POLITECNICO DI TORINO

Master's Degree in Communications engineering

Master's Degree Thesis

Intrusion Detection Systems in Vehicle Controller Area Networks via Hardware Performance Counters

Supervisors

Prof. STEFANO DI CARLO

Prof. ALESSANDRO SAVINO

Prof. FRANCO OBERTI

Candidate

Gaspard MICHEL

April 2024

Summary

In the domain of automotive embedded systems, the Controller Area Network (CAN) protocol stands as a crucial communication mechanism. Originally designed in the 1980s, its architecture is robust but exhibits important vulnerabilities in the face of the complexity and the advent of autonomous vehicles, broadening the potential for cyber-attacks. Addressing the inherent security deficiencies has become imperative. Particularly, its lack of native attack mitigation features.

Traditional security improvements, such as payload encryption and message authentication, offer partial solutions. However, this thesis experiments with a novel approach: implementing an Intrusion Detection System (IDS) specifically adapted for the CAN environment. Hardware Performance Counters (HPCs) inherently monitor and signal hardware event occurrences. Our proposed IDS has as its purpose to detect anomalous activities indicative of potential cyber threats on the CAN bus.

This research is grounded in the simulation of a CAN receiver on a RISC-V architecture using the Gem5 simulator. It focuses on the processing of CAN frame payloads through standard operations known to trigger HPC responses, like convolution operations and AES-128 encryption and decryption.

The methodology relates to the extraction of HPC data post-simulation, followed by a rigorous selection process to identify pertinent counters. Aiming to refine the dataset for enhanced classifier efficiency, initial transformations standardize the HPC data, succeeded by correlation analysis to reduce the feature set.

Subsequently, the study evaluates various classification algorithms and their parameters, ranging from binary to multiclass, to find the most effective to distinguish benign and malicious activities.

This thesis contributes a novel perspective on CAN protocol security, advocating for a dynamic IDS framework that exploits the predictive capacity of HPCs within a vehicular context. Finding alternatives to traditional security measures helps to develop a more resilient automotive communication infrastructure against evolving cyber threats.

Table of Contents

List of Tables

List of Figures

Acronyms

CAN
Controller Area Network

HPC
Hardware Performance Counter

IDS
Intrusion Detection System

ECU
Electronic Control Unit

CAN L
CAN Low

CAN H
CAN High

SOF
Start of Frame

ID
Identifier

RTR
Remote Transmission Request

DLC
Data Length Code

CRC

Cyclic Redundancy Code

ACK

Acknowledgment

EOF

End of Frame

CSMA/CR

Carrier Sense Multiple Access, Collision Resolution

TEC

Transmit Error Count

REC

Receiver Error Count

DOS

Denial of Service

PMU

Performance Monitoring Unit

TLB

Translation Lookaside Buffer

U-mode

User mode

S-mode

Supervisor mode

M-mode

Machine mode

CSR

Control and Status Resgisters

IPS

Intrusion Prevention System

SoC

System on Chip

SE

Syscall Emulation

FS

Full System

RTOS

Real-Time Operating System

AES

Advanced Encryption Standard

SVC

Support Vector Classification

TP

True Positive

FP

False Positive

FN

False Negative

Chapter 1

Introduction

With the rise of electronic devices and automation in vehicles, a communication protocol standard was developed in the 1980's: the Controller Area Network (CAN). As the autonomous car and the entertainment devices have proliferated, the usage of CAN grew significantly and became a central part of the vehicles information system.

This protocol offers a large set of error management mechanisms, ensuring reliable operation even in an electromagnetically noisy environment. Additionally, the CAN standard is characterized by its simplicity; it can run on low-performance devices and functions at quite low bitrate.

However, it does not include built-in security measures, leaving the protocol vulnerable to exploitation for malicious purposes such as sending unauthorized traffic or disrupting access to the channel for legitimate devices. These inherent vulnerabilities needs to be patched or additional security measures must be implemented to prevent potential attacks. Unauthorized access to the vehicle via the CAN network is particularly critical due to its dangerousness and its high value. Indeed, complete control over CAN devices may lead to the theft of the car or the malfunctioning to vital components, such as brakes or motor controller.

Various security mitigations can be set up to overcome to the weaknesses of the protocol. Encryption algorithms and signatures could efficiently ensure confidentiality and authenticity of the CAN frames. However, it's important to note that additional processes are required to achieve comprehensive security, and encryption and signatures alone do not address the underlying flaws of CAN or mitigate control of malicious traffic. Consequently, another solution has been retained; an IDS will be implemented. This program is designed to detect intrusions regardless the type of attack.

Various machine learning models have been tested to implement the IDS. These models fall into two main groups: the dual-classes and the multiclasses. The first ones are able to differentiate a normal traffic and a traffic under one specific type

of attack unlike the latter which have the capacity to classify normal traffic and various kinds of intrusion. Within these categories, different models were studied : the One-class classifier, an unsupervised model, and the SVC and the Random forest classifiers, supervised models.

The IDS relies on metrics that differ between normal traffic context and when the device is under-attack. In this study, the metrics chosen are the HPCs which are counters that increments when specific hardware events occur. The HPCs offer the advantage of requiring a high level of privilege to be modified, making them very robust against attacks.

Chapter 2 outlines the Theoretical background required for the study, particularly security characteristics of CAN, the HPC in RISC-V and the theoretical description of the IDS.

Chapter 3 describes the different tools which have been tested during the study even if they were not used for the final IDS. These tools are the parts of the CAN controller on which the HPCs have been taken.

Chapter 4 depicts the classifiers, the functions and the parameters used in the IDS. It presents also the results for each situation.

Finally, chapter 5 concludes this study by discussing the developed system and showing the improvements which can be integrated in the future.

Chapter 2

Theoretical background

2.1 CAN

2.1.1 History of the protocol

The Controller Area Network (CAN) protocol was developed by Bosch in 1986. It serves as a serial bus protocol specifically designed for the transportation sector and has become the standard protocol for passenger cars. This protocol facilitates communication among various devices known as Electronic Control Units (ECUs), achieving connectivity through a two-wire configuration forming the CAN bus.

The CAN protocol is designed with resistance to electromagnetic disturbances, enabling robust performance. It also supports hotplugging, allowing devices to be connected or disconnected without disrupting the network. Additionally, the protocol incorporates efficient error detection mechanisms within the frame using different control mechanisms. CAN permits bit-rates of up to 1 Mb/s.

Other versions of the CAN protocol were introduced in the 1990s and 2000s to enhance its capabilities. These versions introduced features such as an extended frame format, accommodating 29-bit long identifiers, and elevating the potential bit rates.

2.1.2 Technical description of the protocol

Physical description of the protocol

The CAN protocol transmits frames over the CAN bus, this communication medium consisting of a pair of two simple wires terminated by two 120 Ohms impedances. The first wire is designated as CAN Low (CAN L) and the second one is referred to as CAN High (CAN H). Each ECU establishes a connection by linking its high

port to the CAN H and its low port to the CAN L.

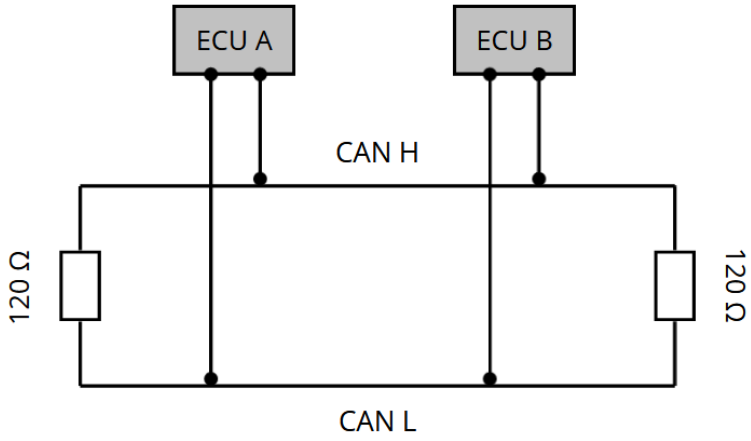

Figure 2.1: CAN bus

The bits are sent by the transceiver using a differential voltage. Each bit is composed of a dominant part and a recessive part. The dominant part of the signal is transmitted on the CAN H and the recessive one on the CAN L. The 1 bits are referred to as recessive bits and the 0 ones as dominant bits. During the transmission, when a 1 is sent the voltage difference between the two wires is null and when a 0 is sent, it is maximal. In consequence, if an ECU sends a 0 on the CAN bus and another sends a 1, the first one will erase the 1, and a 0 will be received by all the ECUs connected to the CAN bus. This collision resolution mechanism is integral to the CAN protocol, facilitating shared access to the bus among multiple ECUs.

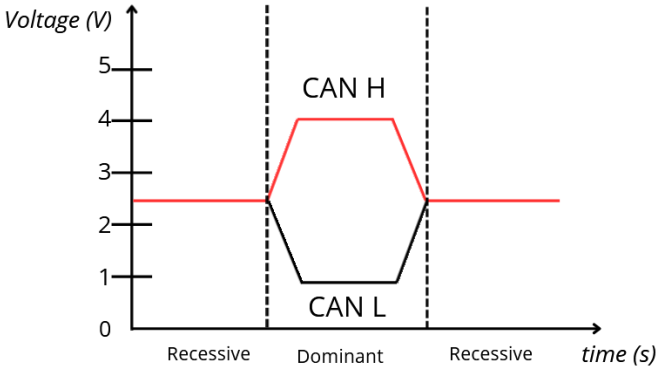

Figure 2.2: Dominant and recessive bits

Each ECU of CAN is segmented into 3 components, the CAN transceiver, the CAN controller, and the "microprocessor". This division of functionality allows great modularity and efficiency.

The CAN transceiver is the simplest part of the device, it converts straightforwardly the ones and the zeroes into voltage and vice-versa. Directly interfacing with the CAN bus, the transceiver operates at the bitwise level, managing the transmission and reception of individual bits. Additionally, it plays a crucial role in maintaining bit synchronization within the CAN network.

The CAN controller assumes the core of the CAN protocol software implementation. It is in charge of the error signaling mechanisms, the frame synchronization, the bus access, the acknowledgment mechanism, and the protocol control mechanisms. It transforms the raw bit stream received by the CAN transceiver into error-free identifiers and payloads. This critical function ensures the integrity and reliability of the data transmitted across the CAN network.

Finally, the "microprocessor" serves as the core of the device. It processes the payloads and the identifiers and, then, sends them to the CAN controller the information to communicate to the other devices.

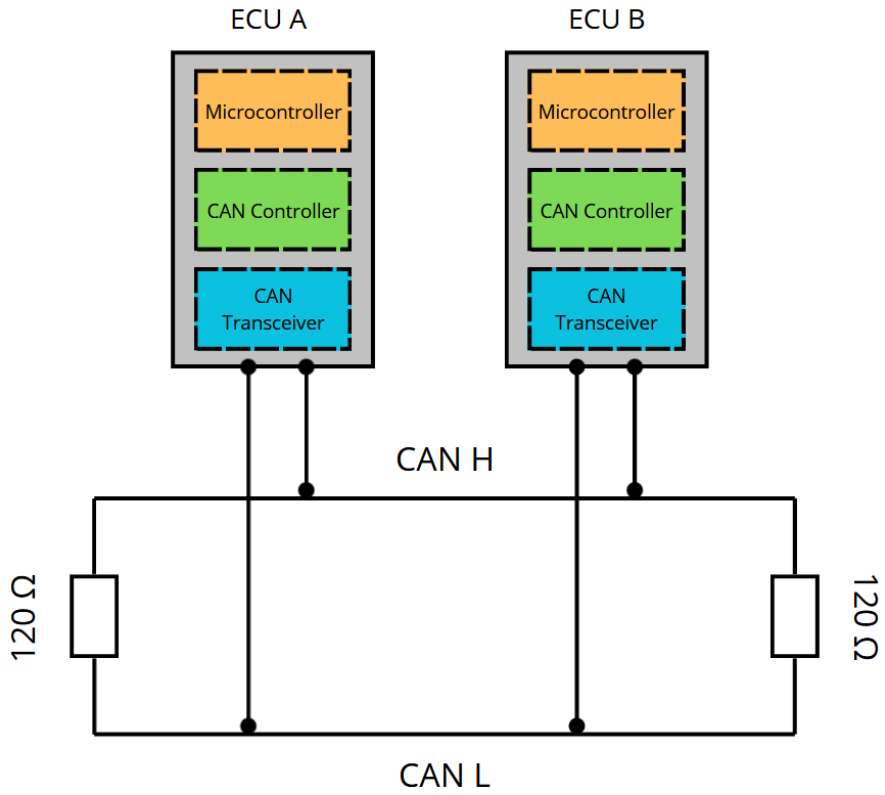

Figure 2.3: ECU description

CAN frame

Figure 2.4: CAN frame

The standard CAN frame is divided into different fields. Its size can change depending on the size of the payload and the number of stuffed bits.

- Start of the frame (SOF): To start the frame, the CAN transceiver sends a 0. It signals to the other ECUs, the beginning of the frame
- Identifier (ID): In standard mode, the identifier is 11 bits long. CAN does not use explicit addressing but functional addressing. Indeed the address of a device is not fixed as a traditional MAC address but depends on the function of the device. This field is also used during the arbitration process.
- Remote Transmission Request (RTR): This field is set as 0 for a data frame and as 1 for a request frame
- IDE, r0: These are reserved bits
- Data Length Code (DLC): The DLC is the length (in bytes) of the payload
- Data: It is where the payload is sent. Its size is between 0 bytes and 8 bytes.
- Cyclic Redundancy Code (CRC): This section is composed of a 15-bit error detection code and a delimiter of 1-bit, which is always recessive. The code is computed on the SOF, ID, control, and data fields.
- Acknowledgment (ACK): The acknowledgment bit is sent recessive by the transmitter and the receiver who is interested in the frame will respond by sending on the same field a 0. The 0 is dominant, so it will erase the recessive bit and the transmitter will detect it.
- End of frame (EOF): To announce the end of the frame, the transmitter will send 7 1s.
- Interframe: To slow down the frame rate and to delimitate the frames, the transmitter will send 3 recessive bits

Arbitration mechanism

In CAN, the access to the bus is distributed using the CSMA/CR protocol, which stands for Carrier Sense Multiple Access with Collision Resolution. This protocol employs a method of channel sharing by checking if it is idle or busy.

The Collision Resolution is accomplished through the identifier field of the frame. When two transmitters simultaneously attempt to send a frame at the same time, they transmit the first bits of the identifier sequentially. If they are different, the transmitter with a 0 to send will overwrite the 1 of the other transmitter. At this point, the second transmitter detects by checking the bus that it lost the arbitration, delayed its frame, and passed into received mode.

Stuffed bits

Due to its support of hot-plugging and the reception of information by the ECU bit by bit, the CAN protocol requires the value of the bits to change regularly. However, the transmitter could have to send a long sequence of 1s or 0s. To avoid desynchronization, a mechanism of stuffed bits has been implemented.

When a sequence of 5 consecutive similar bits needs to be sent, the transmitter employs a mechanism known as bit stuffing. In this process, the device adds a bit, named stuffed bit with a value opposite to the five previously sent bits. This bit does not affect the value of the identifier or the payload and is recognized and removed at the receiver.

Consequently, a sequence of 6 or more similar bits, is immediately detected as an error.

Other mechanisms

Error frames :

When an ECU detects an error in a frame, it initiates the transmission of an error frame to alert the other devices that an error has occurred. This frame consists of 6 dominant bits if it is an active error frame, or 6 recessive bits if it is a passive error frame. Following this, an additional 8 recessive bits are transmitted as a delimiter.

Upon detecting an error frame, the other ECUs respond by sending also an error frame. Indeed, the size of the error frame can be increased by these devices with a maximum size of 12 bits, without the delimiter.

ECU states :

When an ECU detects an error during the frame transmission, it will increase by 8 its Transmit Error Count (TEC). Respectively, when an error is detected as a receiver, the ECU increases the Receiver Error Count (REC) by 1. The TEC

is increased by a higher value because, in the event of an error detected by the transmitter, it considers itself, with a high probability, as the source of the problem.

When the TEC and the REC are below 128, the ECU is in the error active state. It means that when it detects an error, it responds with an error active frame. However, when the counters reach 128, the device transitions into the error passive state. Consequently, it only sends an error passive frame.

When one of the two counters reaches 256, the ECU enters the bus off-state and ceases transmitting frames anymore.

Overload frames :

Sometimes, the devices need to delay the sending or the receiving of the frames. In that case, the ECU will send a series of 0s during the interframe field which will not be detected by the others as an error.

This approach allows for intentional pauses or delays between frames without triggering error detection mechanisms within the CAN network. ECUs can efficiently manage the timing of frame transmission and reception without compromising the integrity of the communication system.

2.1.3 Attacks on CAN

DOS attacks

A simple way to perform denial of service attack (DOS) is to only send 0 on the bus. Since dominant bits can overwrite recessive bits, the attack saturates the CAN bus and no other device will be able to communicate on it. However, this attack is overt and can be easily detected.

Another form of DOS attack takes advantage of the arbitration process of the CAN protocol. The exploited weakness is created by CSMA/CR; indeed, the lowest identifiers are prioritized. Consequently, the attack consists of sending valid frames with the lowest possible identifiers. This prevents other legitimate ECUs from successfully transmitting their frames, leading to a disruption in communication.

Payload spoofing attacks

Due to the lack of authentication mechanisms in the CAN protocol and the relative ease of connecting to the CAN bus, spoofing attacks are very common.

The attacker can gain access to the CAN bus of a vehicle using the OBD-II, by simply connecting his device to the wires or through a hijacked ECU. Certain ECUs used for comfort and entertainment, like Wifi or Bluetooth devices, can be compromised without having physical access to the vehicle. Consequently, they may serve as entry point for an attack.

A straightforward method employed to achieve a spoofing attack involves connecting a malicious device and sending the payload the attacker wants with the

identifier used by a legitimate ECU. The attack can target a unique specific device or multiple ones.

The method described addresses some of the issues. The first one is that the receiver also receives the legitimate frames. Indeed the attacker is not assured his false payloads are taking account. Secondly, if the legitimate ECU and the malicious one send a frame at the same time, it will not be detected by the arbitration mechanism and the ECUs will raise an error in the received frame.

A manner to fix these problems is to adapt the sending of the malicious frame to the sending of the legitimate one. The attacker will listen to the channel, waiting for the sending of the true frame, and then will immediately send the malicious one. Consequently, from the receiver's point of view, the most recent "correct" payload is the malicious one. This tactic removes the risk of a clash between the malicious and the legitimate frames.

More specific attacks

Error Passive Spoofing Attack

The error passive spoofing attack leverages the error control mechanism of the CAN protocol to place a legitimate ECU into the error passive state. When an ECU is in this state, it is constrained in its capacity to interrupt the traffic by sending an error active frame, even if it detects an error.

The attacking strategy involves deliberately inducing errors in the sending frames to force the target device into the error passive state. At this point, the malicious device listens to the bus and waits for the next frame emission of the target. Once the transmission starts, the attacker changes one bit in the target frame. The legitimate sending ECU detects the error but cannot stop the traffic to warn the other legitimate devices. The intruder can, now, just overwrite the other fields of the frame to inject malicious data.

Bus-off Attack

The bus-off attack is a class of DOS attack that only targets one device. The purpose is to create errors when the ECU is sending a frame. Since the TEC increasing faster than the REC, the target enters into the bus-off state first.

Once the target is in this state, it is not able to communicate on the CAN bus anymore. The goal of the attack is reached.

Freeze Doom Loop Attack

This attack is another type of DOS that affects all the devices connected to the CAN bus exploiting the overload frames mechanism.

During the interframe field, the attacker transmits a dominant bit to send an overload frame, and then during the overload frame delimiter it will again send a 0 which will initiate the sending of an overload frame. The intruder repeats this procedure again and again, totally blocking the access of the CAN bus to the other

devices.

This attack does not trigger CAN error control mechanisms and does not change the error state of the devices; consequently, it is complicated to detect.

2.2 HPC

2.2.1 HPC description

Hardware Performance Counters (HPC) serve as specialized processor counters that increment in response to specific events occurring during the operation of a processor. These counters are commonly employed for monitoring hardware behavior and optimizing performance. The Performance Monitoring Unit (PMU) is the functional unit responsible for managing and handling HPCs.

These counters can be separated into two categories, architectural events and non-architectural ones. The first ones are the same across the processor architectures, and the others are specific to each architecture.

In most architectures, the HPC must be accessed using a kernel mode. It makes it difficult for an attacker to voluntarily modify them. This makes HPCs reliable resources in case of an intrusion.

2.2.2 Examples of HPC

Architectural events

Architectural events are common to all architectures, they are, then, the most prevalent HPCs. As examples, architectural events counters can be the processor cycles counter, the instruction counter, or the branch counter.

A processor cycle typically refers to a clock cycle, which serves as the fundamental unit of time during the operation of a processor. It is the elementary time unit of the processor, it is the time of an elementary operation. This cycle is triggered by the clock pulse.

The instruction counter is responsible for counting the instructions executed by the processor. An instruction refers to an elementary operation carried out by the processor and is typically expressed in machine code. An instruction is composed of different elements. The first is the Opcode which defines the type of instruction to be performed. Secondly, there are the operands which are the data on which the operation is applied. These operands can take the form of registers, constants, or memory spaces. Lastly, there is the addressing mode.

The branch counter increments when a branch instruction alters the sequential execution flow of the processor. A branch instruction induces a change in this

flow by jumping to another address where an instruction is stored. The branches are constituents of the proper functioning of loop and "if-else" statements. These branches can be categorized into two types of branches: the conditional and the unconditional. The firsts are activated when a specific condition is completed. In contrast, the second ones are always triggered.

Non-architectural events

The non-architectural events are architecture-dependent and vary according to the different units attached to the processor (caches, TLB, MMU...). These events can be cache hit/miss, branch prediction, TLB hit/miss, memory operation retired,... In the next section, some of them will be described.

The cache is a small high-speed memory device. In comparison to the RAM, it is smaller and faster. This memory is specialized in storing and loading small instructions or pieces of data that are regularly re-used by the processor. By keeping a subset of frequently accessed information close to the processor, the cache significantly reduces the time required for the CPU to fetch data or instructions, thus optimizing overall system performance.

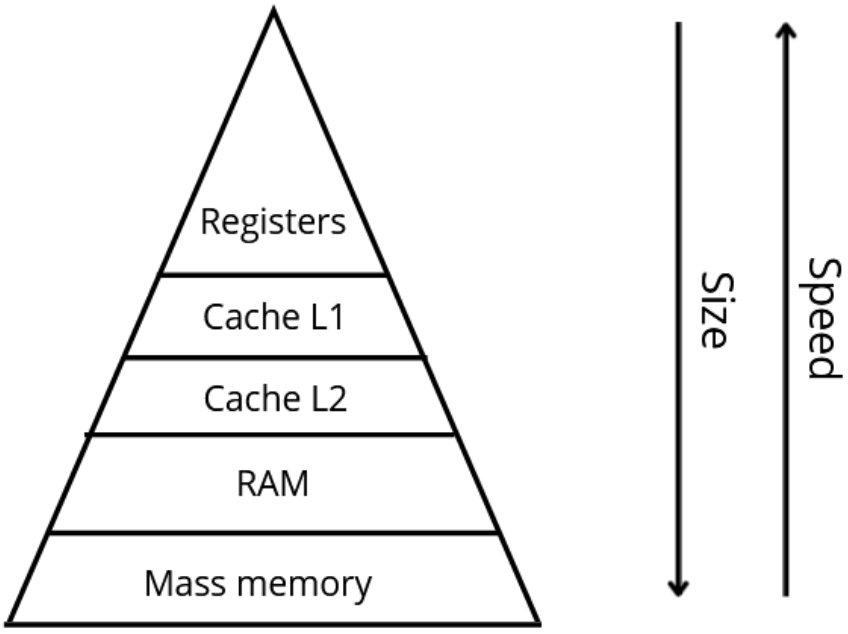

Figure 2.5: Different types of memory

In modern processors, various types of cache are employed. The caches are organized into separate levels (L1, L2, L3).

The first one, L1, is directly linked to the processor. It is split into 2 segments, the data cache and the instruction cache. The data L1 cache is dedicated to storing the data frequently used by the processor. It is directly integrated into the core to be as fast as possible. The instruction L1 cache is also attached to a core, it keeps the instructions of the executing program.

The second level, L2, is connected to the L1 cache through a memory bus. Unlike the L1 cache, it is not divided into separate data and instruction sections; the two are stored in the same memory space. In numerous architectures, the L2 cache is shared between the various processor cores. Additionally, it has often a larger capacity than the L1 cache.

The last level, L3, is generally larger than the two preceding levels. Similar to L2, it stores the data and the instruction in the same memory space. This cache is particularly adapted to multicore processors because it is shared between all the cores.

One of the key concepts with cache is "cache hit/miss". A cache hit occurs when a processor or other data processing element attempts to access data that is already present in its cache. This scenario may happen when a processor is running a program and needs to access data that has already been used recently. Processors use their cache to speed up data access by temporarily storing the most frequently used data. When a processor needs to access a piece of data, it first checks to see if it is in its cache. If the condition is completed, this means that a cache hit occurs, and access to the data is much faster than if it needed to be fetched from the main memory or the hard disk.

The branch prediction aims to anticipate the result of a branch instruction. The objective is to improve the efficiency of the instruction pipeline and to avoid the delays led by the branch instructions.

When a processor gets a branch instruction, it must solve the condition, and if it is true, it takes the branch and if it is false, it does not take it. The branch prediction occurs before the condition is checked. A good prediction reduces a lot the influence of the branch instruction but a misprediction may cause a flush of the instruction pipeline and slow done the computation process.

There are two categories of branch prediction. The first one is the static prediction, it is done during the compilation and stays constant during the program's execution. The second is the dynamic prediction. These predictors adapt their decisions to the behavior of the executing program.

A Translation Lookaside Buffer (TLB) is a special cache memory used to manage the virtual memory of a processor. Its purpose is to accelerate the virtual-to-physical

memory translation mechanism when a program needs to access the memory.

The virtual memory is used to allow the system on which the program is executing to manage its physical memory as it wants while permitting the program to choose its memory mapping.

The TLB divides the virtual memory into fixed-size tables. Each virtual page is mapped to a physical one. When a program accesses a virtual address, the TLB finds the corresponding physical address on the physical page and returns it.

A TLB hit happens when the physical memory request is already stored in the TLB. Indeed, the TLB is a type of cache memory, so looking into it speeds up the memory process. When a TLB miss occurs, the processor must find the correct address in the page table which is situated in the much slower main memory.

2.2.3 HPC in RISC-V

RISC-V architecture encompasses 3 different distinct privilege modes: the User mode (U-mode), the Supervisor mode (S-mode), and the Machine-mode (M-mode). The M-mode is the mode with the most privileges and the U-mode is characterized by the most restricted access. These privilege modes dictate varying levels of access to the Control and Status Registers (CSR), some operations are not permitted in all the modes.

In rudimentary systems, only the M-mode is available. However, the 3 modes are used simultaneously in complex systems like systems running with Unix-like OS. In these complex systems, if an application needs to access restricted resources it must call an interface named ABI to connect the application (running in U-mode) and the OS (running in S-mode) or a SBI to connect the OS and the firmware (running in M-mode).

The CSRs are registers that permit to retrieval of information from the hardware or enable and manage some mechanisms, like interruption, counters, or exceptions. Normally, they are named differently following the mode in which they are called. For example, the title of the registers reserved to the M-mode starts with an "m" and the ones reserved to the S-mode start with an "s".

Certain CSRs are specifically reserved for HPCs. Two of them are dedicated to architectural events and 29 for non-architectural events.

The 2 architectural CSRs are "cycle" and "instret". The first one counts the number of clock cycles that the processor executed and the second registers the number of instructions which has been retired.

The other events increments the "hpmcounterX" CSRs, here "X" is an integer between 3 and 31. The events tallied by these counters are selected by the CSRs "hpmeventX" and are enabled by the register "countinhibit".

To manage these registers, the program must be in M-mode. To authorize access to these CSRs to a lower privilege mode, it must be set in the register "counteren".

2.3 IDS

Attacks against information systems have become commonplace and increasingly prevalent over the years. In response, developers have implemented various mitigation techniques to minimize the impact of these attacks and enhance the security of devices. These techniques operate on distinct components of the system such as the messages, the access to the data, the network traffic, and more

Established methods for securing communication encompass encryption and message signatures. Widely recognized and proven algorithms, like AES-128 and SHA-256, are employed to ensure the confidentiality and integrity of messages. Security measures can also be placed at the network level, technologies like firewalls can block unauthorized packets and prevent malicious users from accessing the network.

An additional solution to protect the systems is to implement an Intrusion Detection System (IDS). The objective of this detection system is to detect abnormal activities within a network or a system. Various tools can be employed to achieve this purpose, including machine learning algorithms. This method is often used in complementarity with other mechanisms to create a multi-layered defense strategy, improving the overall security level.

IDSs can be categorized into two main types: signature detection and anomaly detection. The first one allows the identification of known attacks by analyzing various parameters. The key component of this type of IDS is the attack database which should contain all the situations the system has to detect. Continuous updates are essential to ensure the system can identify the latest threats. The other model of IDSs focuses on identifying the anomalies in comparison to a normal situation. Therefore, this model must be adapted to each system on which it is implemented. Hybrid IDSs also exist which take advantage of the flexibility of anomaly detection and the knowledge of signature detection.

In response to detected attacks, an Intrusion Prevention System (IPS) can be implemented. Upon identifying an intrusion, IPSs can take containment measures to limit the effects of an attack like blocking certain types of traffic. It serves as an extension of a traditional IDS which adds the ability to respond to abnormal situations.

In this study, particular attention is directed toward the security aspects of the CAN protocol. Even if CAN communication could be ensured using payload

encryption and message authentication, an IDS implementation seems to fit better with the CAN characteristics. Indeed, the CAN protocol has been designed as a low bitrate protocol, constrained to a maximum of 1 $Mbit.s^{-1}$. Implementing message authentication involves adding a signature to each message sent, reducing the available bitrate for useful data. Payload encryption, on the other hand, requires high-performance ECUs which have to dedicate a portion of their computing capabilities to payload securitizing. Furthermore, the most efficient encryption algorithms are symmetric algorithms; thus the problem of key sharing must be solved. Some CAN vulnerabilities cannot be fixed using only encryption and authentication measures. The arbitration mechanism, the overload frames weakness, or a large number of error detection processes provide a broad panel of potential attacks, especially DoS attacks. These vulnerabilities are a great challenge to patch because they are inherent to the fundamental design of the CAN protocol. An appropriate IDS could offer a solution to all these difficulties without adding traffic to the CAN bus or altering the protocol mechanisms. In addition, it only requires the ECUs to reserve a part of the computational power to the detection system.

The placement of the IDS has a major influence on its performance. If it is positioned at the network level, the IDS has access to are metrics such as the packets and the timestamps... Conversely, if it is placed inside a node the detection system gains visibility into the HPCs, the execution time of the functions, the OS scheduling, and syscalls... It is also possible to put the system at multiple points to merge and compare the metrics.

In the current study, the IDS is positioned inside a node, focusing exclusively on HPCs as key metrics.

Chapter 3

Environment of work

3.1 Tested RISC-V boards

The initial experiment was conducted using real hardware boards. However, due to challenges in accessing the HPCs, this specific setup was not exploited for training and testing the classifiers.

This experimental configuration consists of a RISC-V board and an Arduino board connected to a CAN bus. Each board functions as a CAN controller and is linked to a CAN transceiver (an SN65HVD230) through its GPIO pins. The Arduino assumes the role of the attacker and the RISC-V's board the defender on which the HPCs extraction is performed.

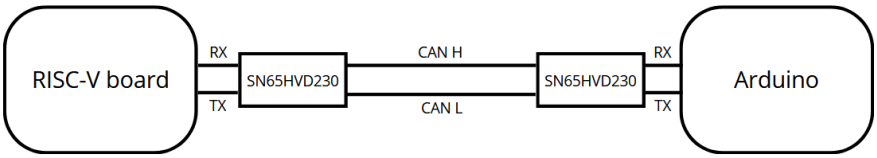

Figure 3.1: Scheme of the boards

3.1.1 K210

The K210 board stands out as a Kendryte System-on-Chip (SoC) designed for machine vision and machine learning applications. It is equipped with a RISC-V dual-core processor and multiple IO devices. The board also disposes of specialized devices for security, audio processing, and image processing, like an FFT accelerator, an AES accelerator, and an audio processor...

To facilitate development and resource management, two SDKs are provided with the K210. One operates without OS and the other includes a FreeRTOS port.

These software packages facilitate access to the devices and the management of the resources.

The decision not to use the K210 in the project is caused by the absence of non-architectural HPCs. The probe would not be sufficient for the detection and classification of the attacks. Consequently, this solution was not retained.

3.1.2 Milk-V Duo

The Milk-V Duo is an embedded board featuring the CVITEK CV1800B chip, designed with specific devices for networking, image processing, and general IO ports. The CV1800B chip incorporates two C906 CPUs, with the first one clocked to 1.0 GHz and the second to 700 MHz.

The provided SDK is a Linux-based software tailored to the CV1800B. In this configuration, the first 1.0 GHz core runs the Linux kernel, while the second 700 MHz one runs a port of FreeRTOS. The two cores communicate with a simple mailbox driver. Additionally, a lightweight version of the OpenCV library (opencv-mobile) is provided as a specialized library for computer vision.

Similar to the situation with the K210, the access to the HPCs was restricted which caused the non-use of the board during the experiments. Although the Milk-V Duo platform incorporates a Performance Monitoring Unit (PMU), monitoring the HPCs is infeasible due to the restricted reading rights of the RISC-V U-mode. Specifically, the HPCs control registers necessitate to be initialized and modified in M-mode during the booting process to permit the user to read into the HPCs registers.

3.2 General workflow

To provide an overview of the project, a workflow will be outlined, and segmented into three components: the CAN Controller Transmitter, the CAN Controller Receiver, and the Classifier.

Initially, the dataset is parsed to extract the identifier, the DLC, and the payload of the frames. The data are collected by the CAN controller transmitter and are transformed into authentic CAN frames. These frames are transmitted to the CAN controller receiver which operates on a RISC-V system emulated by the Gem5 simulator.

Upon completion of the CAN Controller receiver program, the simulator generates log files in which the HPCs are stored. These files are, then, transferred to the Python classifier script. The HPCs are extracted and transformed to train and test the classifier models.

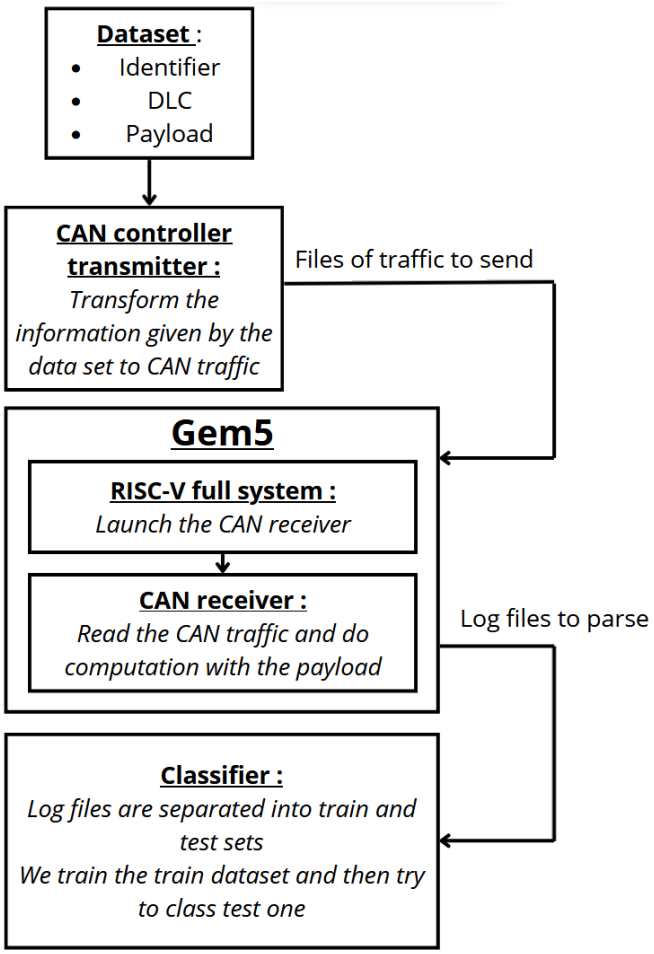

Figure 3.2: Global workflow

3.3 Gem5

3.3.1 General description of Gem5

To acquire the HPCs, it is necessary to simulate the behavior of a real RISC-V system in response to receiving CAN frames. This simulation is achieved through the utilization of Gem5.

Gem5 is an open-source simulator designed to facilitate the emulation of modular platforms, particularly for architecture research. It models the behavior of processors, memories, and interconnection systems... Gem5 serves as a platform for testing and evaluating the behavior of the architecture and the performances of various programs and hardware configurations. Notably, it supports a lot of

different ISAs, like x86, ARM, MIPS, or RISC-V. In the current study, only the RISC-V architecture will be used.

3.3.2 Communication between two systems

To conduct the attack simulation, the first strategy was to emulate two separate CAN controllers. The first controller would execute the attack and the second one would send normal traffic and process the malicious frames. This full-duplex setup was chosen to enable the implementation of complex attacks, including bus-off attacks or freeze-doom loop attacks.

Several solutions were explored to establish communication between two Gem5 systems but no one was effective. Consequently, the chosen approach involves implementing a half-duplex system in which a first system generates CAN frames and a second emulated system receives and processes these frames. While this configuration limits the complexity of attacks that can be simulated, it still allows for the execution of simpler attacks, such as DoS and frame spoofing

3.3.3 Work configuration elements

Full system mode

Gem5 has two pre-configured files that provide users with the flexibility to select configurations via command lines without necessitating direct modifications to the configuration file

The first configuration file facilitates launching Gem5 in Syscall Emulation (SE) mode. It is named this way because SE mode only emulates Linux system calls. This mode focuses on emulating the CPU and the memory systems. Its simplicity is caused by the non-requirement to set up many hardware devices.

In this work, the second mode is employed: the Full System (FS) mode. In FS mode, Gem5 emulates all the hardware systems, providing results that closely resemble real-world scenarios. This mode is chosen to simulate a more realistic environment and to integrate FreeRTOS as the operating system.

Used configuration

This project's configuration file is the "riscv/fs_linux.py" file. The program runs on a very simple bare-metal system composed of a RiscvTimingSimple CPU, a DDR4_2400_8x8 RAM, an L1d cache, and an L1i cache with a size of 64 kB, and an L2 cache with a size of 256 kB. The system is clocked at 1.0 GHz.

m5term

To help debugging, the user can connect itself to a simulated console interface called m5term in localhost.

Log files

Gem5 produces various log files throughout the simulation process. The first ones are the "config.ini" and "config.json" files which contain the list of every simulation object and their parameters.

The other file, "stats.txt", the file registers all the Gem5 statistics. This file will be parsed to extract the HPCs essential for the analysis.

3.4 FreeRTOS

3.4.1 General description of FreeRTOS

A Real-Time Operating System (RTOS) is an OS in which the scheduler is designed to schedule the tasks in a predictable or deterministic manner. An RTOS must guarantee that the system completes the task execution before a certain time limit. This form of OS is widely used in embedded systems with strict real-time requirements, including vehicles for example.

FreeRTOS is an open-source RTOS developed in C and tailored for microcontrollers. Its advantages are its small size and its portability to a large set of hardware architectures. Consequently, this OS is employed in a wide range of fields, like automotive, and aviation... FreeRTOS supports efficient time management and permits preemptive multitasking.

Additionally, FreeRTOS incorporates various OS mechanisms including semaphores, interruption handlers, and time and memory management. These features enable the programs to safely share resources between tasks.

3.4.2 FreeRTOS Gem5 implementation

While FreeRTOS is supported by numerous hardware architectures, no official implementation of FreeRTOS for Gem5 is available. To integrate the OS into the simulator, an adaptation of the "RISC-V-Qemu-virt_GCC" was performed. The bitness from 32 to 64 bits, the CPU clock frequency, and the tick rate frequency have been changed.

Some other small modifications have been introduced to simplify the porting of the OS, like removing the fake ROM. A new debug function has been added to send integers through the debug terminal.

3.5 CAN implementation

In this project, a CAN controller has been implemented using the C language and runs over FreeRTOS. This CAN implementation is divided into two main components. The first part is the CAN controller transmitter and the second is the CAN controller receiver.

Several Python scripts have been introduced to the project to simplify the execution of multiple CAN frame generation programs or CAN frame receiving/processing programs. These scripts serve also the dual purpose of parsing and shaping the dataset to adapt it to the transmitter.

3.5.1 CAN controller transmitter

The primary objective of the CAN controller transmitter is to generate CAN frames based on the information stored in the dataset.

The initial step involves extracting payloads, Data Link Control (DLC) values, and identifiers from the dataset files. Then, the data is written in different ".h" files with the correct shape. These files serve as inputs for building the transmitter program. This process is reiterated for each set of frames that needs to be generated.

The CAN library is divided into two sections, corresponding to a CAN controller and a CAN transceiver. The controller part contains the CAN control mechanisms implementation, like arbitration, and error management... It is the major branch of this program. The transceiver section is just the part that writes into an array. Then, the tab is printed into a text file which will be parsed in the receiver program.

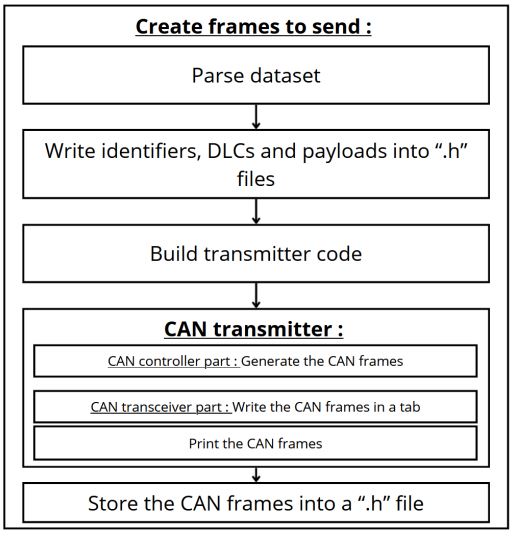

Figure 3.3: Scheme of the CAN transmitter

3.5.2 CAN controller receiver

The text files which contain the frames are parsed and the frames are extracted and copied to a ".h" file named "frame_CAN.h". This file encapsulates the frames in the form of a tab of 0s and 1s, serving as a crucial component during the compilation of the CAN controller receiver. The receiver program is built using a 64-bit RISC-V cross-compiler, enabling the generation of an executable binary with an RISC-V system built on an x86 computer.

The produced executable is run within the Gem5 simulator. The CAN transceiver part reads binary data represented by the 1s and 0s contained in the frame_CAN.h file tab. The received bits are then analyzed by the CAN controller section. These bits are assembled in CAN frames and, next, the payload and the identifier are extracted and employed in the computing functions.

The computation of the payloads triggers various hardware events that increment the HPCs. These counters are retrieved in the log files generated by Gem5 at the end of the simulation. Following a selection and reshaping process, these counters are used to train and test the detection models.

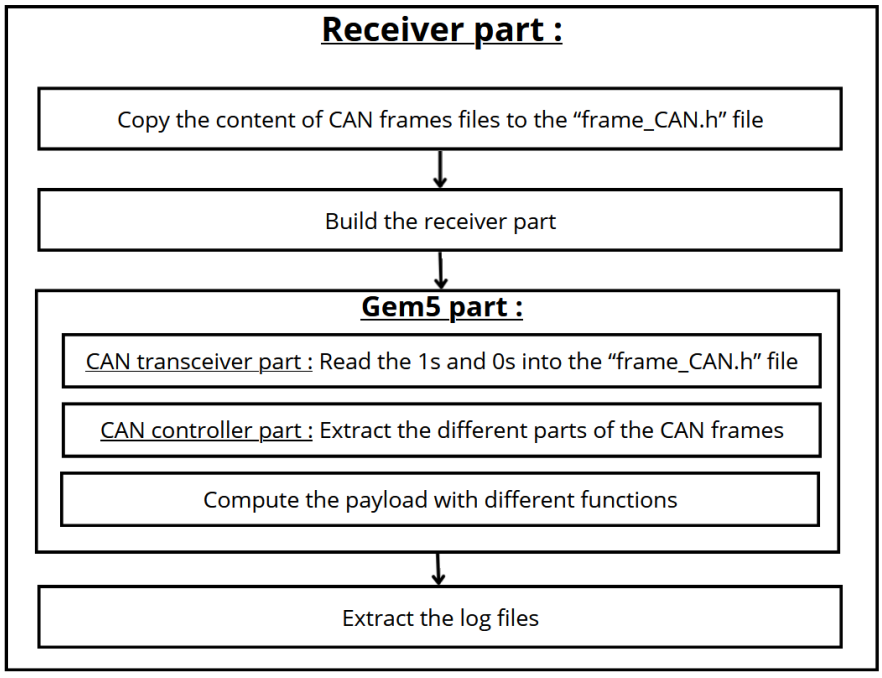

Figure 3.4: Scheme of the CAN receiver

3.5.3 Implemented mechanisms

Specific CAN mechanisms have been implemented to emulate real CAN communication as accurately as possible. Both the transmitter and the receiver can generate, receive and process not-extended CAN frames.

