

Article

Federated Machine Learning to Enable Intrusion Detection Systems in IoT Networks

Mark Devine, Saeid Pourroostaei Ardakani *, Mohammed Al-Khafajiy and Yvonne James

School of Engineering and Physical Sciences, University of Lincoln, Lincoln LN2 7TS, UK; 25242748@students.lincoln.ac.uk (M.D.); malkhafajiy@lincoln.ac.uk (M.A.-K.); yjames@lincoln.ac.uk (Y.J.)

* Correspondence: spourroostaei@lincoln.ac.uk

Abstract: Intrusion detection systems for internet-of-things devices are becoming more relevant as the international reliance on internet-of-things devices grows. Federated learning is one of the most promising areas of study in AI-driven intrusion detection systems in the internet of things and networking, being able to mitigate some of the more severe hardware requirements. Using a federated learning framework, we trained and evaluated several machine learning models to identify distributed denial-of-service attacks in IoT systems. Our framework introduces a novel approach to data preparation for federated learning, incorporating new processing techniques to maximise performance on real-world non-synthetic data. The results show that our proposed first-of-its-kind federated SVM model is highly effective for intrusion detection and matches or outperforms the benchmark algorithms in terms of the attack prediction accuracy, while demonstrating its feasibility for deployment on edge devices. We also compare the physical metrics to conduct one of the first comprehensive evaluations of model suitability for resource-constrained IoT networks, providing valuable insights into the trade-offs between detection accuracy and computational efficiency.

Keywords: intrusion detection; support vector machine; internet of things; federated learning



Academic Editor: Antoni Morell

Received: 30 January 2025

Revised: 12 March 2025

Accepted: 14 March 2025

Published: 17 March 2025

Citation: Devine, M.; Ardakani, S.P.; Al-Khafajiy, M.; James, Y. Federated Machine Learning to Enable Intrusion Detection Systems in IoT Networks. *Electronics* **2025**, *14*, 1176. <https://doi.org/10.3390/electronics14061176>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Cyber security as a research field has grown as the reliance on data increases; the backbone of society exists within servers and devices, all of which need to be secured and protected against threat actors. The extent to which modern businesses use cyber security tools as a shield can be clearly seen by the vast up-tick in cyber insurance usage [1], revealing the need to protect our data and systems from threat actors. In recent years, tools have been developed and deployed in a bid to counteract cyber threats. One of the earliest tools was intrusion detection system (IDS) software, which detects cyber attacks based on either network traffic or internal actions. Historically, these consisted of vast databases of attack signatures that would analyse packets and system behaviour to detect malicious activity or cyber attacks [2], a prevalent type being a distributed denial of service (DDoS) attack [3], where vast amounts of data overload a system from a vast array of sources. IDS platforms have grown in complexity, resulting in a tool which can conduct anomaly-based detection, where machine learning models are trained to identify suspicious behaviour patterns, blocking the source and initiating alerts. Both signature- and anomaly-based approaches exist within industry [3], but signature-based are more common, as they are less likely to result in false intrusion detections (false positives), which can cause a system

to go down; for a company that makes its revenue via an online service, this can cause serious revenue loss. False positives are a common occurrence in classification tasks and suggest that a model is oversensitive. For a complex task such as IDS, it is expected that there will be false positives, but they can be reduced.

These are systems primarily targeted at businesses, who have vast networks of IoT-enabled devices that can be targeted by threat actors. A new era of smart devices has seen the proliferation of internet-of-things (IoT) devices into our homes and businesses [4] including, networked fridges, coffee machines, and even toothbrushes. Historically, IoT security has been a neglected area of research within cyber security [5], partially due to the lack of attacks surrounding it and partially due to the airgap around IoT devices (although the advent of cloud storage has made this less viable [6,7]). This means that IoT security (and by extension IoT IDS) research is a relatively new and vibrant field, with plenty of gaps and unanswered questions to be addressed, with recent work showing flaws in all layers of their operations [5]. IoT introduces new challenges that otherwise do not need to be considered to the same extent; namely, the tight resource restrictions mean that both training delay and memory usage need to be considered. This has led to a recent push in academia for a deeper understanding of IoT attacks and a concerted effort for more effective countermeasures [8–10].

Signature-based IDS platforms tools are generally accepted as blocking only the lowest level of threat actors, modifying the attacks bypassing the signature detection. Most IDS research focuses on machine learning models to detect intrusions. Popular candidates for this process include artificial neural networks (ANN), random forest classifiers (RFC), and naive Bayes classifiers (NBC) [3]; regression models were not considered, as IDS data are categorical and not continuous. While there are other algorithms capable of classification, those relevant to this work excel at anomaly detection, including isolation forest and distance-finding machine learning algorithms such as support vector machines [11]. Support vector machines provide unique benefits in machine learning, such as easy interpretability [12], and a vast body of literature. Their interpretability and generalised nature [13] make them ideal for IDS platforms, which need to balance security with reliability. IoT devices, broadly speaking, lack the memory to produce models from a multi-gigabyte dataset [14]. There have been multiple approaches to solve this, such as data streaming or federated learning. With adequate data preprocessing and optimisations, SVM-based classifiers are theoretically able to combine good resource usage with the reliability and accuracy that has been shown in research on centralised SVM IDS platforms [15,16].

Even with such tools available, IoT security still faces large issues that are not present in other areas of cyber security. One of the most prominent is the resource restrictions; these make it so that IDS platforms, activity monitors, and firewalls cannot take up too much memory and cannot have too much execution time, as disrupting typical operations is largely unacceptable. IoT devices also have cyber attacks that are totally unique to them such as Mirai, which means that the development of datasets and test sets have to be conducted in parallel with the wider body of cyber security research.

Federated learning (FL) as a category of machine learning, first appeared in 2016 [17]. It outlines a method to securely use data to train machine learning models, a modern often cited example being patient data. Instead of transferring the data, each data centre trains a model, and these are then aggregated. There are two possible methods for this: centralised or decentralised. In the centralised form, a central ‘parent’ node averages the parameters from worker nodes to create a master model, which is then distributed, with the process repeating to iteratively optimise the model. For decentralised methods, the parameters are passed around and tweaked with each model’s dataset. Federated learning is ideal for

IoT platforms, as a comprehensive model can be trained on smaller subsets of the original dataset, overcoming the resource requirements inherent to IoT [18], as shown in Figure 1.

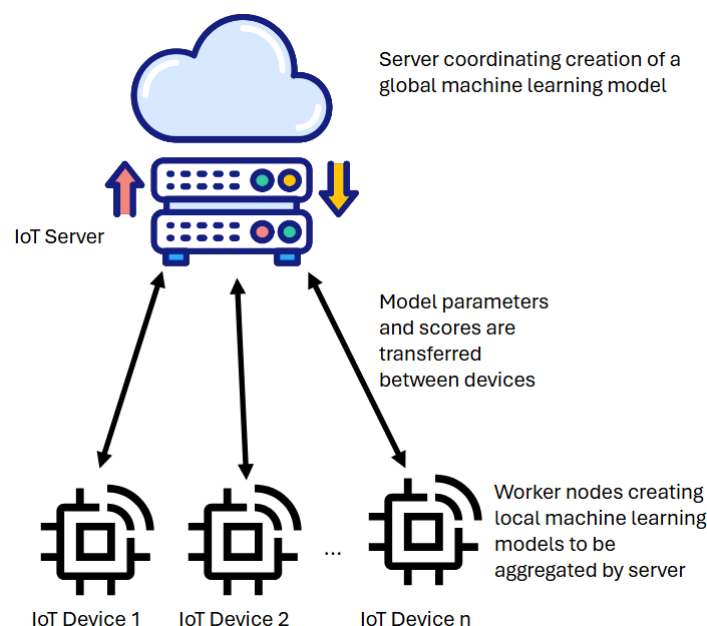


Figure 1. A top-level diagram of federated learning.

