwang Raymond Choo. A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting. *Future Generation Computer Systems*, 85, 2018. ISSN 0167-739X. doi: 10.1016/j.future.2018.03.007. URL https://www.sciencedirect.com/science/article/pii/S0167739X1732486X.

[20] Kyle Haefner and Indrakshi Ray. ComplexIoT: Behavior-Based Trust For IoT Networks. In *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 56–65, 2019. doi: 10.1109/TPS-ISA48467.2019.00016.

[21] Dan Hendrycks and Kevin Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *arXiv:1610.02136 [cs]*, 2018. URL http://arxiv.org/abs/1610.02136.

[22] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006. doi: 10.1126/science.1127647. URL https://www.science.org/doi/10.1126/science.1127647.

[23] Sydney M. Kasongo and Yanxia Sun. Performance Analysis of Intrusion Detection Systems Using a Feature Selection Method on the UNSW-NB15 Dataset. *Journal of Big Data*, 7:105, 2020. ISSN 2196-1115. doi: 10.1186/s40537-020-00379-6. URL https://doi.org/10.1186/s40537-020-00379-6.

[24] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[25] Roopha Kollolu. A Review on Wide Variety and Heterogeneity of IoT Platforms. SSRN Scholarly Paper ID 3912454, Social Science Research Network, Rochester, NY, 2020. URL https://papers.ssrn.com/abstract=3912454.

[26] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Jill Slay. Towards Developing Network Forensic Mechanism for Botnet Activities in the IoT Based on Machine Learning Techniques. In *Mobile Networks and Management*, pages 30–44. Springer International Publishing, 2018. ISBN 978-3-319-90775-8. doi: 10.1007/978-3-319-90775-83.

[27] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. *Future Generation Computer Systems*, 100:779–796, 2019. ISSN 0167-739X. doi: 10.1016/j.future.2019.05.041. URL https://www.sciencedirect.com/science/article/pii/S0167739X18327687.

[28] Rafał Kozik, Marek Pawlicki, and Michał Choraś. A new method of hybrid time window embedding with transformer-based traffic data classification in IoT-networked environment. *Pattern Analysis and Applications*, 24(4):1441–1449, 2021. ISSN 1433755X. doi: 10.1007/s10044-021-00980-2. URL https://www.mendeley.com/catalogue/92cc3e51-9dc9-3c8e-9e05-aeea7382b93c/.

[29] Elizaveta Levina and Peter J. Bickel. Maximum Likelihood estimation of intrinsic dimension. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 777–784, Cambridge, MA, USA, 2004. MIT Press.

[30] Lisa Liu, Daryl Essam, and Timothy Lynar. On quantifying the complexity of iot traffic. In *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pages 379–382, 2021. doi: 10.1109/LCN52139.2021.9525007.

[31] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E. Houle, and James Bailey. Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. *arXiv:1801.02613 [cs]*, 2018. URL http://arxiv.org/abs/1801.02613.

[32] Souhail Meftah, Tajjeeddine Rachidi, and Nasser Assem. Network Based Intrusion Detection Using the UNSW-NB15 Dataset. *International Journal of Computing and Digital Systems*, 8:478–487, 2019. ISSN 2210-142X. doi: 10.12785/ijcds/080505. URL https://journal.uob.edu.bh:443/handle/123456789/3580.

[33] Mujahid Mohsin, Zahid Anwar, Ghaith Husari, Ehab Al-Shaer, and Mohammad Ashiqur Rahman. IoTSAT: A formal framework for security analysis of the internet of things (IoT). In *2016 IEEE Conference on Communications and Network Security (CNS)*, pages 180–188, 2016. doi: 10.1109/CNS.2016.7860484.

[34] Warren Morningstar, Cusuh Ham, Andrew Gallagher, Balaji Lakshminarayanan, Alex Alemi, and Joshua Dillon. Density of States Estimation for Out of Distribution Detection. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 3232–3240. PMLR, 2021. URL https://proceedings.mlr.press/v130/morningstar21a.html.

[35] Nour Moustafa. A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_iot datasets. *Sustainable Cities and Society*, 72: 102994, 2021. ISSN 2210-6707. doi: 10.1016/j.scs.2021.102994. URL https://www.sciencedirect.com/science/article/pii/S2210670721002808.

[36] Nour Moustafa and Jill Slay. UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6, 2015. doi: 10.1109/MilCIS.2015.7348942.

[37] Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Do Deep Generative Models Know What They Don't Know? 2018. URL https://openreview.net/forum?id=H1xwNhCcYm.

[38] 18th October 2021. Tapping AI for Intrusion Detection Systems. URL https://www.iotworldtoday.com/2021/10/18/tapping-ai-for-intrusion-detection-systems/.