Arbitration process

The CAN protocol incorporates a channel access mechanism, known as CSMA/CR, functioning as an arbitration process when two ECUs want to transmit a frame at the same time. This process's purpose is to resolve collisions during the emission of two simultaneous frames without canceling the sending of the first bits of the frame. The arbitration occurs during the emission of the frame's identifier.

In the presented implementation, the ECU emits onto the channel a bit of the identifier, waits a certain fraction of the time bit, and subsequently checks the channel to verify if the received bit matches the one it transmitted. If the condition is satisfied, the transmission continues, or else the received stops the sending.

Stuffed bits

CAN is a hot-plugging communication protocol where each bit is sent one by one and simultaneously to all the devices connected to the channel. Therefore, each ECU needs to maintain synchronization with the others to track the beginning and the end of each bit. Each time a sequence of 5 similar bits is sent, the stuffed bits mechanism inserts and emits a bit that is opposite to the 5 previous ones.

The stuffed bits are inserted after the creation of the CAN frames. The program browses the entire CAN frame to add a stuffed bit every time 5 same consecutive bits are found. Once the process is completed, the frame is ready for transmission.

Others mechanisms

Additional CAN control and error management mechanisms have been implemented. The first one is the CRC testing. Upon the complete reception or transmission of CRC, the device computes the CRC with the beginning of the frame and compares it to the received one. If the twos are equal, the transmission or the reception continues normally; otherwise, the ECU raises an error.

The second implemented mechanism is a fixed-bit controller. In a CAN frame, certain bits are always the same regardless of the payload and the identifier sent, such as the SOF bit, and the delimiters... Therefore, it is possible to detect errors by checking if the received fixed bits match the expected values.

3.5.4 Mechanisms to be implemented in the future

The proposed CAN library is not yet completed, as it still needs to include processes, particularly error reaction, acknowledgment, extended frames, and overload frames. These functions are only usable in a full-duplex communication to report an error, to confirm the reception of a frame, or to delay the receiving of other frames. In the proposed work, only a half-duplex CAN communication has been developed, making these features unnecessary for the current implementation.

In future work, improving the library in allowing full-duplex communication is a turning point. In this case, the presented mechanisms need to be incorporated to establish this type of transmission correctly.

3.6 CAN frames datasets

3.6.1 General description

The CAN frames dataset utilized in this study is a collection of tuples, where each tuple comprises an identifier, a DLC, and a payload.

The dataset has been generated by monitoring the OBD-II port of a KIA SOUL car.

3.6.2 Attacks

The dataset is divided into four datasets. The first one is an attack-free CAN frames dataset which serves as a reference for comparing the "normal traffic" state with the attack state. The three others are the attack datasets; each records a different type of attack.

The first attack involves a basic DoS attack. In this dataset, the attacking device injects CAN frames with an ID equal to 0. Consequently, the injected frames always win the arbitration process, causing the delay of the legitimate frames.

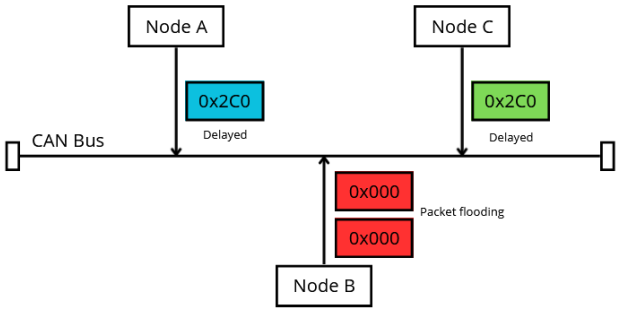

Figure 3.5: Scheme of the DoS attack

The second attack is a Fuzzy attack. In this scenario, the malicious device injects spoofed CAN frames with random IDs and payloads. The intention is to introduce non-legitimate traffic to disrupt communications between the devices.

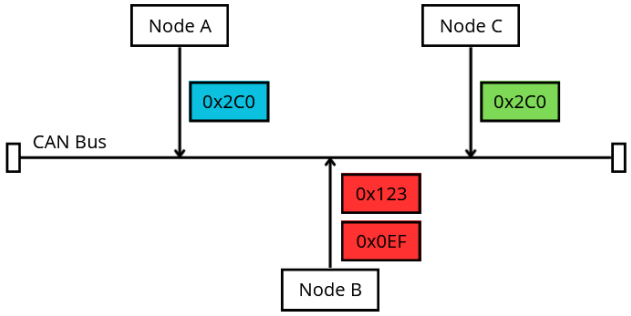

Figure 3.6: Scheme of the Fuzzy attack

The final attack is a simple Impersonation attack. The purpose of this attack is to send a false payload with the identifier of a legitimate device. The ID of the under-attack device is "0x164".

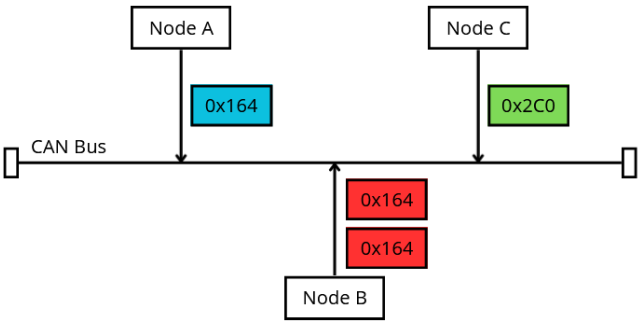

Figure 3.7: Scheme of the Impersonation attack

Chapter 4

Experimental results

4.1 Implemented functions

In this section, the functions will be presented that compute the payload. These are simple functions commonly used in computer sciences. These functions are implemented in C language to modify the HPCs.

4.1.1 Convolution

The convolution is a mathematical function applied to two functions and has numerous application fields, particularly in signal processing. The convolution product can be applied in both continuous and discrete domains.

For two continuous functions f and g, the convolution product is defined as :

$$(f * g)(x) = \int_{-\infty}^{+\infty} f(x-t)g(t)dt \tag{4.1}$$

For two discrete functions f and g, the convolution product is defined as :

$$(f * g)(n) = \sum_{m=-\infty}^{+\infty} f(n-m)g(m)dt \tag{4.2}$$

The second equation is widely used in signal filtering. The function f could be considered as the signal and the function g as the filter.

The convolution product implemented is very basic. It only divides the payload into two parts and computes a discrete convolution product between the two parts.

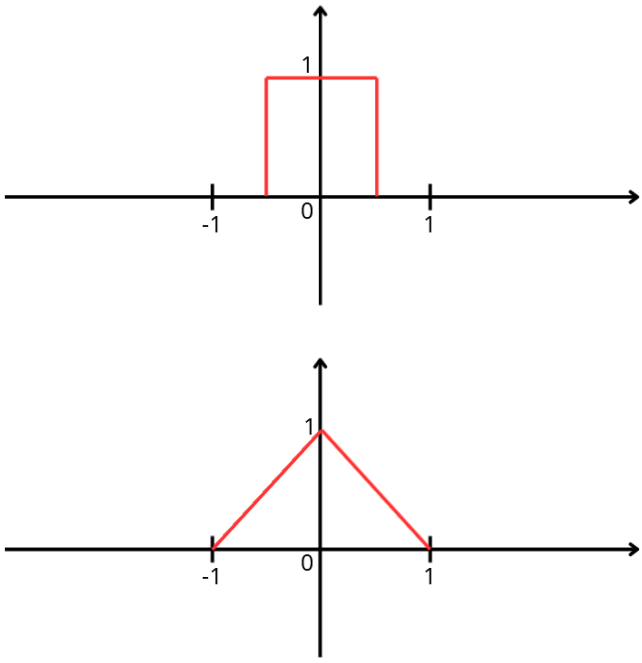

Figure 4.1: Convolution product of a rectangular function by itself

```
1   void convolution(uint8_t* payload, uint8_t payload_size) {
2       uint8_t u1[payload_size/16];
3       uint8_t u2[payload_size/16];
4       uint16_t res[payload_size/8-1];
5       for (int i = 0; i < payload_size/16; i++) {
6           u1[i] = payload[i];
7           u2[i] = payload[i+4];
8       }
9       for (int i = 0; i < payload_size/8-1; i++) {
10          res[i] = 0;
11      }
12      for (int i = 0; i < payload_size/8-1; i++) {
13          for (int j = 0; j < payload_size/16; j++) {
14              if ((i-j >= 0) && (i-j <= (payload_size/16)-1)) {
15                  res[i] = res[i] + u1[j]*u2[i-j];
16              }
17          }
18      }
19  }
```

Figure 4.2: C code of the implemented convolution product

4.1.2 AES-128

The Advanced Encryption Standard 128 bits (AES-128) is the standard symmetric encryption algorithm largely used in cryptography. It was validated in 2000 based on the Rinjdael algorithm to establish a new standard symmetric encryption algorithm. AES-128 encrypts a 128-bit block with a 128-bit key but also exists variations (AES-192 and AES-256) that allow 192-bit and 256-bit keys.

AES-128 plays a crucial role in securing communications between users sharing a common key and in protecting data at rest.

Description of AES-128

The algorithm takes in a 16-byte plaintext block and a 16-byte key and out a 16-byte ciphertext block. It starts with an initialization round with the key, continues with 9 "normal" rounds, and lasts with a simplified round.

Each step is too long to be described here, the following figure shows a general description of the AES-128 algorithm.

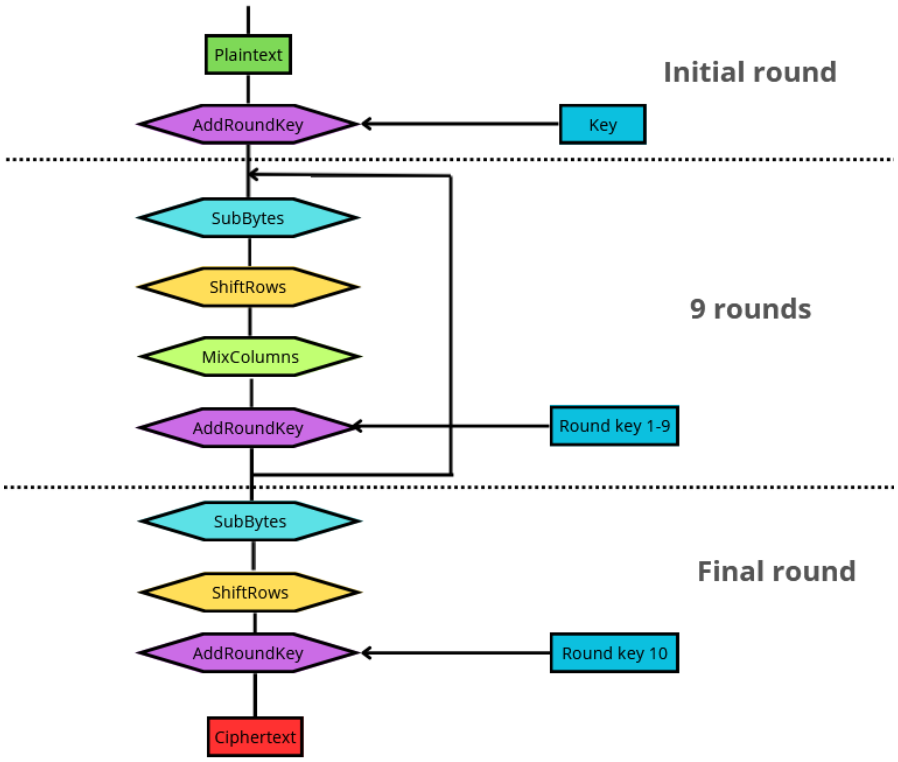

Figure 4.3: General description of AES-128

The AES-128 implementation C code will not be explained here in its entirety

but the algorithm usage is described below. The payload is the plaintext to encrypt and decrypt and an arbitrary key has been chosen.

```
void aes(uint8_t* payload, uint8_t payload_size) {
    if (payload_size/8 != 8) {
        return;
    }
    unsigned char block[16];
    for (int i = 0; i < payload_size/8; i++) {
        block[i] = payload[i];
    }
    for (int i = payload_size/8; i < payload_size/4; i++) {
        block[i] = payload[i-payload_size/8];
    }
    unsigned char key[16] = {
        0x02, 0x03, 0x01, 0x01,
        0x01, 0x02, 0x03, 0x01,
        0x01, 0x01, 0x02, 0x03,
        0x03, 0x01, 0x01, 0x02
    };
    unsigned char roundKey[176];
    keyExpansion(key, roundKey);
    encrypt(block, roundKey);
    decrypt(block, roundKey);
    displayBlock(block);
}
```

Figure 4.4: C code of the usage of the AES-128 algorithm

4.2 Parameters transformations

The parameters employed for training and testing the various models exhibit considerable diversity and quantity. In this study, the aim is to scale the parameters set to reduce its size and its heterogeneity. This is achieved through two methods which proceed one after another, at first the "mean and scale" method and secondly, the "correlation reduced" one.

Another parameter selection is run independently of others, the reduced dataset.

4.2.1 Mean and scale

To address the wide range of input values, spanning from 10^{-7} to 10^{10}, the parameters will be standardized by centering them around 0 and setting the standard

deviation to 1. This operation aims to normalize the data presented in the model.

4.2.2 Correlation reduced

For each input parameter, correlation coefficients will be calculated with the output parameter. Parameters for which the correlation coefficients cannot be computed or fail to reach a threshold (0.9 for traditional classifiers and 0.1 for multiclass classifiers) will be excluded.

Ultimately, only about half of the parameters will be retained for training and testing the model.

4.2.3 Arbitrary parameter selection - reduced dataset

This last selection is independent from the others. It is an arbitrary selection of the HPCs related to the cache memory. 16 parameters have been selected :

1. system.cpu.commitStats0.numInsts
2. system.cpu.fetchStats0.numBranches
3. system.cpu.dcache.demandHits::cpu.data
4. system.cpu.dcache.demandMisses::cpu.data
5. system.cpu.dcache.ReadReq.hits::cpu.data
6. system.cpu.dcache.ReadReq.misses::cpu.data
7. system.cpu.dcache.WriteReq.hits::cpu.data
8. system.cpu.dcache.WriteReq.misses::cpu.data
9. system.cpu.icache.demandHits::cpu.inst
10. system.cpu.icache.demandMisses::cpu.inst
11. system.cpu.icache.ReadReq.hits::cpu.inst
12. system.cpu.icache.ReadReq.misses::cpu.inst
13. system.l2.demandHits::cpu.data
14. system.l2.demandMisses::cpu.inst
15. system.l2.demandMisses::cpu.data
16. system.l2.demandMisses::total

4.3 Classifiers theoretical explanation

The IDS component of this work involves a classifier implemented in a Python script using the sklearn library. The program takes log files containing HPCs produced by the Gem5 simulator as input and produces predictions from the classifier. Specifically, it determines whether there is an attack, and if so, what type of attack it is.

4.3.1 One class classifier

The first classifier tested in this study is the simple OneClassSVM from the sklearn library. This unsupervised model is capable of distinguishing samples of a particular class from other samples that do not belong to this class. It is firstly trained by a dataset exclusively containing objects of the specified class and then tested with objects included or not in this class. Such classifiers are employed when, in a system, only the normal operating is known and the purpose is to detect anomalies or malfunctions.

In this study, the training class points are the normal traffic samples while the outlier points are the under-attack traffic samples.

The model can be trained with variations by adjusting the ν-parameter. This parameter controls the upper bound of acceptable errors during the training step. The lower ν is, the lower the amount of tolerable errors will be. Increasing it will make the model more flexible but, thus, may also result in more misclassifications.

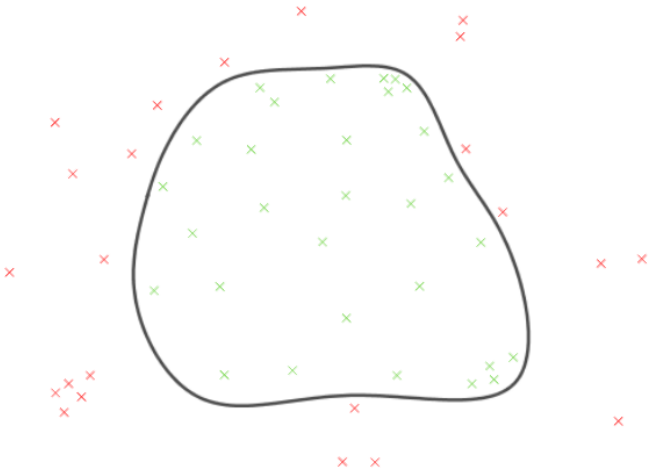

Figure 4.5: Example of a result of a one-class classification. The green points correspond to the selected class and the red ones to the points outside of this class

The model aims to establish a boundary, materialized in the example by the

black line, that separates the class points from the others. The shape of this boundary is determined by the kernel function used as a model parameter.

4.3.2 SVC

The Support Vector Classification (SVC) is a type of supervised classifier designed to distinguish two classes of objects. As a supervised model, it requires training with samples from all classes. The training step needs to link each class to a certain label, the "normal traffic" class is associated with the label 0, and the "under-attack traffic" class with the label 1.

The primary objective of the SVC is to identify a hyperplane that effectively separates the different classes and maximizes the margin between them. In cases when it is impossible to linearly divide the data, the model applies a kernel function to project the sample onto a higher dimensional space. Afterward, the classifier determines the hyperplane that maximizes the margin between the different classes, which is the distance between the hyperplane and the elements closest to the other class.

An essential parameter of the SVC model is the C-parameter which regulates the trade-off between the maximization of the margin and the minimization of the number of errors. Increasing C reduces the number of errors and the size of the margin, conversely lowering the C-parameter makes the classifier more tolerant to errors but enlarges the margin.

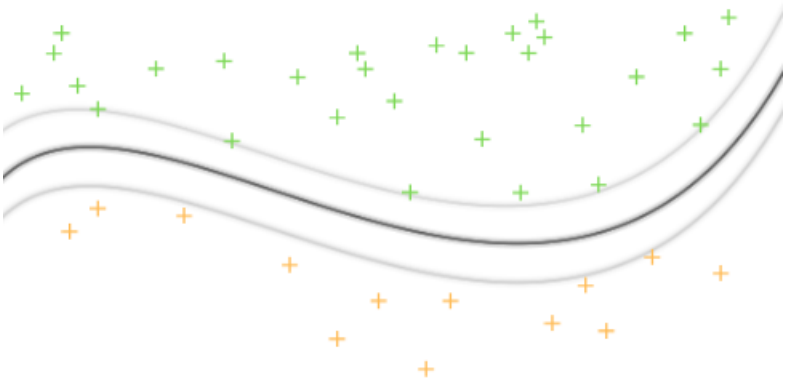

Figure 4.6: Example of a result of an SVC classification. The green points correspond to one class and the orange ones to the other class. The black line is the hyperplanes and the gray lines are the margins

Unlike the previous one-class unsupervised classifier, the SVC is unable to detect a new attack case that differs from the normal traffic situation. Therefore, the

classifier must be trained on each attack the ECU is expected to handle. This limitation could especially be a withdrawal if a large set of attacks could affect the device. Hence, the training complexity increases quadratically with the number of samples.

4.3.3 Random forest

Random forest is a classifier based on the decision tree technology. A decision tree is a series of different conditions arranged in the shape of a tree, where each condition is a node between two branches (condition completed or condition not completed), and the classification results are the leaves of the tree. This is a straightforward type of classifier capable of easily categorizing a two-class or multiclass dataset.

The random forest model generates a large quantity of decision trees during the training step. Each tree is constructed using a subset of the training dataset and a subset of all the parameters. During testing, the trees individually provide a prediction, and the most frequently predicted class is selected as the final prediction.

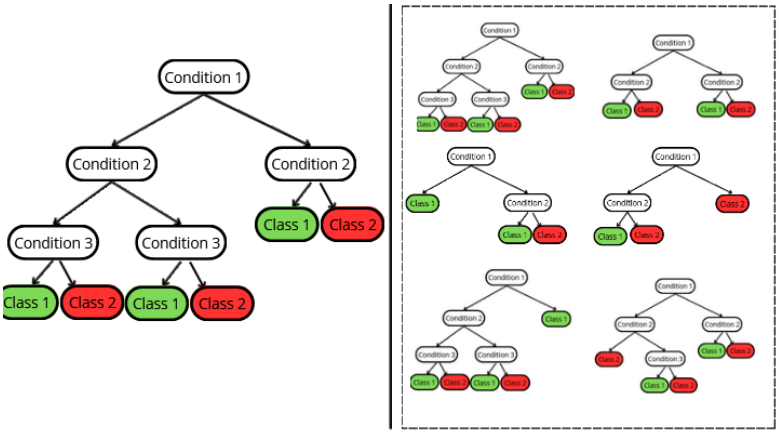

Figure 4.7: Example of a Decision tree (at left), and of a Random forest (at right)

In this study, the two explore parameters are the n_estimators and the criterion. The different tested criteria will be presented in detail in a next section. The n_estimators parameter is the number of trees generated during the training of the classifier. Increasing this parameter improves the results given by the model; however, the complexity also grows, which extends the training duration.

The random forest offers multiple advantages over a simple decision tree. At first, it limits overfitting. Each decision tree is formed with a different section of the dataset and different features. In contrast, the decision tree adapts perfectly itself to the particularities of the training samples. Consequently, the random forest allows a better generalization to new data and is more stable in these responses to

small perturbations.

Although the dataset is complete in this study and thus this property will not be utilized here, the random forest classifier can easily handle the missing values. Effectively, the trees are built using different features, therefore the trees can compensate for the deficiencies of another.

4.3.4 Multiclass

In this project, two types of classification will be attempted for the SVC and the Random forest: two-class and multiclass classification (more than two classes). The two-class classification differentiates between normal traffic and traffic under one type of attack. On the other hand, the multiclass classification assigns a label to each type of traffic (normal, DoS, Fuzzy, and Impersonation attacks).

The multiclass classifier model used is the One-versus-One classifier combined with another classifier (SVC or Random Forest). For each pair of classes, this model creates a classifier; in the study case, 6 classifiers will be generated. During the testing step, a vote is organized between the classifiers, and the class that obtains the most votes is the final result.

4.3.5 Kernel functions

One crucial parameter of the One class classifier and the SVC is the kernel function. In SVM models, the goal is to identify a hyperplane that separates two classes. When the two classes are not linearly separable, a technique known as "kernel trick" can solve the problem. This technique involves transforming the space of the input space into a higher dimensional space, where a linear separation can be found. The kernel function is responsible for performing this transformation.

Various kernel functions are being tested to train the classifiers. The first and simplest one is the "linear" kernel which corresponds to a model that remains in the same space and without applying the kernel trick. This serves as a control model for comparison.

The second one is the "polynomial" kernel which, is governed by the following function. The parameter d corresponds to the degree of the polynomial. In the study, only odd degrees between 1 and 11 are tested, and r is a setting set to 0. Lastly, γ will be explained in the following section.

$$K(x, x') = (\gamma\langle x, x'\rangle + r)^d \tag{4.3}$$

The "rbf" kernel is a kernel function for which the space of the inputs is increased to an infinite number of dimensions. It is described by the next equation :

$$K(x, x') = exp(-\gamma||x - x'||^2) \tag{4.4}$$

The last kernel function is the "sigmoid" function. The parameter r is set to 0 here.

$$K(x, x') = tanh(\gamma\langle x, x'\rangle + r) \tag{4.5}$$

4.3.6 Gamma

For One-class classifier and SVC kernel functions, two types of gamma parameters are used, the "auto" and the "scale". The "auto" is equal to $\frac{1}{n\_features}$ and the "scale" is equal to $\frac{1}{n\_features \times Var(x)}$.

4.3.7 Criteria

The criteria are parameters used to train the Random forest classifiers, three have been employed in this study: "gini", "entropy" and "log_loss". These parameters are functions that gauge the quality of separation between two branches.

In a branch split at node m, the purpose is to find the feature j and the threshold t_m for which the impurity is minimal. The impurity is materialized by the equation :

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} H(Q_m^{left}(\theta)) + \frac{n_m^{right}}{n_m} H(Q_m^{right}(\theta)) \tag{4.6}$$

With :

- $\theta = (j, t_m)$
- Q_m, the data at node m
- n_m, the number of samples at node m
- n_m^{left}/n_m^{right}, the number of samples at node m respectively for the left and the right branches
- $H()$, the loss function, which is determined by the criterion
- Q_m, the data at node m
- Q_m^{left}/Q_m^{right}, the data at node m respectively for the left and the right branches

For a node splitting in a decision tree, the loss function H for the criterion "entropy" and the "log_loss" is the same.

Gini :

$$H(Q_m) = \sum_{k}^{n_{class}} p_{mk}(1 - p_{mk}) \tag{4.7}$$

Entropy and log_loss :

$$H(Q_m) = -\sum_{k}^{n_{class}} p_{mk} log(p_{mk}) \tag{4.8}$$

With $p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k)$

4.3.8 Observed metrics

To assess the performance of the classifiers, multiple metrics are employed. They are designed to evaluate the quality of the classification models tested in this study.

Accuracy

This metric measures the overall correctness of the model. It is the most straightforward metric utilized in the project, representing the ratio between the number of correct predictions to all the predictions. The accuracy is equal to 0 when the classifier is consistently wrong and to 1 when it is consistently correct.

The accuracy is particularly valuable when dealing with balance class occurrences. However, in real-world scenarios, attacks often occur much less frequently than in normal traffic, resulting in imbalanced classes. This imbalance can lead to a misinterpretation of the performances of the classifier. Consequently, additional metrics must be collected and analyzed to provide a more accurate description of the results.

Precision

Precision offers a solution to the accuracy limitation. Indeed, this metric measures how a model predicts properly a certain class. It is computed using the following equation :

$$Precision = \frac{TP}{TP + FP} \tag{4.9}$$

When the precision reaches 1, the model predicts the class without errors. Conversely, when it drops to 0, the classifier is incapable of making accurate predictions for this class.

The precision now provides a manner to figure out how well the model is working even if the classes are strongly imbalanced. Moreover, false positives have

a significant impact on the result, which is highly valuable in a detection case. Nevertheless, precision does not specifically capture the influence of false negatives which is required for an effective IDS. In practice, an element of the given class may be misclassified.

Recall

The recall takes into consideration the false negatives, providing the performance ratio of how well the model can identify all the elements of a given class. A recall equal to 1 indicates that all the elements of a certain class are correctly labeled. This metric is calculated with the next equation :

$$Recall = \frac{TP}{TP + FN} \tag{4.10}$$

Like precision, recall offers very good results in measuring imbalanced classes but it does not consider false positives. As a consequence, it has to be used complementary to the precision metric.

F1

The F1 is a metric that represents a compromise between the recall and the precision. It takes into account both false positives and false negatives in its calculation.

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{TP}{TP + \frac{1}{2}(FN + FP)} \tag{4.11}$$

An F1-score equal to 1 indicates that there is no error in the classification in the target class and all the elements of this class are correctly classified.

An alternative to the F1-score is the Fβ-score which introduces a parameter, β, to adjust the influence of the precision in the score. It is calculated as follows :

$$F\beta = (1 + \beta^2)\frac{precision \times recall}{(\beta^2.precision) + recall} = \frac{TP}{TP + \frac{1}{1+\beta^2}(\beta^2 FN + FP)} \tag{4.12}$$

The F1-score is a particular case of Fβ when $\beta = 1$. If $\beta > 1$, the recall has a greater impact on the computation. Conversely, if $\beta < 1$, the precision is more significant.

Log loss

The log loss is a metric based on the predicted probability of a class. In other words, the classifier provides, for each sample, the probabilities of belonging to each class. Log loss measures how incorrect the probability prediction is. If the model is completely certain (with a probability of 1) and correct, the log loss will be equal to 0. Therefore, if the classifier is wrong or uncertain about its classification, the log loss increases. The metric is computed in the following way :

$$Log\_loss(y, p) = -(y.log(p) + (1 - y).log(1 - p)) \tag{4.13}$$

With :

- y, the true label
- $p = Pr(y = 1)$, the probability estimated to be in a certain class

4.4 Results

Initially, a first search for the best parameters for the classifiers will be conducted by simply counting the number of errors. In this scenario, one-half of the dataset is used to train the model, and the other part to test it except for the one-class classifier. In this case, 95% of the normal traffic part is used for training, 5% for testing, and all the attack traffic part is used for testing as well. Indeed, the one-class is an unsupervised classifier which only train on the normal traffic dataset.

The results of this research are outlined in the "Raw results, research of adapted parameters" section for each classifier.

Subsequently, a more detailed study will be carry out with the optimized parameters. The size of the training dataset will be gradually increased from 2% until it encompasses 95% of the entire dataset. The output metrics include accuracy, precision, recall, F1, and log loss.

This research will be applied to 4 different types of datasets. The first dataset contains all the significant HPCs provided by the Gem5 simulator without modification or re-selection. The other ones are presented in the 4.2 section; they are the "mean and scale", the "correlation reduced" and the "reduced" datasets.

Before the training of the models, the datasets are randomly shuffled and then divided into two parts, the training dataset and the testing one. Multiple training sessions are conducted varying the size of the datasets. In each iteration, the first part is increased while the second is decreased.

Certain datasets were not evaluated on all the classifier models, such as the "correlation reduced" dataset which was not tested on the one class classifier, due

to the consistently bad results given by this type of classification in this scenario. Additionally, datasets with a smaller number of frames per sample, like 75 frames dataset for AES or 100 frames for Convolution, were assessed on the "reduced" dataset. Indeed, the number of frames is too small to trigger all the hardware events present in the "full" dataset on which the "correlation reduced" and the "mean and scale" ones are constructed.

The log loss metric has not been computed for all the classifiers, as the metric requires the prediction probabilities of each class. However, certain classifiers, such as the one-class classifier and the multiclass ones, lack a function in the sklearn library to calculate these probabilities.

An interesting observation is that the results of the multiclass models for the Fuzzy attack sharply decrease to 0 when the ratio training set size on all dataset sizes reaches 68%. This issue is due to an error in the computation of this ratio because the size of the Fuzzy attack class is lower than the size of the other classes.

For the one-class classifier, the searching range for nu is between 0.001 and 1 with a step of 0.001. For the SVC, it is between 0.01 and 1 with a step of 0.01 and between 1 and 100 with a step of 1. Lastly, for the random forest one, the search is made between 1 and 1000 with a step of 1.

4.4.1 One class classifier

Raw results, research of adapted parameters

Globally, the research conducted in this study reveals poor results for the one-class classifier to distinguish attacks from normal traffic in this particular situation. Consequently, even with the best-adapted parameters, the effectiveness of the classifier is limited.

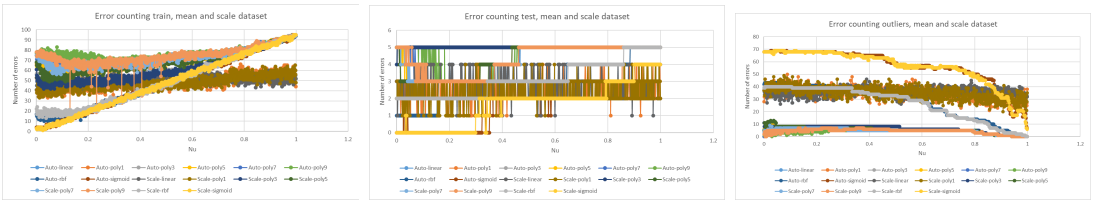

Figure 4.8: Raw results, Fuzzy attack, Convolution 2000 frames mean and scale dataset

The selected parameters are 0.2 for *nu*, rbf for the kernel, and auto for the gamma.

Final results

As precedently explained, the performances of the model are not good. Therefore, the presented results are shown as an example.

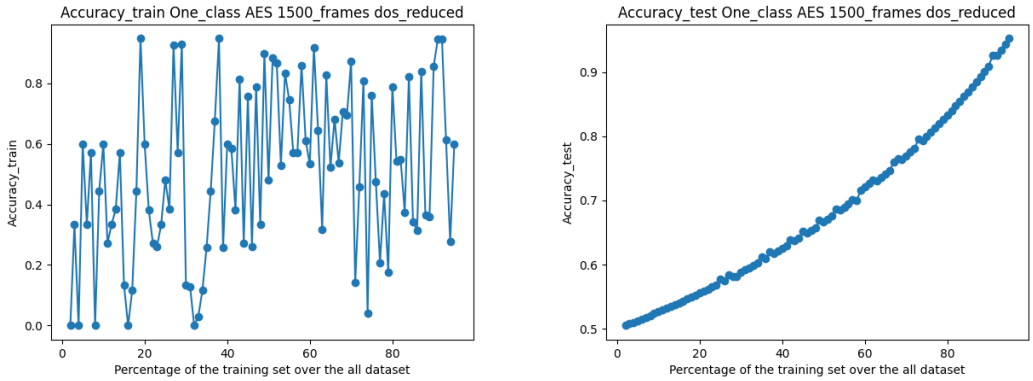

Figure 4.9: Accuracy final results, DoS attack, AES 1500 frames reduced dataset

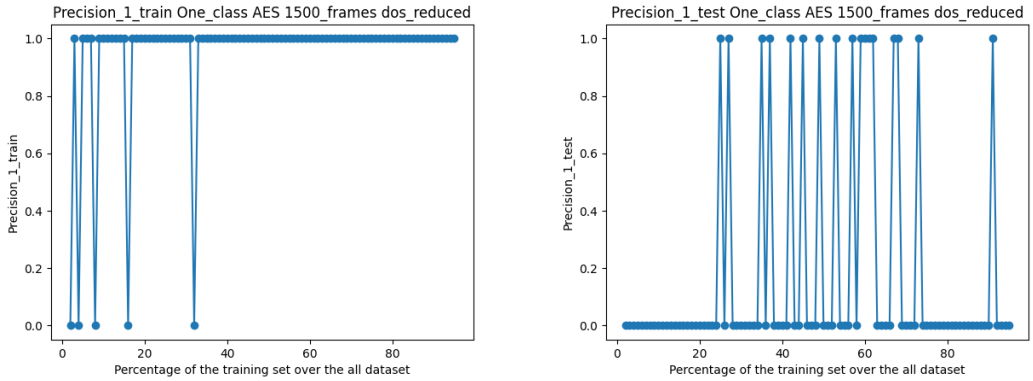

Figure 4.10: Precision normal traffic final results, DoS attack, AES 1500 frames reduced dataset

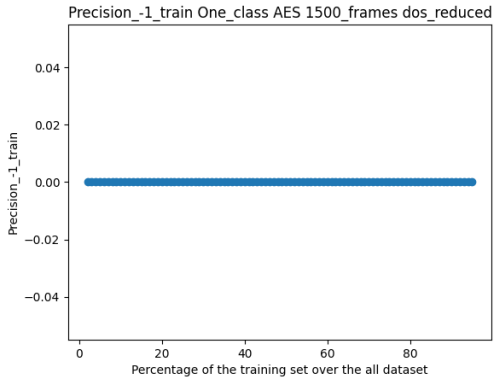

Figure 4.11: Precision malicious traffic final results, DoS attack, AES 1500 frames reduced dataset

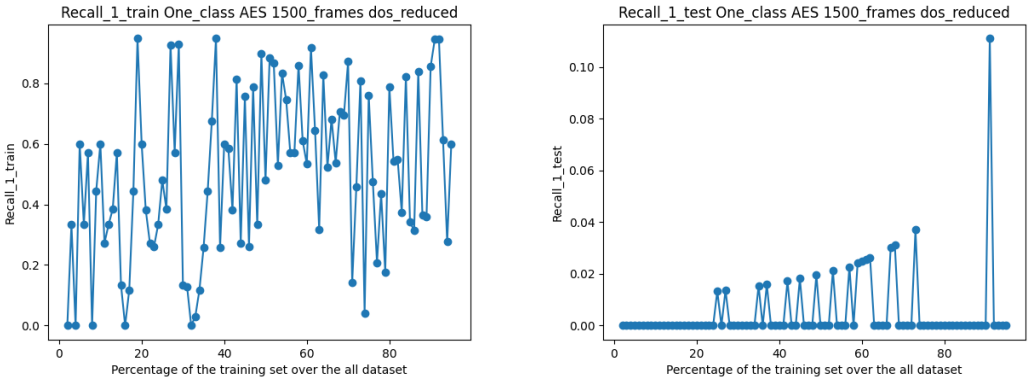

Figure 4.12: Recall normal traffic final results, DoS attack, AES 1500 frames reduced dataset

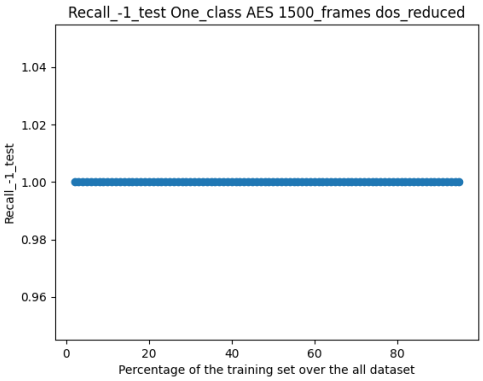

Figure 4.13: Recall malicious traffic final results, DoS attack, AES 1500 frames reduced dataset

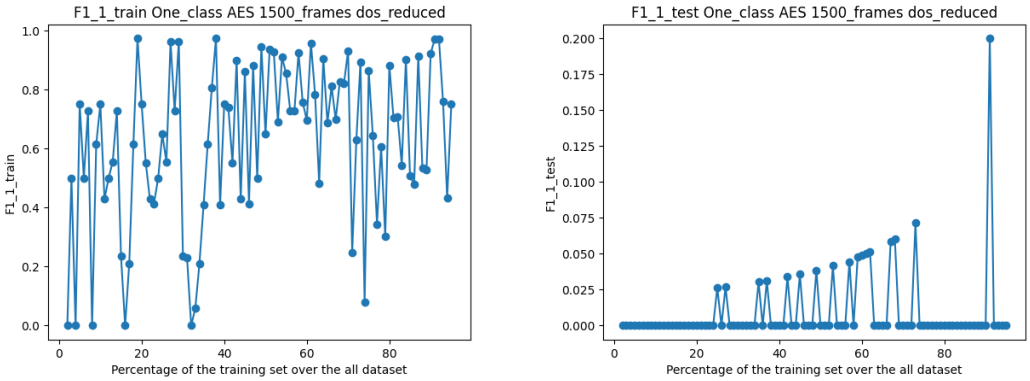

Figure 4.14: F1 normal traffic final results, DoS attack, AES 1500 frames reduced dataset

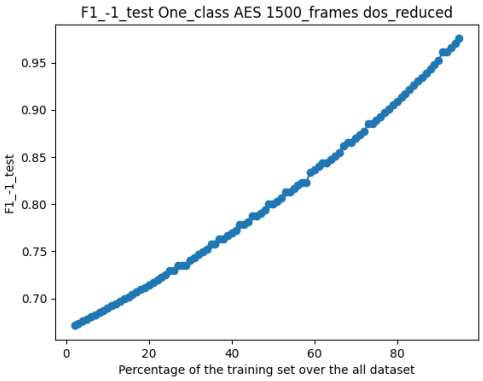

Figure 4.15: F1 malicious traffic final results, DoS attack, AES 1500 frames reduced dataset

	DoS	Fuzzy	Impersonation
Accuracy	0.66666	0.5185	0.66
Precision Normal traffic	0	1	0
Precision Malicious traffic	0.66666	0.5149	0.66
Recall Normal traffic	0	0.015	0
Recall Malicious traffic	1	1	1
F1 Normal traffic	0	0.0298	0
F1 Malicious traffic	0.8	0.6798	0.795

Table 4.1: Table of test results for AES 1500 frames reduced dataset, where the ratio is equal to 50%

The results indicate that one class classifier is not well-suited to the situation. Although the accuracy and the F1 metric for malicious traffic may appear acceptable, the other metrics remain very low. Indeed, the classifier is not able to correctly class normal traffic which is not contained in the training dataset.

4.4.2 SVC

Raw results, research of adapted parameters

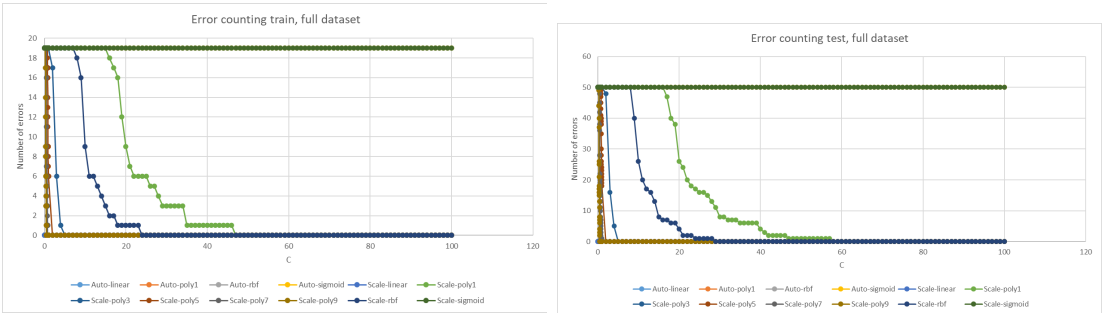

Figure 4.16: Raw results, Fuzzy attack, AES 1500 frames full dataset

The SVC model successfully classifies both attacks and normal traffic, with carefully chosen parameters resulting in very few errors and, in some cases, achieving a perfect classification, 0 errors.

Across the different datasets and attacks, the parameters that produce the best results are 96 for C, rbf for the kernel, and scale for the gamma.