As a distance-measuring algorithm, linear SVMs (specifically support vector classifiers (SVCs)) excel at anomaly detection [19], their $O(n \cdot d)$ space and time complexity mean that they are largely not suitable for IoT device training models on data with high-dimension data, due to the sparsity of data in high dimensions and the memory usage required with storing more support vectors. To solve this, we propose using a federated SVM as an IoT IDS to explore machine learning and IoT-specific performance and whether it can overcome the associated limitations. Using the *CIC-IoT2023* [20] dataset, with adequate preprocessing to make it suitable for SVM by reducing the dataset to binary classification, we aim to leverage the strong anomaly-detection capabilities of an SVM to create an efficient DDoS detection system. Federated networks are tested with different amounts of worker nodes to see how the metrics change in order to determine the level at which optimisation can be achieved and to form a comprehensive view of federated SVM network behaviour. The novelties of this research are in the federated SVM, the measurement of physical metrics for multiple federated models, and the application of SVM for an IDS.

The contributions of this paper are as follows:

- We propose an FL-enabled IDS framework for identifying security attacks (i.e., distributed denial of service) in IoT ecosystems.
- We train the proposed framework using several machine learning models and evaluate their performance to find the best-fitting method for security attack recognition in IoT networks.
- We provide a comprehensive outline of the data preparation process required for FL models, with new considerations for data processing in order to maximise the results with non-synthetic data.
- We develop the first federated SVM for IDS research, measuring how effective it may be for attack detection on edge devices.

The remainder of this paper is arranged as follows: Section 2 covers the related work and the current state of the research. Section 3 provides the methodology, experimental setup, and evaluation methods and metrics. Section 4 includes the presentation and

interpretation of the results. Section 5 discusses the results, attempting to justify them, leading into Section 6, which provides the conclusions and future directions.

2. Related Work

Before discussing the methodology it is important to understand the state of the literature relevant to this topic, which encompasses the state of research in IDS, IoT, and federated learning. Some landmark pieces will be analysed, but the main goal is to explore the frontiers of federated learning, where there are less-discussed and -explored models.

2.1. Machine Learning Approaches for IDS

Saranya et al. [21] present an overview of the models used in IDS research, although they do not cover the efficacy of the models within industry. Although it is never stated, they show that other authors report extremely high results across a range of datasets, suggesting either overfitting or a trivial task. They provide a comparison of models, which is useful for demonstrating the performance differences; they do not differentiate between datasets, which reduces the usability of the figures. Different datasets have different flaws; they use KDD99, which is widely accepted to have major flaws [22]. The discussion regarding the implementation framework (such as TensorFlow) is useful when moving forwards with IDS research, as it may affect the results in unpredictable ways. It is a good introduction to the field, and the results explored can be used to compare with other authors; however, it is far too broad to be used as a benchmark or baseline.

Ahmad et al. [23] present similarities to [21] but with some key improvements. They discuss the different models that are widely used in research, along with papers that innovate using these. They do not discuss the drawbacks of the models in detail, which gives them less depth in the survey. They discuss the origins and structure of the datasets, but again, they do not critique them, giving them less depth. The largest improvement over [21] is the differentiation between the datasets and results, giving a much clearer view of how the models perform on different datasets and attack types. One of the insights they present, which provides a good baseline for this research, is that deep learning makes up 60% of IDS research; they do not comment on the black-box nature of these models, which significantly impedes model interpretability, an essential attribute for IDS deployment. Ahmad et al.'s summary identification of research trends and gaps provide meaningful insights; but a more nuanced exploration of the weaknesses of the models outside of just performance metrics would strengthen its contributions.

As mentioned when discussing the work conducted by Ahmad et al. [23] and Saranya et al. [21], the dataset is a very important component. Ahmad et al. come to the conclusion that KDD Cup'99 is the most common IDS dataset. Tavalee et al. [22] discuss this dataset in great depth; their analysis shows that a lot of the impressive results (authors can often report accuracies greater than 0.97) are due to deep flaws in the dataset including redundant records and outdated attacks. They produce the NSL-KDD dataset through heavy data cleaning, but it still contains a lack of coverage for modern attacks. This work's key contributions are twofold: it establishes that a large body of the literature in the IDS field is flawed, and it also establishes that KDD Cup'99 and to a more limited degree NSL-KDD should be avoided within IDS research.

2.2. Federated Learning

Our proposal uses federated learning for an IoT-based IDS; this is an emerging field, first developed in 2016, that still presents numerous research gaps. To understand the landscape, Banahbilah et al. [24] offer an overview of methods commonly used in federated learning and their primary uses. They present the important observation that while

federated learning is much more secure than centralised learning, it is still not inherently secure; IDS platforms require some level of security, as transferred data could still be used for a man-in-the-middle attack, although machine learning parameters are less likely to be useful for this. Banahbilah et al. do not highlight specific research gaps or trends and only summarise the papers; furthermore, they do not discuss the models in detail. It is a good source for finding benchmarks and other authors, but it falls short of summarising the field.

Tamil, Selvi [25] proposes a method of using blockchain in conjunction with federated learning in order to process medical data in vast quantities, while still maintaining the necessary anonymity and data safety. Healthcare informatics was one of the prime areas of interest during the development of federated learning, and articles such as this paint a clear case for sensitive use-cases. It goes hand in hand with Goswami [26], a 2024 paper that aims to utilise federated learning for secure informatics on IoT devices; this provides a nice landscape and proven setup to work with, even if it is far removed from IDS platforms.

Qi et al. [27] present a survey focused on the various methods employed in aggregating federated learning models and parameters, dedicated to discussing the various methods that have been used within the research to aggregate federated learning models and parameters. The most common is FEDAVG, but Qi et al. explore more novel methods. There is a good comparison of performances, network overheads, and resource consumption; however, it fails to mention non-neural models, which are becoming more important in resource-constrained environments or fields where interpretability is important.

The focus of this paper, support vector machines, are uncommon within the federated learning literature, at least in part due to the unintuitive nature of their federation. Navia-Vázquez et al. [28] investigated the federation of support vector classifiers (SVCs) and produce insights into why they have not gained traction in the field, chiefly the computational overheads and the awkward aggregation techniques. They create a model called BDSVM, which obtains >0.99 AUC scores against common benchmark datasets; however, they never discuss whether this is a sign of overfitting. Their results suggest that, if overfitting is not an issue, SVMs could perform competitively with neural networks and ensemble methods in federated settings. However, the practical challenges of federating SVMs underscore why they remain less popular despite their potential.