[39] Phil Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The Intrinsic Dimension of Images and Its Impact on Learning. 2020. URL https://openreview.net/forum?id=XJk19XzGq2J.

[40] B. M. Rashma, Suchitha Macherla, Achintya Jaiswal, and G. Poornima. Handling Heterogeneity in an IoT Infrastructure. In *Advances in Machine Learning and Computational Intelligence*, Algorithms for Intelligent Systems, pages 635–643, Singapore, 2021. Springer. ISBN 9789811552434. doi: 10.1007/978-981-15-5243-460.

[41] Shahadate Rezvy, Yuan Luo, Miltos Petridis, Aboubaker Lasebae, and Tahmina Zebin. An efficient deep learning model for intrusion classification and prediction in 5G and IoT networks. In *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2019. doi: 10.1109/CISS.2019.8693059.

[42] Syed Rizvi, RJ Orr, Austin Cox, Prithvee Ashokkumar, and Mohammad R. Rizvi. Identifying the attack surface for IoT network. *Internet of Things*, 9:100162, 2020. ISSN 2542-6605. doi: 10.1016/j.iot.2020.100162. URL https://www.sciencedirect.com/science/article/pii/S2542660520300056.

[43] Bipraneel Roy and Hon Cheung. A Deep Learning Approach for Intrusion Detection in Internet of Things using Bi-Directional Long Short-Term Memory Recurrent Neural Network. pages 1–6, 2018. doi: 10.1109/ATNAC.2018.8615294. ISSN: 2474-154X.

[44] Amiya Kumar Sahu, Suraj Sharma, M. Tanveer, and Rohit Raja. Internet of Things attack detection using hybrid Deep Learning Model. *Computer Communications*, 176:146–154, 2021. ISSN 0140-3664. doi: 10.1016/j.comcom.2021.05.024. URL https://www.sciencedirect.com/science/article/pii/S0140366421002164.

[45] Mohanad Sarhan, Siamak Layeghy, Nour Moustafa, and Marius Portmann. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In *Big Data Technologies and Applications*, pages 117–135, Cham, 2021. Springer International Publishing. ISBN 978-3-030-72802-1. doi: 10.1007/978-3-030-72802-19.

[46] Joan Serrà, David Álvarez, Vicenç Gómez, Olga Slizovskaia, José F. Núñez, and Jordi Luque. Input Complexity and Out-of-distribution Detection with Likelihood-based Generative Models. 2019. URL https://openreview.net/forum?id=SyxIWpVYvr.

[47] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. A Deep Learning Approach to Network Intrusion Detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2:41–50, 2018. ISSN 2471-285X. doi: 10.1109/TETCI.2017.2772792.

[48] Praneet Singh, Jishnu Jaykumar, Akhil Pankaj, and Reshmi Mitra. EdgeDetect: Edge-centric Network Intrusion Detection using Deep Neural Network. *arXiv:2102.01873 [cs]*, February 2021. URL http://arxiv.org/abs/2102.01873.

[49] Bernadette J. Stolz, Jared Tanner, Heather A. Harrington, and Vidit Nanda. Geometric anomaly detection in data. *Proceedings of the National Academy of Sciences*, 117(33):19664–19669, 2020. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.2001741117. URL https://www.pnas.org/content/117/33/19664.

[50] Samuel Tweneboah-Koduah, Knud Erik Skouby, and Reza Tadayoni. Cyber Security Threats to IoT Applications and Service Domains. *Wireless Personal Communications*, 95(1):169–185, 2017. ISSN 1572-834X. doi: 10.1007/s11277-017-4434-6. URL https://doi.org/10.1007/s11277-017-4434-6.

[51] A.R. Vasudevan, E. Harshini, and S. Selvakumar. SSENet-2011: A Network Intrusion Detection System dataset and its comparison with KDD CUP 99 dataset. In *2011 Second Asian Himalayas International Conference on Internet (AH-ICI)*, pages 1–5, 2011. doi: 10.1109/AHICI.2011.6113948.

[52] Qizhou Wang, Sarah M Erfani, Christopher Leckie, and Michael E Houle. A Dimensionality-Driven Approach for Unsupervised Out-of-distribution Detection. page 9, 2021.

[53] Kai Zhao and Lina Ge. A Survey on the Internet of Things Security. pages 663–667, 2013. doi: 10.1109/CIS.2013.145.

[54] Shuo Zhou, Antoinette Tordesillas, Mehdi Pouragha, James Bailey, and Howard Bondell. On local intrinsic dimensionality of deformation in complex materials. *Scientific Reports*, 11(1):10216, 2021. ISSN 2045-2322. doi: 10.1038/s41598-021-89328-8. URL https://www.nature.com/articles/s41598-021-89328-8.