Final results

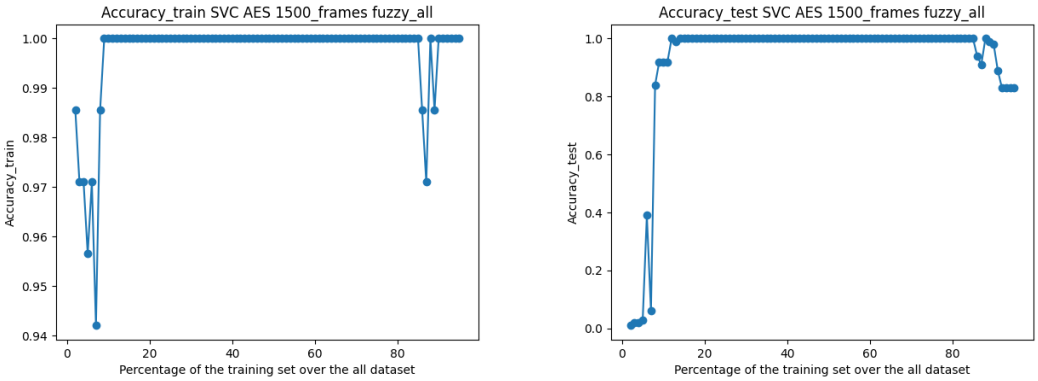

Figure 4.17: Accuracy final results, Fuzzy attack, AES 1500 frames full dataset

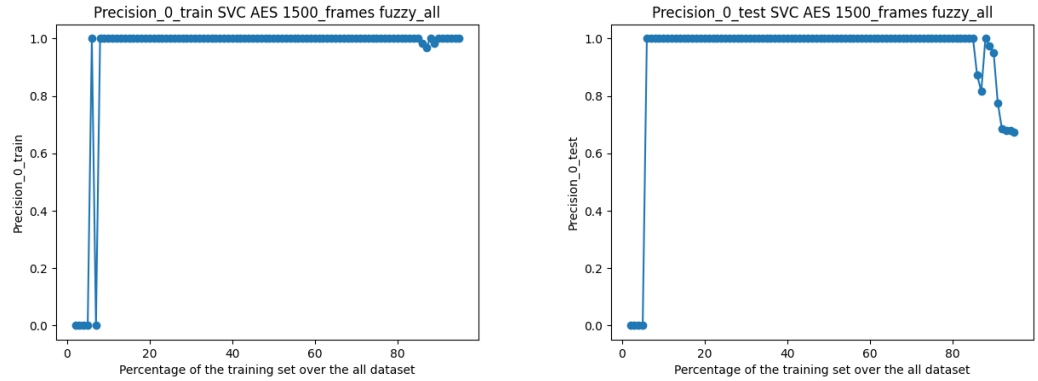

Figure 4.18: Precision final results normal traffic, Fuzzy attack, AES 1500 frames full dataset

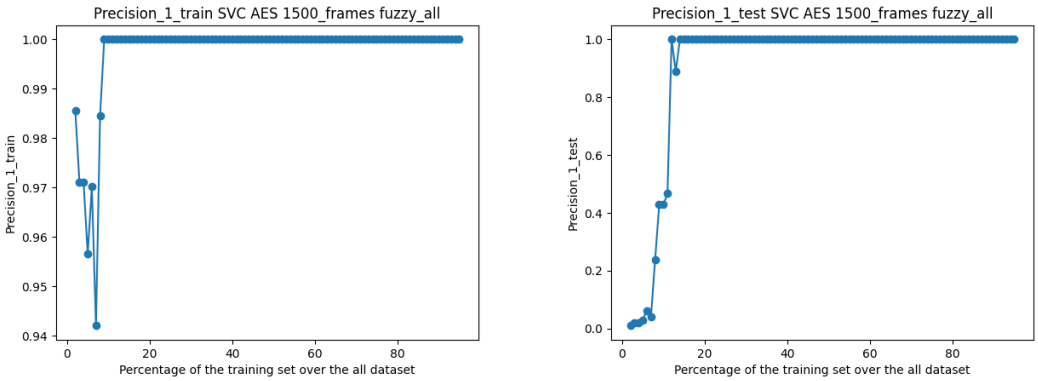

Figure 4.19: Precision final results malicious traffic, Fuzzy attack, AES 1500 frames full dataset

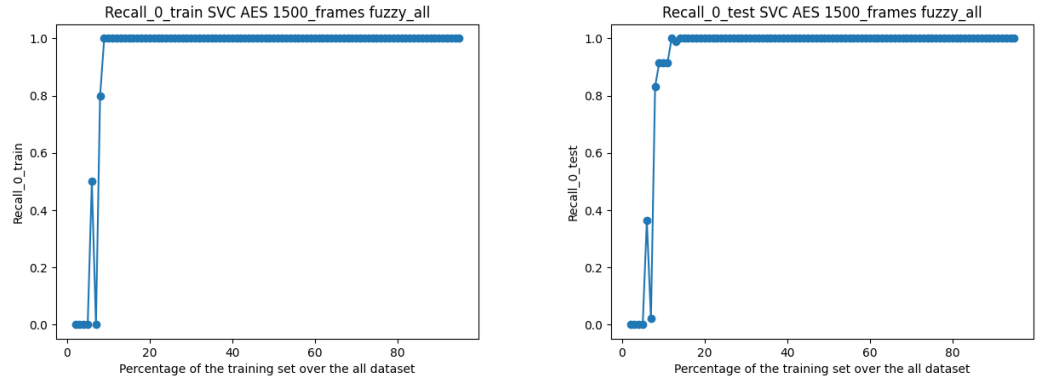

Figure 4.20: Recall final results normal traffic, Fuzzy attack, AES 1500 frames full dataset

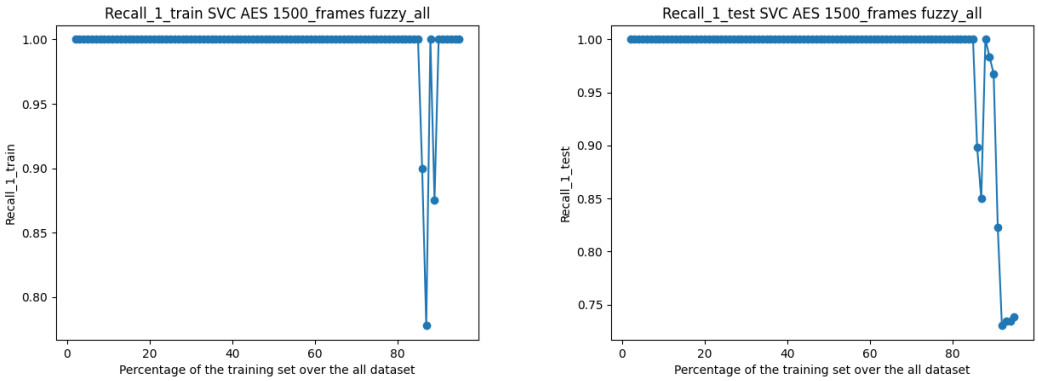

Figure 4.21: Recall final results malicious traffic, Fuzzy attack, AES 1500 frames full dataset

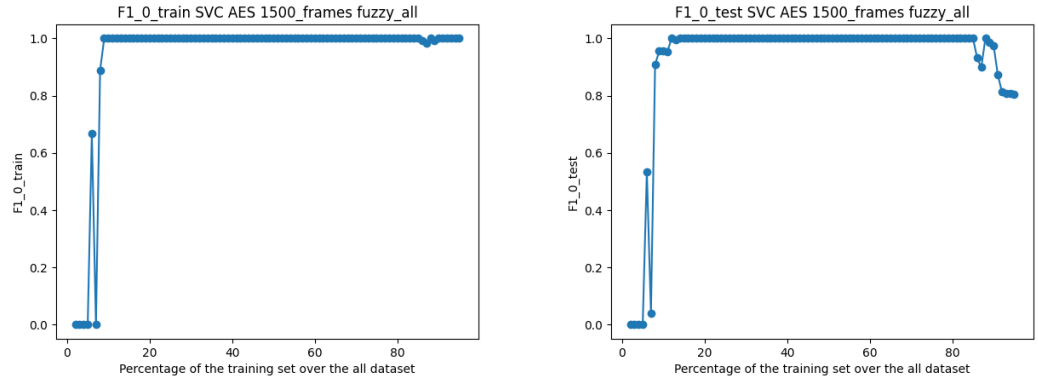

Figure 4.22: F1 final results normal traffic, Fuzzy attack, AES 1500 frames full dataset

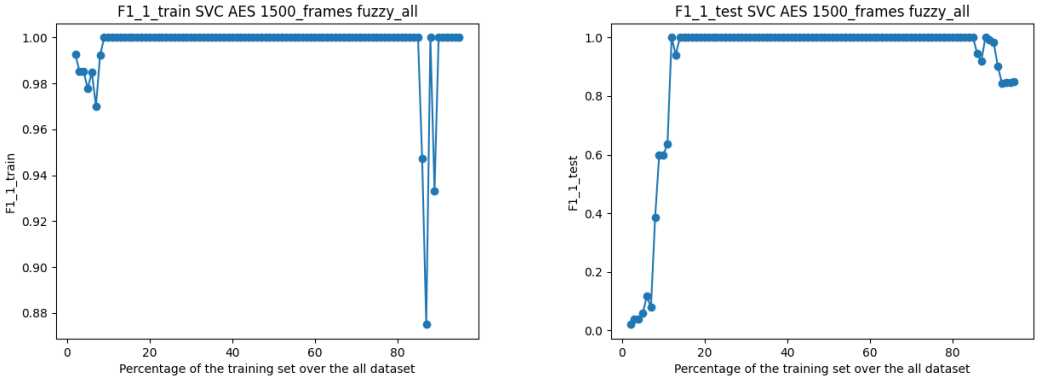

Figure 4.23: F1 final results malicious traffic, Fuzzy attack, AES 1500 frames full dataset

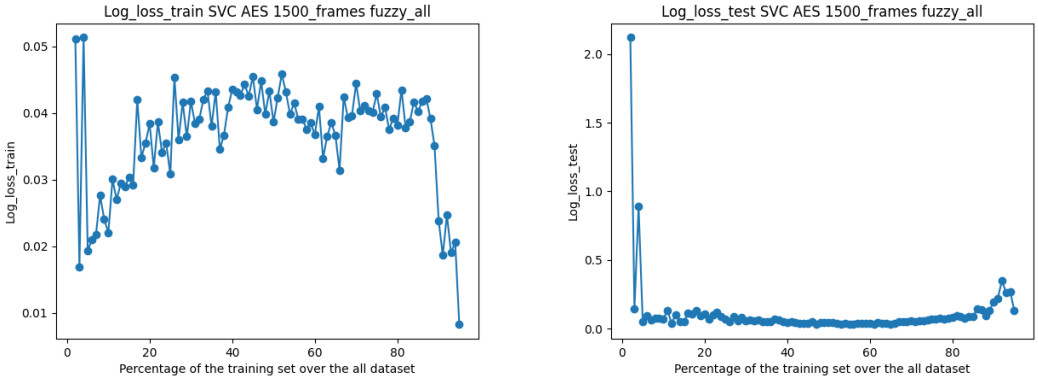

Figure 4.24: Log loss final results, Fuzzy attack, AES 1500 frames full dataset

	DoS	Fuzzy	Impersonation
Accuracy	0.97	1	0.85
Precision Normal traffic	0.943	1	0.81
Precision Malicious traffic	1	1	0.905
Recall Normal traffic	1	1	0.92
Recall Malicious traffic	0.94	1	0.776
F1 Normal traffic	0.971	1	0.86
F1 Malicious traffic	0.969	1	0.84
Log loss	0.0888	0.04687	0.368

Table 4.2: Table of test results for AES 1500 frames full dataset, where the ratio is equal to 50%

The graphs show significantly improved results compared to the one-class. A model trained with a dataset containing between 20% and 80% of the overall samples consistently returns correct answers. Outside of these boundaries, there is evidence of under-fitting and over-fitting. However, the influence of over-fitting is far less pronounced compared to under-fitting.

The log loss metric permits to measure of how confident the model is when making a correct classification. The results suggest that, for the training dataset, the proportion of the training part over the entire set has not an important effect, and, for the testing set, the log loss increases beyond the 20% and 80% limits with a more significant rise below 20%. This confirms that under-fitting is a more substantial issue than over-fitting.

4.4.3 Random forest

Before discussing the results, the first point to tackle is the training time of the Random forest classifier. Effectively, training a random forest model takes significantly longer than training an SVC or a one-class classifier. Even though it was not a concern in this study, it could pose a challenge in another scenario with larger datasets or when an exhaustive search for the parameters has to be carried out.

Raw results, research of adapted parameters

For each criterion, a slight increase in the n_iterators parameter results in the number of errors dropping below 2. There is an exception for the test set of the "Convolution 100 frames under Impersonation attack reduced" dataset, where the

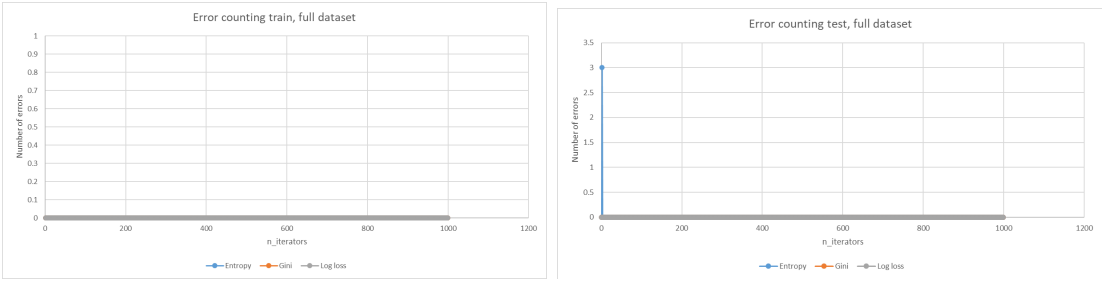

Figure 4.25: Raw results, Fuzzy attack, AES 1500 frames full dataset

n_iterators must be significantly increased before the model converges and the number of errors remains at 6.

Hence, the selected criterion is the gini one and the n_iterators is set to 400. This last parameter is chosen particularly high to ensure the model converges correctly.

Final results

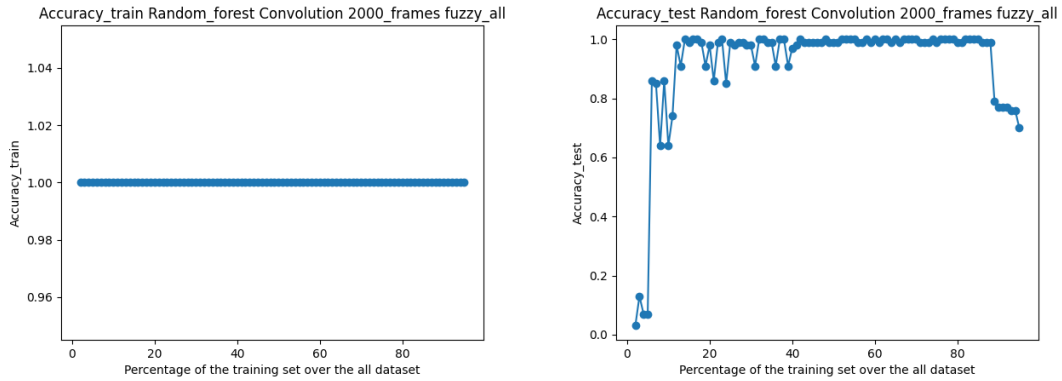

Figure 4.26: Accuracy final results, Fuzzy attack, Convolution 2000 frames full dataset

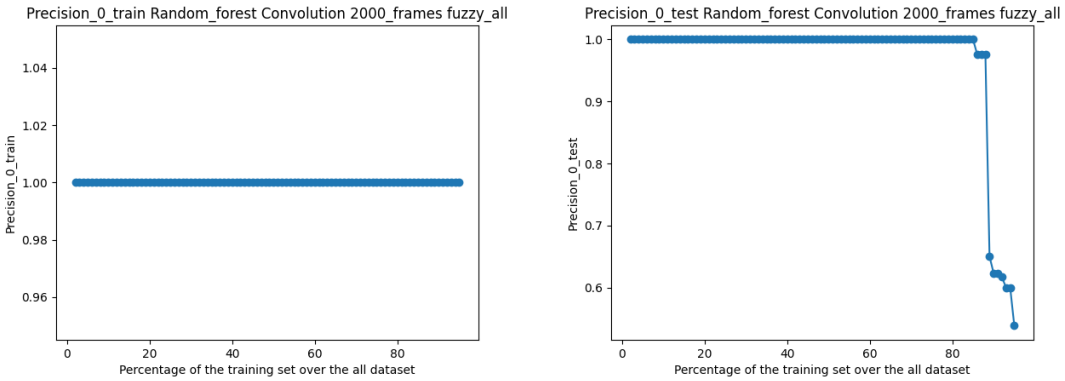

Figure 4.27: Precision final results normal traffic, Fuzzy attack, Convolution 2000 frames full dataset

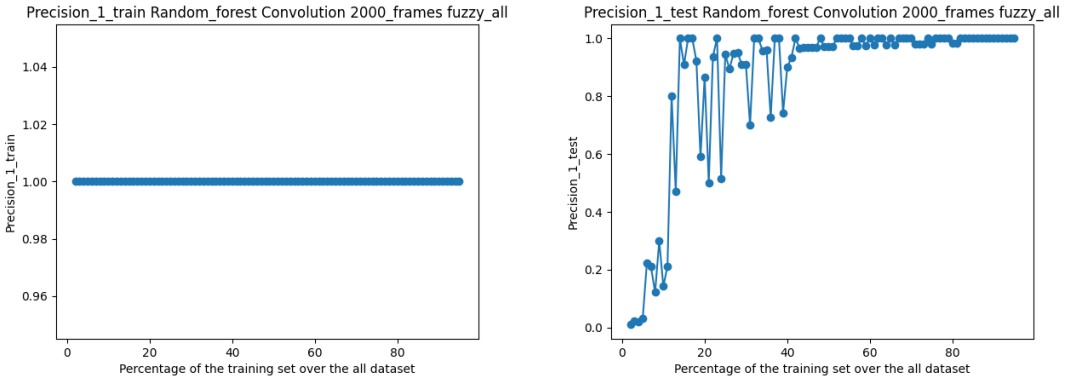

Figure 4.28: Precision final results malicious traffic, Fuzzy attack, Convolution 2000 frames full dataset

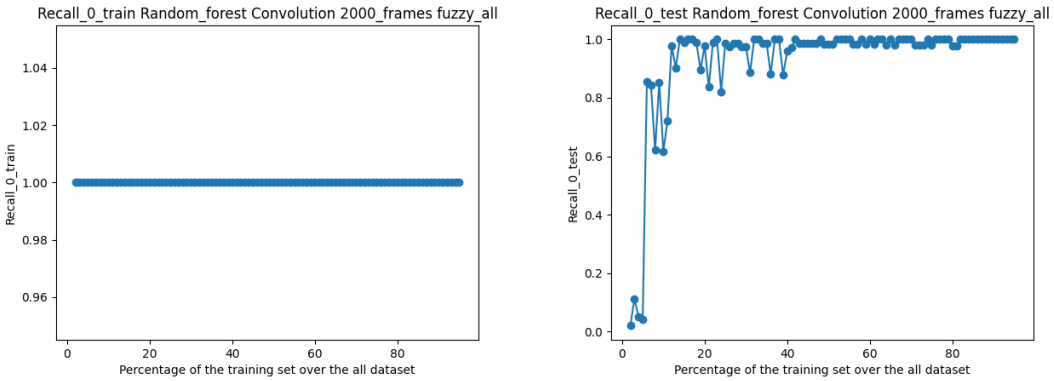

Figure 4.29: Recall final results normal traffic, Fuzzy attack, Convolution 2000 frames full dataset

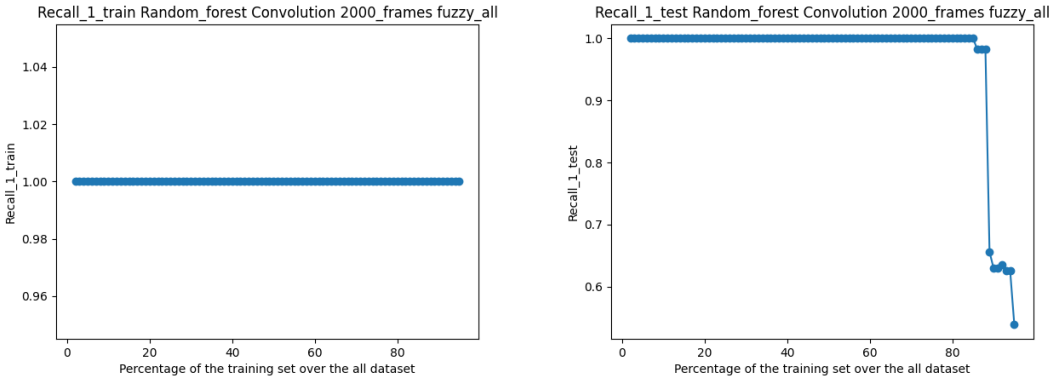

Figure 4.30: Recall final results malicious traffic, Fuzzy attack, Convolution 2000 frames full dataset

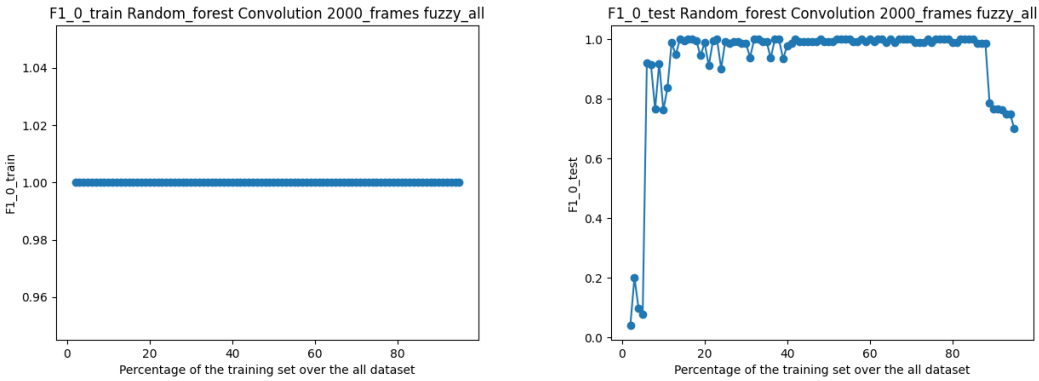

Figure 4.31: F1 final results normal traffic, Fuzzy attack, Convolution 2000 frames full dataset

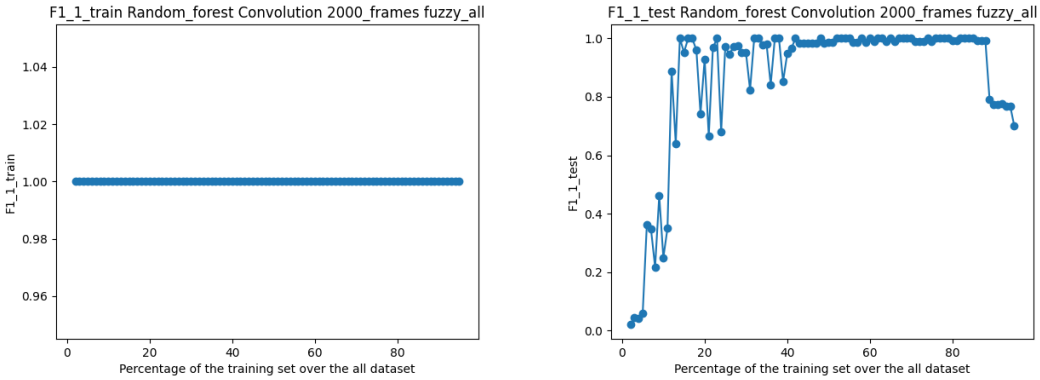

Figure 4.32: F1 final results malicious traffic, Fuzzy attack, Convolution 2000 frames full dataset

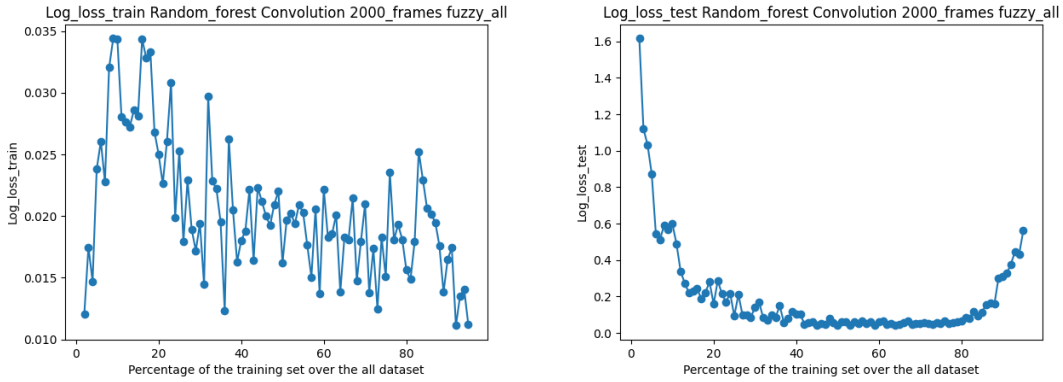

Figure 4.33: Log loss final results, Fuzzy attack, Convolution 2000 frames full dataset

	DoS	Fuzzy	Impersonation
Accuracy	1	0.99	0.99
Precision Normal traffic	1	1	0.98
Precision Malicious traffic	1	0.971	1
Recall Normal traffic	1	0.985	1
Recall Malicious traffic	1	1	0.98
F1 Normal traffic	1	0.992	0.99
F1 Malicious traffic	1	0.986	0.99
Log loss	0.001085	0.0444	0.00777

Table 4.3: Table of test results for Convolution 2000 frames full dataset, where the ratio is equal to 50%

The Random forest demonstrates superior results across all the datasets compared to the SVC. Nonetheless, both under-fitting and over-fitting processes are still noticeable on the test set, with a more pronounced influence for the under-fitting; these phenomena are imperceptible on the train set. The manifestation of these tendencies is also apparent through the log loss metric which increases when the fraction of the training set is too low or too high.

4.4.4 Multiclass SVC

Raw results, research of adapted parameters

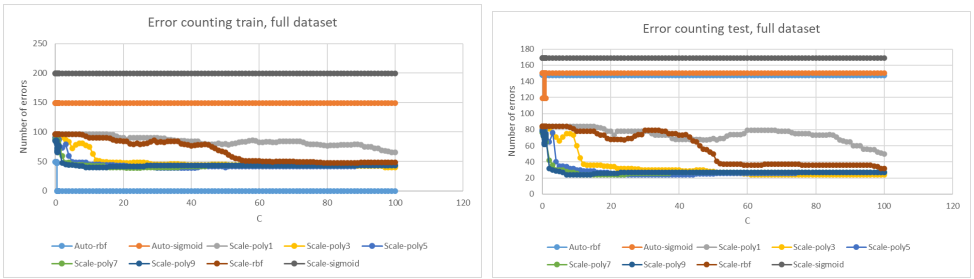

Figure 4.34: Raw results, AES 1500 frames full dataset

Across all the datasets, the parameters offering the greatest results are 100 for C, rbf for the kernel, and scale for the gamma. While these parameters proved to be the most suitable on average, certain tested datasets reveal instances where the classifier is unable to produce accurate results.

Final results

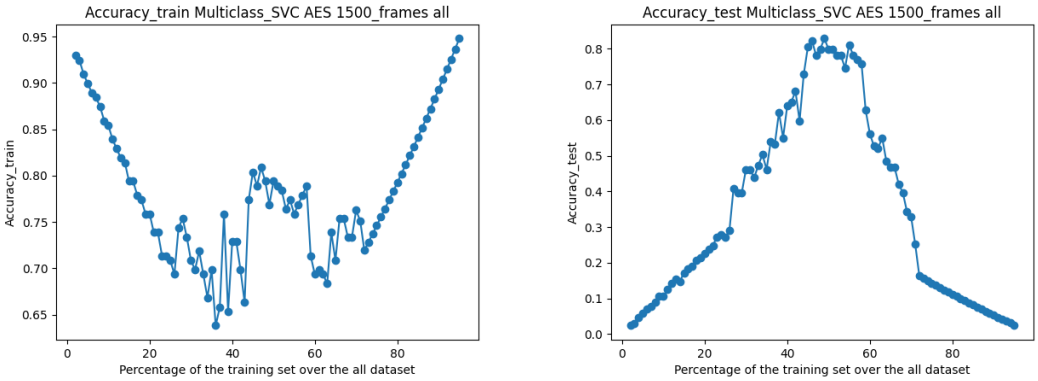

Figure 4.35: Accuracy final results, AES 1500 frames full dataset

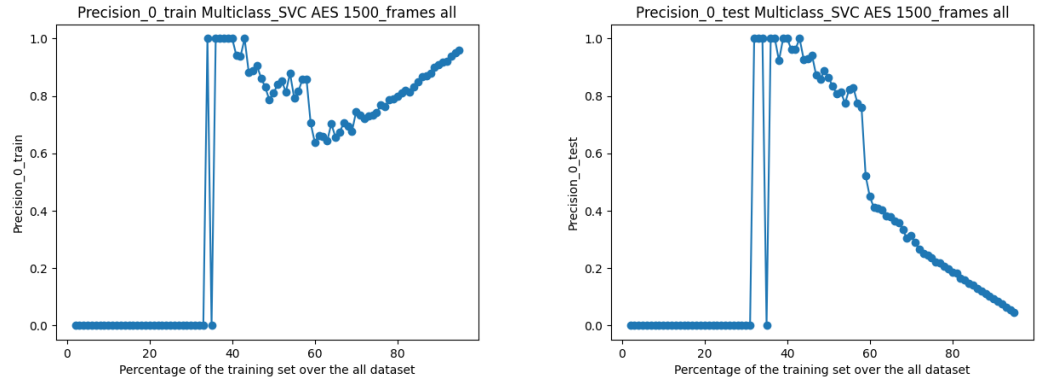

Figure 4.36: Precision final results normal traffic, AES 1500 frames full dataset

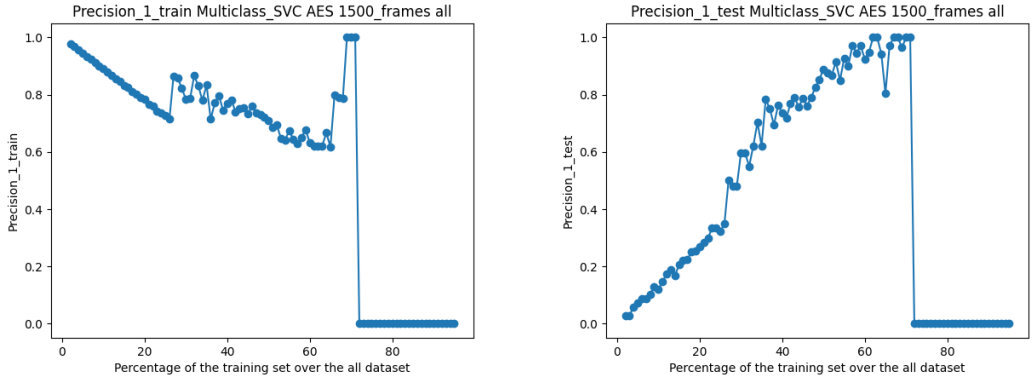

Figure 4.37: Precision final results DoS attack, AES 1500 full dataset

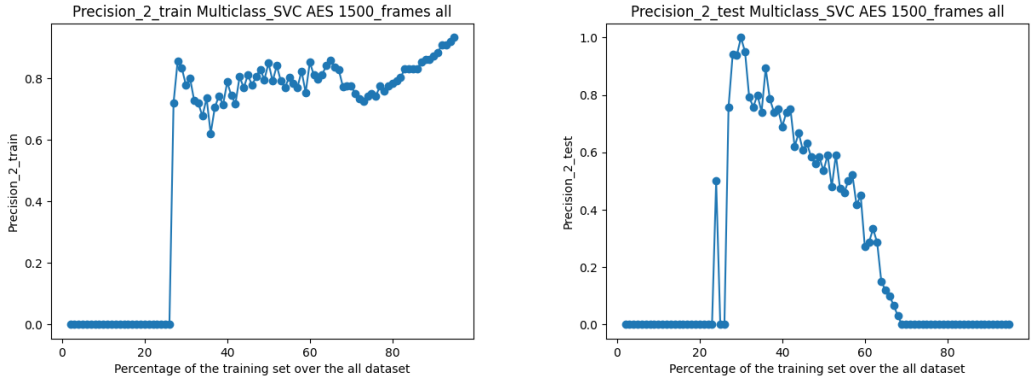

Figure 4.38: Precision final results Fuzzy attack, AES 1500 full dataset

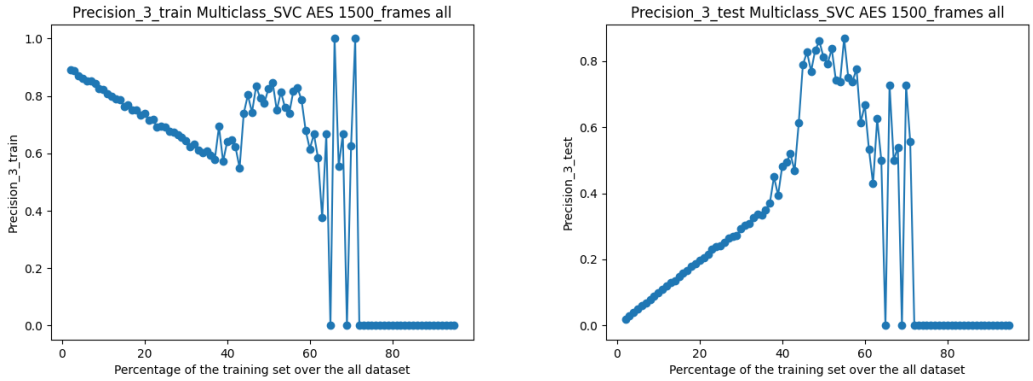

Figure 4.39: Precision final results Impersonation attack, AES 1500 full dataset

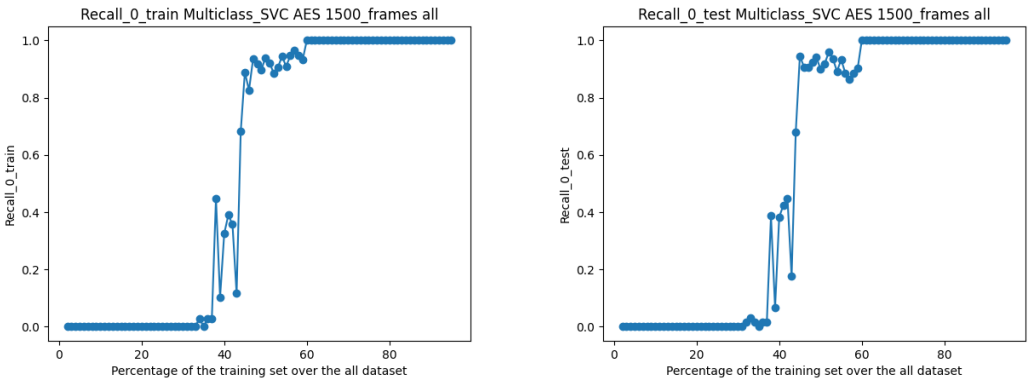

Figure 4.40: Recall final results normal traffic, AES 1500 frames full dataset

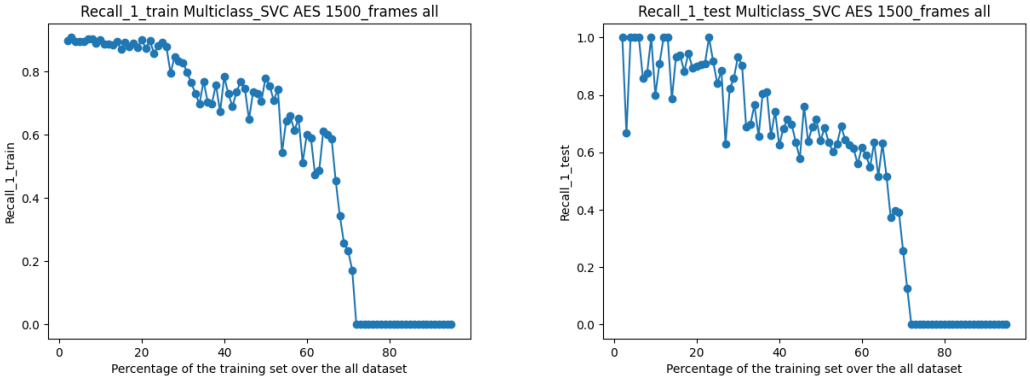

Figure 4.41: Recall final results DoS attack, AES 1500 frames full dataset

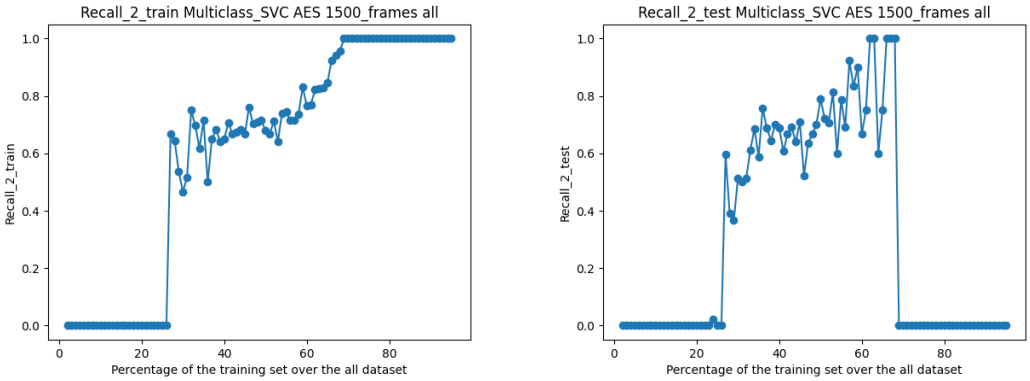

Figure 4.42: Recall final results Fuzzy attack, AES 1500 frames full dataset

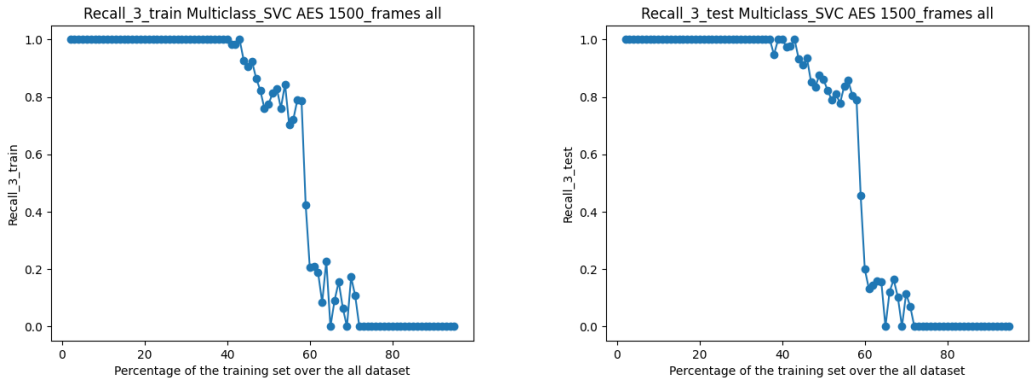

Figure 4.43: Recall final results Impersonation attack, AES 1500 frames full dataset

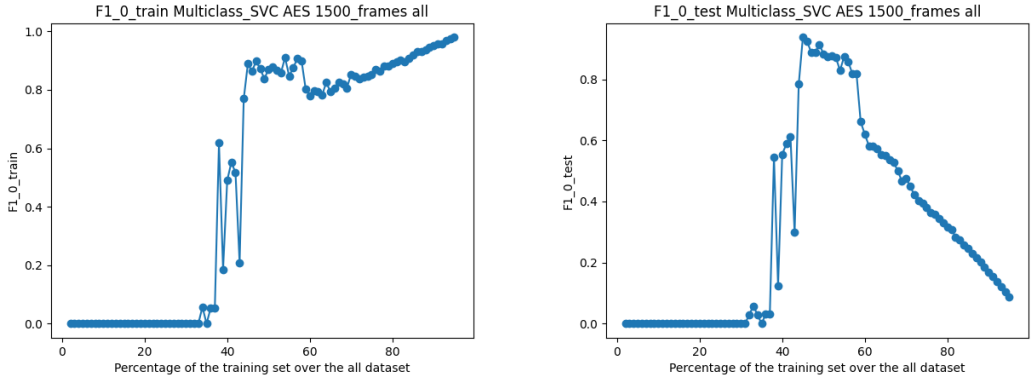

Figure 4.44: F1 final results normal traffic, AES 1500 frames full dataset

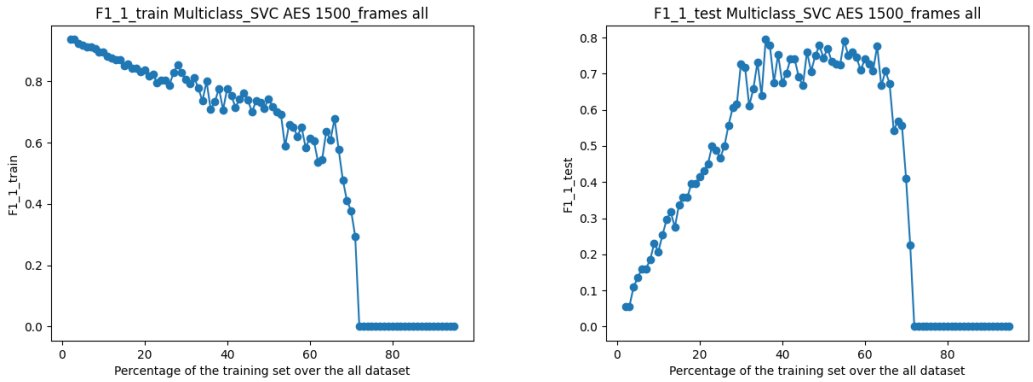

Figure 4.45: F1 final results DoS attack, AES 1500 frames full dataset

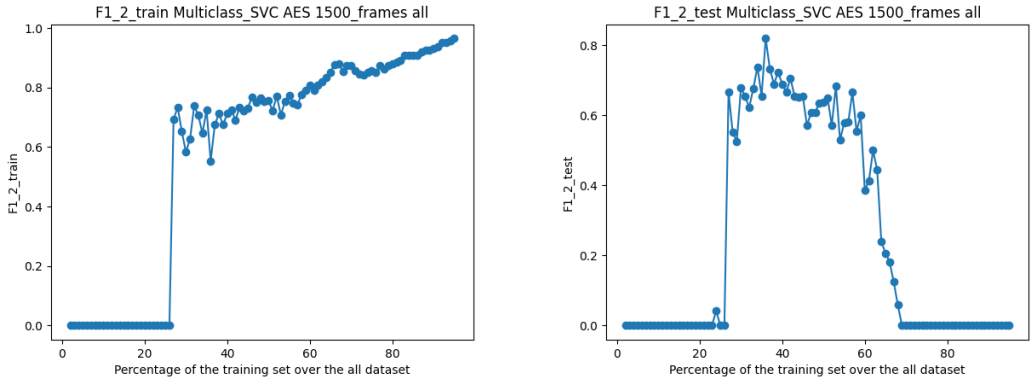

Figure 4.46: F1 final results Fuzzy attack, AES 1500 frames full dataset

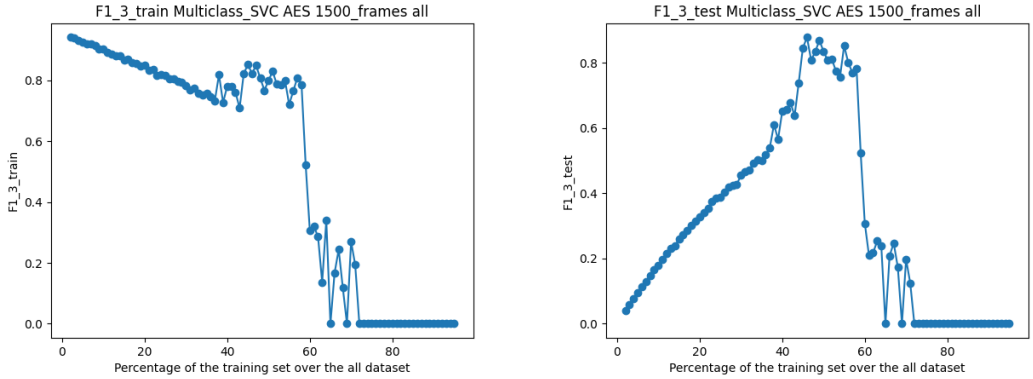

Figure 4.47: F1 final results Impersonation attack, AES 1500 frames full dataset

	Normal traffic	DoS	Fuzzy	Impersonation
Accuracy	0.7989	same	same	same
Precision	0.865	0.889	0.536	0.811
Recall	0.9	0.64	0.789	0.86
F1	0.882	0.744	0.638	0.835

Table 4.4: Table of test results for AES 1500 frames full dataset, where the ratio is equal to 50%

An initial observation regarding this classifier is that the results vary based on the dataset and the function used. Specifically, when convolution is applied the results are optimal for both full and reduced datasets. In contrast, for AES, the mean and scale one performs the best, followed by the full dataset, and lastly, the reduced dataset exhibits poor efficacy. Furthermore, the results obtained from the correlation reduced dataset are not viable due to the insufficient number of data points.

As a result, the conclusions drawn in this paragraph are only applicable on average and may not be relevant in every studied situation. It is noticeable that the normal traffic is better classified than the traffic under attack. Among the various attacks, Fuzzy is the most easily detectable, then the DoS, and finally, the Impersonation. One can note that the results of DoS and Impersonation are quite close, reflecting the similarity in their behavior. In both cases, the receiver observes new attack frames with a specific identifier, where the DoS attack uses 0, and the Impersonation 0x164.

However, a general comment can be made across all the observed situations. The best results are constantly achieved when the ratio between the training set size and the entire dataset size is close to 50%. Similar to the other presented classifiers, this phenomenon could be caused by the under-fitting and over-fitting mechanisms.

4.4.5 Multiclass Random forest

Similar to the dual-class random forest and SVC models, the training time for the multiclass random forest is significantly higher than the one for the multiclass SVC.

Raw results, research of adapted parameters

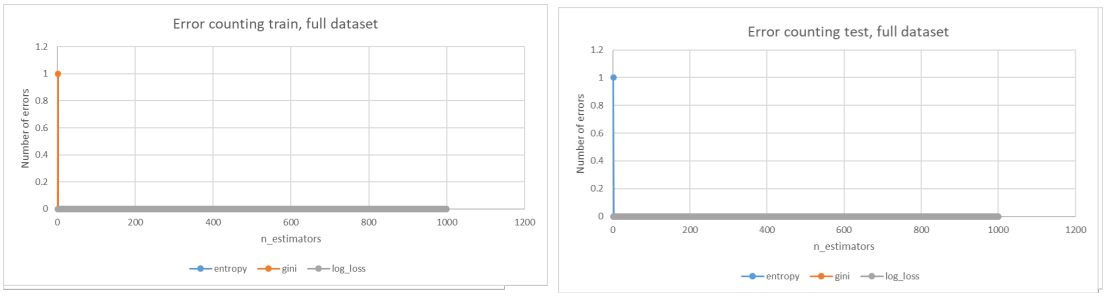

Figure 4.48: Raw results, AES 1500 frames full dataset

Alike the dual-class random forest for the dual-class SVC, the multiclass random forest classifier outperforms the multiclass SVC. Additionally, as seen in the precedently discussed Random forest, the number of errors decreases significantly with a slight increase in n_iterators, except for two situations (Convolution, 2000 frames, mean and scale, and correlation reduced datasets) where the number of errors remains high.