There is a large overlap between the body of IDS literature and the federated learning literature; authors such as Doriguzzi-Corin, Siracusa [29], and Agrawal et al. [30] have explored this overlap. Agrawal et al. present a survey that describes and discusses the state of federated intrusion detection systems. They discuss at great length information security and draw the conclusion that federated intrusion detection systems provide a new weak point in secure networks, a problem which is not adequately discussed by other authors. Their results from other authors show ubiquitously that federated IDS platforms are performant; this dual nature of their conclusion on both the performance and vulnerability hints at the nuanced nature of federated learning.

The literature presents an optimistic view of both the future of federated learning and of federated IDS platforms. It shows that certain algorithms tend to perform better than others, and there are a large amount of unanswered questions. One of the key gaps noticed was the lack of IoT IDS platforms; there are a few extant studies on the matter, but they are not as well-explored as other areas. Furthermore, there is a large gap in federated learning research regarding support vector machines, which are a model that is commonly used when justifiability is required. As a result, we aim to plug these two gaps, showing both how effective federated IDS platforms can be and how well SVMs can fit into that role.

3. Methodology

The process comprises three stages: preprocessing, development, and evaluation. The preprocessing stage consists of cleaning the dataset to match the requirements and optimisations for SVM. This process involved the selection of a single attack type with the closest balance to benign data and stripping away the rest of the non-benign data. Both Pearson correlation and linear discriminant analysis were used in order to reduce the feature space. Cross validation was performed on the reduced dataset to determine whether any overfitting existed. The general structure of the preprocessing code can be found in Algorithm 1. Figure 2 shows a general overview of the methodology with the proposed network structure.

Algorithm 1 Preprocessing

```

dataset ← raw_dataset
proportion_benign ←  $\frac{\text{total\_benign}}{\text{total\_size}}$                                 ▷ Get attack type
for Number of labels do
    feature_proportions ←  $\{label, \frac{\text{total\_attack}}{\text{total\_size}}\}$ 
end for
attack ← proportion closest to benign
dataset ← attack + benign
pearson_correlation_matrix ← pearson(dataset)
correlated ← empty_array
for features do                                                            ▷ Get the Pearson correlated features
    if pearson_correlation_matrix[i] > 0.2 then
        correlated ← pearson_correlation_matrix[i]
    end if
end for
LDA ← lda(dataset)                                                         ▷ Get LDA features
dataset ← dataset − features not in LDA and Pearson
Label encode (1, 0) labels
if 0.9 < 10-fold cross validation score < 0.97 then                        ▷ Check for overfitting
    Export dataset to csv
end if

```

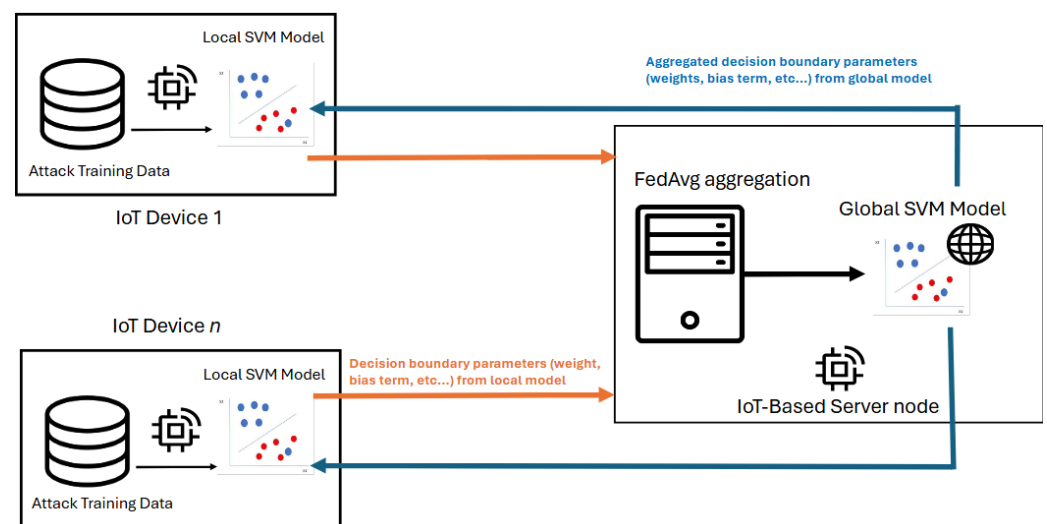


Figure 2. A top-level diagram showing the structure and function of the network and components.

The selected dataset was the CIC-IoT2023 [20], which is publicly accessible via the Canadian Institute of Cybersecurity. It contains data collected by an IoT lab, which was hit with various different cyberattacks. This is a large dataset, about 13 GB with 47 features,

which is considered to be too large for reasonably-sized IoT networks [14]. Models such as SVM scale in complexity with the number of features, making this too large for usage. Furthermore, there are over 46 million records, which is far too many to be used in machine learning training on an IoT device. The reason it is used for IoT attack detection, despite this, is the nature of the attacks; it includes attack families such as Mirai, which are IoT exclusive and represent the newest era of IoT attacks. For this reason, the decision was made to target DDoS attacks only, the removal of which reduced a decent amount of data from the dataset, around 10%. The distribution of classes in the original dataset can be seen in Table 1' the post-removal distribution remained largely the same with random forests and non-federated SVM models having a <0.01 change in machine learning metrics before and after the change. This led us to be confident that an oversampling method such as SMOTEEN was not required. The vast majority of the dataset was denial of service (a large amount of Mirai traffic is DDoS) or benign traffic anyway, and the average change across labels was 2.14%, which was reduced further when all the DDoS types were folded into one label.

Table 1. The distribution of attack types within the dataset. Data provided by CIC [20]. These are further distributed into subcategories within the dataset, but this research focuses on all DDoS attacks.

Data Class	Quantity
DDoS	33,984,560
DoS	8,090,738
Mirai	2,634,124
Benign	1,098,195
Spoofing	486,504
Recon	354,565
Web	24,829
BruteForce	13,064

All of the experiments were run on a workstation equipped with an AMD Ryzen 9 7900X3D CPU and ASUS B650-A motherboard, featuring 24 threads and a 4.9 GHz clock speed. A 6600 MHz DDR5 memory was used on the workstation, and while a powerful GPU was present, none of the experiments were run on the GPU. This processor was chosen to simulate a network, as it can run the worker nodes concurrently, and it has a faster clock speed than can be expected from an IoT device, but the relative difference between the models is the same, so cross-model comparisons are still meaningful. The flower network was centralised, with one server node and every other node being a worker node. All the code was written in python, and all data processing was performed using the numpy and pandas libraries.

We used the flower federated learning framework; it abstracts away the parts of federated learning that are not of interest, while allowing control over the parts that are important. For this research, we let it handle the coordination and communication between worker nodes; this reduced the work to just the parts relevant to the research. For the implementation of federated learning, we had to consider a few key aspects: how the model would be implemented, the federated averaging strategy, and the structure of the network. We decided on a client/server model for federated learning, where each node would send their model, weights, biases, and tuning parameters to a central node, which aggregates them and redistributes them. This maintains information security by never distributing any part of the dataset and is therefore a completely valid way of handling federated learning.