The chosen parameters are 400 for n_iterators to ensure that the model can converge and log_loss as a criterion which has slightly better results.

Final results

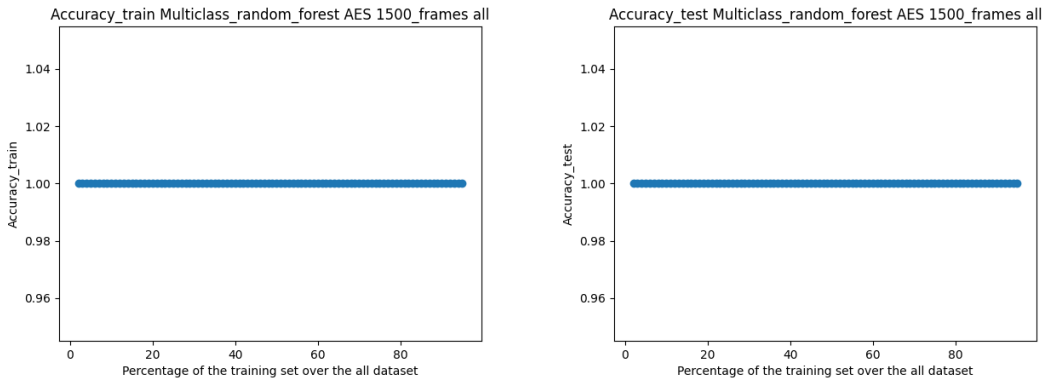

Figure 4.49: Accuracy final results, AES 1500 frames full dataset

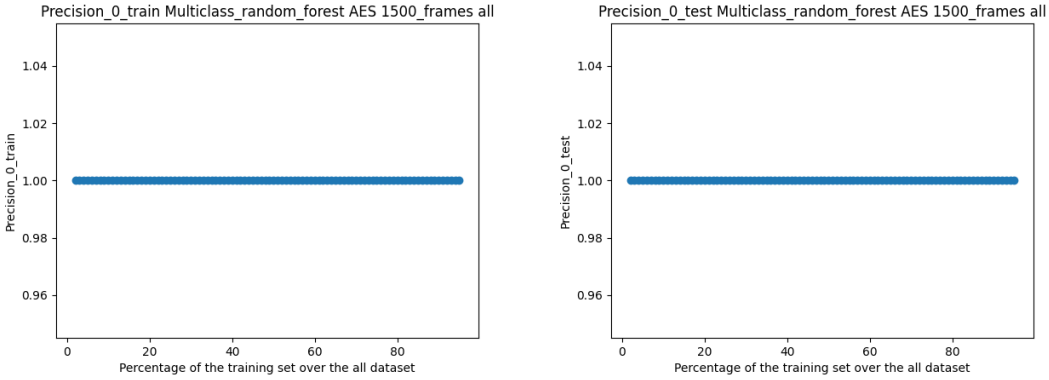

Figure 4.50: Precision final results normal traffic, AES 1500 frames full dataset

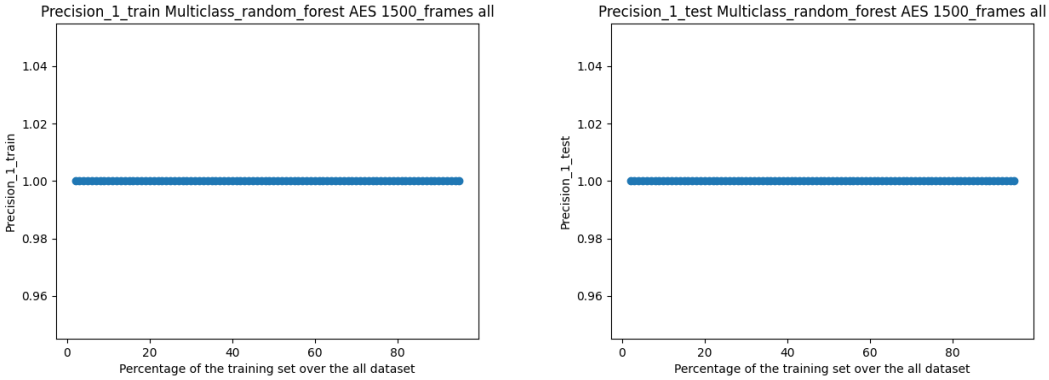

Figure 4.51: Precision final results DoS attack, AES 1500 full dataset

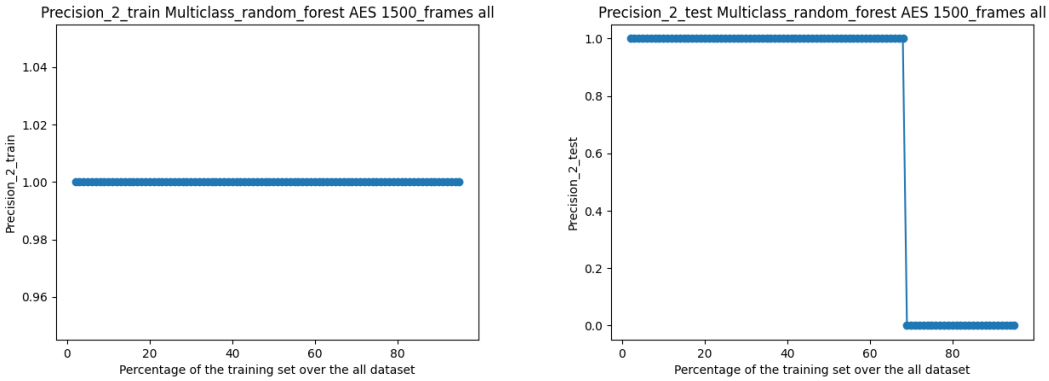

Figure 4.52: Precision final results Fuzzy attack, AES 1500 full dataset

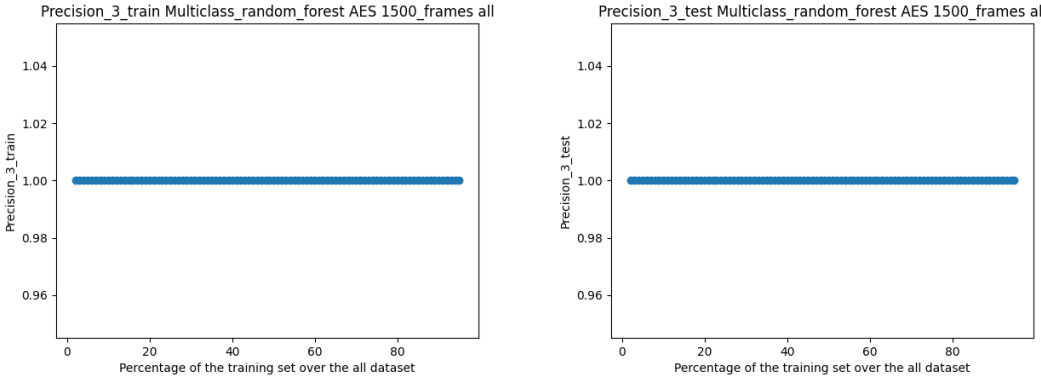

Figure 4.53: Precision final results Impersonation attack, AES 1500 full dataset

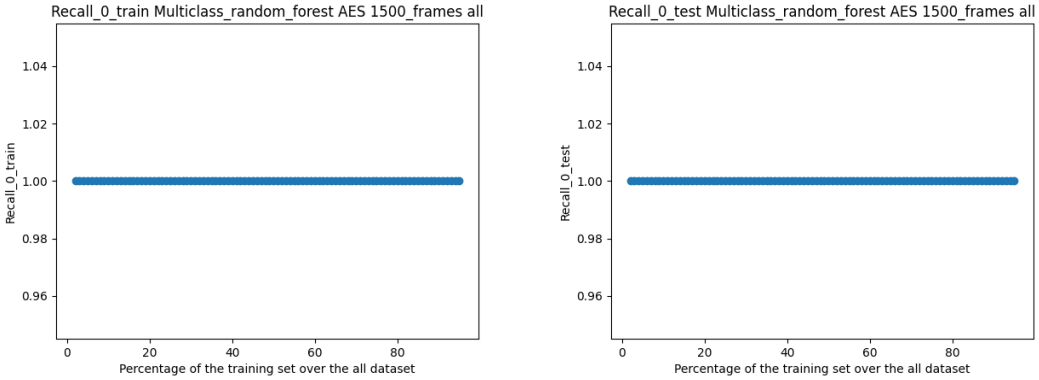

Figure 4.54: Recall final results normal traffic, AES 1500 frames full dataset

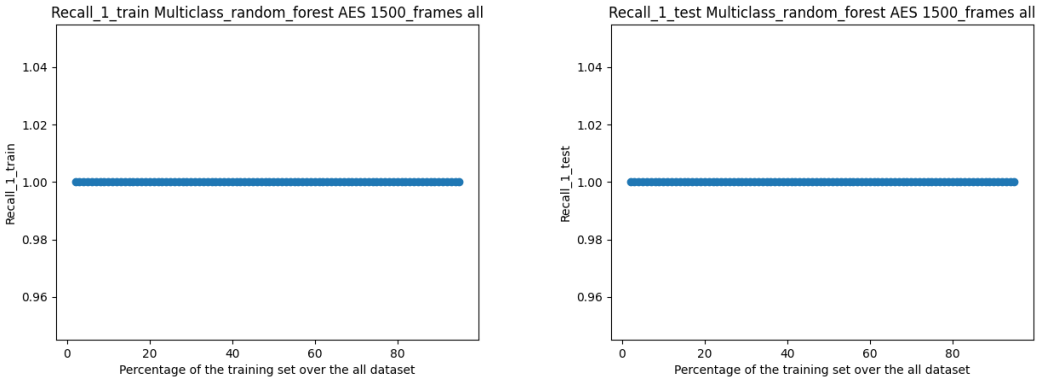

Figure 4.55: Recall final results DoS attack, AES 1500 frames full dataset

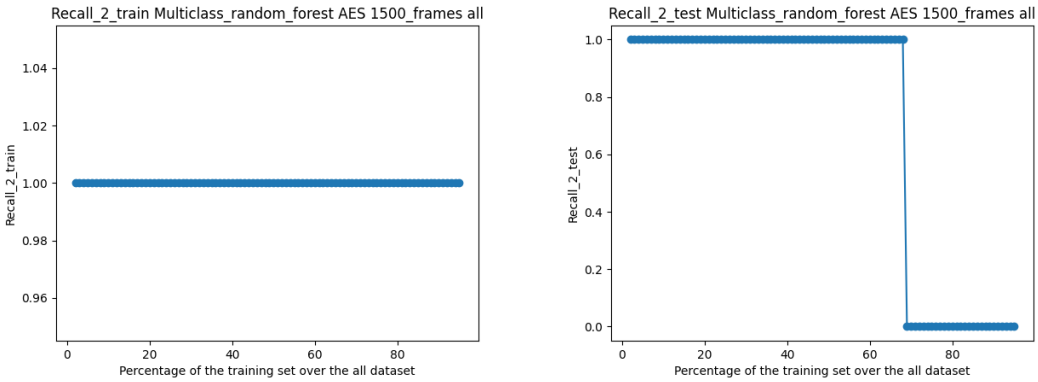

Figure 4.56: Recall final results Fuzzy attack, AES 1500 frames full dataset

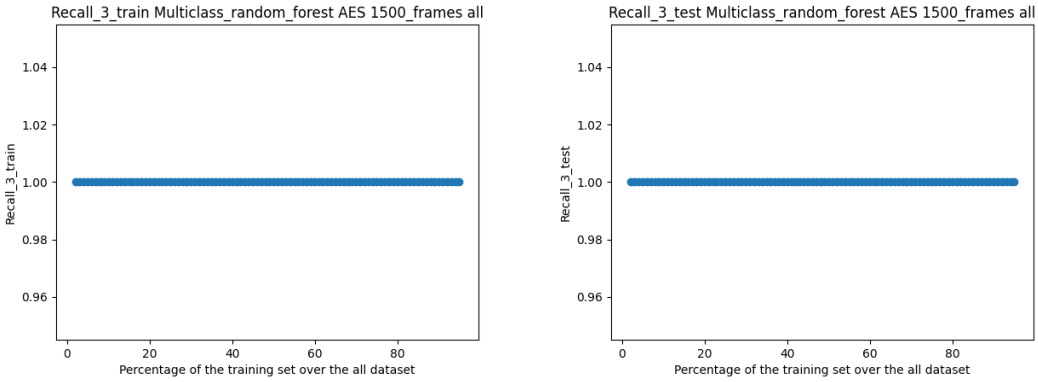

Figure 4.57: Recall final results Impersonation attack, AES 1500 frames full dataset

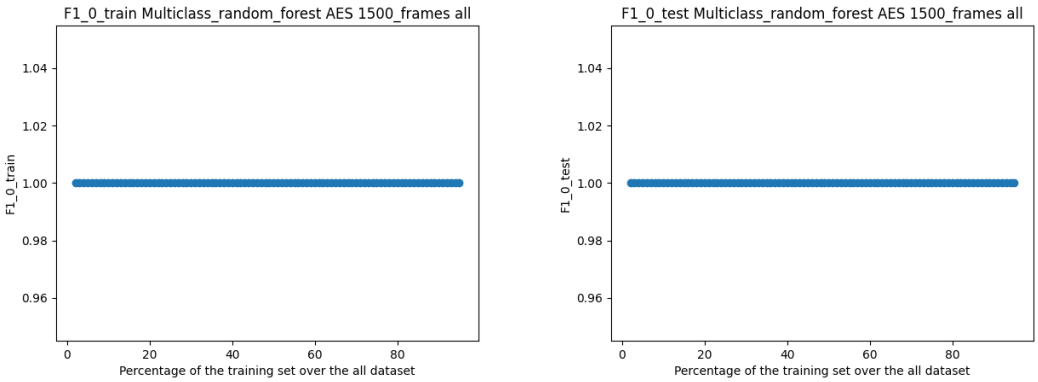

Figure 4.58: F1 final results normal traffic, AES 1500 frames full dataset

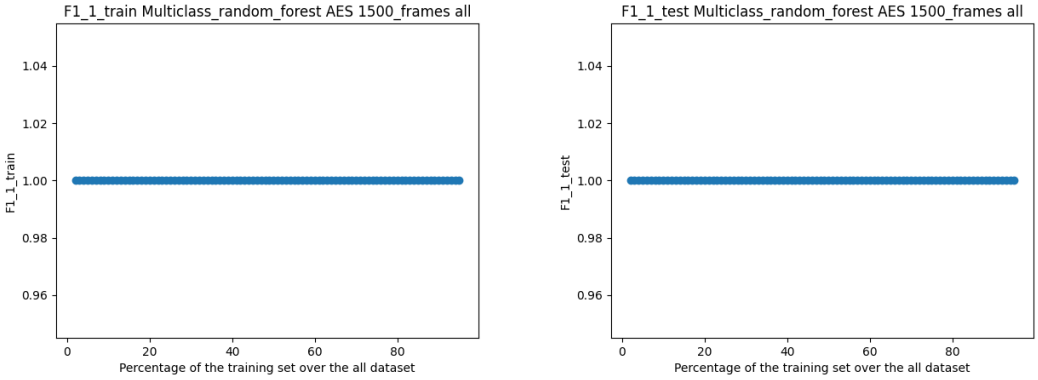

Figure 4.59: F1 final results DoS attack, AES 1500 frames full dataset

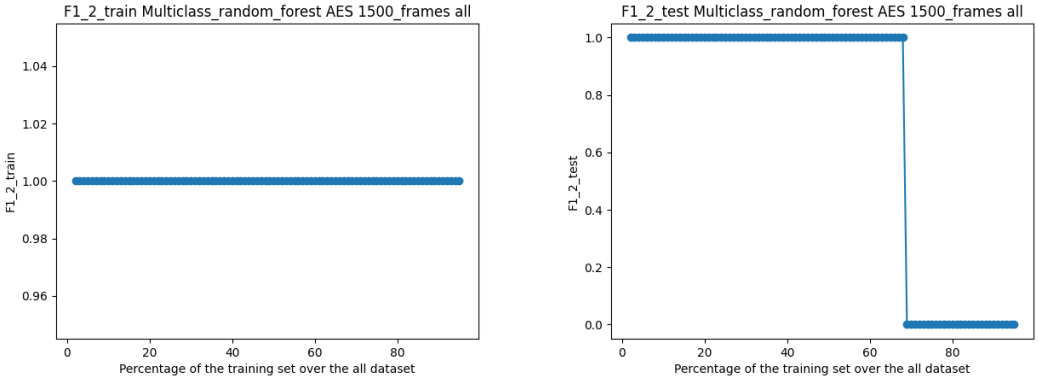

Figure 4.60: F1 final results Fuzzy attack, AES 1500 frames full dataset

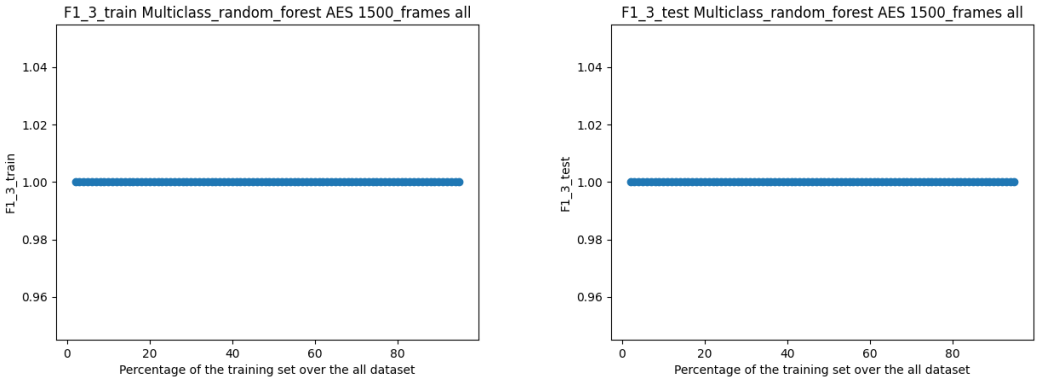

Figure 4.61: F1 final results Impersonation attack, AES 1500 frames full dataset

	Normal traffic	DoS	Fuzzy	Impersonation
Accuracy	1	same	same	same
Precision	1	1	1	1
Recall	1	1	1	1
F1	1	1	1	1

Table 4.5: Table of test results for AES 1500 frames full dataset, where the ratio is equal to 50%

The multiclass random forest surpasses the multiclass SVC achieving excellent classifications for most tested datasets and nearly all ratio combinations of the training set size over the entire set size, except for two scenarios. The first one is observed in the mean and scale dataset using the AES function where the results are good only when the ratio is around 45%. The second problematic dataset is the mean and scale dataset using the Convolution function; this time, the classifications are consistently incorrect.

Moreover, the under-fitting and over-fitting processes are apparent in a few situations but they have less importance than in the previously presented multiclass model.

Similar to the multiclass SVC, the correlation reduced dataset cannot be exploited due to the insufficient number of points.

4.4.6 Comparison results attacks

The objective of this section is to evaluate the quality of the classifications concerning the different attacks. The main used metric is F1 which synthesizes information from both precision and recall metrics for each class.

The comparison will be conducted for each type of classifier.

Comparision for dual-class classifiers

As previously presented, the results for the One-class classifier are generally not satisfactory. The variations in the results based on the ratio between the training set size and the entire set are approximately consistent for each kind of attack. On average, the results of DoS and Impersonation are close and the ones of the Fuzzy intrusion are slightly worse than for the two other attacks.

Mechanisms of under-fitting and over-fitting are also noticeable. The models trained on DoS and Impersonation exhibit slight overfitting, while an under-fitting process is evident in certain datasets, such as the AES 1500 frames mean and scale dataset. This pattern has a more negative effect on the normal traffic classification

than on the under-attack classification.

The SVC results are significantly better than the One-class ones on average. However, some datasets, such as the AES 75 frames reduced dataset and AES 1500 frames reduced dataset, exhibit poor classification capabilities, and in these cases, the Fuzzy attack models outperform the other attacks.

For certain datasets, especially AES full and reduced, the performances are higher for the Fuzzy and DoS attacks than for the Impersonation, even though they are acceptable when the percentage of the training set is around 50%. In cases where the results are good for Impersonation, the log loss metric is very high indicating that the model is not confident in its predictions.

In the Convolution Full and Reduced datasets, the results are strongly less encouraging for the Fuzzy attack. The Mean and scale and Correlation reduced operations allow for improvement and similar performances on each type of attack.

The Random forest results are generally very good for all kinds of attacks. However, over-fitting appears for DoS and Fuzzy attacks but this phenomenon is not consistent across all datasets.

Comparision for multiclass classifiers

The results for the Multiclass SVC for the AES function are not optimal. When the training ratio is around 50%, with the normal traffic and the Fuzzy attack the obtained results are relatively correct but they remain underwhelming for the DoS and poor for the Impersonation for the AES 75 frames Reduced dataset. Some datasets exhibit acceptable results for the Full dataset and good results for the Mean and scale one when the ratio is close to 50%. These performances are slightly better for the normal traffic than for the other attacks.

For the Convolution function, the classifications are accurate for the Reduced and the Full dataset and slightly less satisfactory for the Mean and scale dataset. Upon analyzing the attack, it can be observed that all the classes are properly classified except for the Fuzzy.

In the case of the Random forest, the elements are classed precisely for every set excluding the Mean and scale one. In this particular set, the normal traffic is particularly mislabeled. For the other ones, the normal traffic and the Fuzzy are less properly classified compared to the other classes.

Conclusion

On average, there is a similarity in the performances between the DoS and the Impersonation. The closeness between the attacks can explain this similarity in

the results. Indeed, in both cases, the receiver observes "normal" frames and one specific type of malicious frames with the ID 0 for the DoS and the ID 0x164 for the Impersonation. On the other hand, the Fuzzy attack varies the ID of the attacker frames leading to a different traffic "shape". Additionally, the normal traffic has frequently different results from the attacks.

4.4.7 Comparison results functions

Comparision for dual-class classifiers

No major influence of the functions can be noticed for the One-class classifier, except for the Convolution in the Mean and scale dataset for the Impersonation attack.

The classification performances of datasets with a small number of frames (75 frames for AES and 100 frames for Convolution) are significantly influenced by the function used to train the SVC model. While the classification for the Convolution function is well performed, it is not as satisfactory for AES. A notable observation is that, for the Impersonation with the Convolution function, good results can only be achieved when the ratio is very close to 50%.

The classification results for datasets with a large number of frames (1500 frames for AES and 2000 frames for Convolution) exhibit significant dependence on the dataset type and the specific attack being studied. When using the Full dataset, both functions achieve correct performances when the training ratio is near 50% for DoS and Impersonation, with Convolution slightly outperforming. The models trained on the Fuzzy attack perform well with the AES function but not with the Convolution. The Mean and scale and the Correlation reduced datasets, for both functions, yield good results; however, the ratio needs to be around 50% for the Impersonation attack when using the Convolution. For the reduced datasets, no correct results can be achieved for the AES function. Convolution permit to obtaining good performances for DoS and Impersonation attacks but not for the Fuzzy one.

For the Random forest with the small number of frames set, no significant difference can be observed between the two functions.

With the large number of frames set, the same results are obtained for the both Full and the Reduced datasets, although they are slighty lower with the Convolution function. The same comment can be made with the Mean and scale and Correlation reduced sets, except for the Impersonation, for which the ratio needs to be very close to 50%.

Comparison for multiclass classifiers

In the case of Multiclass SVC and the small number of frames datasets, Convolution demonstrates superior results compared to the AES when the ratio is approximately 50%.

For the large number of frames datasets with a Full dataset, all the attacks exhibit great performances for both AES and Convolution, especially around 50%, except the Fuzzy attack with Convolution. While AES performs well with the Mean and scale dataset, Convolution does not yield satisfactory results. In the case of the Reduced dataset, the Convolution and the AES functions work optimally when the ratio is very close to 50% except for the Fuzzy attack for Convolution and the Impersonation for AES.

There is no significant discrepancy observed for the Multiclass Random forest with the small number of frames set.

For the large number of frames set, using the Full and Reduced datasets very good results are achieved for both functions, even though the AES performs slighty better. However, the Mean and scale set enables good performances for AES when the ratio is centered on 50% but performs poorly when the Convolution function is used.

Conclusion

Overall, the Convolution function tends to perform better for the SVC classifiers with few exceptions such as the Multiclass SVC where the models trained with this function sometimes fail to perform adequately. Conversely, with Random forest, generally, the AES provides better results.

An important difference between the two functions is that the proposed AES only processes payloads with a size of 8 bytes while the Convolution operates on all the different payload sizes.

4.4.8 Comparison results number of frames

The comparison made in this section will focus only on the small number of frames Reduced dataset and the large number of frames Reduced dataset. This limitation arises from the availability of only Reduced datasets for the small number of frames.

Comparison for dual-class classifiers

The One-class classifier continues to obtain consistently poor results overall. Although the difference between the two observed sets remains small, the small number of frames demonstrates slightly improved performances in the classification

of the normal traffic and the Impersonation attack.

For the SVC model, there is no particular discrepancy between the datasets for the AES function and the Convolution with DoS and Impersonation attacks. However, it is noticeable that the small number of frames set performs largely better with the Convolution for Fuzzy attack.

In all the studied scenarios with the Random forest model, the results are consistently very good. The only notable variations between the sets are the occurrences of the under-fitting and over-fitting processes.

Comparison for multiclass classifiers

In both situations, when using AES with the Multiclass SVC classifier, only poor results are obtained. However, with Convolution, better performance is observed for the small number of frames set, but the ratio needs to be close to 50%. Moreover, the classification for the Fuzzy attack is not accurately achieved with a large number of frames dataset.

For the Multiclass Random forest, the small number of frames set achieved lightly smaller performances.

Conclusion

When using the Convolution function, to train the SVC models, the small number of frame datasets tends to yield better results. Conversely, for the Random forest ones, no major difference can be observed apart from the emergence of under-fitting and over-fitting phenomena. In the Multiclass case, the small number of frames set shows slightly better performances. Overall, the small number of frames dataset has better performance but it is situational and not the most influential parameter.

4.4.9 Comparison results datasets

The analysis in this section will be done solely on the large number of frames set because it is the only dataset ensemble that contains various types of results datasets for comparison.

Comparison for dual-class classifiers

For the One-class model, there is no significant difference between the Reduced and the Full datasets. However, the Mean and scale operation enhances the detection

of normal traffic while compromising the attack classification.

The SVC classifier exhibits excellent performance with the AES function across the Full, Mean and scale, and Correlation reduced datasets, but struggles with the Reduced set. Interestingly, the results for the Mean and scale and Correlation reduced datasets are quite similar.

When utilizing the Convolution function, notable performance can be noted for the Full and the Reduced datasets when the model is trained on the DoS and the Impersonation attacks. However, the performance largely declines for the Fuzzy attack. Both the Mean and scale and the Correlation reduced sets obtained approximately the same results except that for Impersonation a second peak of good results is visible around 20%.

For the Random forest model, both the Full and the Reduced demonstrate the same results, likewise between the Mean and scale and the Correlation reduced sets. There is a notable deviation for the Mean and scale using the Convolution function where a second peak of correct results emerges near 15%.

Comparison for multiclass classifiers

For the Multiclass SVC, using the AES function the Reduced dataset gets largely worse results than the Full dataset, especially for the attacks classifications. The Mean and scale one permits performance improvement but only when the ratio is around 50%. Conversely, with the Convolution function, comparable results are observed for the Full and the Reduced datasets. Moreover, the Mean and scale operation reduced strongly the performances.

Overall, the datasets perform very well for the AES function, even if it can be observed that the Mean and scale dataset needs the ratio to be centered on 50%. Similar trends are visible with the Convolution function for the Full and the Reduced sets. However, the Mean and scale one severely decreases the performances.

Conclusion

A general observation can be achieved, a significant similitude is present between the Full and the Reduced datasets. Similarly, the Mean and scale and the Correlation reduced sets yield similar results.

Moreover, on average with multiclass classifiers, the Mean and scale datasets reduce the performances.

Chapter 5

Conclusion

5.1 Discussion of the developed system

The IDS was developed on a simplify and half-duplex version of the CAN protocol. Consequently, the conclusions drawn from this analysis should be interpreted with caution. The same experiences with a full-duplex communication may trigger different hardware events and alter the behaviour of the tested models.

Moreover, the study was only conducted on an unique KIA SOUL car. Since the ECUs presented on a car varying strongly depending the vehicle's model, the study has to be reproduced on multiple vehicles to adapt the IDS.

The obtained results are strongly dependent of the type of classifier used and the specific situation tested. In this study, the unsupervised classifier exhibits really poor performances, while the supervised ones are globally correct or even very good. Among the dual-class case, the SVC and the Random forest perform well while for the multiclass experiments only the Random forest is usable.

In every situation and for each supervised classifier tested, the highest performances are obtained when the ratio between the number of training samples and the total number of samples is close to 50%. Otherwise in some cases, processes of under-fitting and overfitting appear.

In certain circumstances, classification performances can strongly vary with a small perturbations in the ratio. This may be due to the randomisation of the training and testing sets.

Some datasets present perfect classification results no matter the ratio used. A perfectly adapted classifier could be a model which overfits with the provided data and which will not be able to achieve correct results with another set. A solution could be to train these models with a slightly perturbed dataset to avoid perfect accommodation to the training set.

On average, better results were observed for datasets with a larger number of

samples and shorter simulation times per sample compared to datasets with fewer samples and longer simulation times per sample. Consequently to enhance the IDS, it is imperative to construct larger datasets with a quite short simulation time. However by reducing the simulation time, the number of triggered hardware events decreases. Therefore, less HPCs are incremented during this duration and a new selection of the parameters must be achieved. A solution may be to train the classifiers on set of HPCs for which not all the studied events are triggered.

As a result, the obtained IDS functions as a signature detection algorithm specialized only for the studied attacks. However, a limitation of this project is that it did not encompass enough diverse threat scenarios to be applicable in a real situation. By consequence, the dataset must be enlarged.

Two transformations were applied to the parameters: a mean and scale transformation to reduce the scattering and a reducing of the number of parameter based of the correlation with the class of the traffic. These transformations appears effective in certain situations where the results vary significantly depending the studied attack; the transformations permit to achieving more similar performances. However, both perform approximately the same; consequently it seems that the correlation reducing operation is not very efficient.

The functions considered in the project are not very impactful on the classifiers' results. Even when one performs better than the other, it is not very significant. Nevertheless, it can not be definitively concluded that functions have no effect on the classification, only were experimented. A manner to obtain more practical results would be to use functions of a real CAN device.

On the explored attacks and with a quite large number of frames (between 75 and 2000 frames) for each sample, good results have been obtained. However, it is important to note that real attacks are often more complex and largely shorter in duration.

Moreover, the Gem5 tool only provides logs corresponding to the HPCs. Although they corresponds to simulated hardware events, they may have a different behaviour than the counters presented in a real board. Additionally, in RISC-V, only 29 performance counters are available, which is significantly less than in the studied datasets.

5.2 Milk-V Duo implementation

As described in a previous chapter, a hardware solution was explored but not ultimately selected due to various challenges. Firstly, the CAN library was implemented for K210 but this board lacks a performance management unit for non-architectural events. Subsequently, the Milk-V Duo appeared as a potential solution. Indeed, its advantages include its affordability and its management of

multiple non-architectural HPCs.

Nevertheless, this solution has limitations. The only software available on the Milk-V Duo board is an embedded Linux with multiple levels of execution privileges. Unfortunately, the user privilege mode, in which the application program runs, has no access to the registers storing the HPCs. Several modifications to the Linux OS need to be achieved to access the counters or a new program has to be developed from scratch.

Despite these challenges, a Milk-V Duo CAN implementation could provide full-duplex CAN communication on a real hardware board. The low price and the good performance of the two-core processor make the Duo board a good candidate for CAN controller devices. Three solutions are, thus, proposed hereafter.

5.2.1 SBI modification

The implemented Linux software operates with 3 levels of execution privileges, corresponding to the 3 privilege modes of the RISC-V architecture. Each level has specific tasks to realize and has access to different reachable resources. When a user program requires to access an M-mode-only or S-mode-only register or any other resource of this type, it may utilize interfaces.

The user program employs system calls (syscalls) to get the OS kernel resources typically only attainable in S-mode. Additionally, if the kernel needs to reach the firmware resources another interface is used called the Supervisor Binary Interface (SBI). The SBI of Linux implementation provided for the Milk-V Duo is OpenSBI, an open-source RISC-V SBI implementation. Consequently, it can be modified and enhanced to enable access to the control register that contains the HPCs. By appropriately modifying OpenSBI and syscalls, the user program can access HPCs, rendering this board suitable for the study.

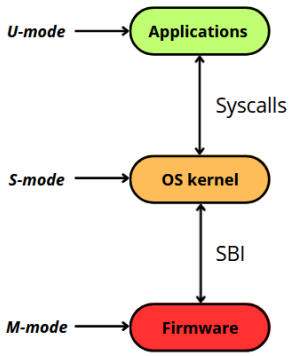

Figure 5.1: Privilege levels and interfaces of a RISC-V Linux port

5.2.2 Bare-metal integration

Another solution to address the problem of accessing the HPCs involves developing a bare-metal adaptation of the CAN library for the Milk-V Duo. This approach offers the benefit of giving the programmer complete access and full control over all the resources and devices on the board.

The bare-metal integration eliminates the need to modify the provided software and only requires developing features necessary for the study. It also enables an analysis of a simpler system that is closer to the hardware.

Nevertheless, this type of development requires "reinventing the wheel" by needing an implementation of simple well-known elements like a scheduler or an IO manager.

5.2.3 Second core with FreeRTOS

Milk-V also offers a FreeRTOS port which can operate on the second core. FreeRTOS has the advantage of running the user code in M-mode, permitting the reading and writing of the control registers. In other words, application programs have complete monitoring access to the HPCs.

Working with FreeRTOS on the second core involves compiling all Milk-V software whenever a program modification is made, which is very time-consuming. Hence, the implementation must be developed and tested on another medium before adapting to the FreeRTOS port.

Secondly, the second core does not have access to the UART peripheral, only the main core does. As a consequence, adjustments need to be made to the driver sharing data between the two cores to match the requirements of the project.

5.3 Real-time IDS

The study successfully implemented an IDS which gets very high performances using specific parameters. However, this current program is only able to detect attacks after a prolonged collection of HPCs; indeed, it cannot identify intrusions on the spot. Consequently, there is a need to develop a real-time IDS.

This system represents a new paradigm compared to the one implemented in the project. The process of collecting the training and the testing data has to be redesigned. Several solutions have to be tested like recovering the counters after a small number of frames, only a few ones, or during the receiving of a frame. A real-time system allows to attach to the counters set the timing at which the frame has been received. This new information may improve the performance of the IDS.

5.4 FreeRTOS implementation

The FreeRTOS implementation utilized in this project is not fully completed, it was just a reuse from another official RISC-V port which was quickly adapted to the Gem5 model. As a result, a lot of mechanisms are dysfunctional or unavailable. Notably, the scheduler does not perform properly; it cannot run two tasks simultaneously.

Enhancing the scheduler and the task management would allow to execution of two distinct tasks, one handling the CAN controller and the other computing the payload. A review of the Gem5 model may be necessary to improve its handling of the IOs and the resources.

5.5 Homemade traffic datasets

The entirety of the traffic dataset on which the analysis was carried out, normal traffic and attacks, came from another study. However, two alternative approaches could be undertaken to generate homemade datasets.

The first modification could be to start from the normal traffic used in the project and then modify or augment it by incorporating frames corresponding to the various attack scenarios. It offers the advantage of enabling the detection of new attacks on real traffic.

Another possibility is to recreate from scratch a whole dataset of frames; both for normal traffic and for attacks. This method offers more modularity and versatility.

5.6 CAN mechanisms not implemented

The mechanisms already in place are adequate for establishing half-duplex CAN communication. The next step involves full-duplex communication enabling which would facilitate the testing of new types of attacks and traffic patterns.

To establish this type of transmission effectively, additional mechanisms need to be integrated into the developed CAN implementation. These processes have various roles in error management, transmission validation, and traffic control. It's important to note that these mechanisms may introduce vulnerabilities that attackers could potentially exploit.

5.6.1 Remote frames

In certain scenarios, an ECU may need to request a message from another device within a network. In such a setup, the node will transmit a remote frame containing

a specific ID corresponding to the requested device. Upon receiving this remote frame, the target device will respond by sending back a frame with the same ID.

These two frames distinguish themself with a special reserved control field: the Remote Transmission Request (RTR) bit. When the RTR bit is equal to 1, the frame is a remote frame and when it is equal to 0, this is a data frame.

Figure 5.2: Position of the RTR field in the CAN frame

5.6.2 Error flags emission

When an error is detected within the CAN network, the Electronic Control Unit (ECU) responds based on its current state. If its state is in the "error active state" the node sends an active error flag, if it is in the "error passive state" it sends a passive error flag and if it is in the "bus off state" it does not respond and stop transmitting.

The active error flag is a sequence of 6 dominant bits sent just after the detected error. It aims to stop the transiting traffic and to signal to all the other ECUs that an error occurred.

The passive error flag is also sent just after the detected error but it transmits a sequence of 6 recessive bits. The purpose is to signal the error without stopping the traffic. The desire to not stop the traffic is due to the multiple errors previously made by the device; the detected error could effectively be caused by this device, in this situation the other nodes do not necessarily need to be warned.

These two flags can sometimes be larger than 6 bits long due to the longer time that the other nodes may need to detect the error. In this situation, the error flag can reach a size equal to 12 bits.

After the transmission of one of these flags, an error delimiter with a size of 8 bits is sent on the CAN bus.

5.6.3 Overload frames

In CAN communications, some devices may process frames at a slower rate than the CAN bus produces them. To address this issue, the ECU can trigger a mechanism named Overload frame to delay the emission of new frames. The purpose is to keep synchronization between the different nodes regardless of their performances and to permit the slowest ones not to miss some frames. While this mechanism was once useful, it has become somewhat obsolete with advancements in CAN device capabilities.

The Overload frame is inserted during the intermission following the transmission of a normal frame. It is composed of two parts: an overload flag and an overload delimiter. The first part is made of 6 dominant bits which can contain stuffed bits and the delimiter of 8 recessive bits. To prevent too long or infinite delay, a legitimate ECU is limited to sending only 2 overload frames.

However, this mechanism can be exploited by malicious nodes. Despite restrictions on overload frame transmission by legitimate ECUs, an attacker can still manipulate the process to cause indefinite delays in traffic. This poses a potential security risk that needs to be addressed in CAN networks.

5.6.4 Extended CAN frames

For heavy vehicles, a specific type of frame is known as the extended CAN frame. In these frames, the IDE bit is passed to 1 and the size of the identifier increases from 11 bits to 29 bits.

Extended frames are in particular used in the SAE J1939 standard, where the identifier is divided as follows:

- Bits 1 to 3: Priority bits
- Bit 4: Reserved bit
- Bit 5: Data page
- Bits 6 to 13: PDU format
- Bits 14 to 21: PDU specific
- Bits 22 to 29: Source address

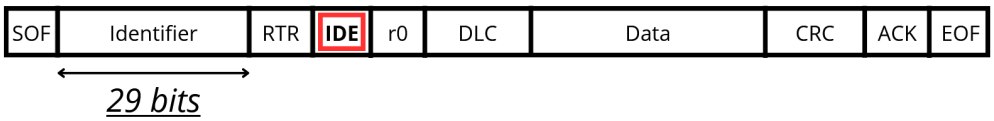

Figure 5.3: Position of the IDE field in the CAN extended frame

5.6.5 Acknowledgment

To confirm the successful reception of a frame, the CAN receiver acknowledges the message by altering the value of a bit within a designated field. Unlike other protocols, CAN does not require additional messages to confirm the reception.

When the sender transmits the frame, it sets the acknowledgment field to 1, indicating a recessive bit, and the receiver answers by changing the value to 0, a dominant bit, which overwrites the first value and warns all the nodes connected to the CAN bus that the message has been correctly received.

Figure 5.4: Position of the ACK field in the CAN frame

5.7 New attacks

As outlined in the preceding section, the implementation of full-duplex communication and additional mechanisms such as error flag emissions or overload frames opens up possibilities for new attacks on the CAN bus.

3 potential attacks could be interesting to study: the Error Passive Spoofing attack, the Bus-off attack, and the Freeze Doom Loop attack. The first one has a purpose to send spoofed data. The second drives offline a specific device and the others can continue to communicate normally. The last one aims to freeze all the communication on the CAN bus.

5.8 Other classifiers not tested

During the project, three different classifiers were evaluated: the One-class, the SVC, and the Random Forest. While the One-class classifier is unsupervised, the others are supervised. However, this study did not conduct exhaustive research overall on the classifiers, leaving room for further experimentation to enhance attack detection capabilities.

Neural networks present an adapted alternative for the IDS due to their great modularity. Plenty of hyperparameters permit to finding of a good trade-off between training time and performance of the model. Furthermore, Neural networks have shown their benefits when applied to other IDS, which suggests they warrant exploration in this context.

The only unsupervised classifier studied in the project, the SVM One-class classifier, did not demonstrate good results. However, a large panel of unsupervised models exists and could perform well. In addition, this type of classifier has great flexibility. Exploring alternative unsupervised classifiers may lead to improved performance.

Additionally, the results show that the operations applied to the samples before the training are efficient in certain scenarios. Experimenting with different reshaping

techniques for the training data could yield further enhancements. Moreover, a reselection of the HPCs could be set up; indeed, the difference between the Reduced dataset and the Full dataset is not very important. In consequence, it is possible to select the HPCs which are the most affected by the attack.

Appendix A

Raw error results

A.1 One class classifier

A.1.1 AES

1500 frames dataset

Full dataset

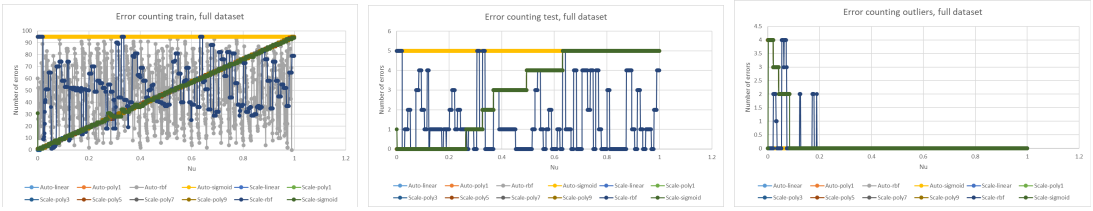

Figure A.1: Error counting DoS attack full dataset

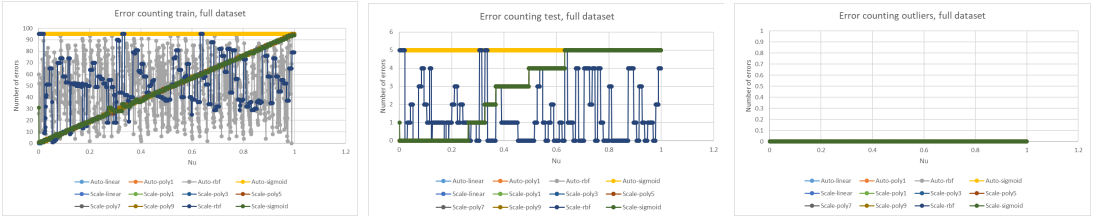

Figure A.2: Error counting Fuzzy attack full dataset

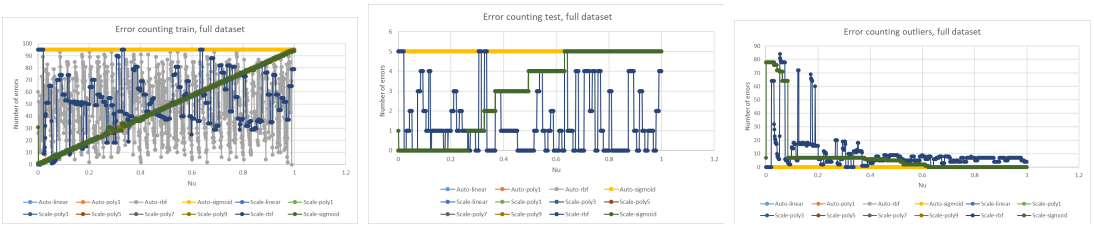

Figure A.3: Error counting Impersonation attack full dataset

Mean and scale dataset

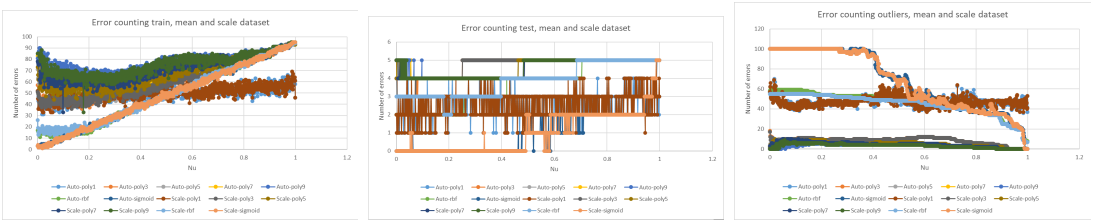

Figure A.4: Error counting DoS attack mean and scale dataset

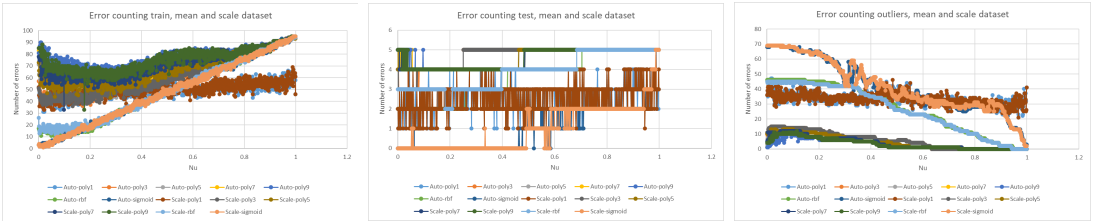

Figure A.5: Error counting Fuzzy attack mean and scale dataset

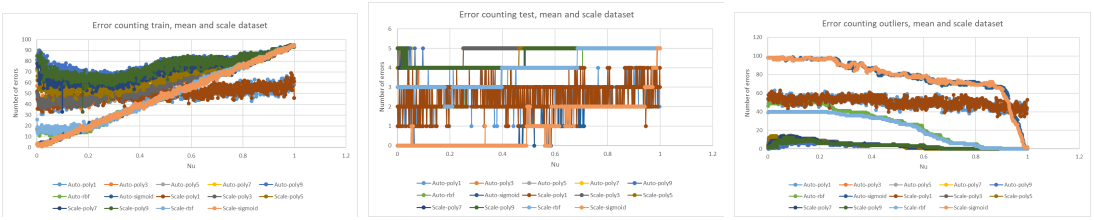

Figure A.6: Error counting Impersonation attack mean and scale dataset

Reduced dataset

Figure A.7: Error counting DoS attack reduced dataset

Figure A.8: Error counting Fuzzy attack reduced dataset

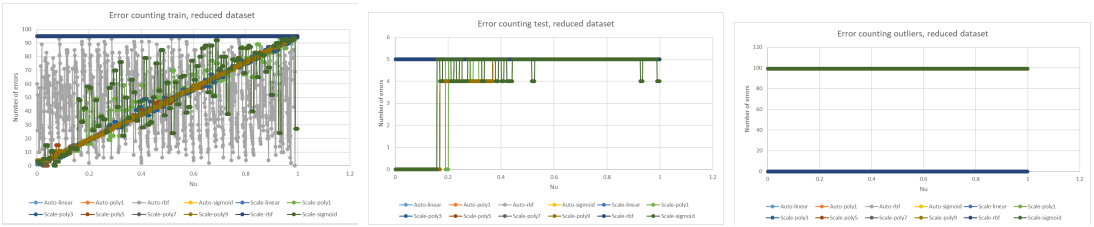

Figure A.9: Error counting Impersonation attack reduced dataset

A.1.2 Convolution

100 frames dataset

REDUCED DATASET

Figure A.10: Error counting DoS attack reduced dataset

Figure A.11: Error counting Fuzzy attack reduced dataset

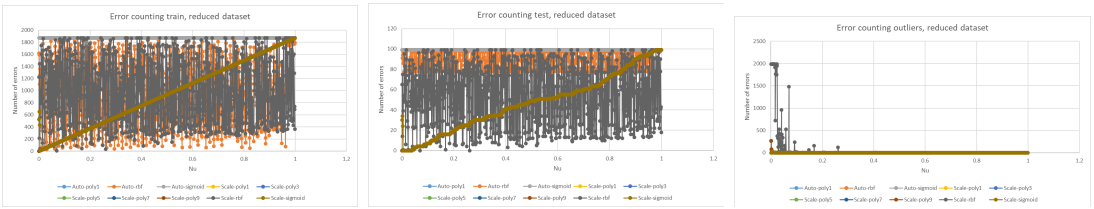

Figure A.12: Error counting Impersonation attack reduced dataset

2000 frames dataset

Full dataset

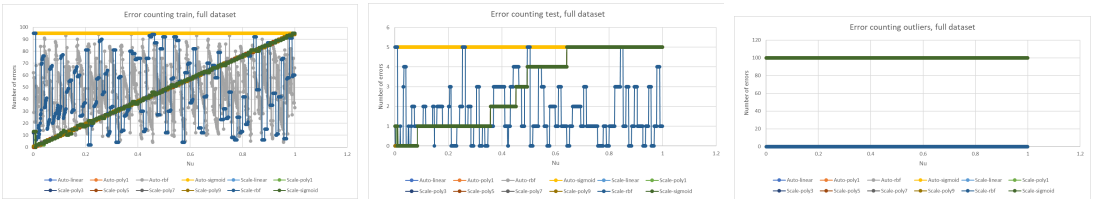

Figure A.13: Error counting DoS attack full dataset

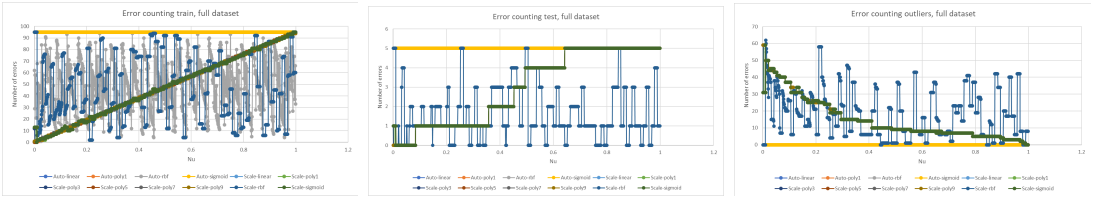

Figure A.14: Error counting Fuzzy attack full dataset

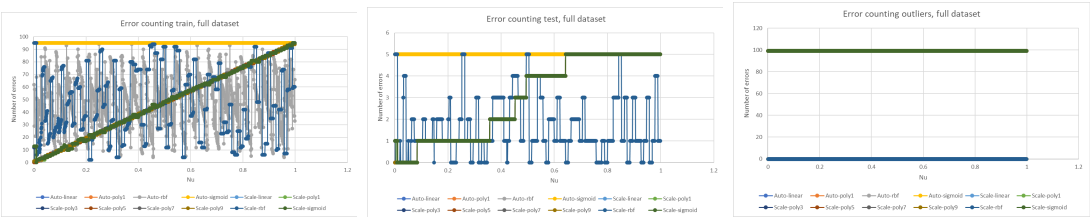

Figure A.15: Error counting Impersonation attack full dataset

Mean and scale dataset

Figure A.16: Error counting DoS attack mean and scale dataset

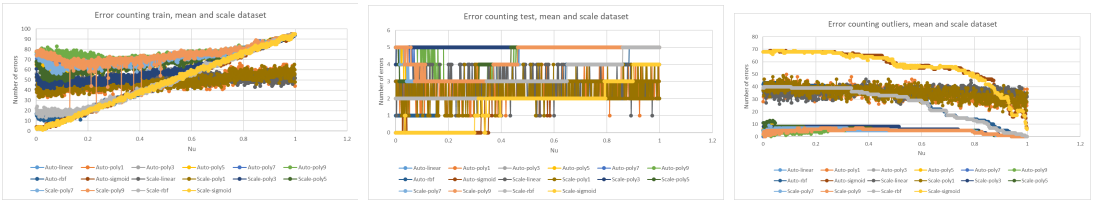

Figure A.17: Error counting Fuzzy attack mean and scale dataset

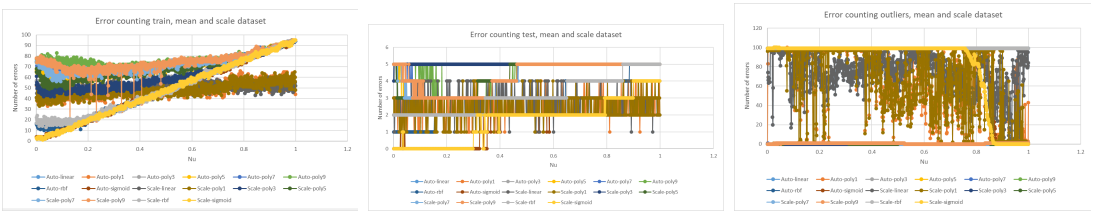

Figure A.18: Error counting Impersonation attack mean and scale dataset

Reduced dataset

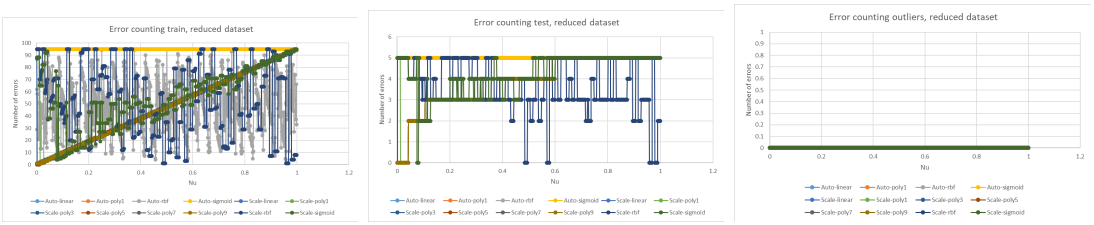

Figure A.19: Error counting DoS attack reduced dataset

Figure A.20: Error counting Fuzzy attack reduced dataset

Figure A.21: Error counting Impersonation attack reduced dataset

A.2 SVC

A.2.1 AES

1500 frames dataset

Full dataset

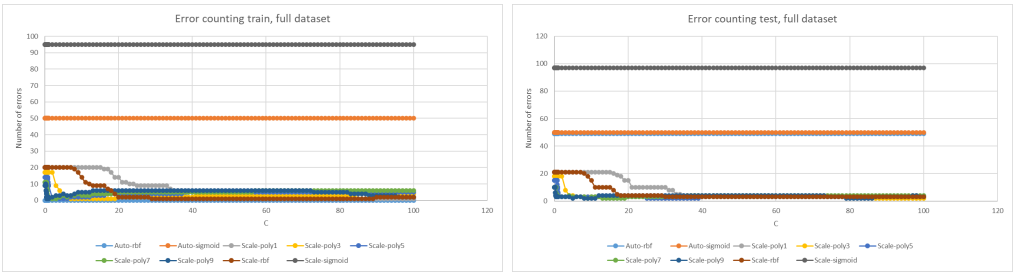

Figure A.22: Error counting DoS attack full dataset

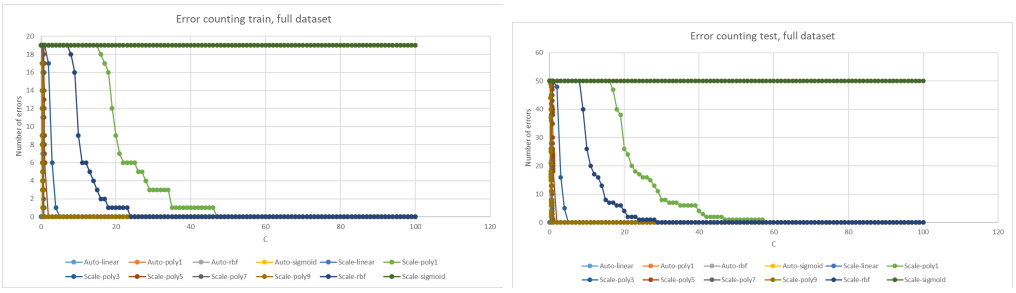

Figure A.23: Error counting Fuzzy attack full dataset

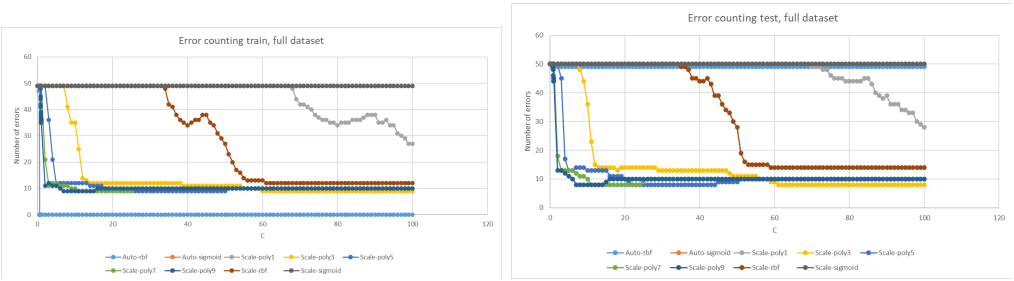

Figure A.24: Error counting Impersonation attack full dataset

Mean and scale dataset

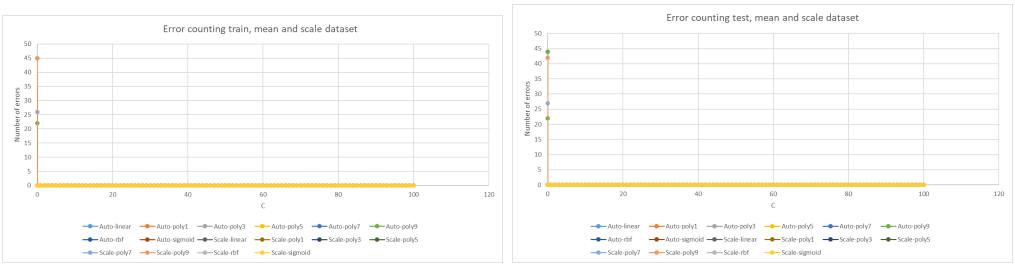

Figure A.25: Error counting DoS attack mean and scale dataset

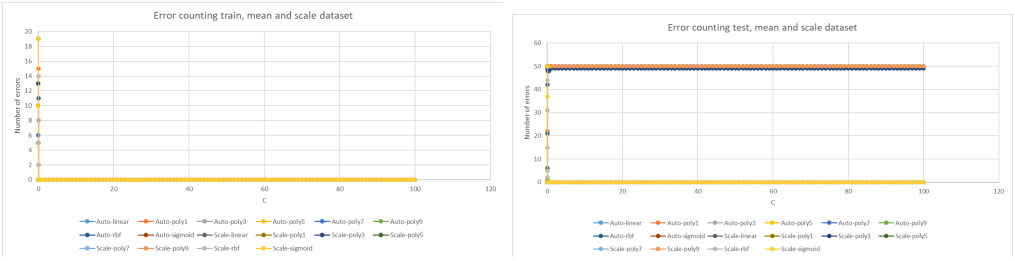

Figure A.26: Error counting Fuzzy attack mean and scale dataset

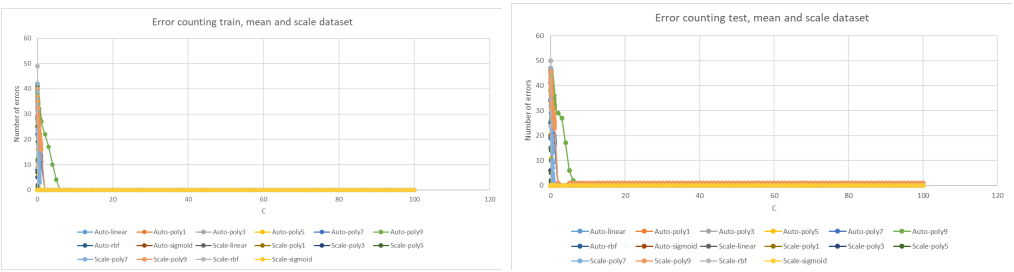

Figure A.27: Error counting Impersonation attack mean and scale dataset

Correlation reduced dataset

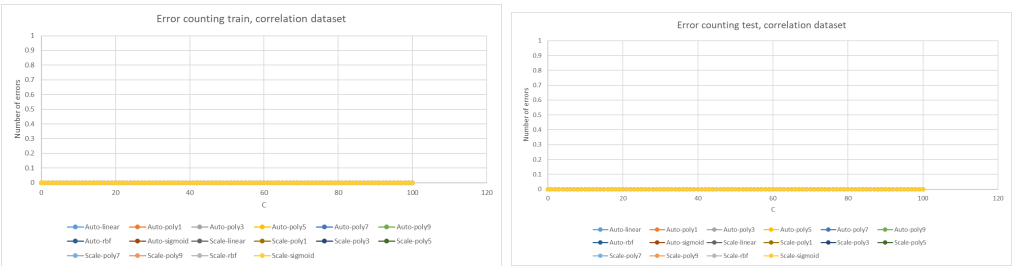

Figure A.28: Error counting DoS attack correlation reduced dataset

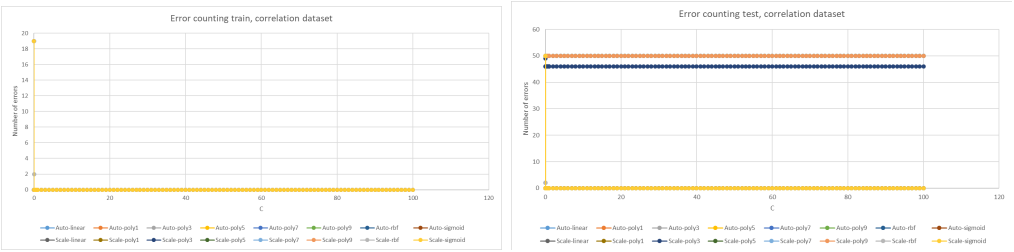

Figure A.29: Error counting Fuzzy attack correlation reduced dataset

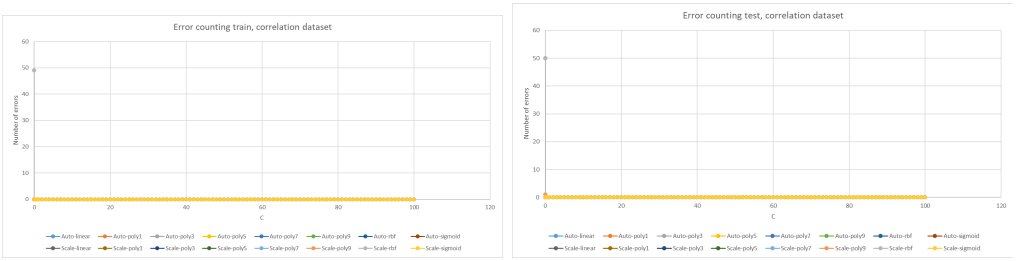

Figure A.30: Error counting Impersonation attack correlation reduced dataset

Reduced dataset

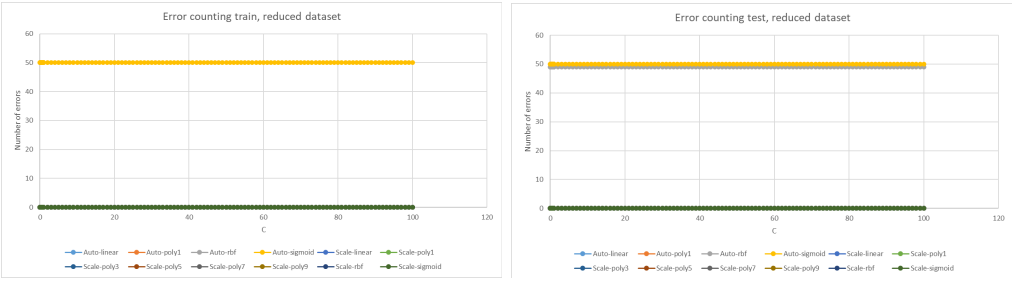

Figure A.31: Error counting DoS attack reduced dataset

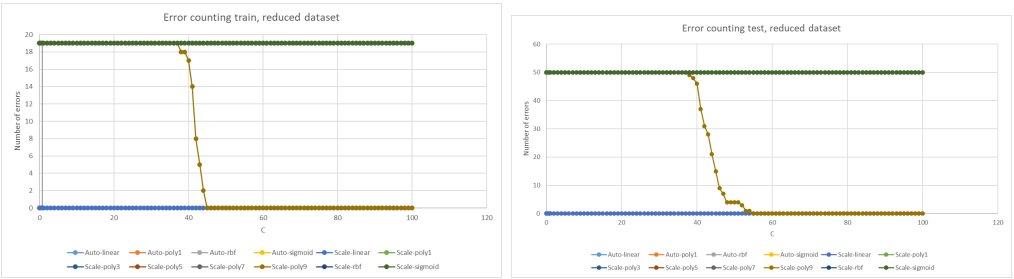

Figure A.32: Error counting Fuzzy attack reduced dataset

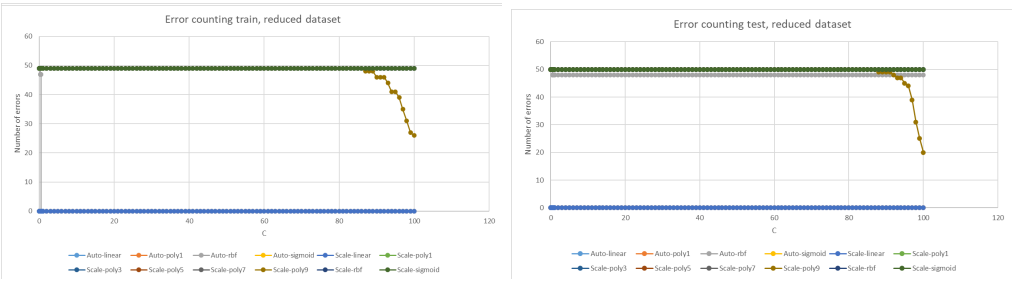

Figure A.33: Error counting Impersonation attack reduced dataset

A.2.2 Convolution

100 frames dataset

REDUCED DATASET

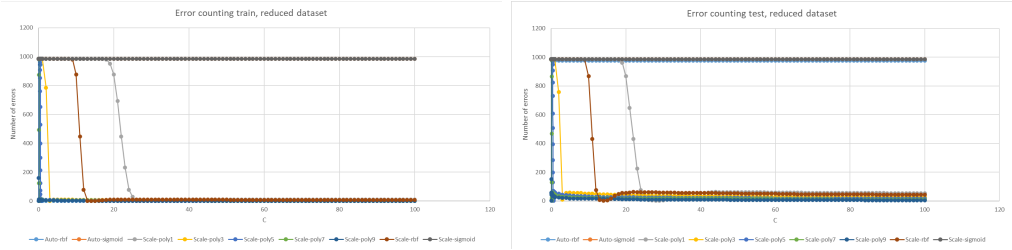

Figure A.34: Error counting DoS attack reduced dataset

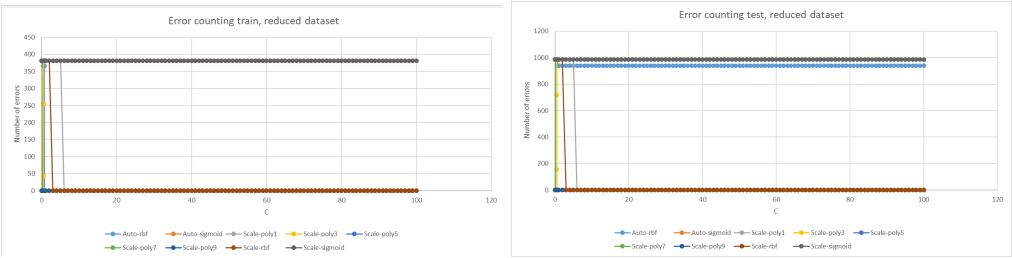

Figure A.35: Error counting Fuzzy attack reduced dataset

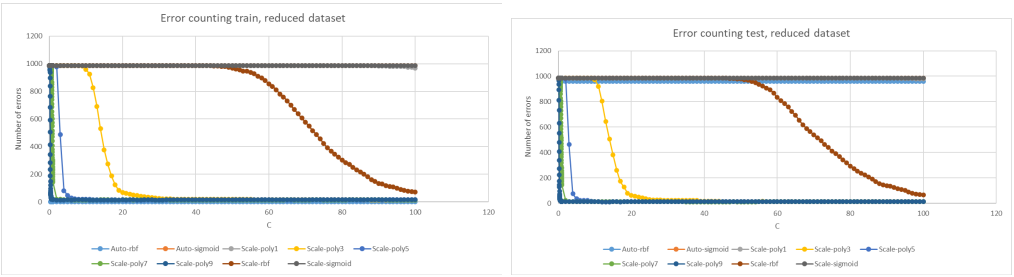

Figure A.36: Error counting Impersonation attack reduced dataset

2000 frames dataset

Full dataset

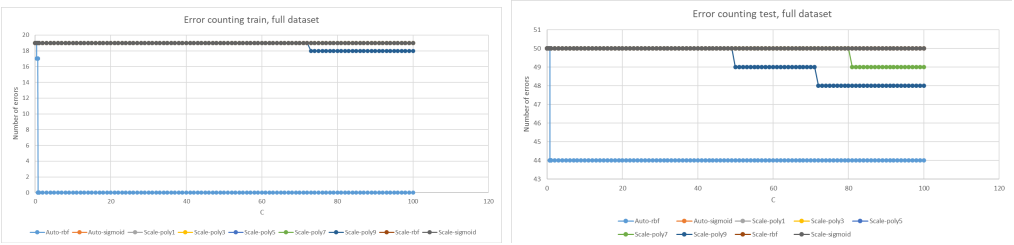

Figure A.37: Error counting DoS attack full dataset

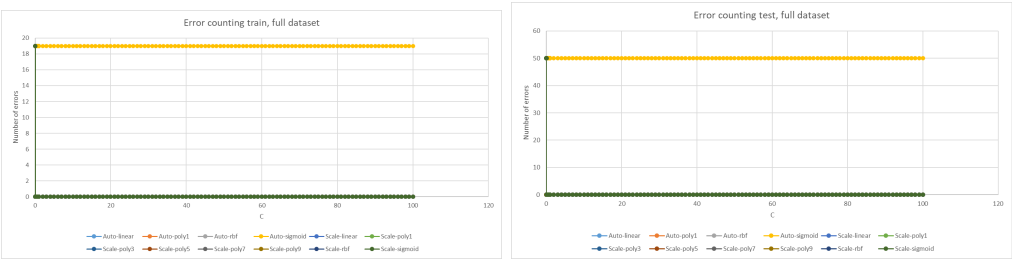

Figure A.38: Error counting Fuzzy attack full dataset

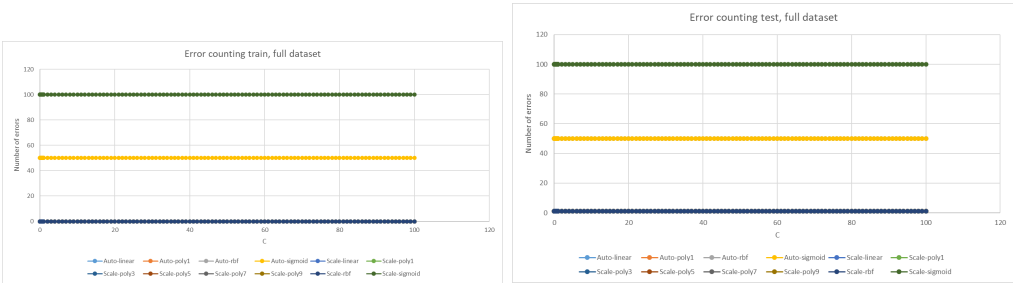

Figure A.39: Error counting Impersonation attack full dataset

Mean and scale dataset

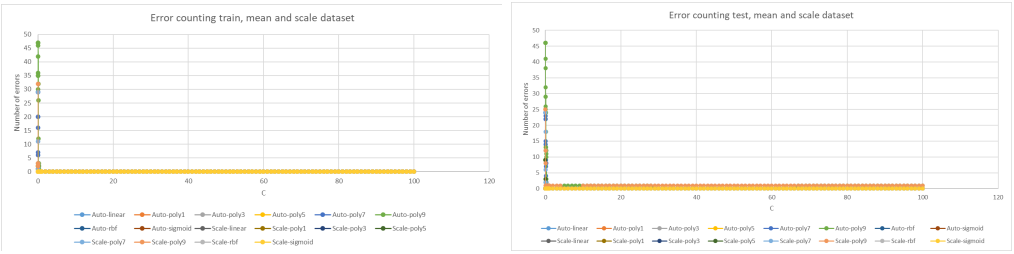

Figure A.40: Error counting DoS attack mean and scale dataset

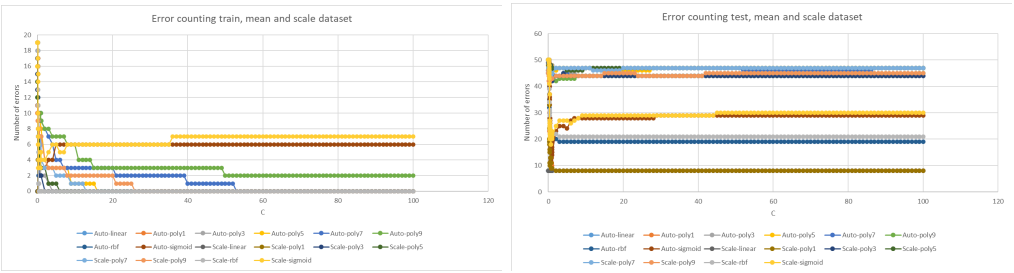

Figure A.41: Error counting Fuzzy attack mean and scale dataset

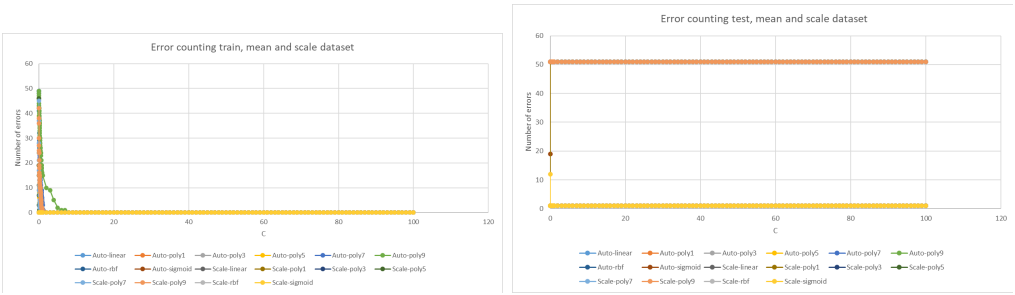

Figure A.42: Error counting Impersonation attack mean and scale dataset

Correlation reduced dataset

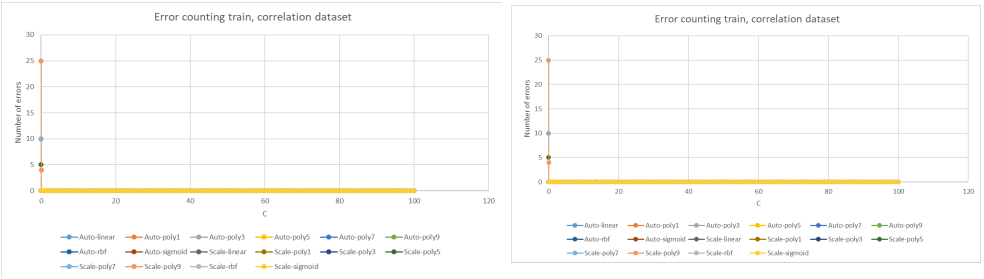

Figure A.43: Error counting DoS attack correlation reduced dataset

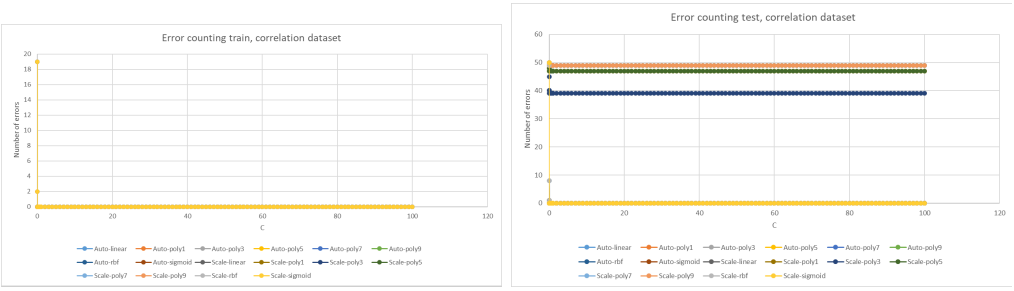

Figure A.44: Error counting Fuzzy attack correlation reduced dataset

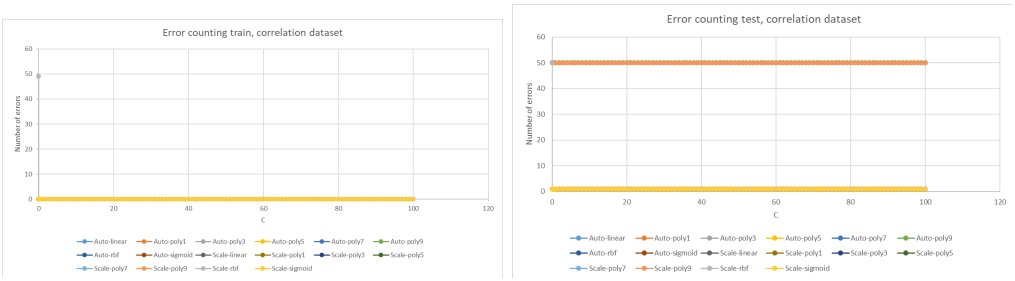

Figure A.45: Error counting Impersonation attack correlation reduced dataset

Reduced dataset

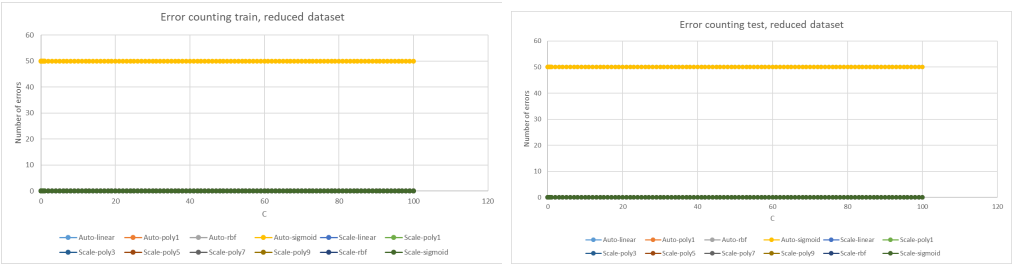