A dataloader was implemented that split the dataset equally into n CSV files, which the worker nodes could read. A lot of federated learning implementations use a central dataloader for this; however, the memory constraints on the development device meant

this was not possible. The partitioning process is not included in the memory consumption or training delay measurements, as it is not a part of a real-world federated learning system. The goal behind this partitioning strategy was to ensure the entire dataset was used to train the global model and therefore provided theoretically maximum coverage. Federated models take a ‘rounds’ approach, where, in each round, specific parameters are tuned to produce the optimal value; we chose for the regularisation parameter (C) to be optimised every round. This parameter determines the importance of misclassification, and the optimal value varies heavily depending on the dataset. The only other parameter that was affected was the iteration cap, which aims to tune parameters. A linear kernel was used for the SVM instead of a possibly more effective radial basis function (RBF). This was chosen, as the linear kernel has the smallest memory footprint of any of the kernels, as well as the lowest amount of computations required, which keeps the requirements and training delay within reason for IoT devices. The train/test split was performed after the dataloader, and we used a 80% training proportion with no validation set.

The model parameters for the SVM can be found in Algorithm 2. All of the models were federated with the FedAvg function, and the parameters for the other models are listed in Table 2.

Algorithm 2 General Process

```

function DATALOADER(id, seed) ▷ Dataloaders are not used in real-world applications
  Get dataset csv
  Shuffle with seed
  Partition into n chunks
  chosen ← idchunk return chosen
end function
Require: seed is constant across all nodes
Require: each node has unique ID int
  startParameters ←  $C = 1$ , Kernel = linear, max_iter = 50    ▷ RBF has not been tested
  for rounds do
    Each node runs DATALOADER and receives their part of data
    model ← LinearSVM
    Train model with data from dataloader
    Evaluate model with data from dataloader
    Tune  $C$  value
    Return parameters and evaluations
    Run FEDAVG
  end for
  finalParameters ← Parameters from nodes
  finalMetrics ← Metrics from nodes
  
```

Table 2. The parameters for the other models used in the tests. Where not specified, parameters are blank.

Model	Parameters
Artificial Neural Network	13 input nodes, 8 hidden layer one, 4 hidden layer two, 1 output. LeakyReLU activation and He uniform initialisation.
Random Forest	50 trees, 40 max depth, 2 minimum samples per split.
Isolation Forest	0.45 C factor, 100 trees

The most common use for federated learning is with ensemble models such as random forest or network models such as neural networks, as these have easily ‘stackable’ properties that mean one worker node can produce a part of the parent model, with simple aggregation at the end. SVM does not have these properties, so the method for aggregating is much less intuitive. A linear SVC produces a line with the equation $y = m \cdot x + c$, which is a simple straight line; however, it produces it with a different equation $w \cdot x + b = 0$, where w is a vector normal to the hyperplane, and b is an offset. The parameters w and b can be averaged each round to produce the parent model, in much the same way that weights and biases are averaged for neural networks; the pseudocode for this can be found in Algorithm 3. This is the only deviation from the generic FedAvg function that is commonly used in federated learning. This approach allows IoT devices to leverage the anomaly detection capabilities of SVM, while hopefully offsetting the heavy performance costs that come with it.

Algorithm 3 Top-level FedAvg algorithm

```

parameters ← [weight, bias]
for rounds do
  Each node ← parameters
  Each node trains
  weight_array ← weight FROM ALL nodes
  bias_array ← bias FROM ALL nodes
  weight ←  $\frac{\sum weight_i}{N}$                                 ▷ Average w, b
  bias ←  $\frac{\sum bias_i}{N}$ 
  parameters ← [weight, bias]
end for
  
```

When the worker nodes acquired the data, they were fed their starting parameters, which are listed below. The specific version of SVM used was a linear SVC, as it both excels at classification and keeps resource usage lower than other SVM kernels. After n rounds, the parent model was tested, and the metrics were collected and displayed; this gave a reading of the effectiveness of the models. The SVM was given a maximum number of optimisation iterations, which kept it within a reasonable timeframe, which came at the cost of ML performance. The starting parameters were regularisation (C) at 1, a maximum of 1000 iterations (otherwise the code took days), and a random state of 42.

The models’ performance was measured with four machine learning metrics: accuracy, precision, recall, and F1-score. An IDS needs to be well-rounded, as false intrusion detections can have serious negative consequences, and these metrics provide a good indication of how suitable the models are for an IDS. The accuracy is defined by Equation (1) and gives an indication into the overall reliability of the models. A high accuracy means most of the predictions are correct, and the model is probably useful.

$$Accuracy = \frac{\text{Correct predictions}}{\text{Total predictions}} \quad (1)$$

The precision is defined in Equation (2) and shows the rate at which positive predictions are correct. This is useful in an IDS as false positives come with a high cost.

$$Precision = \frac{\text{True positives}}{\text{True positives} + \text{False positives}} \quad (2)$$

The recall is defined by Equation (3) and indicates how effective a model is at classifying positive samples. For an IDS, this indicates what proportion of intrusions are detected, making this a very important metric to track.

$$\text{Recall} = \frac{\text{True positives}}{\text{True positives} + \text{False negative}} \quad (3)$$

The F1-score, found in Equation (4), is the harmonic mean of the precision and recall, showing the balance between the two. It has already been established that high precision and recall are vital for an effective IDS; so, this is another way to show these aspects.

$$F_1 = 2 \left(\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \right) \quad (4)$$

In order to rule out overfitting, stratified five-fold cross validation was used; if the mean accuracy score was as high as the individual results, it suggested there was no overfitting. We found an average score of 0.987 for the centralised model, suggesting the model was very well fitted and not overfitted.

The cost of the model in this case is considered either by the memory used per worker node or by the training delay. As stated in the Introduction, IoT devices are often memory-constrained, meaning keeping the memory low is paramount. Equally true is that a lot of IoT devices have batteries or limited power sources; so, the training delay needs to be kept as low as possible. A perfect classifier would not be suitable if it could not run on IoT devices, and therefore both types of measurement were considered.

4. Results

We decided to evaluate the performance along two lines: machine learning performance and physical performance. IoT devices are often subject to resource restrictions that other devices are not. We decided that an extra type of measurement was needed. The tests collected standard metrics, in this case, accuracy, precision, recall, and the F1-score, as well as the running time and peak memory usage per node. These were compared against benchmark federated models: random forest, isolation forest, and artificial neural network. Each benchmark model was tested with five worker nodes and compared to the SVM with five worker nodes. This number created a balance between performance and deep federation. The final dataset had a 51:49 ratio of benign to DDoS traffic, which means that the dataset was effectively balanced for classification purposes.

The RF and ANN were chosen as benchmark models because they make up the majority of federated learning research. Isolation forest was chosen, as it is designed with applications such as IDS in mind, working well in nearly any anomaly detection scenario. ANNs are not applicable to federated systems, however, having no justifiability [31], which is essential when making decisions, which can cost money or safety.

4.1. Federated vs. Non-Federated

In order to maintain fairness between the tests, the non-federated SVM model was given the same parameters and the same iteration cap as the federated versions.

Table 3 shows the steady performance decreases as the node count rises, due to the dataset becoming more fractured. The jump from three nodes to five nodes was much larger than five to ten, suggesting that it had some diminishing effects. It also showed excellent performance metrics, which was expected, as SVM is ideal for anomaly detection, as detailed in Figure 3.

Table 3. A comparison of SVM metrics with different sized federated networks, beginning at a non-federated model. Numbers within parenthesis indicate the number of worker nodes within the federated model.