Figure A.46: Error counting DoS attack reduced dataset

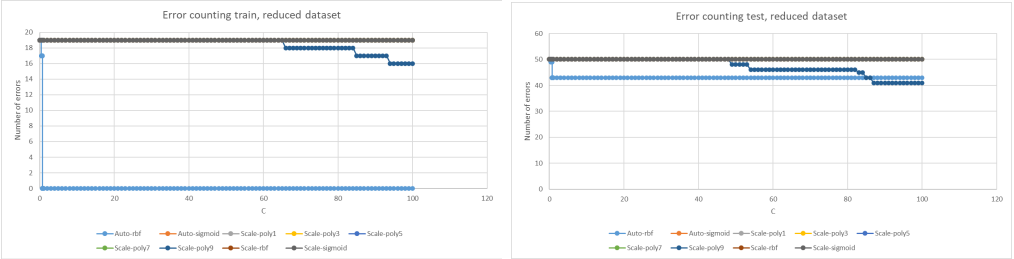

Figure A.47: Error counting Fuzzy attack reduced dataset

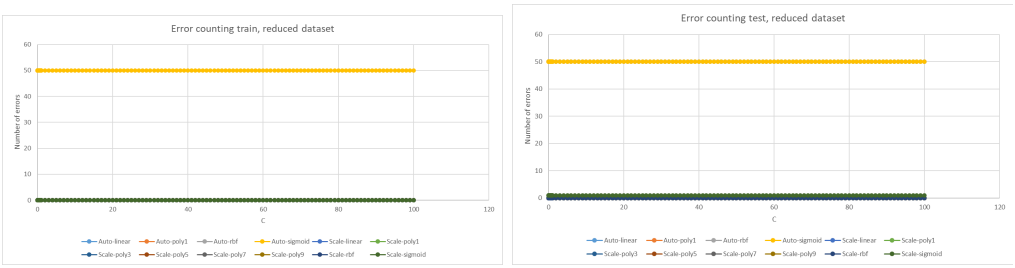

Figure A.48: Error counting Impersonation attack reduced dataset

A.3 Random forest

A.3.1 AES

1500 frames dataset

Full dataset

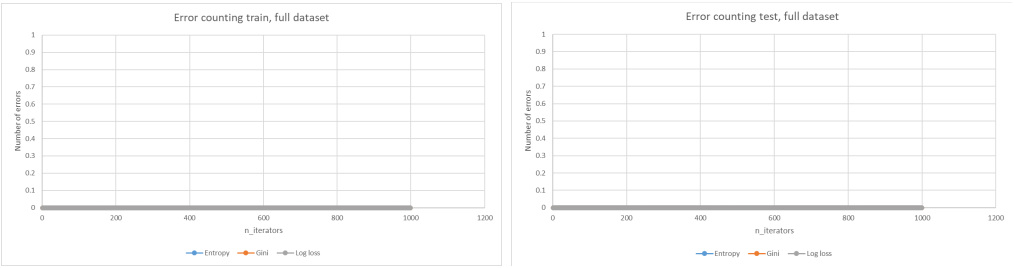

Figure A.49: Error counting DoS attack full dataset

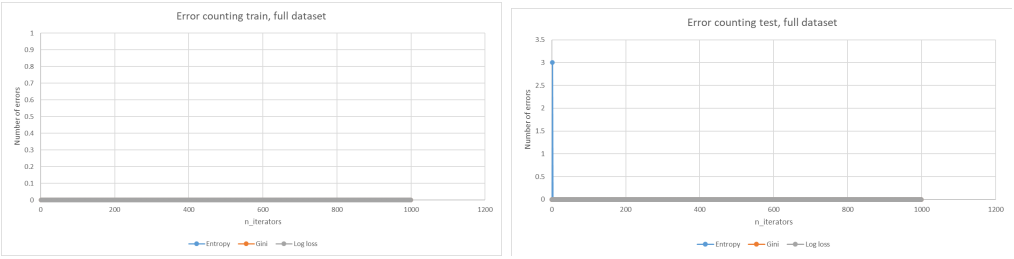

Figure A.50: Error counting Fuzzy attack full dataset

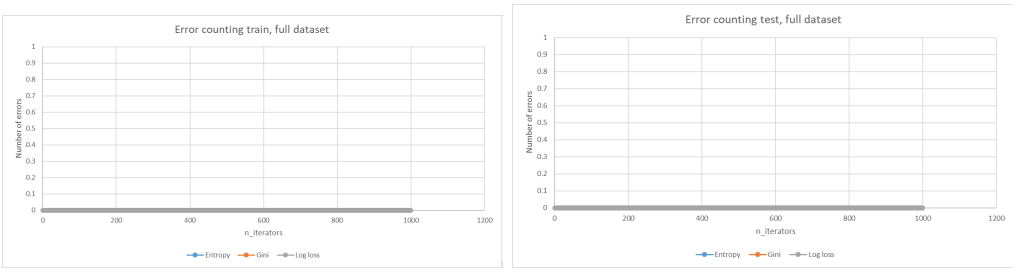

Figure A.51: Error counting Impersonation attack full dataset

Mean and scale dataset

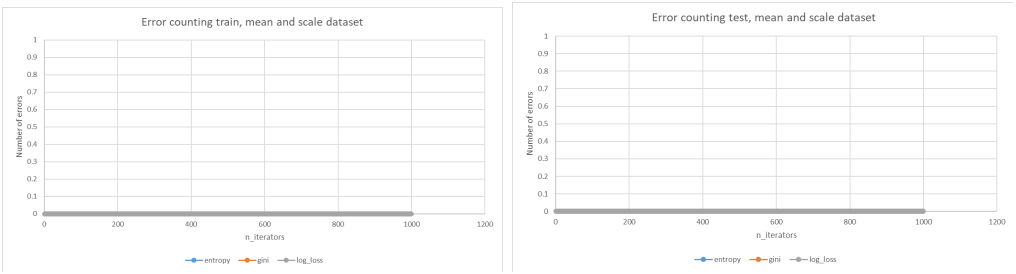

Figure A.52: Error counting DoS attack mean and scale dataset

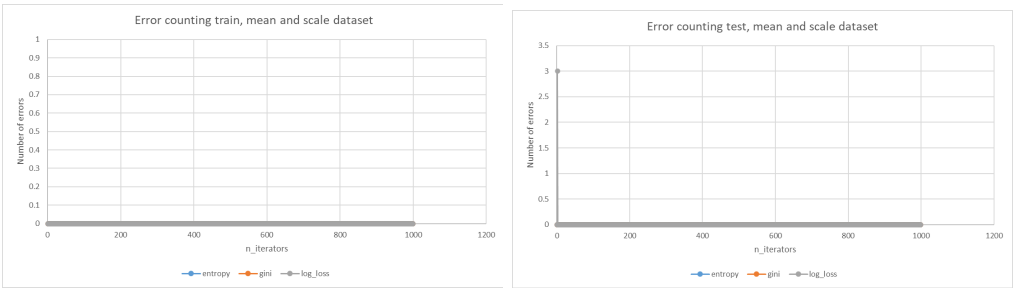

Figure A.53: Error counting Fuzzy attack mean and scale dataset

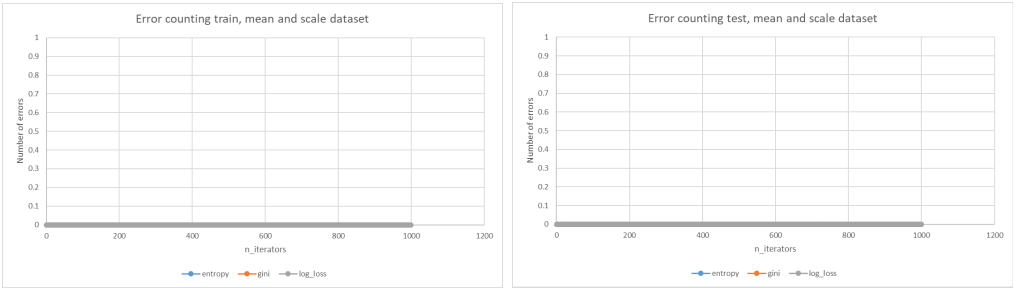

Figure A.54: Error counting Impersonation attack mean and scale dataset

Correlation reduced dataset

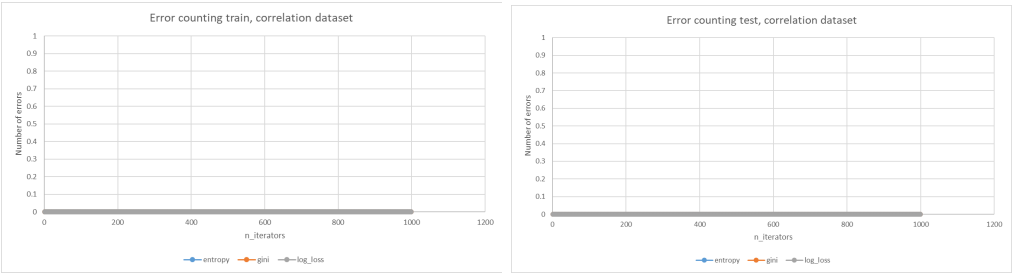

Figure A.55: Error counting DoS attack correlation reduced dataset

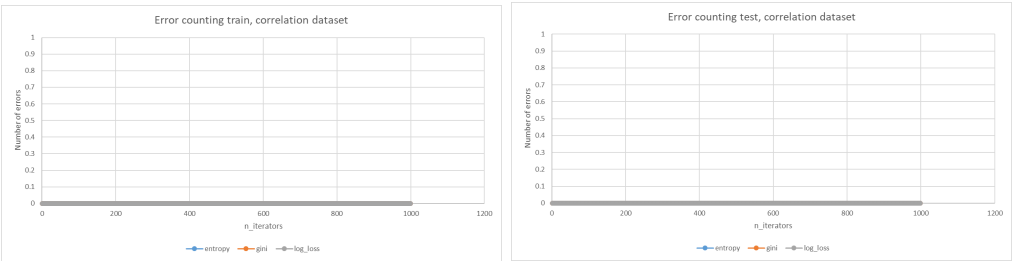

Figure A.56: Error counting Fuzzy attack correlation reduced dataset

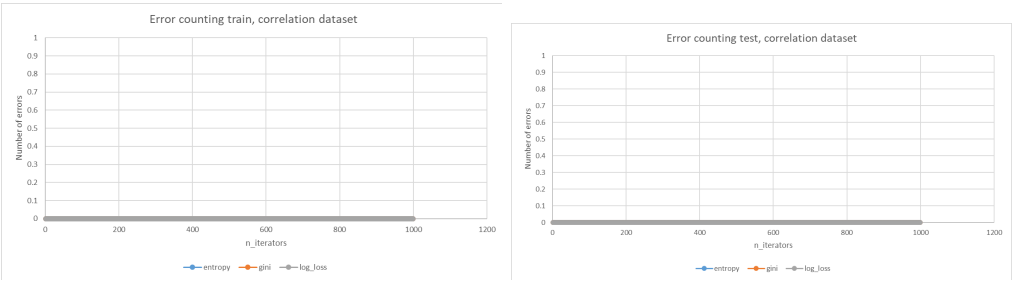

Figure A.57: Error counting Impersonation attack correlation reduced dataset

Reduced dataset

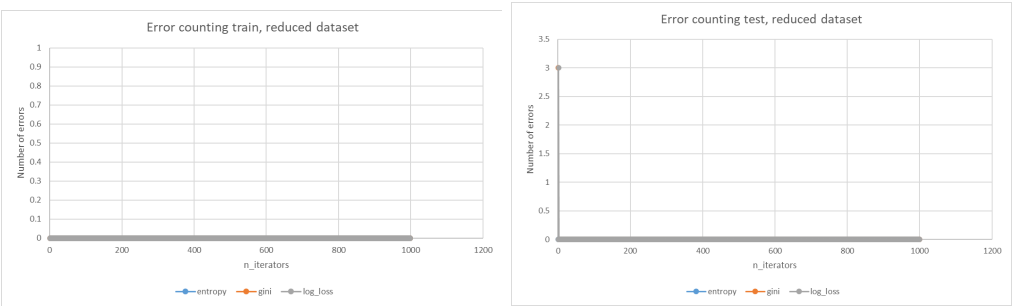

Figure A.58: Error counting DoS attack reduced dataset

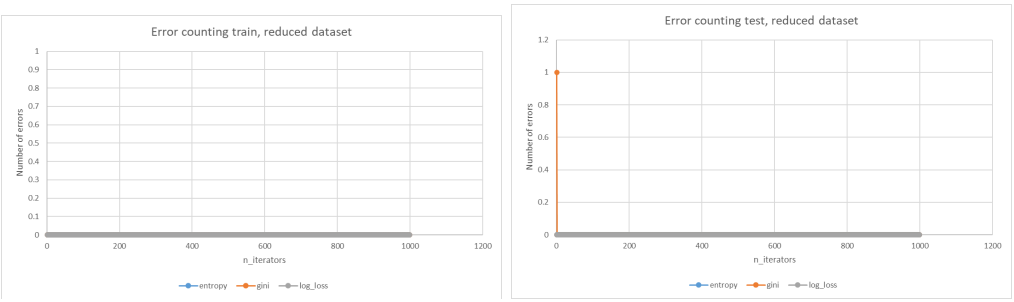

Figure A.59: Error counting Fuzzy attack reduced dataset

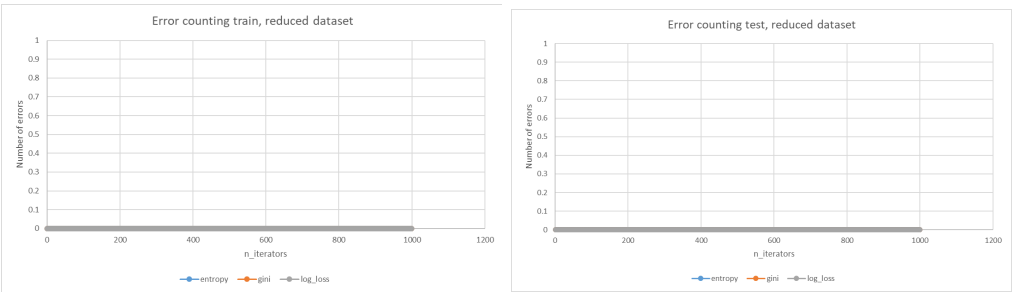

Figure A.60: Error counting Impersonation attack reduced dataset

A.3.2 Convolution

100 frames dataset

REDUCED DATASET

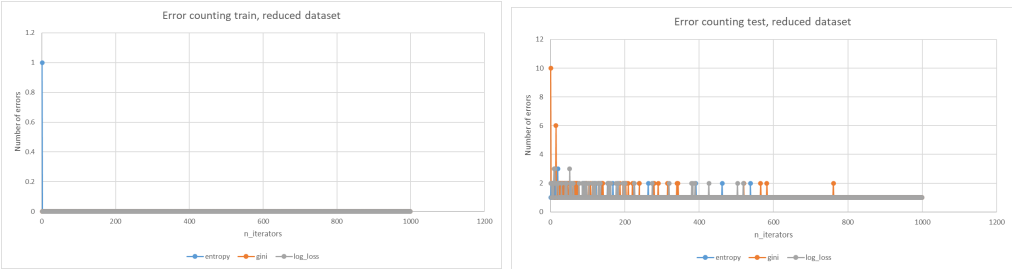

Figure A.61: Error counting DoS attack reduced dataset

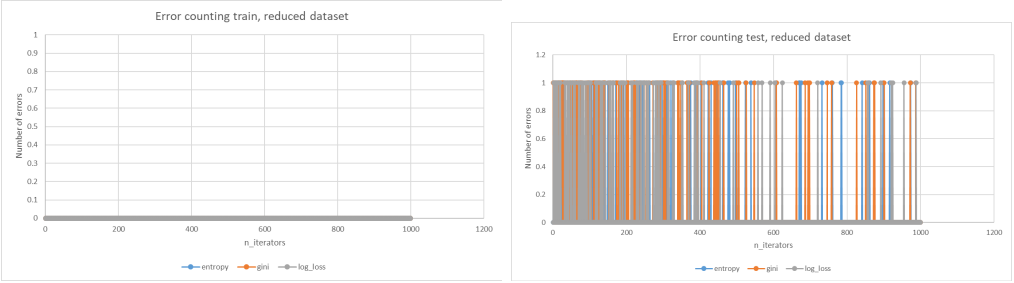

Figure A.62: Error counting Fuzzy attack reduced dataset

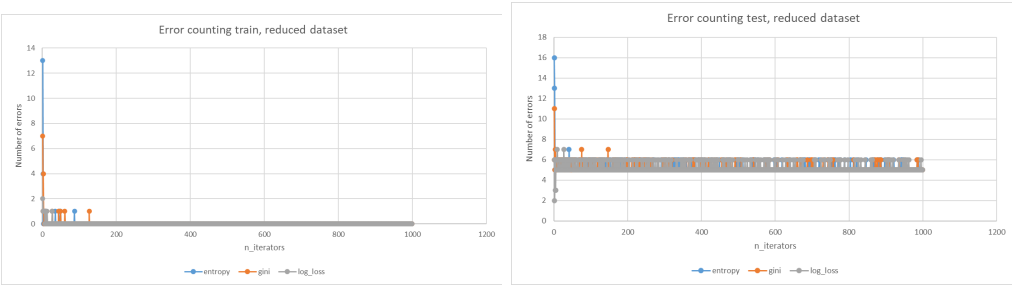

Figure A.63: Error counting Impersonation attack reduced dataset

2000 frames dataset

Full dataset

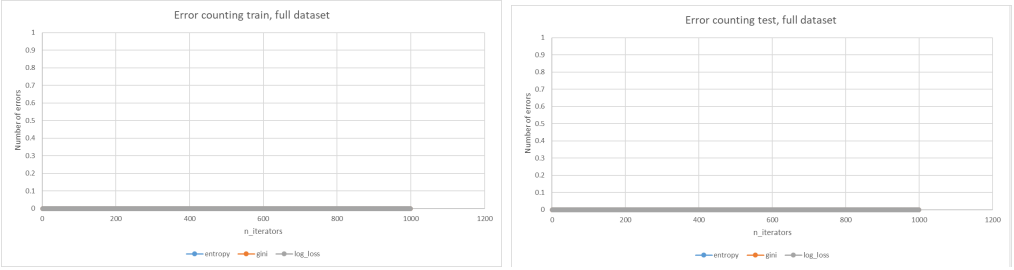

Figure A.64: Error counting DoS attack full dataset

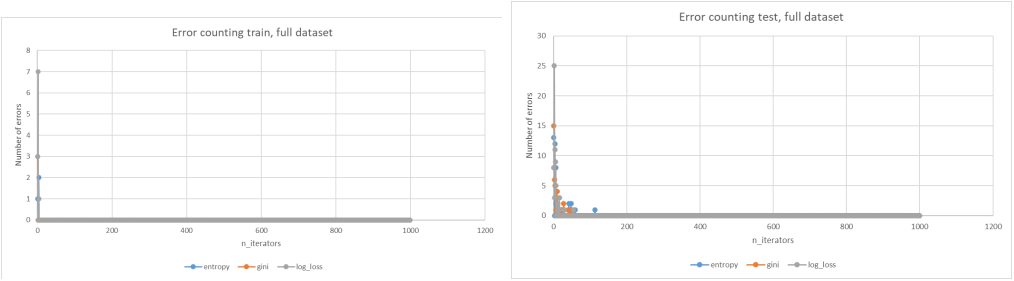

Figure A.65: Error counting Fuzzy attack full dataset

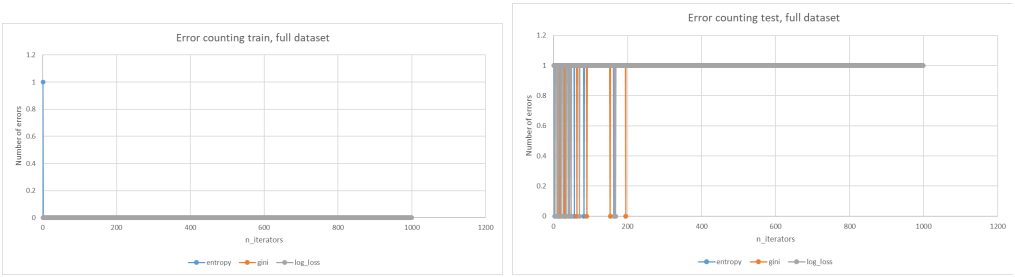

Figure A.66: Error counting Impersonation attack full dataset

Mean and scale dataset

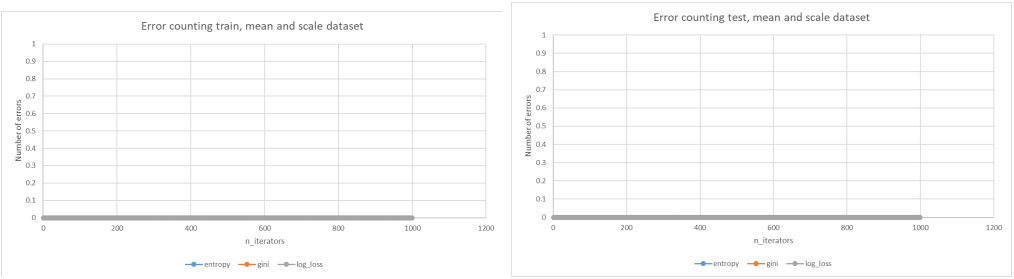

Figure A.67: Error counting DoS attack mean and scale dataset

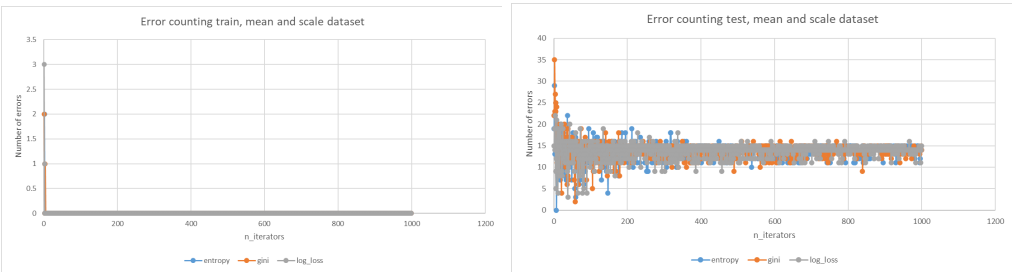

Figure A.68: Error counting Fuzzy attack mean and scale dataset

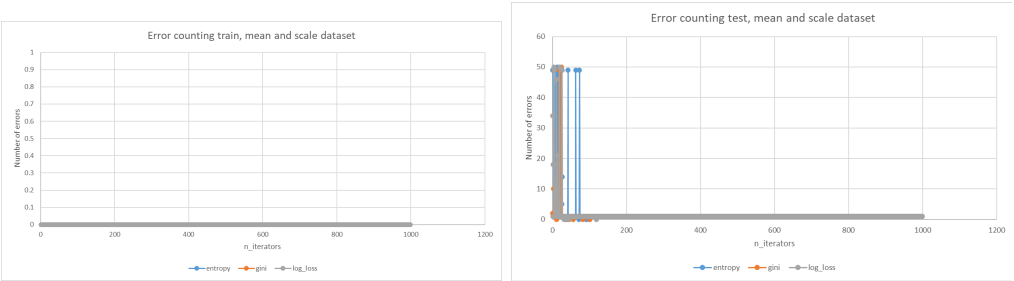

Figure A.69: Error counting Impersonation attack mean and scale dataset

Correlation reduced dataset

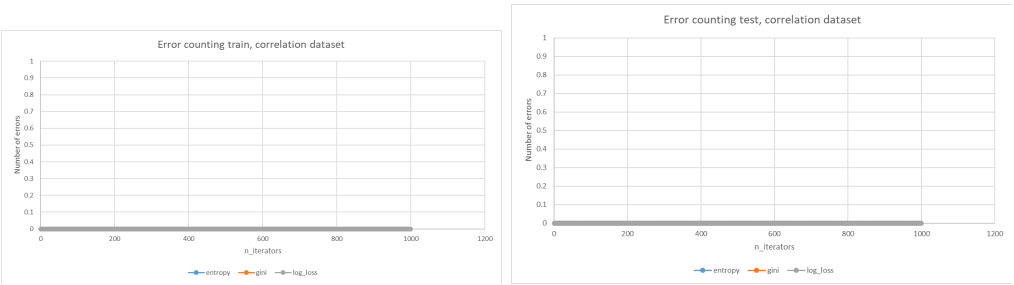

Figure A.70: Error counting DoS attack correlation reduced dataset

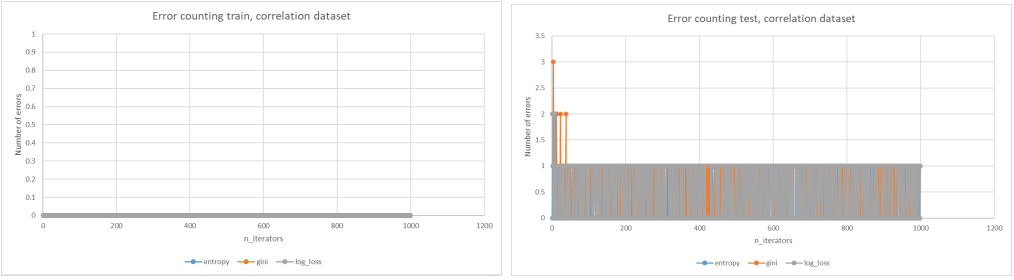

Figure A.71: Error counting Fuzzy attack correlation reduced dataset

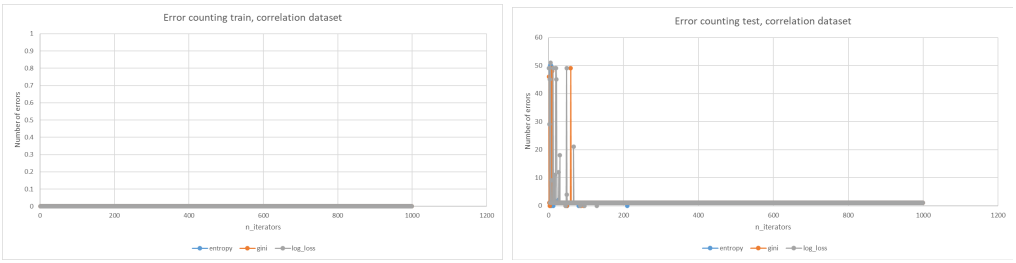

Figure A.72: Error counting Impersonation attack correlation reduced dataset

Reduced dataset

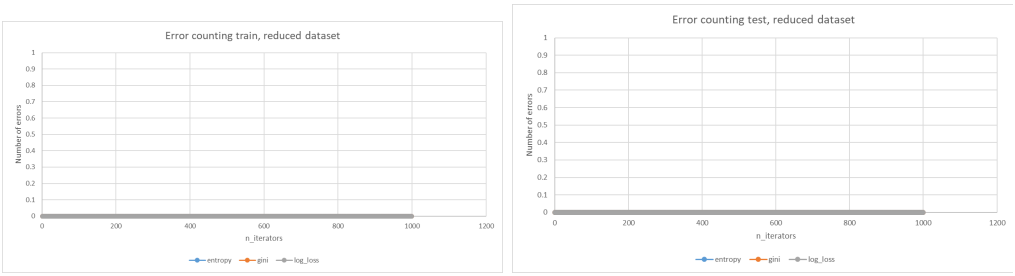

Figure A.73: Error counting DoS attack reduced dataset

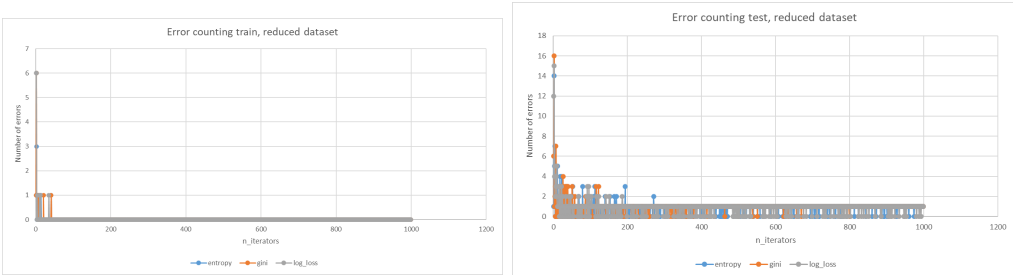

Figure A.74: Error counting Fuzzy attack reduced dataset

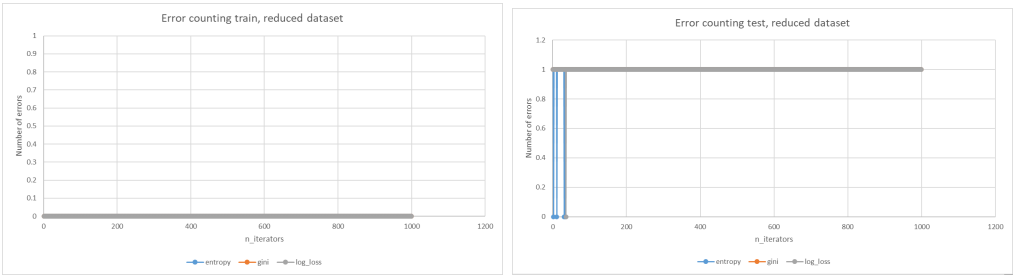

Figure A.75: Error counting Impersonation attack reduced dataset

A.4 Multiclass SVC

A.4.1 AES

1500 frames dataset

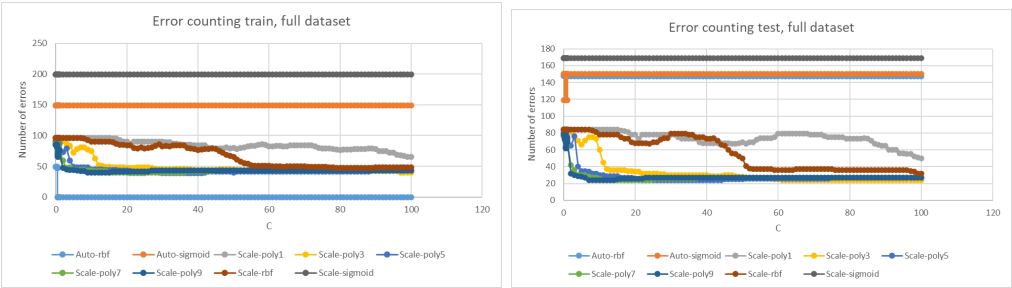

Figure A.76: Error counting full dataset

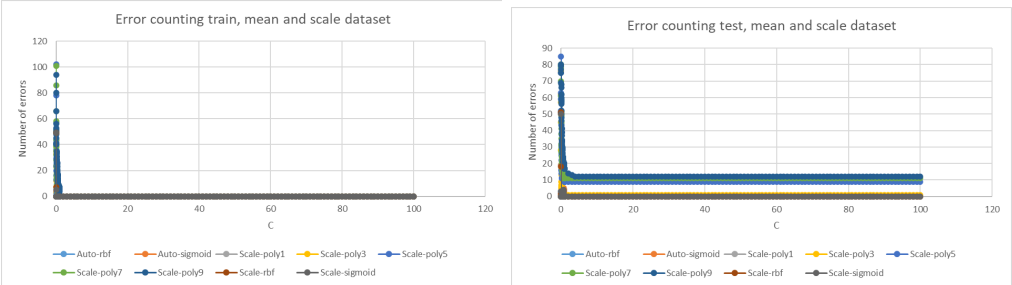

Figure A.77: Error counting mean and scale dataset

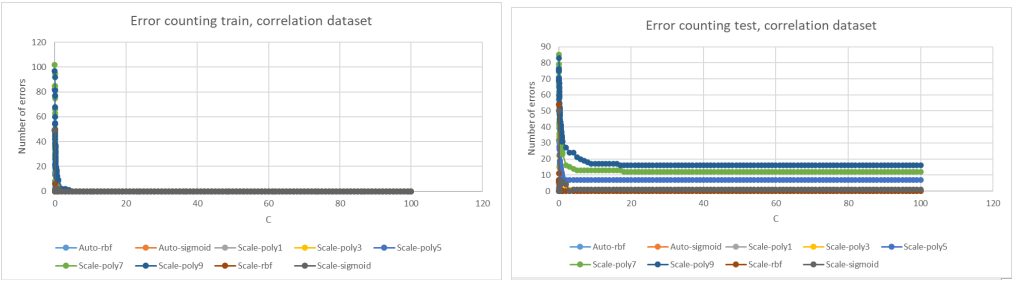

Figure A.78: Error counting correlation reduced dataset

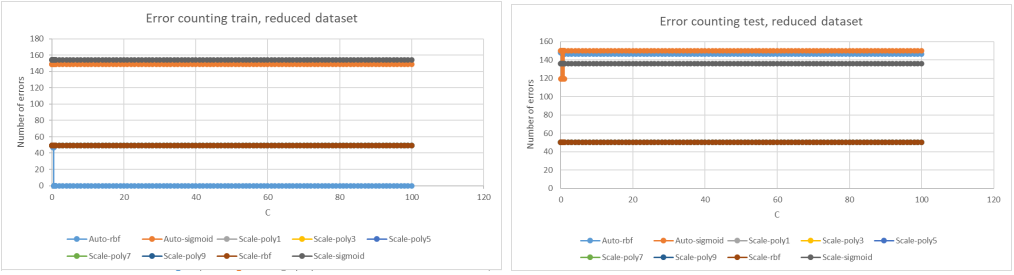

Figure A.79: Error counting reduced dataset

A.4.2 Convolution

100 frames dataset

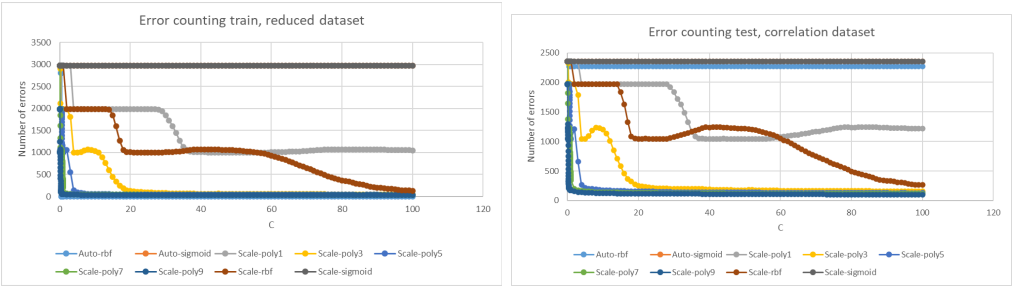

Figure A.80: Error counting reduced dataset

2000 frames dataset

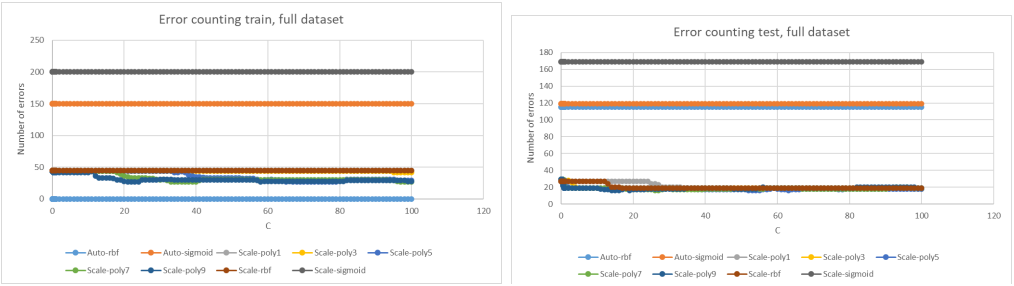

Figure A.81: Error counting full dataset

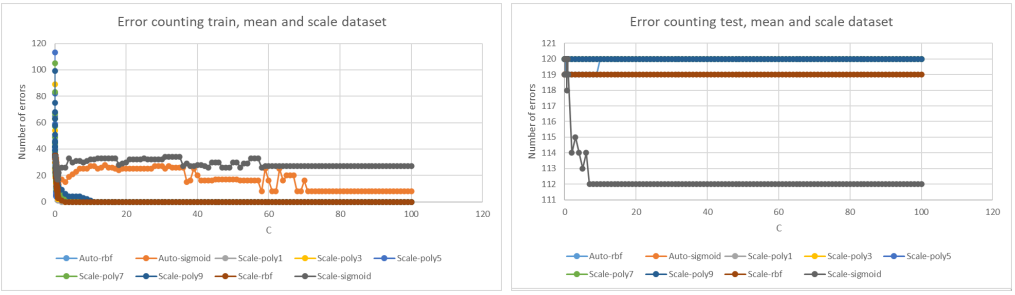

Figure A.82: Error counting mean and scale dataset

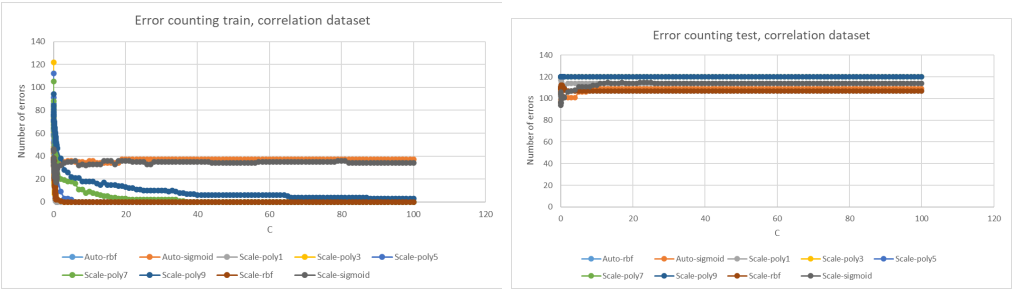

Figure A.83: Error counting correlation reduced dataset

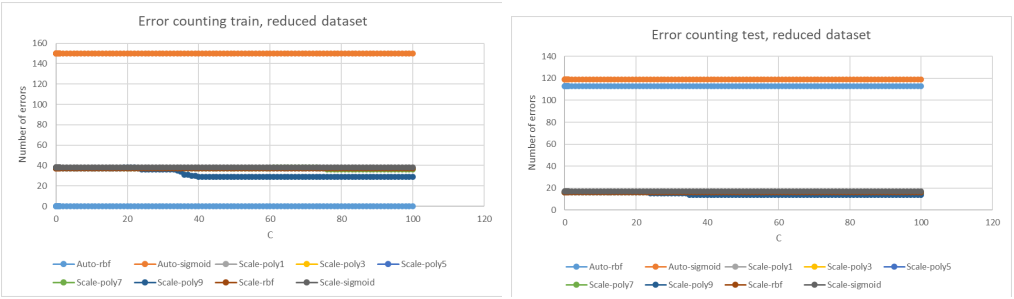

Figure A.84: Error counting reduced dataset

A.5 Multiclass random forest

A.5.1 AES

1500 frames dataset

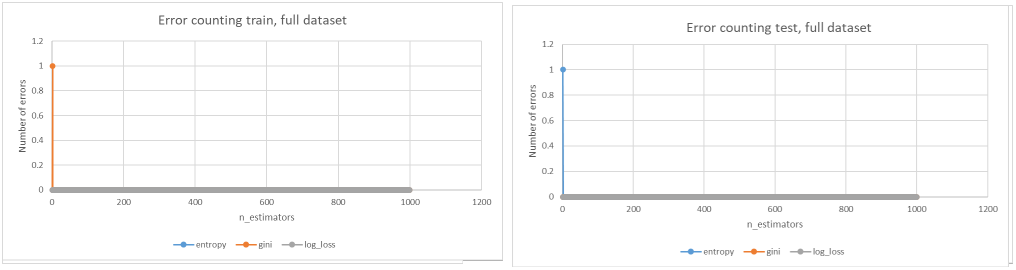

Figure A.85: Error counting full dataset

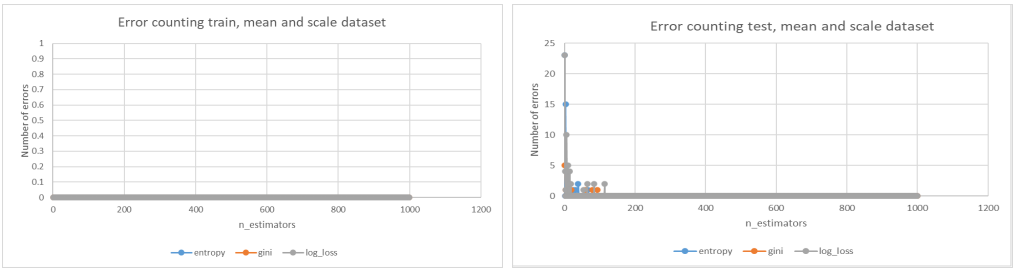

Figure A.86: Error counting mean and scale dataset

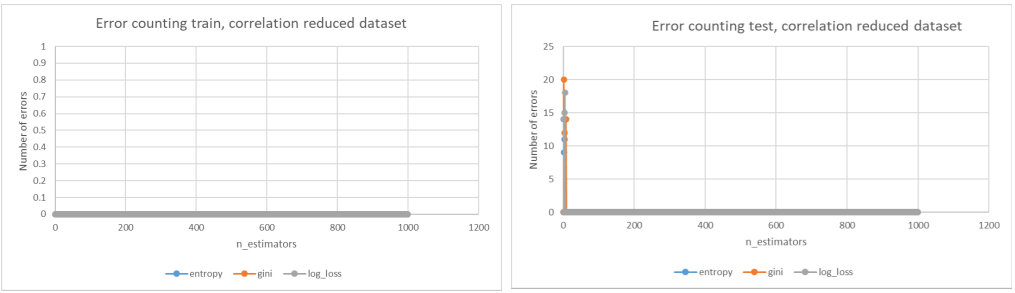

Figure A.87: Error counting correlation reduced dataset

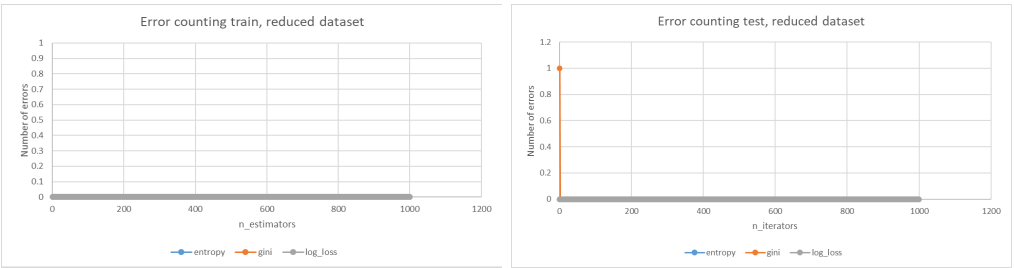

Figure A.88: Error counting reduced dataset

A.5.2 Convolution

100 frames dataset

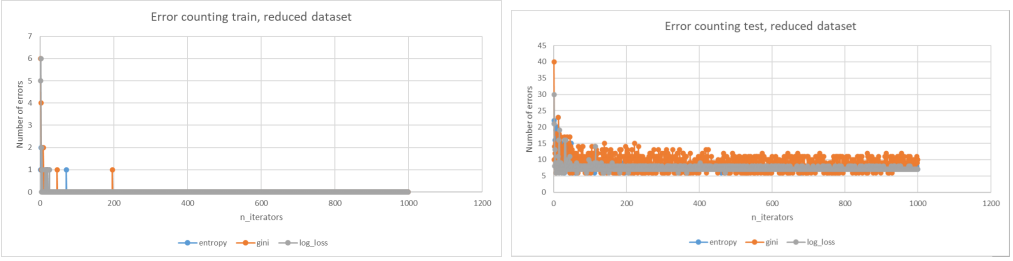

Figure A.89: Error counting reduced dataset

2000 frames dataset

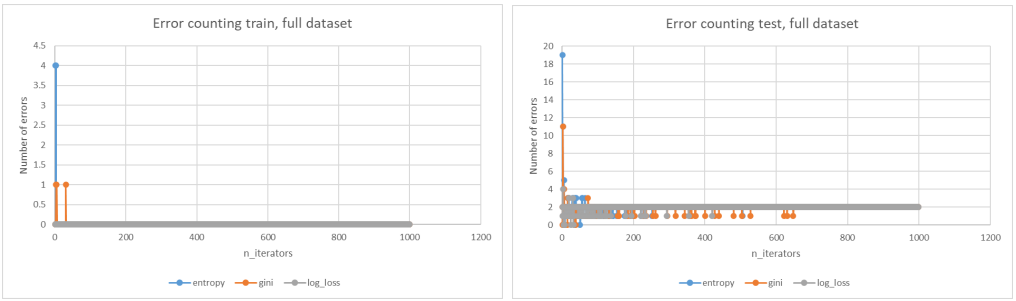

Figure A.90: Error counting full dataset

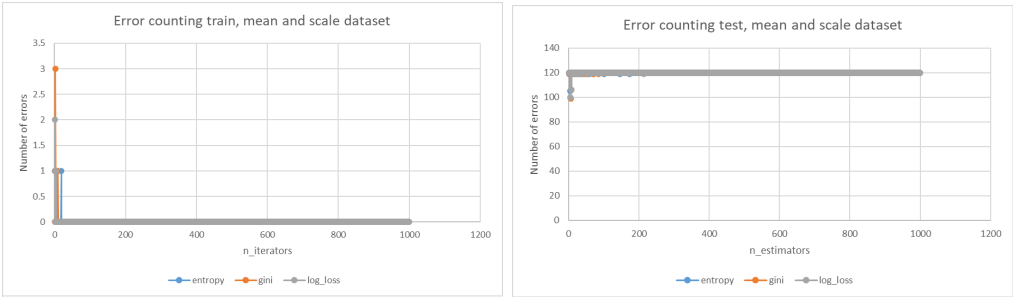

Figure A.91: Error counting mean and scale dataset

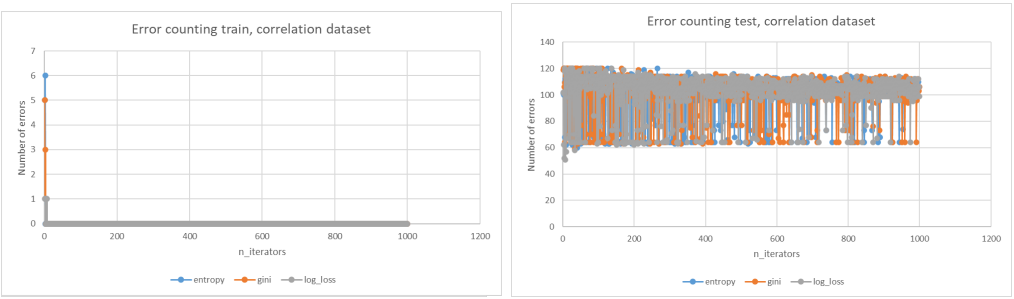

Figure A.92: Error counting correlation reduced dataset

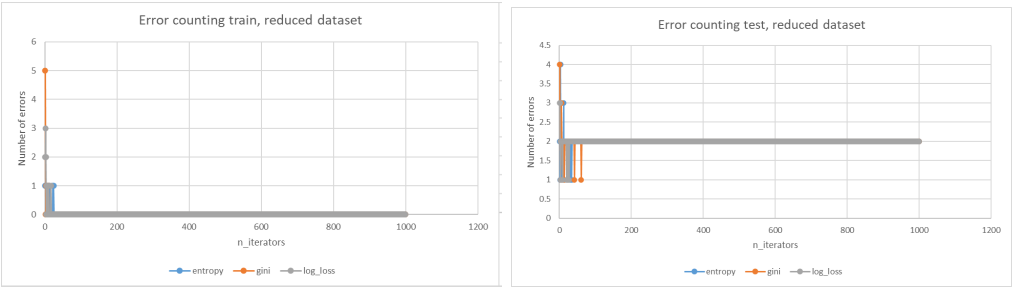

Figure A.93: Error counting reduced dataset

Appendix B

Final results

B.1 One class

B.1.1 AES

75 frames dataset

Reduced dataset

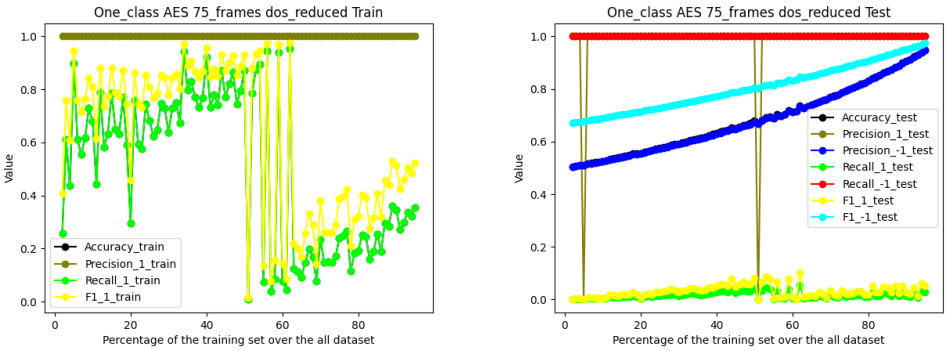

Figure B.1: DoS reduced dataset

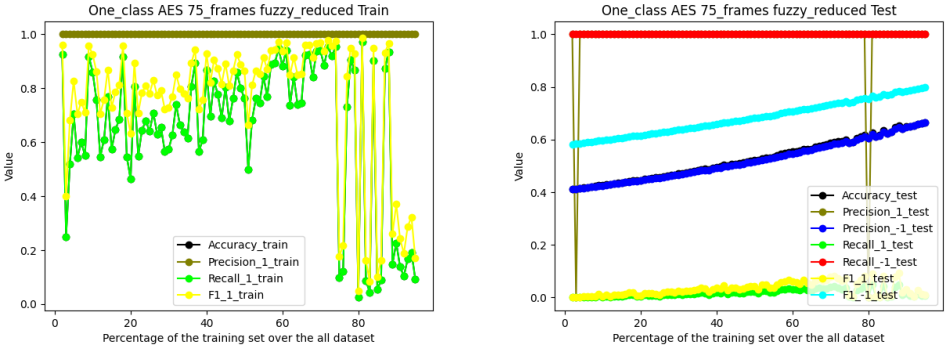

Figure B.2: Fuzzy reduced dataset

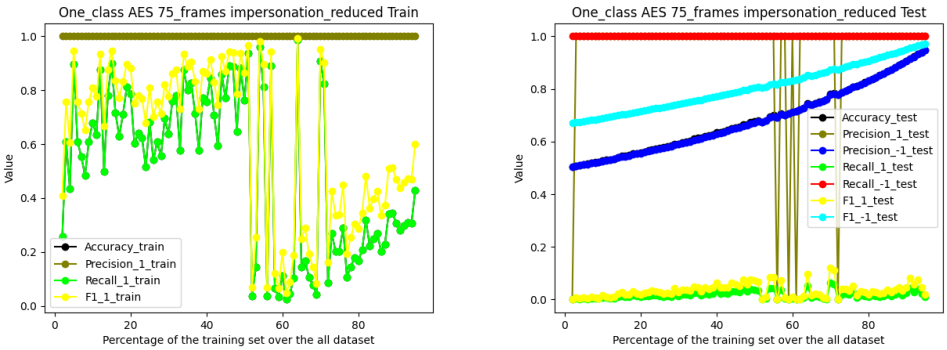

Figure B.3: Impersonation reduced dataset

1500 frames dataset

Full dataset

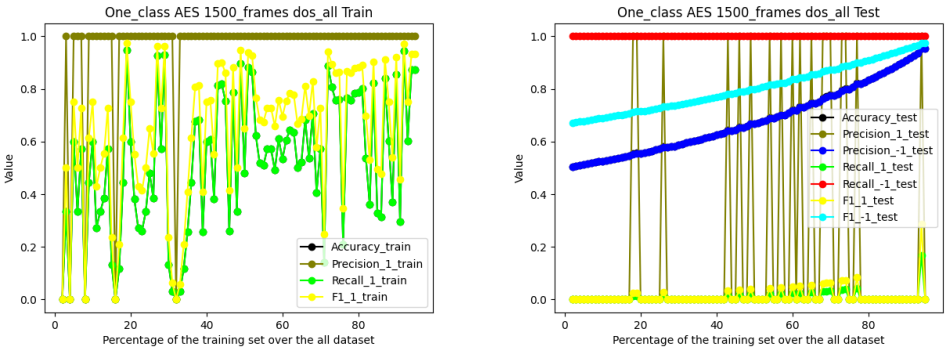

Figure B.4: DoS full dataset

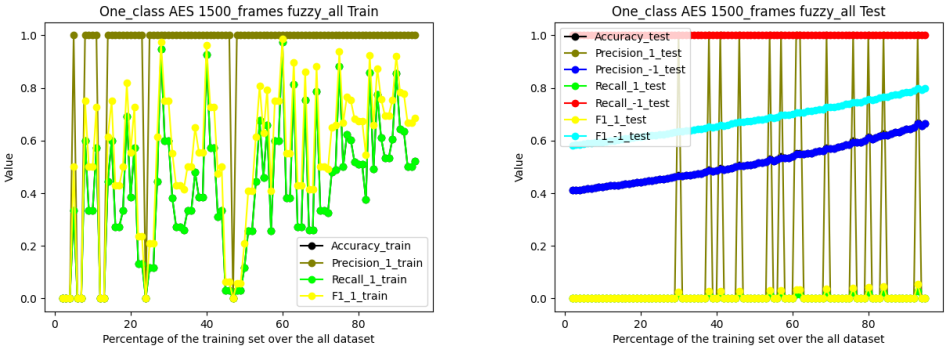

Figure B.5: Fuzzy full dataset

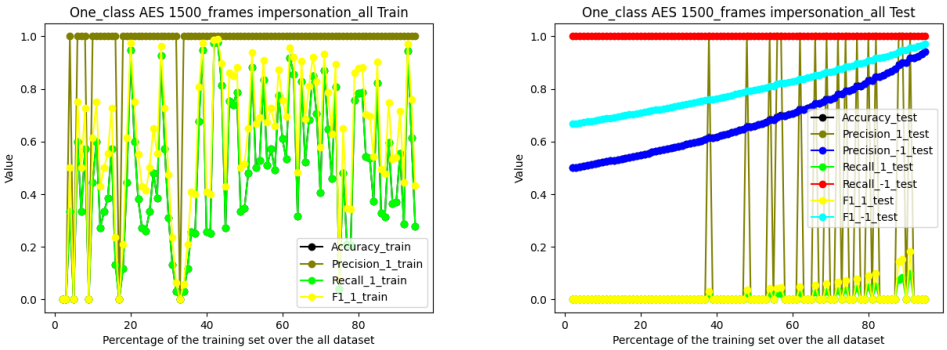

Figure B.6: Impersonation full dataset

Mean and scale dataset

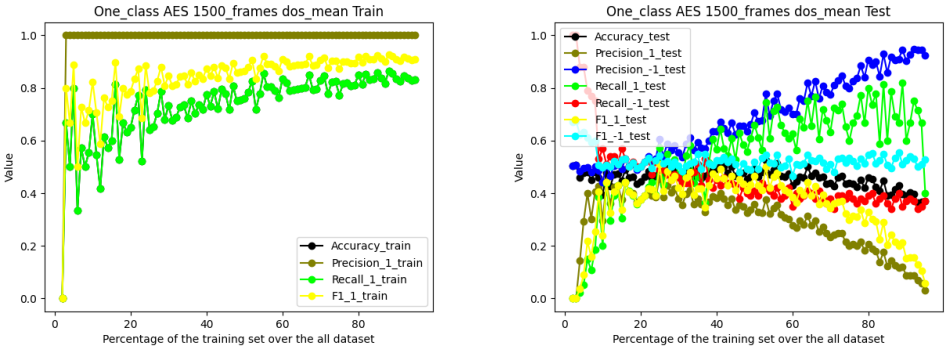

Figure B.7: DoS mean and scale dataset

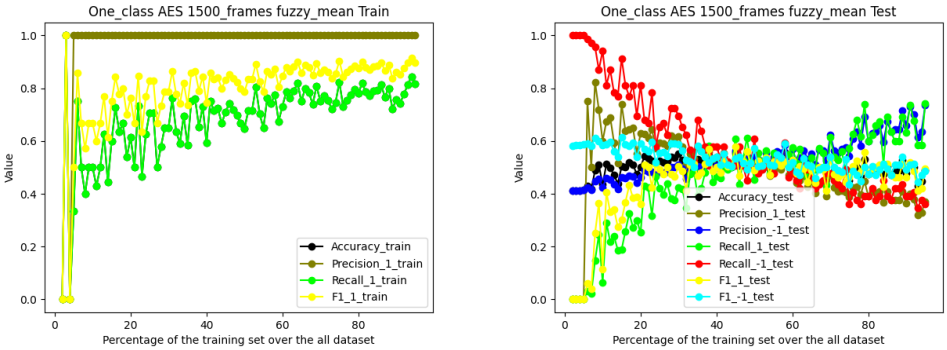

Figure B.8: Fuzzy mean and scale dataset

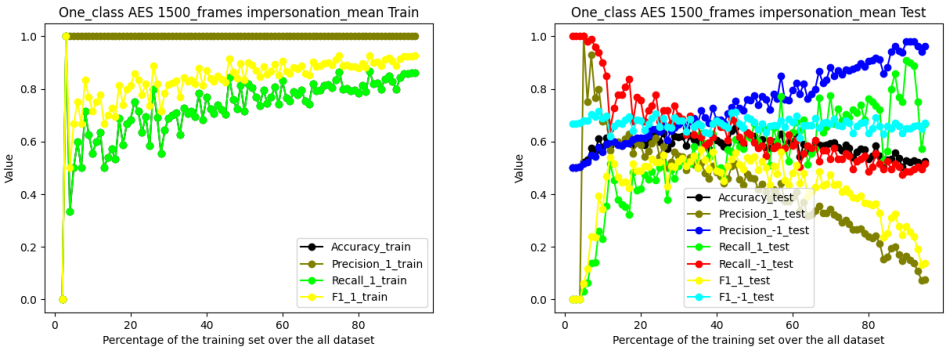

Figure B.9: Impersonation mean and scale dataset

Reduced dataset

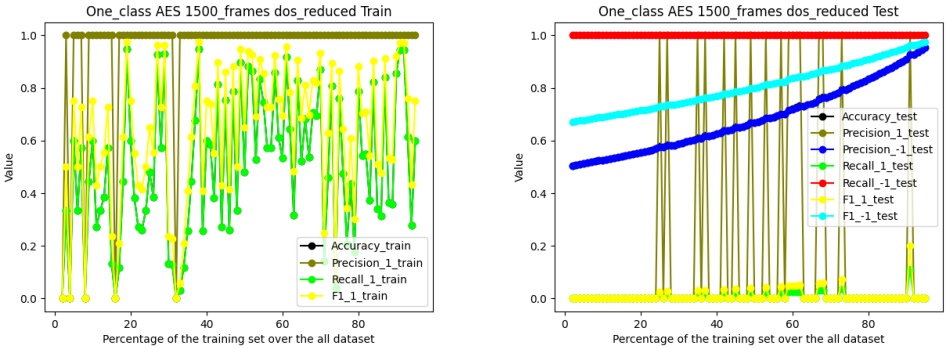

Figure B.10: DoS reduced dataset

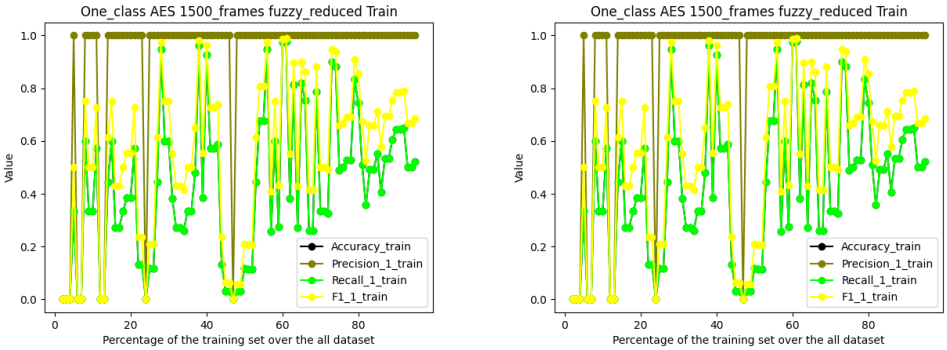

Figure B.11: Fuzzy reduced dataset

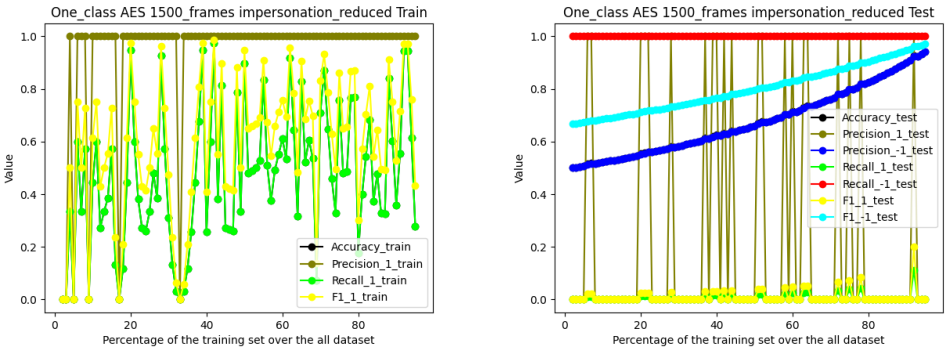

Figure B.12: Impersonation reduced dataset

B.1.2 Convolution

100 frames dataset

REDUCED DATASET

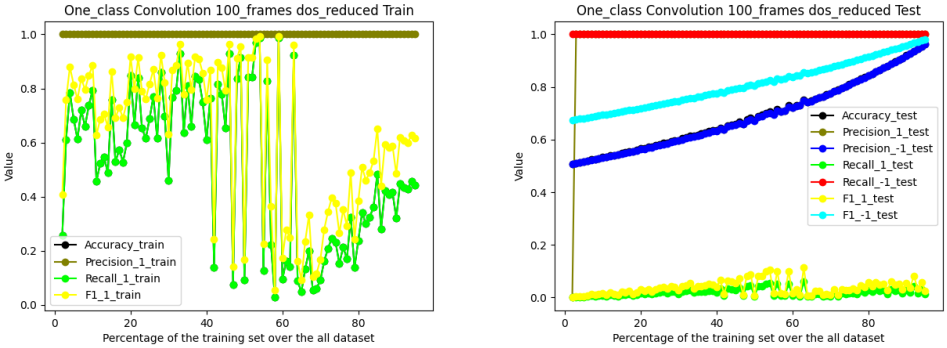

Figure B.13: DoS reduced dataset

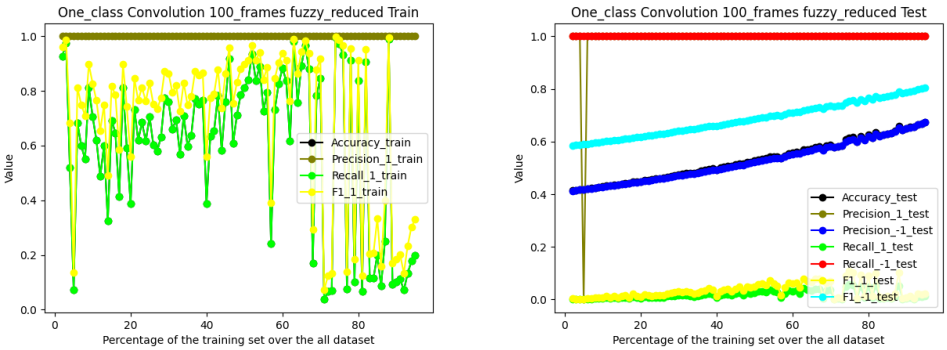

Figure B.14: Fuzzy reduced dataset

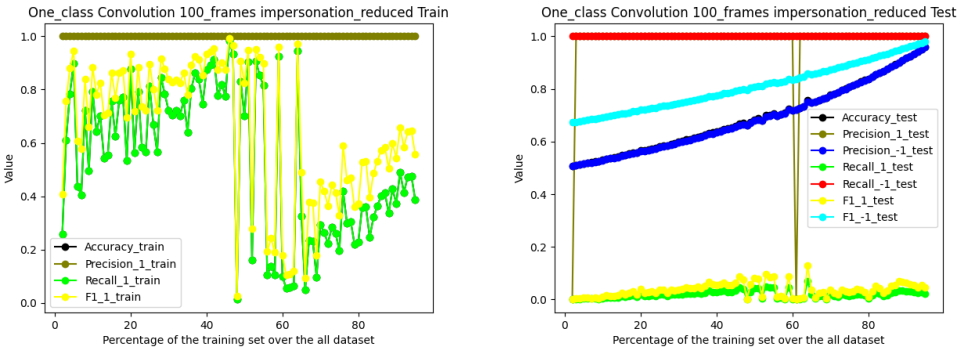

Figure B.15: Impersonation reduced dataset

2000 frames dataset

Full dataset

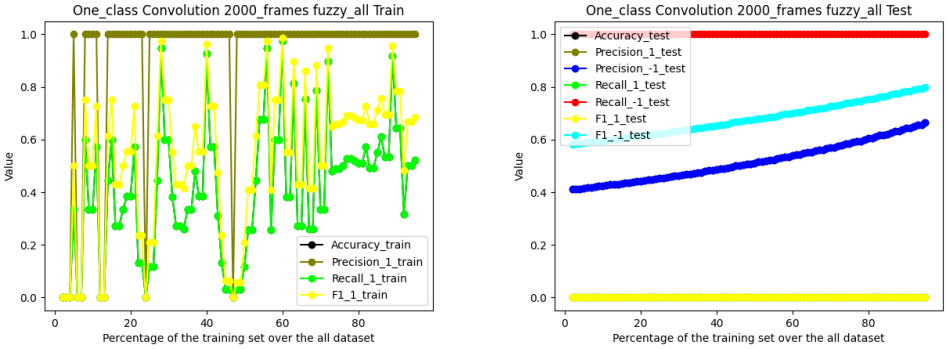

Figure B.16: Fuzzy full dataset

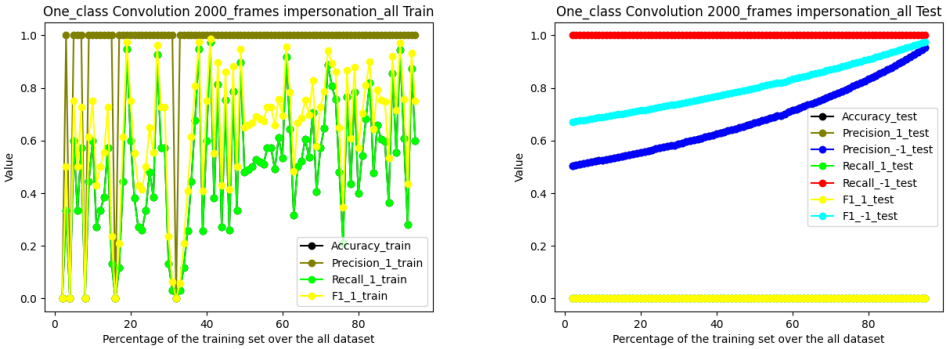

Figure B.17: Impersonation full dataset

Mean and scale dataset

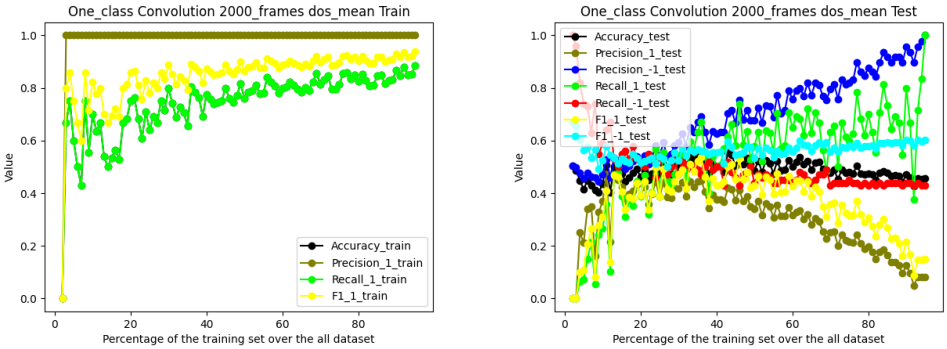

Figure B.18: DoS mean and scale dataset

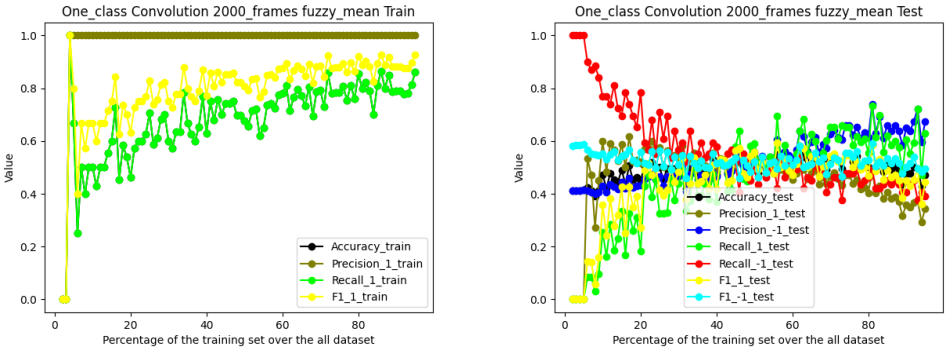

Figure B.19: Fuzzy mean and scale dataset

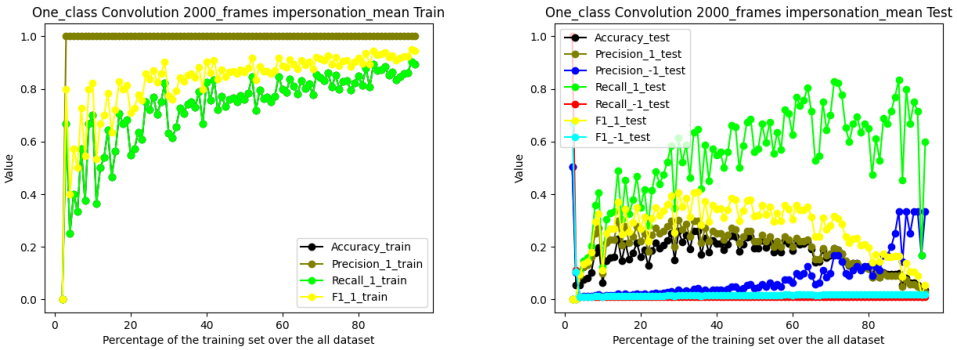

Figure B.20: Impersonation mean and scale dataset

Reduced dataset

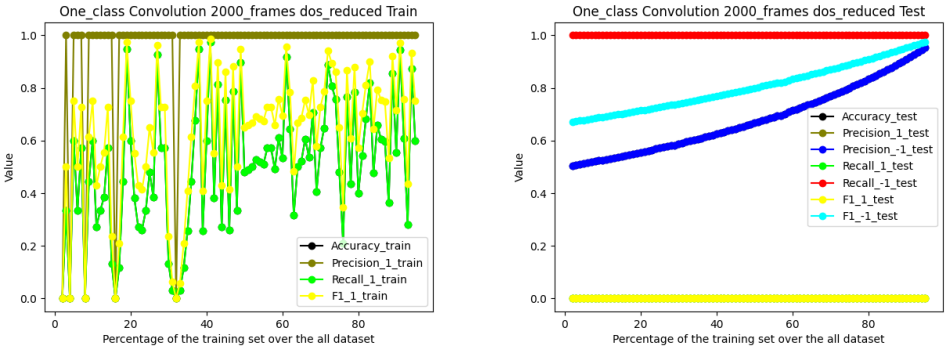

Figure B.21: DoS reduced dataset

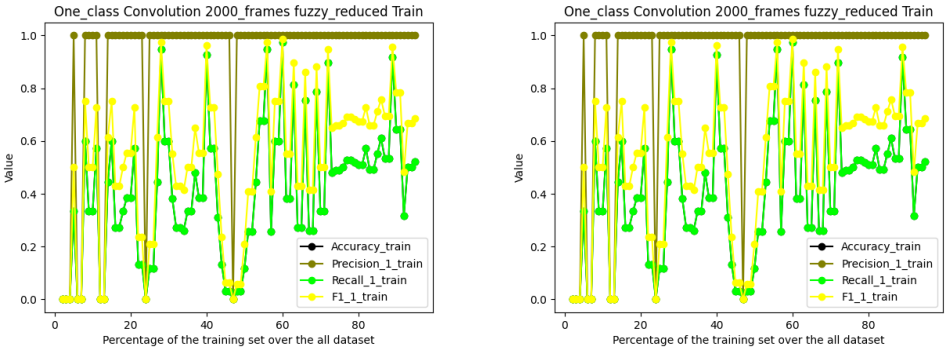

Figure B.22: Fuzzy reduced dataset

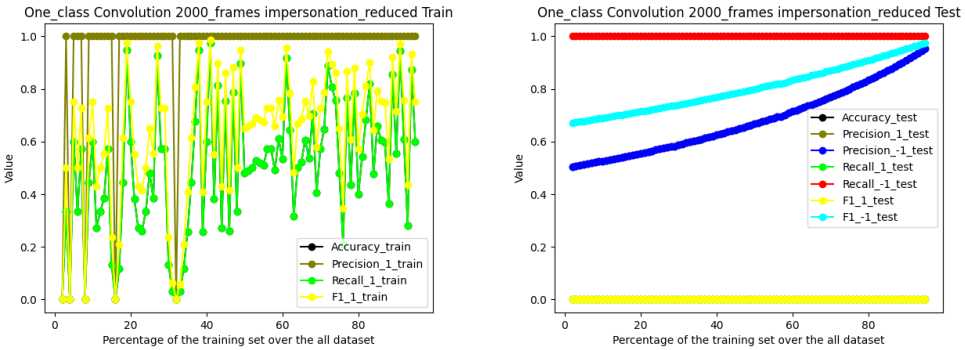

Figure B.23: Impersonation reduced dataset

B.2 SVC

B.2.1 AES

75 frames dataset

Reduced dataset

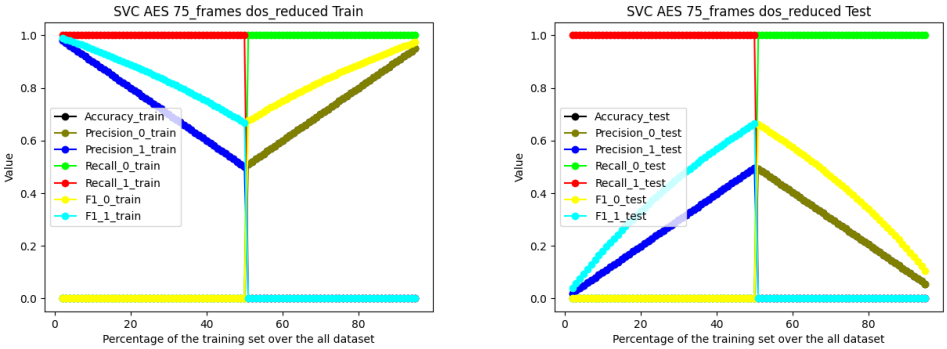

Figure B.24: DoS reduced dataset

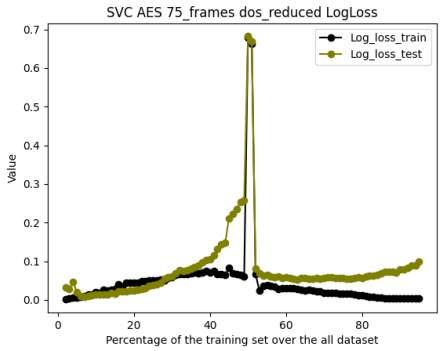

Figure B.25: DoS Log loss reduced dataset

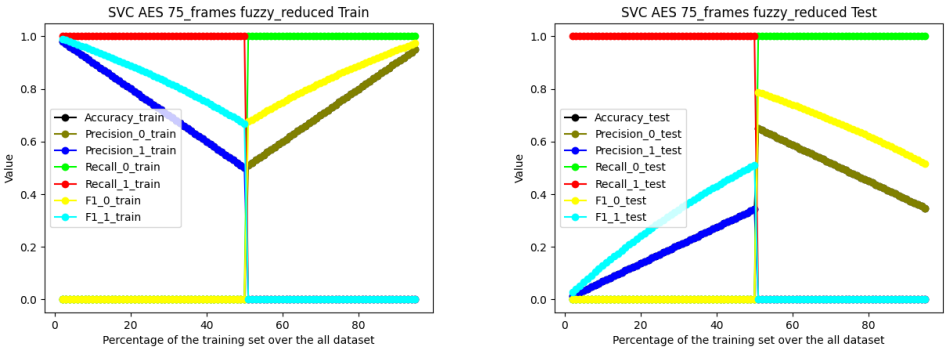

Figure B.26: Fuzzy reduced dataset

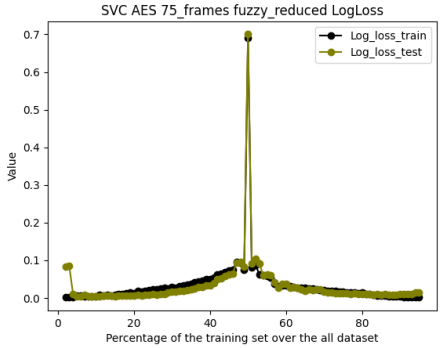

Figure B.27: Fuzzy Log loss reduced dataset

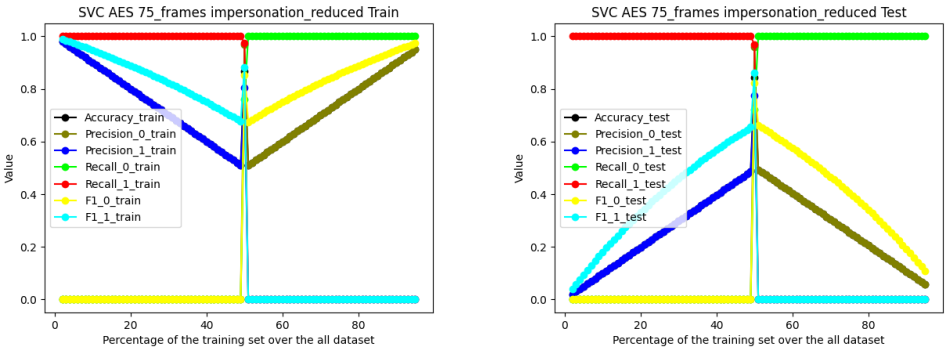

Figure B.28: Impersonation reduced dataset

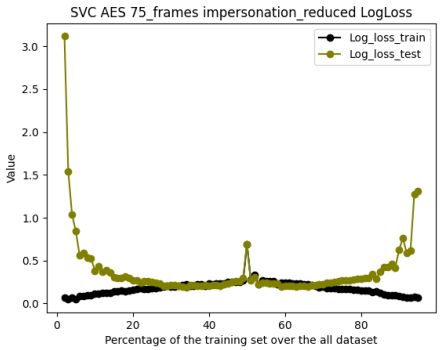

Figure B.29: Impersonation Log loss reduced dataset

1500 frames dataset

Full dataset

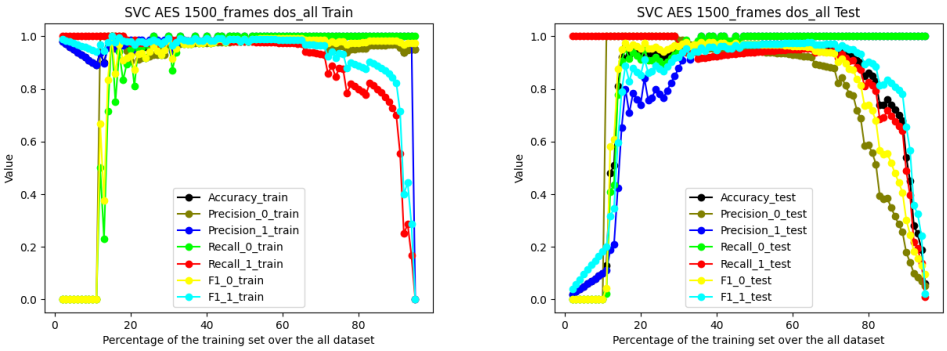

Figure B.30: DoS full dataset

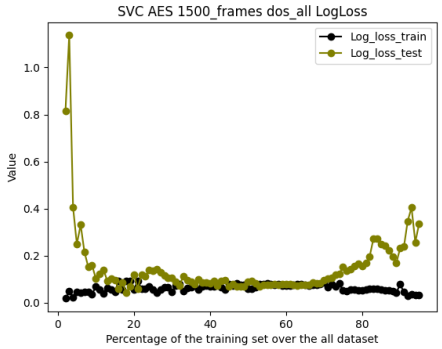

Figure B.31: DoS Log loss full dataset

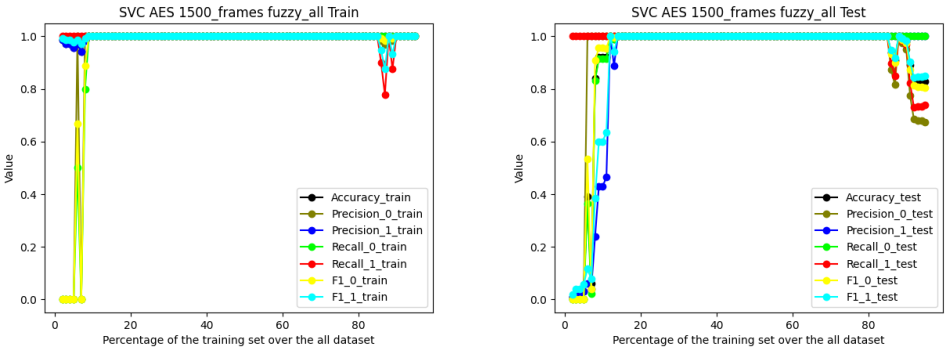

Figure B.32: Fuzzy full dataset

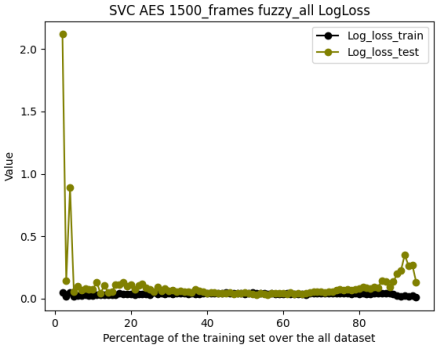

Figure B.33: Fuzzy Log loss full dataset

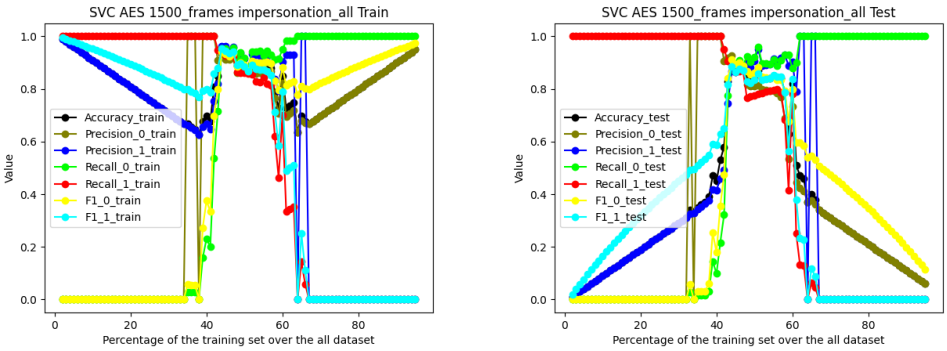

Figure B.34: Impersonation full dataset

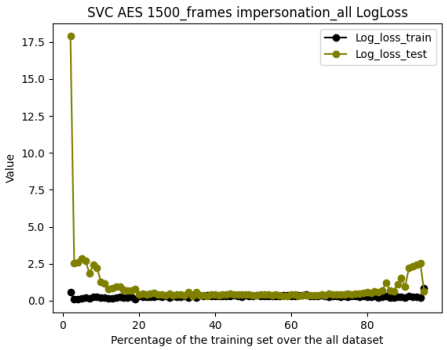

Figure B.35: Impersonation Log loss full dataset

Mean and scale dataset

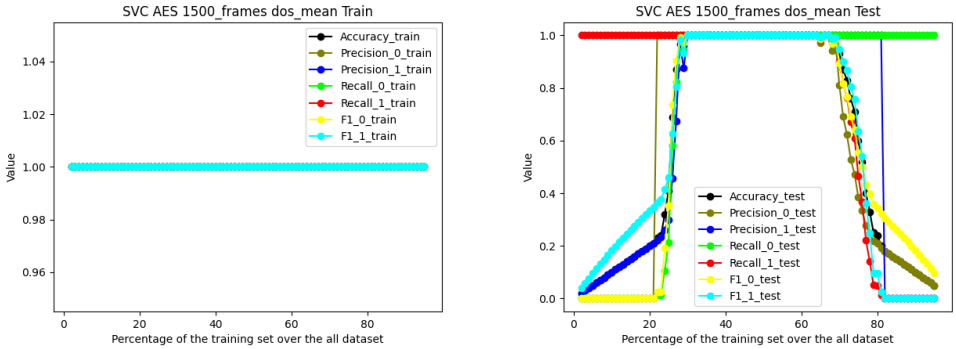

Figure B.36: DoS mean and scale dataset

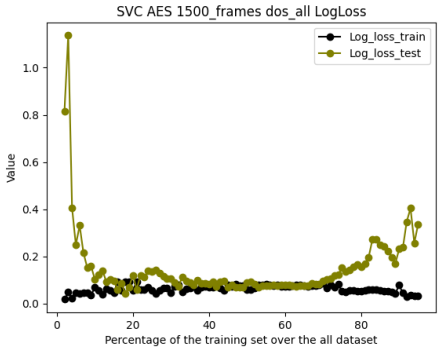

Figure B.37: DoS Log loss mean and scale dataset

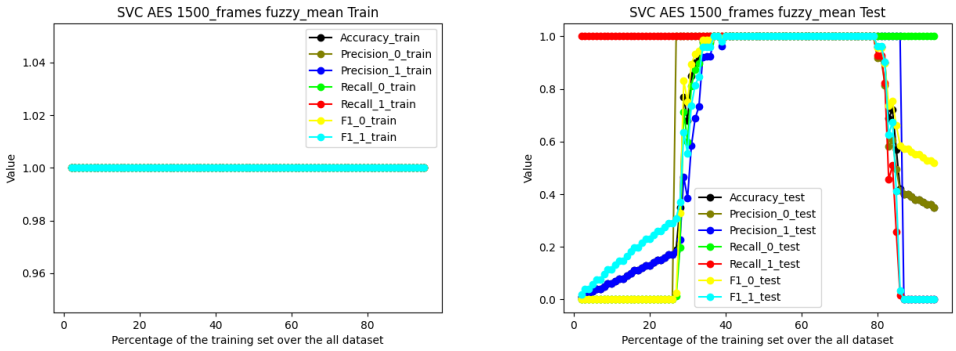

Figure B.38: Fuzzy mean and scale dataset

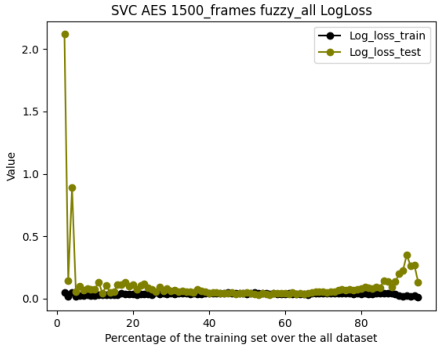

Figure B.39: Fuzzy Log loss mean and scale dataset

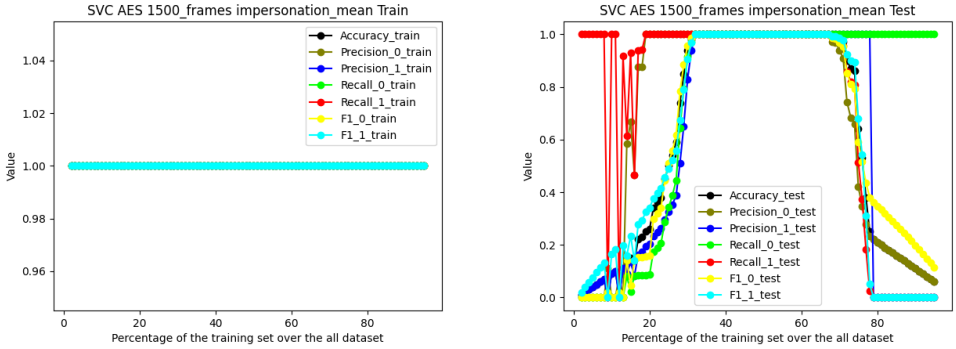

Figure B.40: Impersonation mean and scale dataset

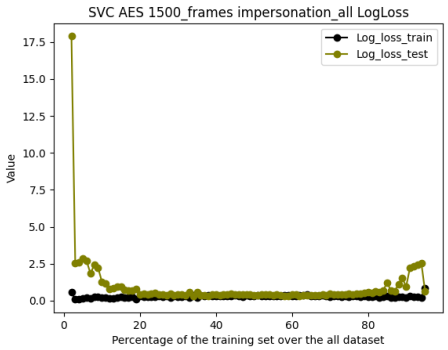

Figure B.41: Impersonation Log loss mean and scale dataset

Correlation reduced dataset

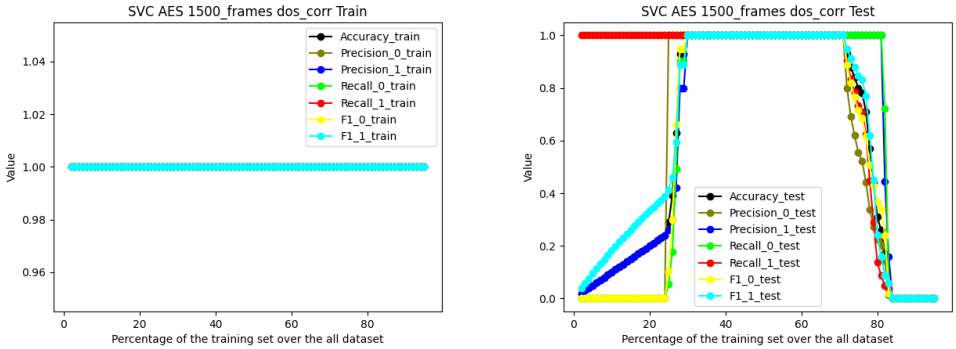

Figure B.42: DoS correlation reduced dataset

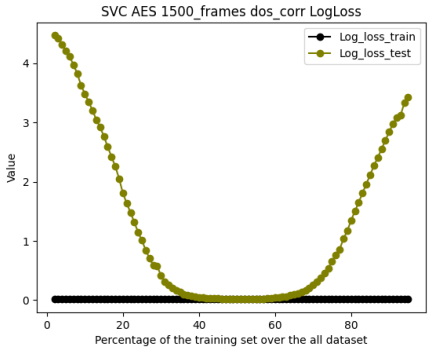

Figure B.43: DoS Log loss correlation reduced dataset

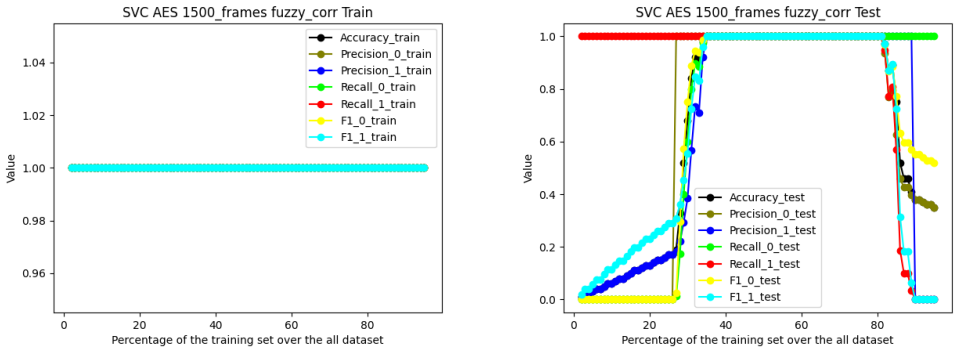

Figure B.44: Fuzzy correlation reduced dataset

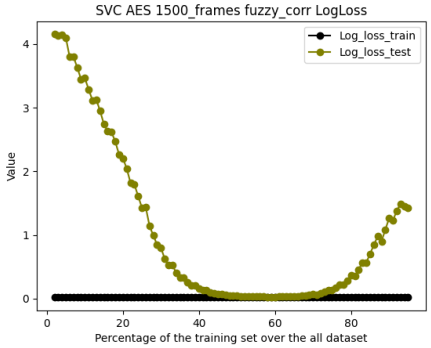

Figure B.45: Fuzzy Log loss correlation reduced dataset

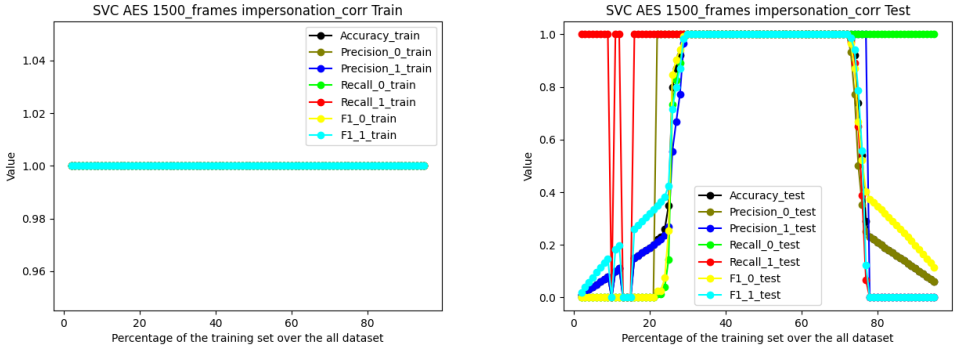

Figure B.46: Impersonation correlation reduced dataset

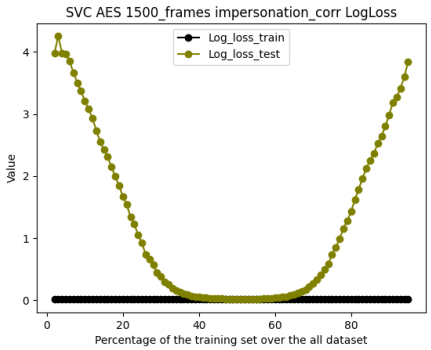

Figure B.47: Impersonation Log loss correlation reduced dataset

Reduced dataset

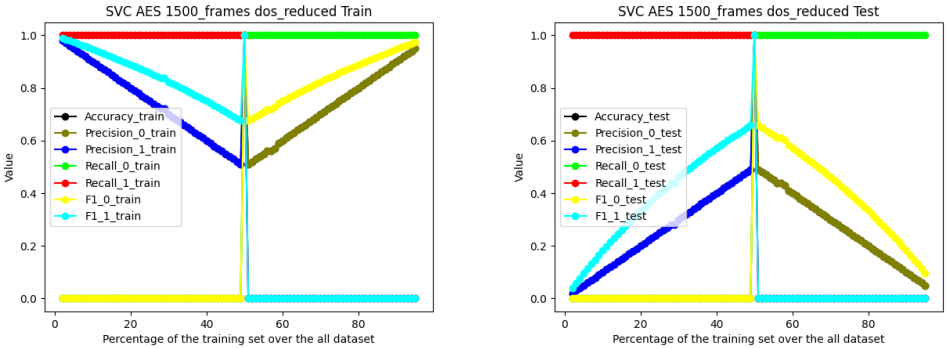

Figure B.48: DoS reduced dataset

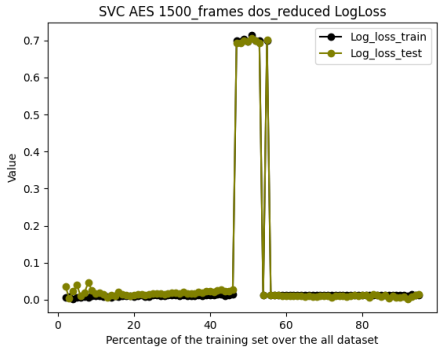

Figure B.49: DoS Log loss reduced dataset

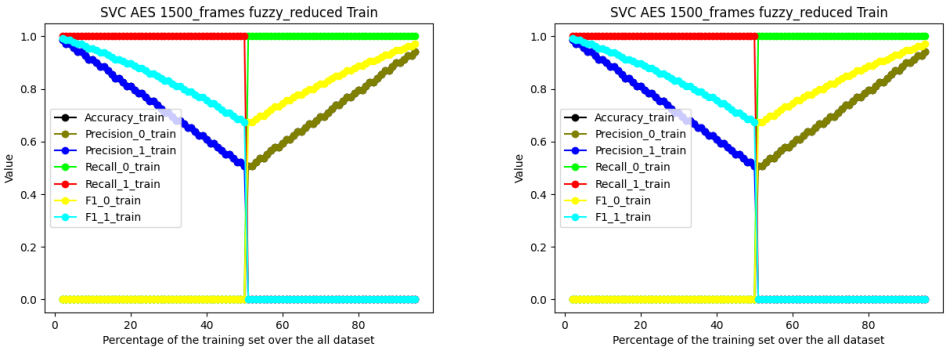

Figure B.50: Fuzzy reduced dataset

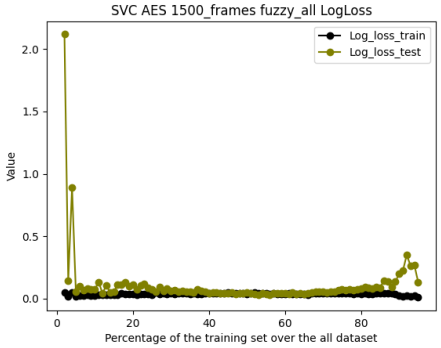

Figure B.51: Fuzzy Log loss reduced dataset

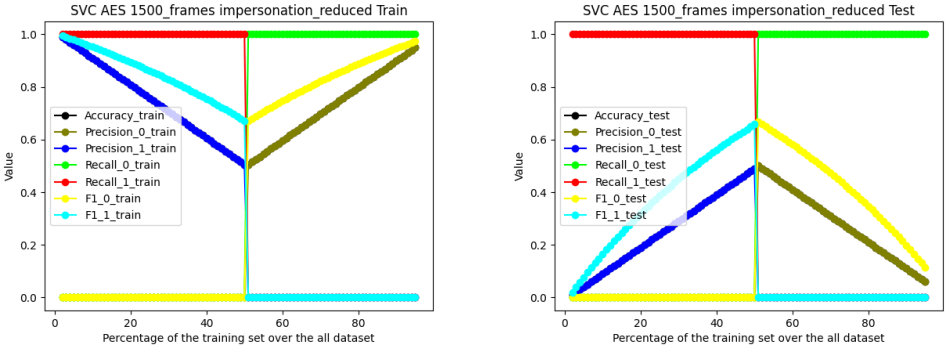