Model	Accuracy	Precision	Recall	F1Score
SVM	0.991	0.979	0.993	0.981
SVM (3)	0.981	0.979	0.988	0.977
SVM (5)	0.974	0.976	0.975	0.973
SVM (10)	0.973	0.974	0.973	0.969

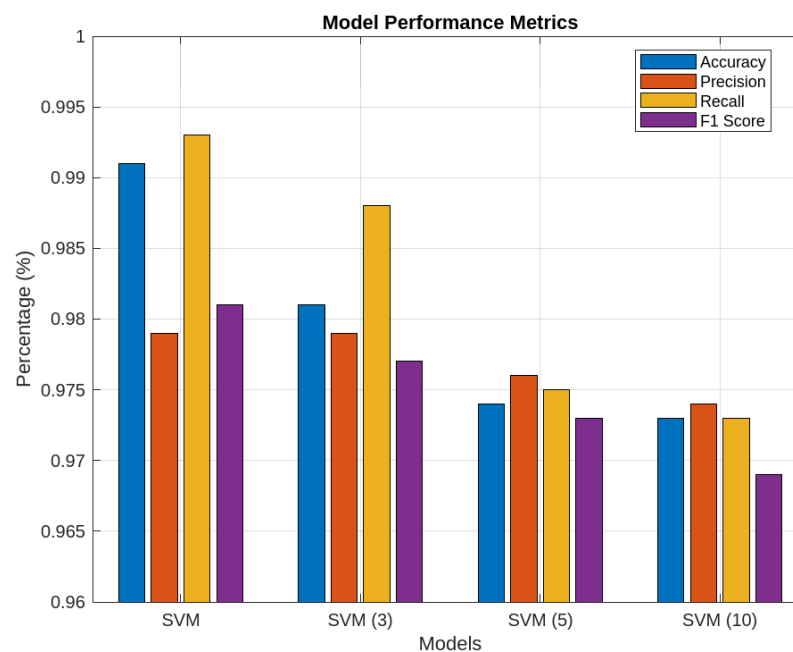


Figure 3. Performance of SVM across different numbers of worker nodes.

4.2. SVM vs. Other Models

The models in Table 4 were all trained on a network size of five worker nodes, with three rounds of federated learning. The five worker node network size was chosen, as it is large enough to provide meaningful federation but small enough to keep overall memory usage within reasonable levels for all models. They all had the same dataset, with dual-class classification and label encoding. Isolation forest had incomplete metrics, due to the fact that it is not a classifier and could not be treated as such; it also treated benign data as the anomalous class, due to its minority state within the dataset. They were all federated with flower, with general optimisation made in order to ensure that they performed as well as possible.

Table 4. SVM compared with other federated models.

Model	Accuracy	Precision	Recall	F1-Score
SVM (5)	0.974	0.976	0.975	0.981
IF (5)	0.732	0.741	0.741	0.742
RF (5)	0.981	0.982	0.982	0.981
ANN (5)	0.974	0.983	0.975	0.975

Table 4 shows that SVM fits in nicely with the top range of models, with the slight variances between ANN, RF, and SVM being explainable by quirks in the dataset and other minor things. Isolation forest performed poorly, as the data balance was roughly equal,

with a 51% balance of benign data; as an anomaly detection model, isolation forest excels at minority class detection and therefore underperformed here. Figure 4 shows this; it is notable that the poor performance of isolation forest skews the entire graph, highlighting the similar results of SVM, random forest, and ANN.

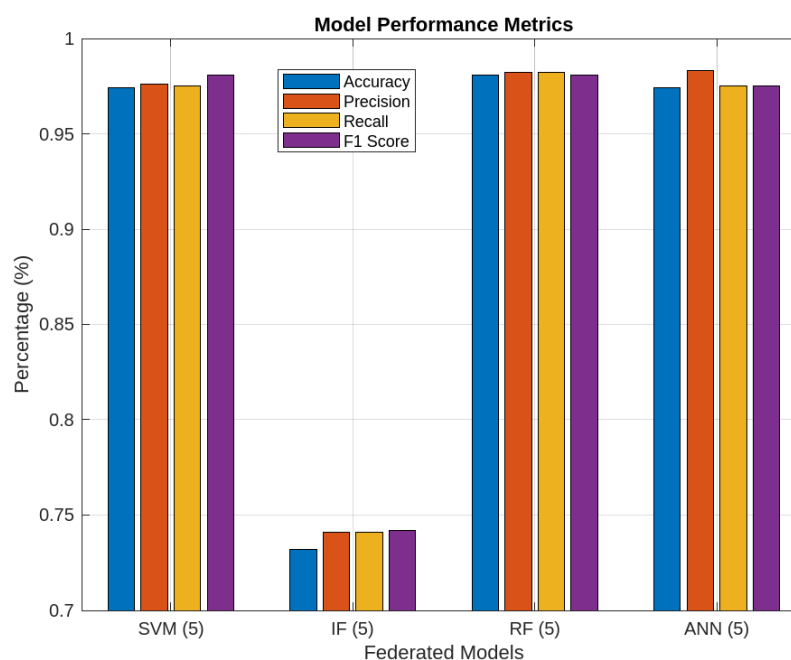


Figure 4. A comparison of federated models with a worker count of five.

4.3. Physical Metrics

Physical metrics constitute the metrics that are used to measure the practicality of an otherwise good model. These are delay and memory usage; the purpose of testing these is to see how prohibitive the resource constraints are. The total delay is the elapsed time required for a federated model to conclude training, testing, and evaluation; this is platform dependent, so it was compared to the total delay of a centralised SVM. The peak memory usage is the highest amount of memory used by the federated platform: this metric is not very useful, as it will continue to grow; so, it was accompanied by the peak memory usage per node. The other models chosen broadly have better complexities, with ANN having a linear memory complexity and isolation forest being sublinear. The idea was that, by using federated learning, it would be possible to reduce the resource cost of SVM to a practical amount for IDS applications. The time was recorded, but this will be subjective to the device being tested; as such, the metric focused on was the ratio to the centralised SVM.

Table 5 shows some interesting trends in both time and memory usage. The first quirk is that the delay seems to rise before it eventually falls below the centralised models, which is due to the fact that federated learning takes place in rounds, meaning that each node must train n (in this case, three) models. The overhead introduced by the federation was deemed to be minimal, especially when compared to the machine learning delay. The memory usage diminished in a non-linear way, which makes sense due to the $O(n \cdot d)$ time complexity of SVM. The memory usage per node was significantly higher than with the other models, nearly double the forest algorithms. Figure 5 shows the memory per node as a bar chart, showing the rapid descent of SVM and the naturally lower values of other models. Figure 6 shows how SVM's memory consumption per node dropped, which provides a very useful visual insight into the effects of federated learning, with clearly visible diminishing returns. SVM had the second best training delay performance, which corresponded with the power consumption, a critical factor for IoT models. When paired

with the fact that SVM had the best F1-score (shown in Figure 4), it became the model with the best training delay-to-performance balance.

Table 5. SVM compared with other federated models.

Model	Total Delay (s)	Ratio to SVM	Peak Memory Per Node
SVM (C)	32	1	2919
SVM (3)	46	1.44	1168
SVM (5)	34	1.06	819
SVM (10)	27	0.843	542
ANN (5)	41	1.28	613
RF (5)	201	6.28	418
IF (5)	6	0.19	410

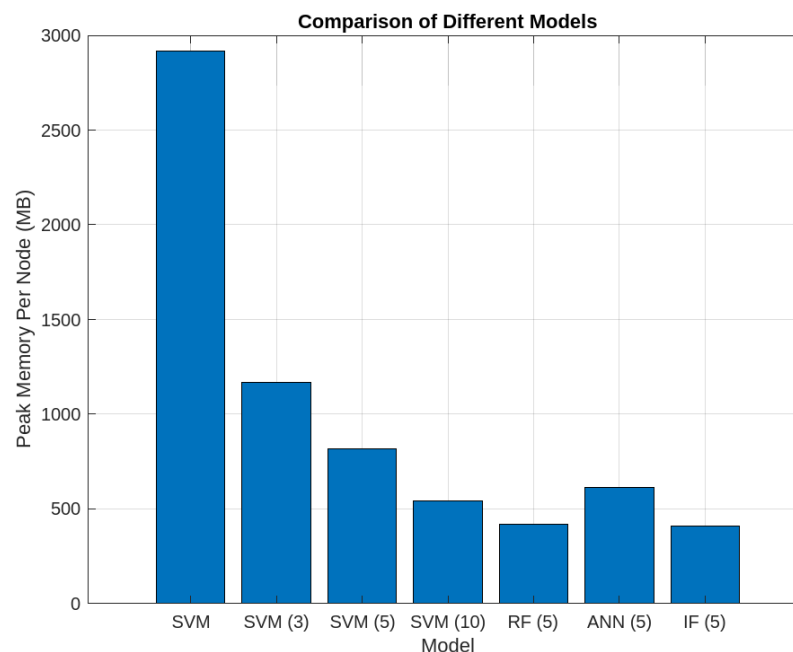


Figure 5. Peak memory usage per node across different federated models.

Figure 7 shows the time ratio of each model to a centralised SVM model. It shows how, even with the rounds used in federation, it hugely improves the overall execution time. It clearly shows the excellent performance of isolation forest and the very poor performance of random forest. The decrease in time for SVM is visible, showing that, even with multiple rounds of learning, it becomes approximately the same speed as its centralised self.

In each round of communication, the models send and receive a dictionary of parameters. In our implementation, this dictionary was always five parameters or less; due to a quirk of python, this means that it was always 240 bytes. This occurred twice per round, meaning the base network overhead per worker node was $480 \cdot R$. The model, network rules, network configuration, and flower configuration will all change this, and this does not account for the parent node, which has a base network overhead of $480 \cdot RN$. The IoT has a broad range of bandwidths, with no single rule dictating what is an acceptable upper limit. Less than a megabyte seems acceptable, especially with IoT devices growing in power and resources.

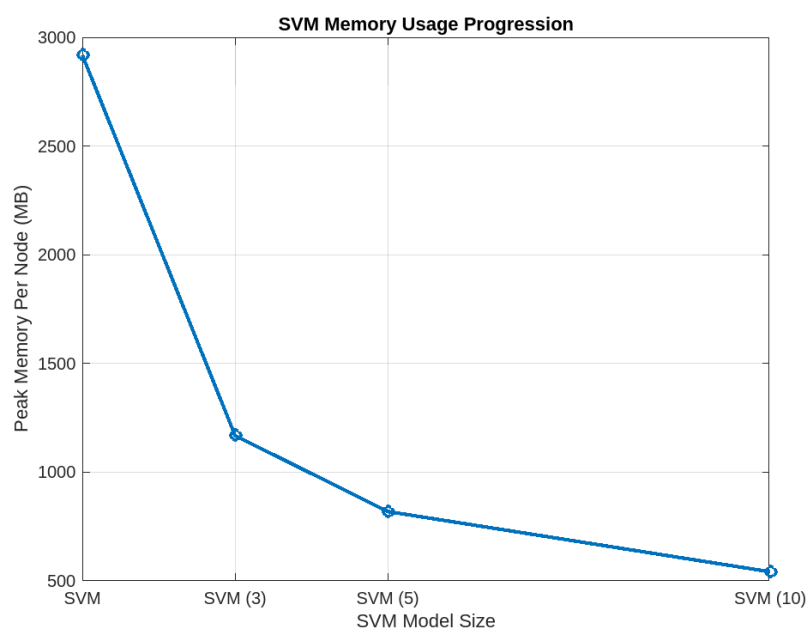


Figure 6. Peak memory usage per node across different federated models.

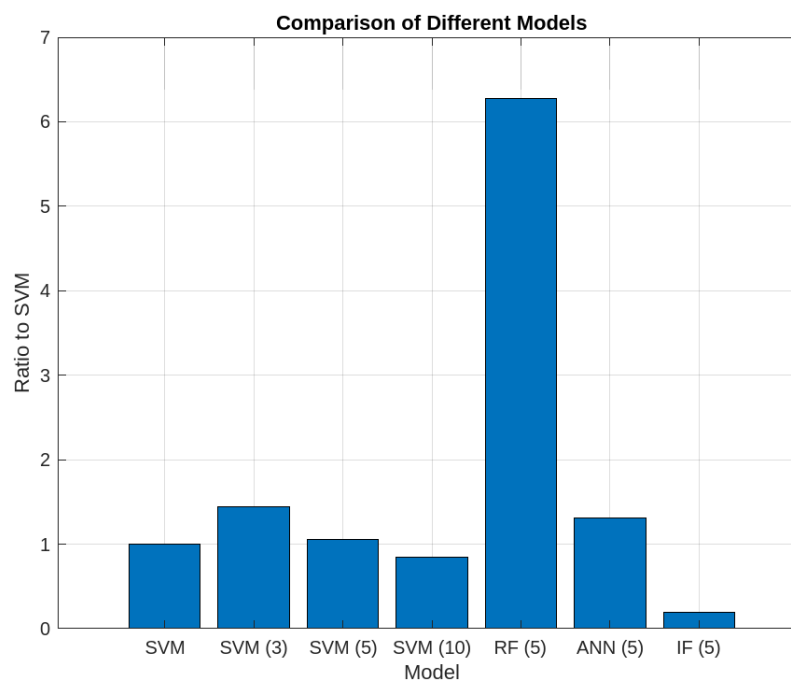


Figure 7. The progression of training delay as network size increases.

Figure 8 shows the decrease in execution time as the federated network becomes larger, which shows that it takes a fairly sizeable network in order to undercut a centralised SVM; however, it is important to remember that federated learning uses a rounds system, so each node is creating multiple SVMs here.

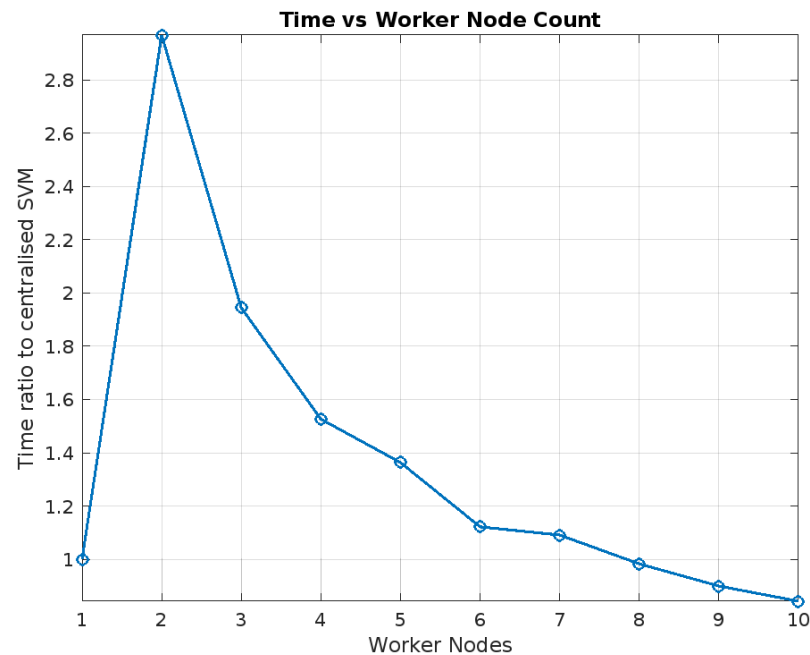


Figure 8. The decrease in time as the worker count increases.

5. Discussion

The first point to address is how the results appear to be overfitted. The metric tables contain accuracies, precisions, and F1-scores in excess of 0.95; this is conventionally what would be considered overfitting, as machine learning models should not be able to consistently predict with such accuracy when the dataset is so complex. We checked the accuracy and AUC-ROC within 10-fold stratified cross validation and found no evidence of overfitting. We found that the metrics per class were similar; so, the data balance was not causing classification issues. The models were just well-fitted. While the results of cross-fold validation suggest the models were not overfitting, this is also supported by other authors, who report extremely high accuracy with SVM models [32,33].

The SVM takes to federation very well, with a marginal decrease in performance when moving from non-federated to federated in a three-node network; this is best seen by how little visual change there is in Figure 3. Table 3 shows this change; it also shows a steady but slow decrease as the network size increases. This is to be expected and falls in line both with other models and with the hypothesis in the beginning. It also compares very well with the other federated models, with very respectable machine learning metrics, comparing well with other models. For each model cross validation, dataset balancing, and metric reporting per class was conducted to ensure that there was no overfitting. Figure 4 shows that isolation forest performed very poorly, which was due to the close balance of the dataset; designated anomaly finding algorithms perform best with strong data imbalances, and the (mostly) equal nature causes the weak showing.

Figure 7 shows random forest performed very poorly, which was because it had to process its entire dataset for every tree that the model created. This means that the number of estimators could be changed, tuning it down to smaller amounts of time; however, this strongly hurts the performance. Finding the right balance is application-specific, and as random forest is not the focus of this research, it falls outside the scope. However, Figure 7 also shows isolation forest performed exceptionally well, due to the fact that it operated entirely within the data space and used the path length in order to detect anomalies. Figure 5 also suggests that it was extremely efficient and would be a promising avenue of future research if the dataset was better geared towards it. Isolation forest's results have

less real-world importance than the other models due to the dataset's construction; the dataset was explicitly geared towards support vector models, which prefer balanced data.

Due to the smaller feature space of the data, it was faster than all of the other models except for isolation forest (which was extremely fast). It suffered badly in terms of memory, nearly doubling all of the other models. This was due to the high dimensionality of the data; the hyperplane had to be calculated in all of these, which resulted in a lot of bloat. IoT devices do not often have a gigabyte of free memory, and therefore SVM seems infeasible unless the network is large enough to ensure that the memory requirement per node is small.

Overall, SVM is still nascent for federated IoT IDS research; it has too many hardware limitations for now, even after optimising the dataset to its fullest extent. It is both possible and likely that further model optimisations can be made and that it may be very strong in different scenarios; therefore, we are confident in saying that this is not the end for federated SVM models. The other models tested, while not the primary focus, still provide useful insights. Random forest and ANN are both extremely effective in a scenario where one may use a SVM; however, random forest comes with severe time penalties. ANN models seem fine, but their black-box nature makes them unsuitable for security-critical applications, where downtime and revenue loss could come from poor decisions that cannot be reviewed by a human.

The main scenario in which SVM would be very useful as an IoT IDS is with higher memory devices, as its high F1-score suggests it is a very robust model, more robust than the other models tested, and the lower training delay than most of the other models means it will consume less power or battery life.

6. Conclusions

The tests created a comprehensive view of how federated SVM models may perform on IoT networks; there are two considerations that come with these results: the application and the data. It is possible that a federated SVM may perform very differently when used elsewhere, with lower-dimensional data shrinking the memory usage of the model. Equally so, it is possible that if it had been given a different IDS dataset then we could see vastly different results. The results collected are a snapshot of the efficacy for IDS research only; they point to it being effective but impractical. This is not an indictment against all federated SVM models, but it is a definitive statement that they do not have a place in IoT IDS research, and it is not a promising avenue for further exploration.

Our results (Figures 3 and 4) show fine machine learning metrics (accuracy, precision, F1-score, and recall), with Figures 5 and 7 showing very poor memory; it is poor enough relative to the other models that they are unsuitable for use within industry without device considerations. We conclude that they have limited use; they are the best model as long as memory is not a consideration, which it often is when IoT is the application. They can be a convenient worst-case scenario for memory usage when benchmarking other federated models. This is not to say that SVMs do not have a future in research, as with low-dimensional data, the IoT metrics become a lot more forgiving, and their interpretable quick nature makes them excellent candidates. As they are interpretable and generalisable, they are superior for security-critical work compared with ANN, and they have better power consumption and robustness than random forests. For this kind of high dimensional data, it is likely that a choice such as logistic regression would be more device-effective, but further feature engineering could make SVM a viable choice. However, extensive proof would be needed to demonstrate that any further feature reduction did not affect the integrity of the IDS' ability to detect cyber attacks.

Three main areas of future work were identified regarding SVM during this research. With such a new emerging field, we chose only the most prominent and interesting areas of research. Firstly, we used a linear kernel, due to the simplicity of averaging the hyperplane variables; however, it is certainly worth investigating the use of RBF kernels in federated learning. Secondly, IDS is not the only platform that federated SVMs could be used in; other applications may yield better physical metrics due to the nature of their datasets, which needs to be explored further before determining the effectiveness of SVM as an IoT model. Lastly, we only explored horizontal partitioning; it is worth seeing the potential effects of vertical partitioning. It is likely that federated transfer learning could reduce the physical metrics into acceptable ranges, leading to useable SVMs.

Author Contributions: Conceptualisation, M.D., S.P.A., M.A.-K. and Y.J.; methodology, M.D., S.P.A., M.A.-K. and Y.J.; software, M.D.; validation, M.D., S.P.A., M.A.-K. and Y.J.; formal analysis, M.D.; investigation, M.D.; resources, M.D.; data curation, M.D.; writing—original draft preparation, M.D.; writing—review and editing, M.D., S.P.A., M.A.-K. and Y.J.; visualisation, M.D.; supervision, S.P.A., M.A.-K. and Y.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available at <https://www.unb.ca/cic/datasets/iotdataset-2023.html>, accessed on 7 March 2025. Experimental results available upon request.

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

Abbreviations

The following abbreviations are used in this manuscript:

IDS	Intrusion Detection System
ANN	Artificial Neural Network
SVM	Support Vector Machine
RF	Random Forest
IF	Isolation Forest
ML	Machine Learning
CIC	Canadian Institute of Cybersecurity
LDA	Linear Discriminant Analysis

References

1. Granato, A.; Polacek, A. The growth and challenges of cyber insurance. *Chic. Fed Lett.* **2019**, *426*, 1–6. [[CrossRef](#)]
2. Iouliaou, P.; Vasilakis, V.; Moscholios, I.; Logothetis, M. A signature-based intrusion detection system for the internet of things. *Inf. Commun. Technol. Form* **2018**, *in press*.
3. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [[CrossRef](#)]
4. Kizza, J.M. Internet of Things (iot): Growth, Challenges, and Security. In *Guide to Computer Network Security*; Springer: Berlin/Heidelberg, Germany, **2024**; pp. 557–573.
5. Mayoral-Vilches, V.; Carbajo, U.A.; Gil-Uriarte, E. Industrial Robot Ransomware: Akerbeltz. In Proceedings of the 2020 fourth IEEE International Conference on Robotic Computing (IRC), Virtual, 9–11 November 2020; IEEE: Piscataway, NJ, USA, **2020**; pp. 432–435.
6. Pei Breivold, H. Towards factories of the future: Migration of industrial legacy automation systems in the cloud computing and Internet-of-things context. *Enterp. Inf. Syst.* **2020**, *14*, 542–562. [[CrossRef](#)]
7. Clark, G.W.; Doran, M.V.; Andel, T.R. Cybersecurity issues in robotics. In Proceedings of the 2017 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA), Savannah, GA, USA, 27–31 March 2017; pp. 1–5. [[CrossRef](#)]

8. Xiao, L.; Wan, X.; Lu, X.; Zhang, Y.; Wu, D. IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security? *IEEE Signal Process. Mag.* **2018**, *35*, 41–49. [\[CrossRef\]](#)
9. Nawir, M.; Amir, A.; Yaakob, N.; Lynn, O.B. Internet of Things (IoT): Taxonomy of security attacks. In Proceedings of the 2016 3rd International Conference on Electronic Design (ICED), Phuket, Thailand, 11–12 August 2016; pp. 321–326. [\[CrossRef\]](#)
10. Butun, I.; Morgera, S.D.; Sankar, R. A Survey of Intrusion Detection Systems in Wireless Sensor Networks. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 266–282. [\[CrossRef\]](#)
11. Kent, M.; Huynh, N.K.; Schiavon, S.; Selkowitz, S. Using support vector machine to detect desk illuminance sensor blockage for closed-loop daylight harvesting. *Energy Build.* **2022**, *274*, 112443. [\[CrossRef\]](#)
12. Salazar, S.; Denton, S.; Salleb-Aouissi, A. Counterfactual Explanations for Support Vector Machine Models. *arXiv* **2022**, arXiv:2212.07432.
13. Cristianini, N.; Shawe-Taylor, J. Generalisation Theory. In *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*; Cambridge University Press: Cambridge, UK, 2000; pp. 52–78.
14. Hussain, F.; Hussain, R.; Hassan, S.A.; Hossain, E. Machine learning in IoT security: Current solutions and future challenges. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1686–1721. [\[CrossRef\]](#)
15. Azimjonov, J.; Kim, T. Designing accurate lightweight intrusion detection systems for IoT networks using fine-tuned linear SVM and feature selectors. *Comput. Secur.* **2024**, *137*, 103598. [\[CrossRef\]](#)
16. Alghushairy, O.; Alsini, R.; Alhassan, Z.; Alshdadi, A.A.; Banjar, A.; Yafoz, A.; Ma, X. An efficient support vector machine algorithm based network outlier detection system. *IEEE Access* **2024**, *12*, 24428–24441. [\[CrossRef\]](#)
17. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the Artificial Intelligence and Statistics, PMLR, Fort Lauderdale, FL, USA, 20–22 April 2017; pp. 1273–1282.
18. Imteaj, A.; Thakker, U.; Wang, S.; Li, J.; Amini, M.H. A survey on federated learning for resource-constrained IoT devices. *IEEE Internet Things J.* **2021**, *9*, 1–24. [\[CrossRef\]](#)
19. Hosseinzadeh, M.; Rahmani, A.M.; Vo, B.; Bidaki, M.; Masdari, M.; Zangakani, M. Improving security using SVM-based anomaly detection: Issues and challenges. *Soft Comput.* **2020**, *25*, 3195–3223. [\[CrossRef\]](#)
20. Neto, E.C.P.; Dadkhah, S.; Ferreira, R.; Zohourian, A.; Lu, R.; Ghorbani, A.A. CICIOT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. *Sensors* **2023**, *23*, 5941. [\[CrossRef\]](#)
21. Saranya, T.; Sridevi, S.; Deisy, C.; Chung, T.D.; Khan, M. Performance Analysis of Machine Learning Algorithms in Intrusion Detection System: A Review. *Procedia Comput. Sci.* **2020**, *171*, 1251–1260.
22. Tavallae, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6. [\[CrossRef\]](#)
23. Ahmad, Z.; Shahid Khan, A.; Wai Shiang, C.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4150. [\[CrossRef\]](#)
24. Banabilah, S.; Aloqaily, M.; Alsayed, E.; Malik, N.; Jararweh, Y. Federated learning review: Fundamentals, enabling technologies, and future applications. *Inf. Process. Manag.* **2022**, *59*, 103061. [\[CrossRef\]](#)
25. Tamil Selvi, K.; Thamilselvan, R. Privacy-Preserving Healthcare Informatics Using Federated Learning and Blockchain. In *Healthcare 4.0*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2022; pp. 1–26. [\[CrossRef\]](#)
26. Goswami, S.A.; Dave, S.; Patel, K.C. Healthcare Informatics Security Issues and Solutions Using Federated Learning. In *Federated Learning for Smart Communication Using IoT Application*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2024; pp. 124–154. [\[CrossRef\]](#)
27. Qi, P.; Chiaro, D.; Guzzo, A.; Ianni, M.; Fortino, G.; Piccialli, F. Model aggregation techniques in federated learning: A comprehensive survey. *Future Gener. Comput. Syst.* **2024**, *150*, 272–293. [\[CrossRef\]](#)
28. Navia-Vázquez, A.; Díaz-Morales, R.; Fernández-Díaz, M. Budget Distributed Support Vector Machine for Non-ID Federated Learning Scenarios. *ACM Trans. Intell. Syst. Technol.* **2022**, *13*, 1–25. [\[CrossRef\]](#)
29. Doriguzzi-Corin, R.; Siracusa, D. FLAD: Adaptive Federated Learning for DDoS attack detection. *Comput. Secur.* **2024**, *137*, 103597. [\[CrossRef\]](#)
30. Agrawal, S.; Sarkar, S.; Aouedi, O.; Yenduri, G.; Piamrat, K.; Alazab, M.; Bhattacharya, S.; Maddikunta, P.K.R.; Gadekallu, T.R. Federated Learning for intrusion detection system: Concepts, challenges and future directions. *Comput. Commun.* **2022**, *195*, 346–361. [\[CrossRef\]](#)
31. Lones, M.A. How to avoid machine learning pitfalls: A guide for academic researchers. *arXiv* **2021**, arXiv:2108.02497.

32. Bhati, B.; Rai, C. Analysis of Support Vector Machine-based Intrusion Detection Techniques. *Arab. J. Sci. Eng.* **2019**, *45*, 2371–2383. [[CrossRef](#)]
33. Alkasassbeh, M. An empirical evaluation for the intrusion detection features based on machine learning and feature selection methods. *arXiv* **2017**, arXiv:1712.09623.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.