Figure B.52: Impersonation reduced dataset

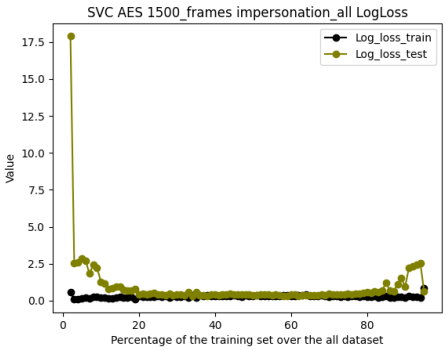

Figure B.53: Impersonation Log loss reduced dataset

B.2.2 Convolution

100 frames dataset

REDUCED DATASET

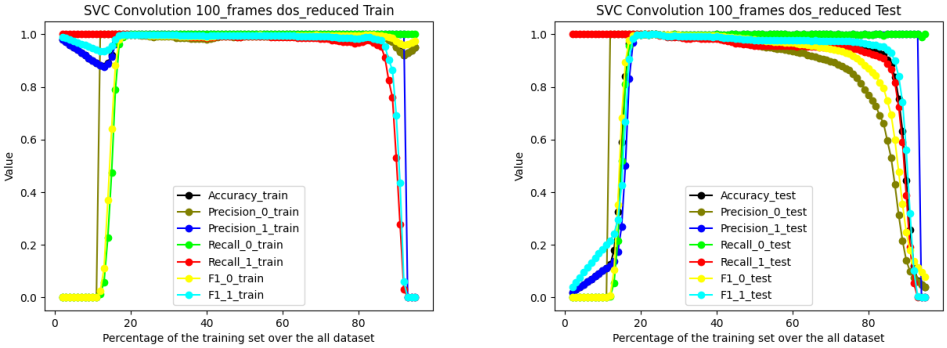

Figure B.54: DoS reduced dataset

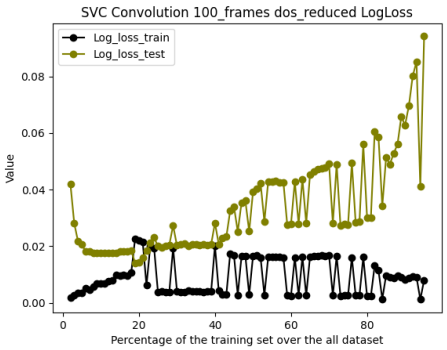

Figure B.55: DoS Log loss reduced dataset

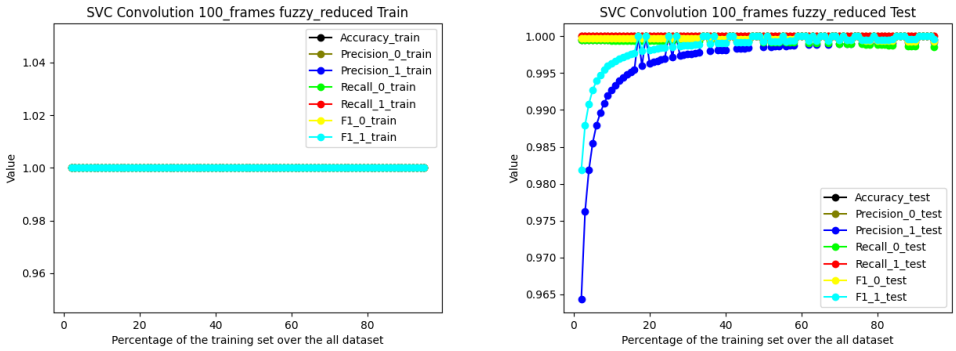

Figure B.56: Fuzzy reduced dataset

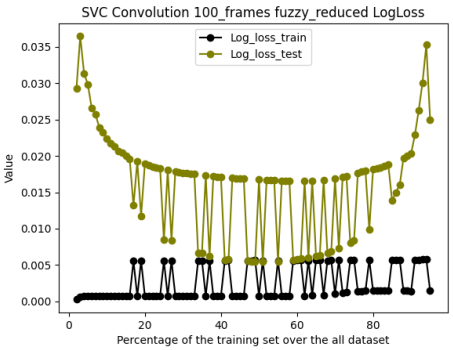

Figure B.57: Fuzzy Log loss reduced dataset

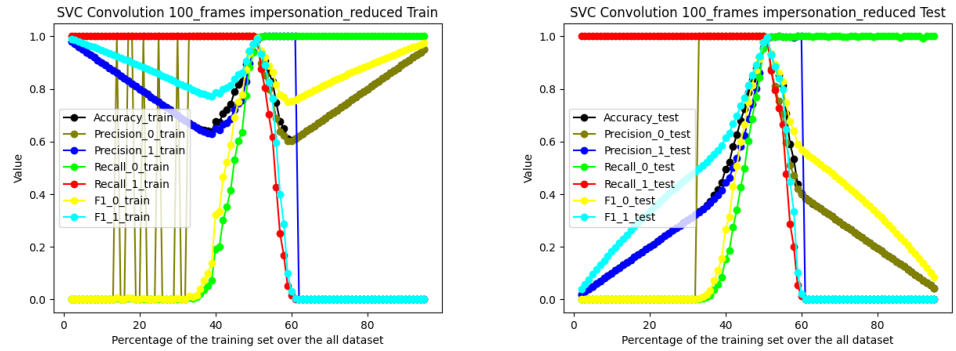

Figure B.58: Impersonation reduced dataset

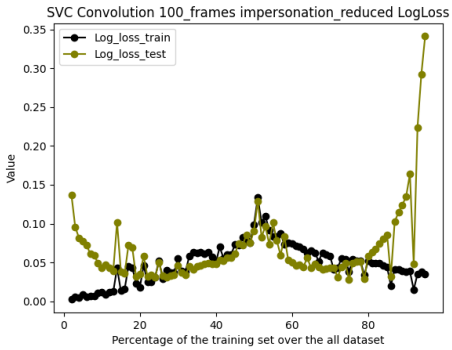

Figure B.59: Impersonation Log loss reduced dataset

2000 frames dataset

Full dataset

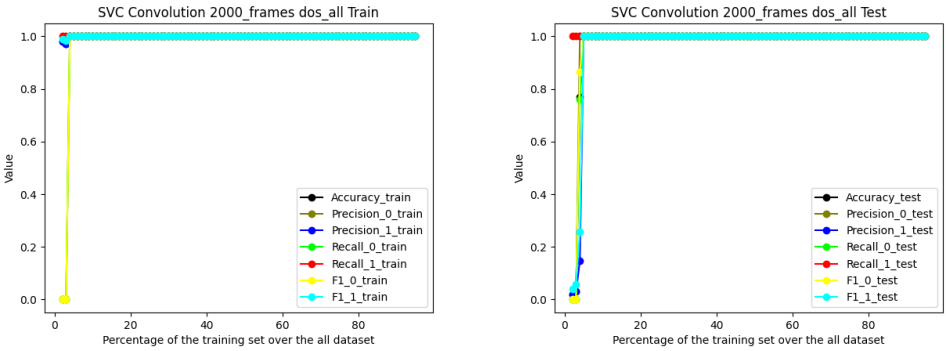

Figure B.60: DoS full dataset

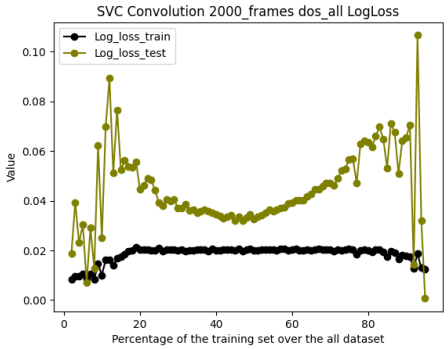

Figure B.61: DoS Log loss full dataset

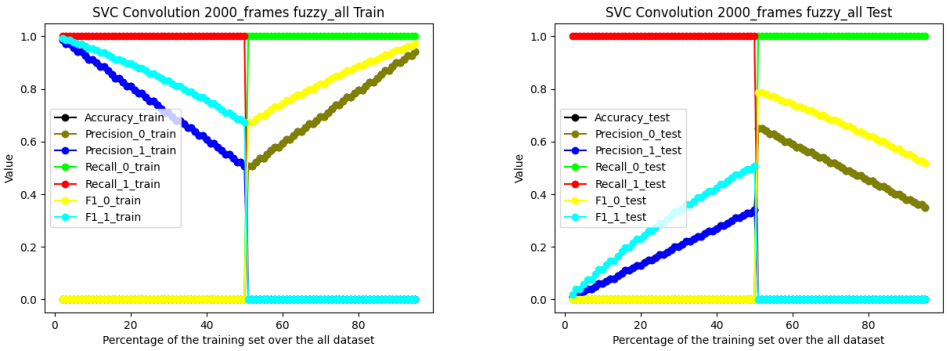

Figure B.62: Fuzzy full dataset

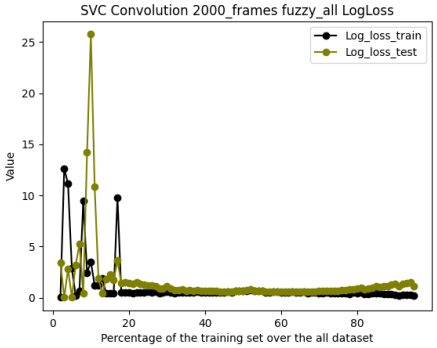

Figure B.63: Fuzzy Log loss full dataset

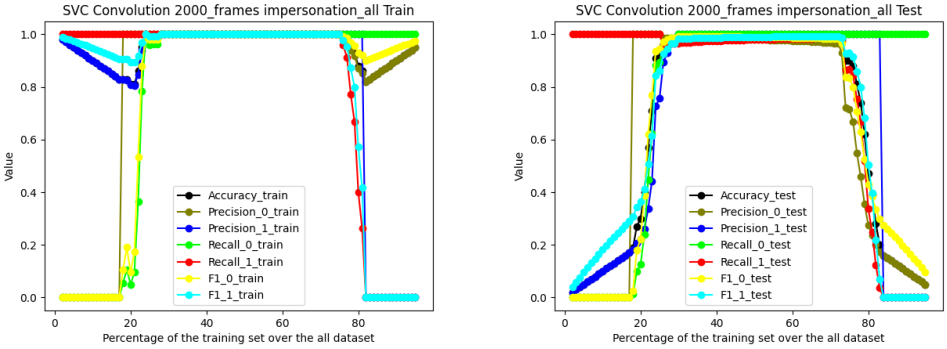

Figure B.64: Impersonation full dataset

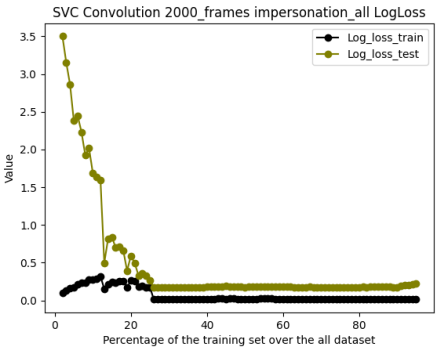

Figure B.65: Impersonation Log loss full dataset

Mean and scale dataset

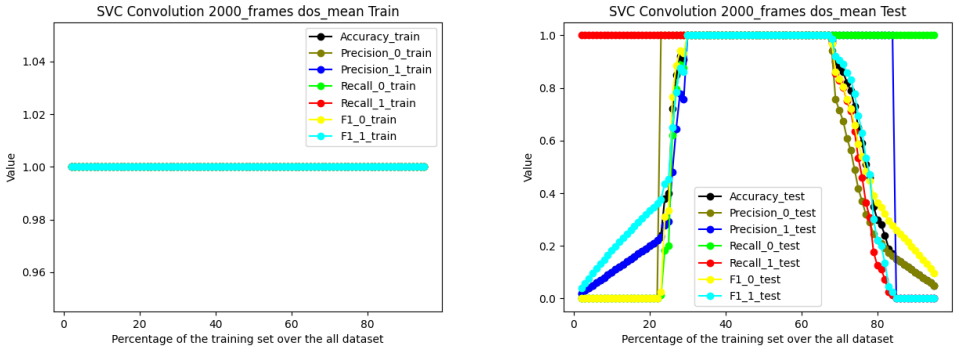

Figure B.66: DoS mean and scale dataset

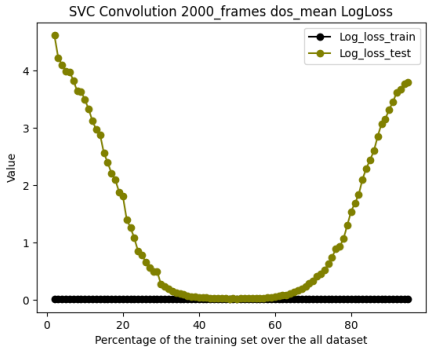

Figure B.67: DoS Log loss mean and scale dataset

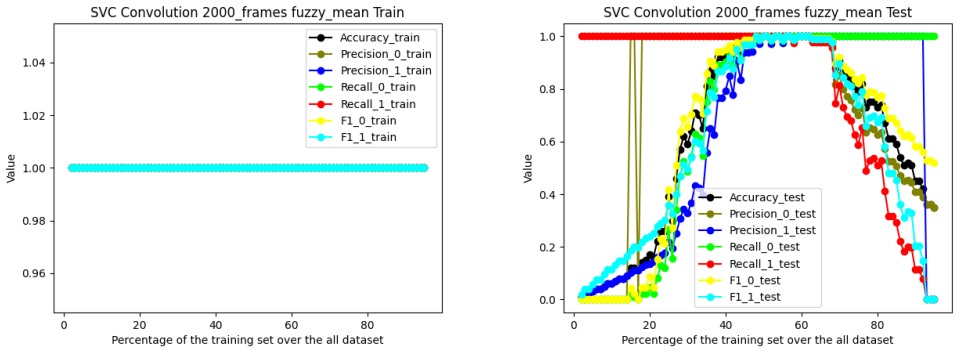

Figure B.68: Fuzzy mean and scale dataset

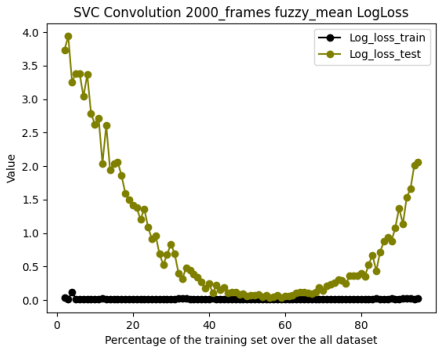

Figure B.69: Fuzzy Log loss mean and scale dataset

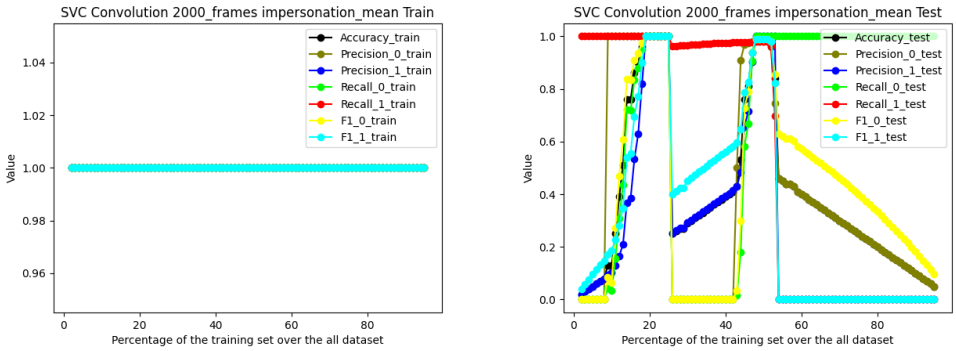

Figure B.70: Impersonation mean and scale dataset

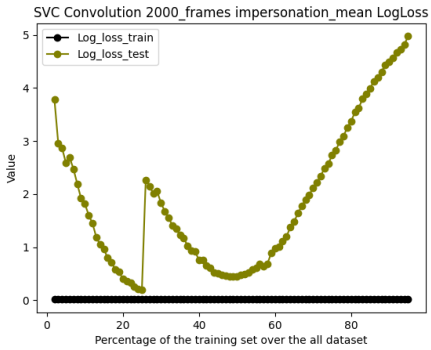

Figure B.71: Impersonation Log loss mean and scale dataset

Correlation reduced dataset

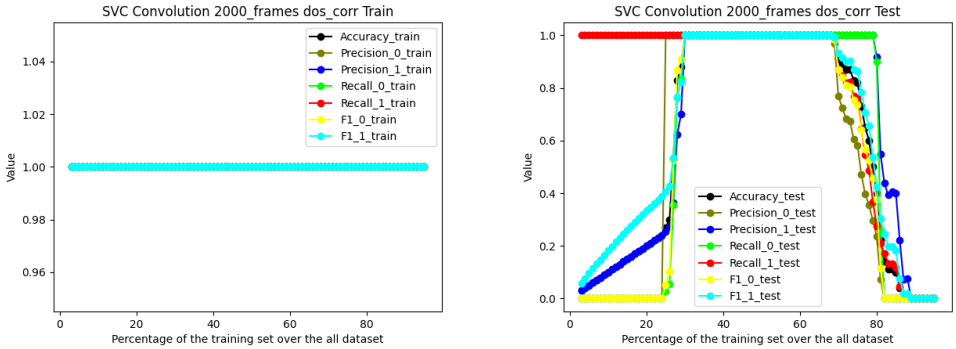

Figure B.72: DoS correlation reduced dataset

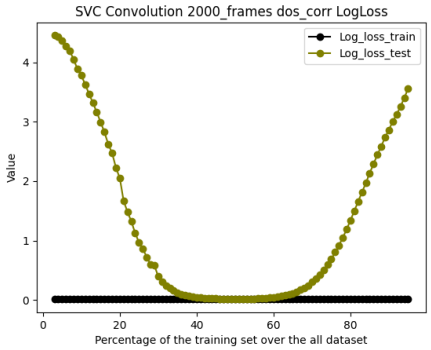

Figure B.73: DoS Log loss correlation reduced dataset

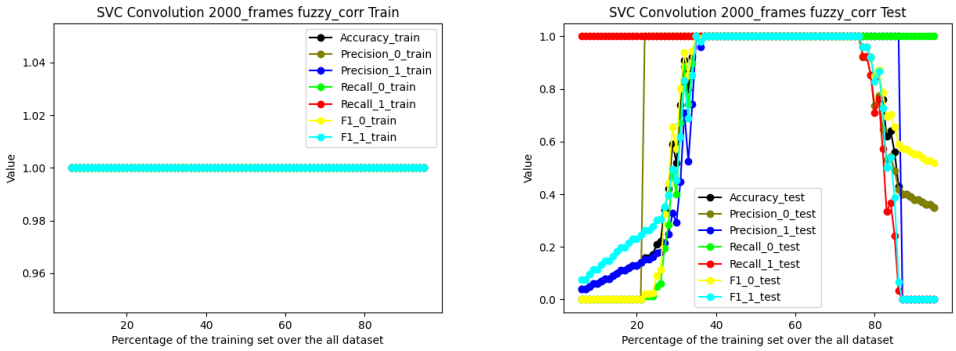

Figure B.74: Fuzzy correlation reduced dataset

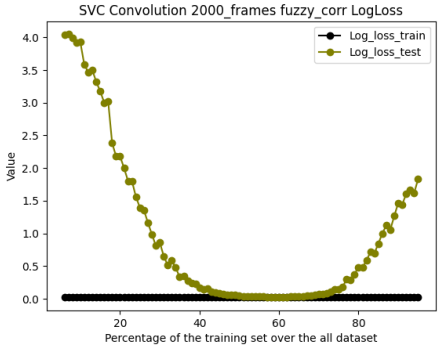

Figure B.75: Fuzzy Log loss correlation reduced dataset

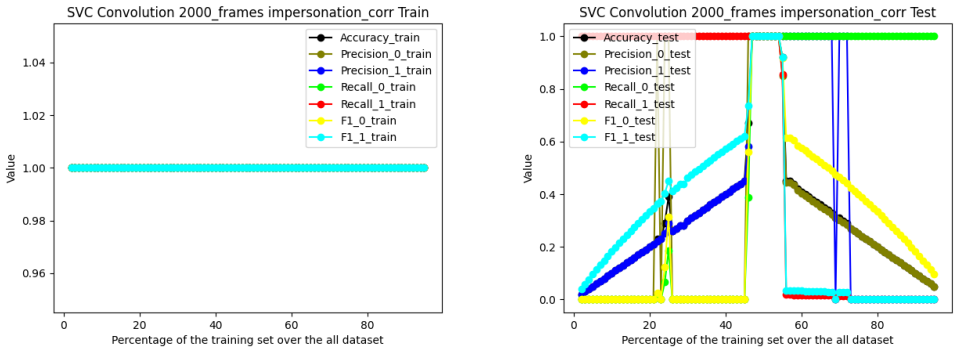

Figure B.76: Impersonation correlation reduced dataset

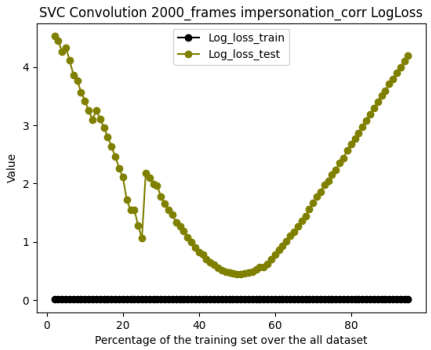

Figure B.77: Impersonation Log loss correlation reduced dataset

Reduced dataset

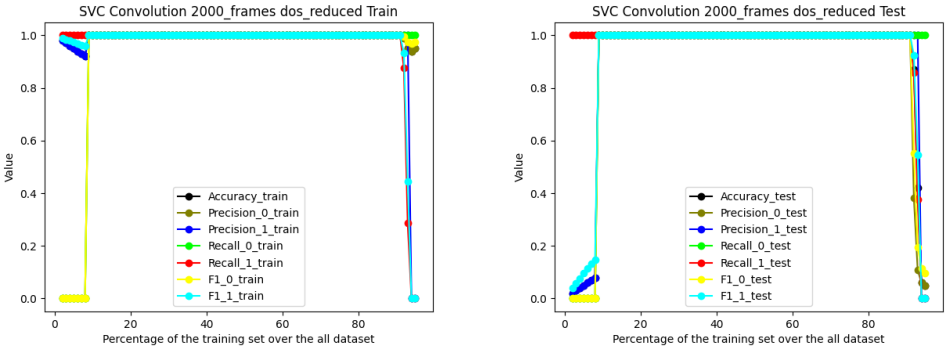

Figure B.78: DoS reduced dataset

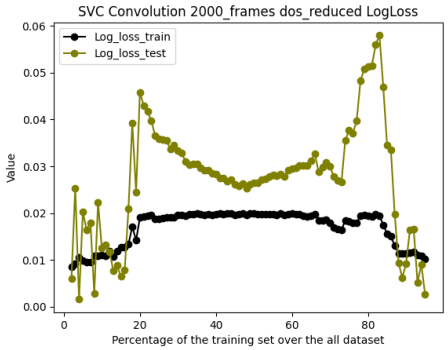

Figure B.79: DoS Log loss reduced dataset

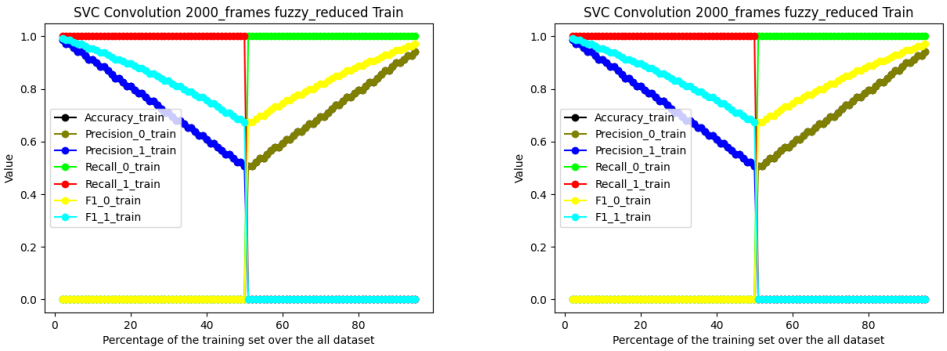

Figure B.80: Fuzzy reduced dataset

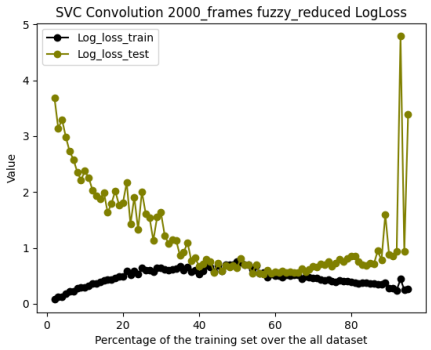

Figure B.81: Fuzzy Log loss reduced dataset

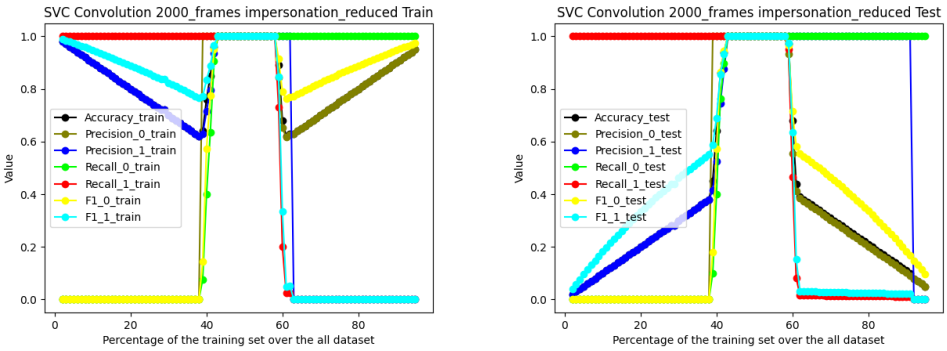

Figure B.82: Impersonation reduced dataset

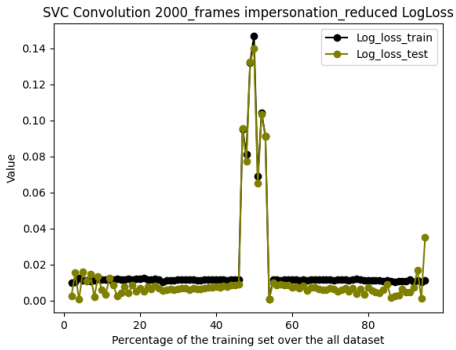

Figure B.83: Impersonation Log loss reduced dataset

B.3 Random forest

B.3.1 AES

75 frames dataset

Reduced dataset

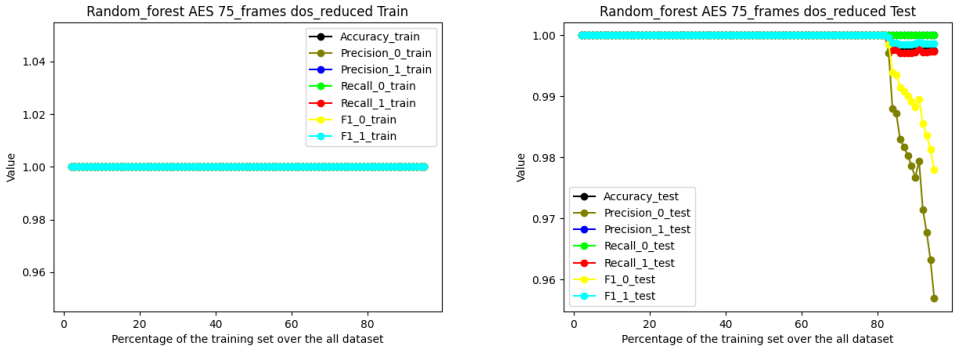

Figure B.84: DoS reduced dataset

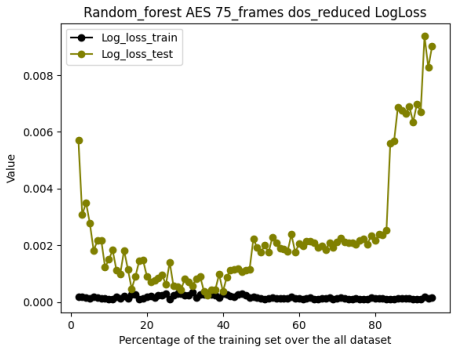

Figure B.85: DoS Log loss reduced dataset

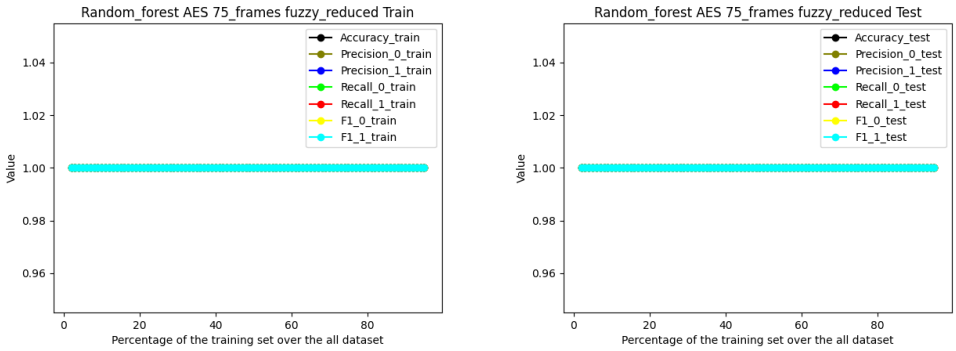

Figure B.86: Fuzzy reduced dataset

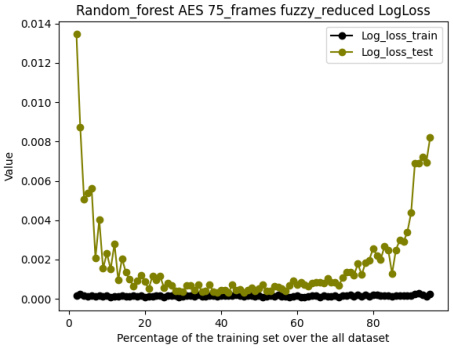

Figure B.87: Fuzzy Log loss reduced dataset

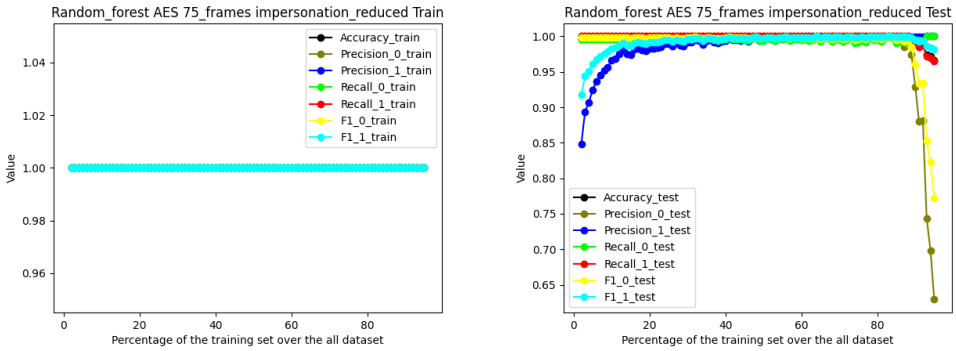

Figure B.88: Impersonation reduced dataset

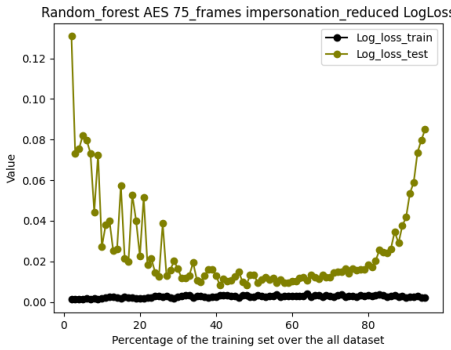

Figure B.89: Impersonation Log loss reduced dataset

1500 frames dataset

Full dataset

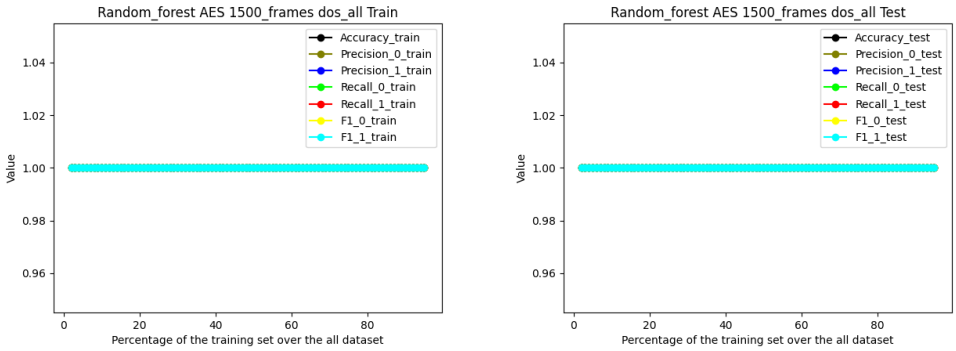

Figure B.90: DoS full dataset

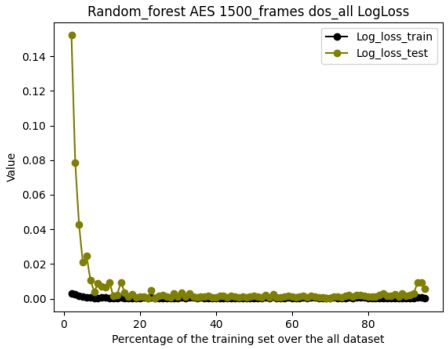

Figure B.91: DoS Log loss full dataset

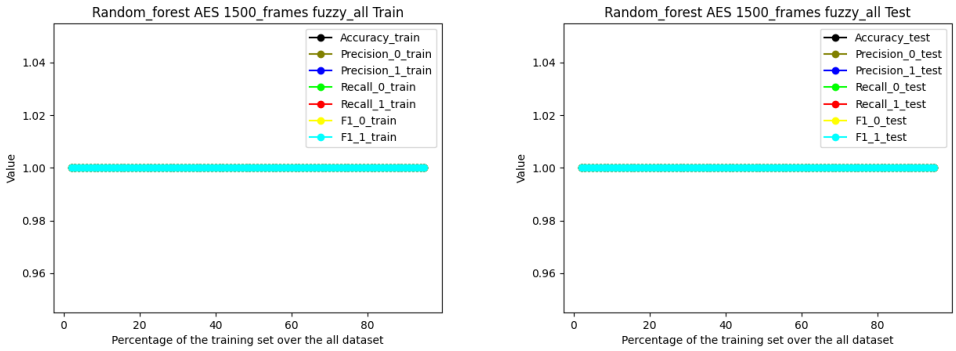

Figure B.92: Fuzzy full dataset

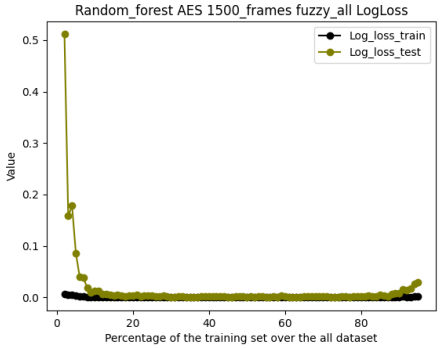

Figure B.93: Fuzzy Log loss full dataset

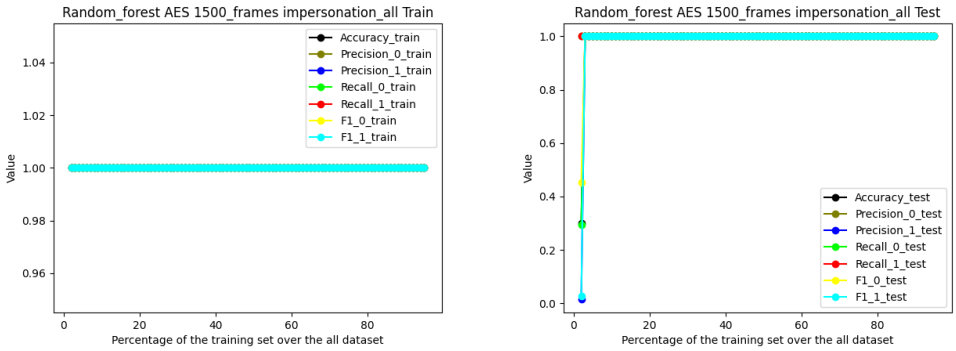

Figure B.94: Impersonation full dataset

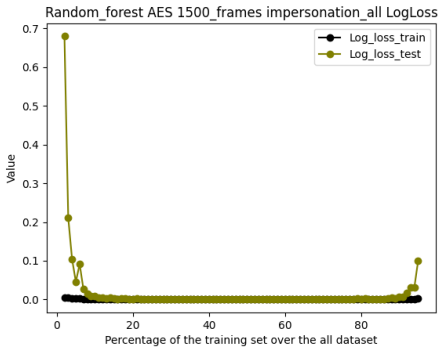

Figure B.95: Impersonation Log loss full dataset

Mean and scale dataset

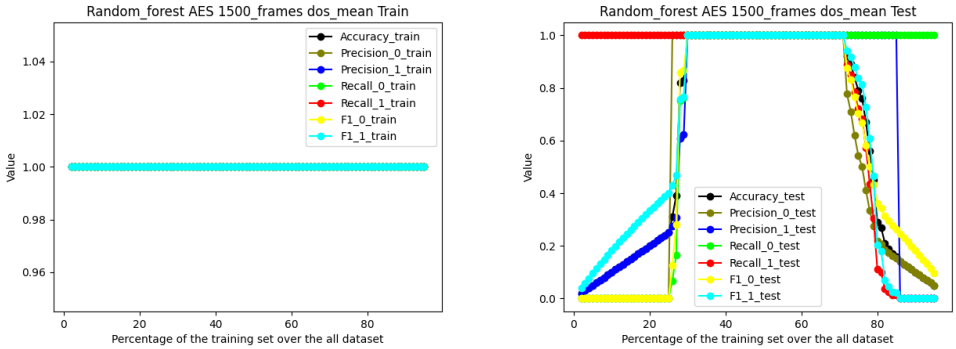

Figure B.96: DoS mean and scale dataset

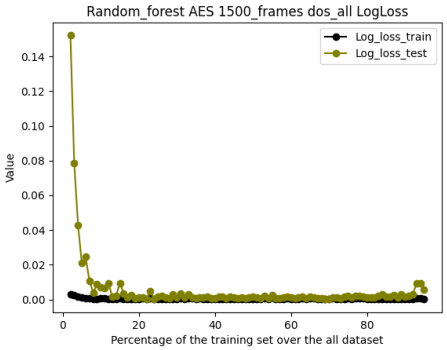

Figure B.97: DoS Log loss mean and scale dataset

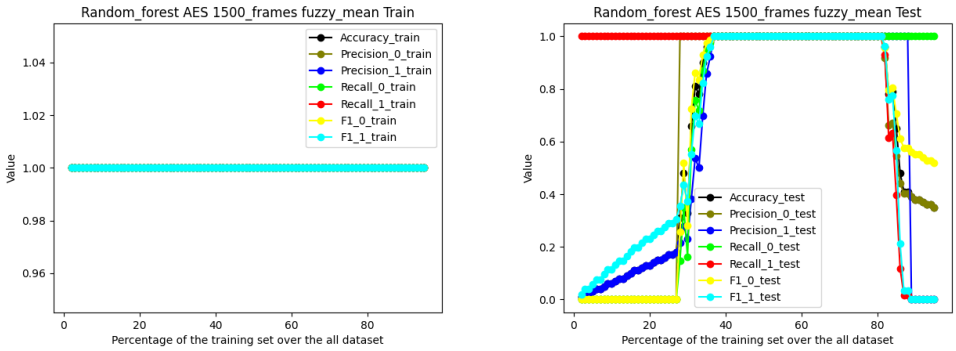

Figure B.98: Fuzzy mean and scale dataset

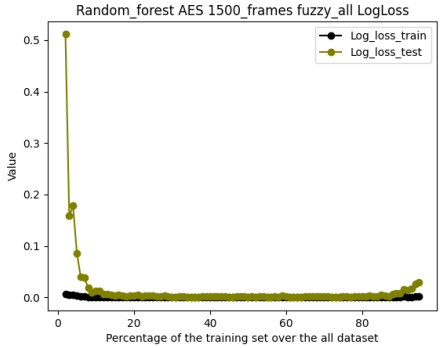

Figure B.99: Fuzzy Log loss mean and scale dataset

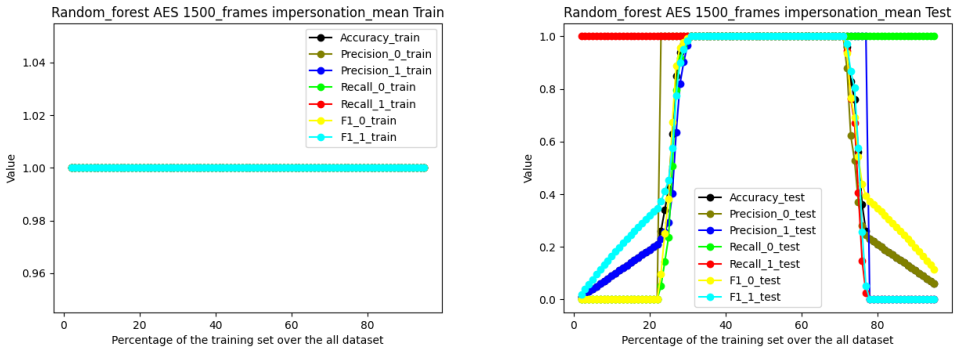

Figure B.100: Impersonation mean and scale dataset

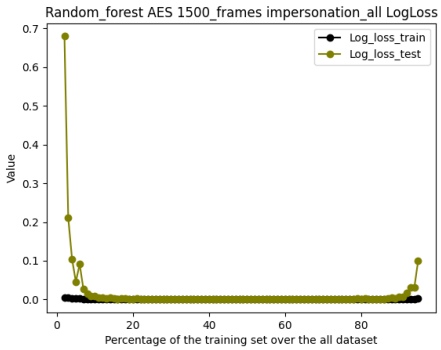

Figure B.101: Impersonation Log loss mean and scale dataset

Correlation reduced dataset

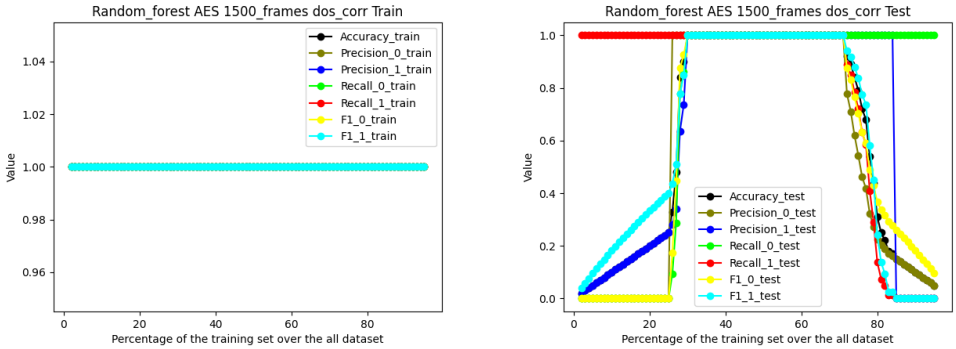

Figure B.102: DoS correlation reduced dataset

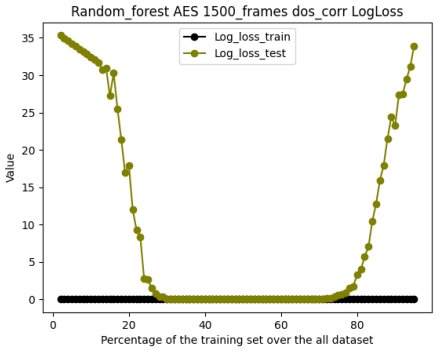

Figure B.103: DoS Log loss correlation reduced dataset

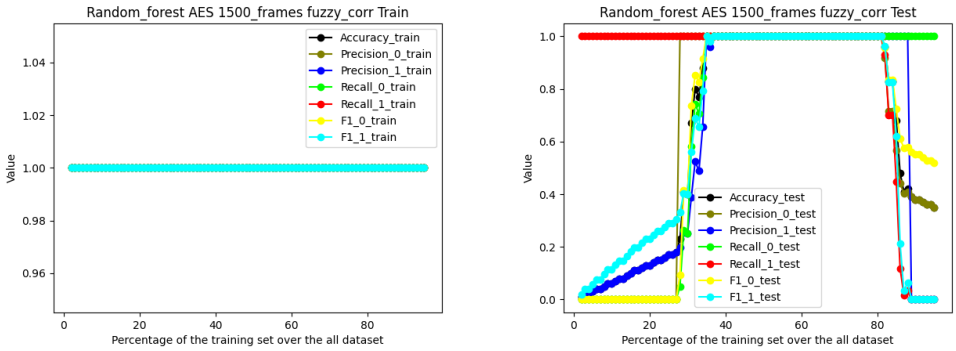

Figure B.104: Fuzzy correlation reduced dataset

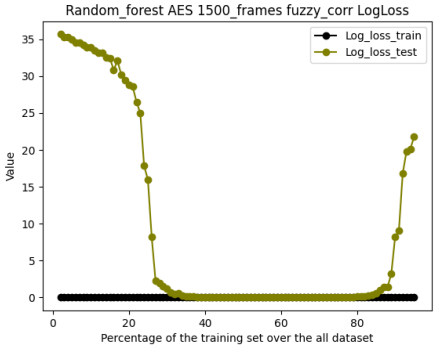

Figure B.105: Fuzzy Log loss correlation reduced dataset

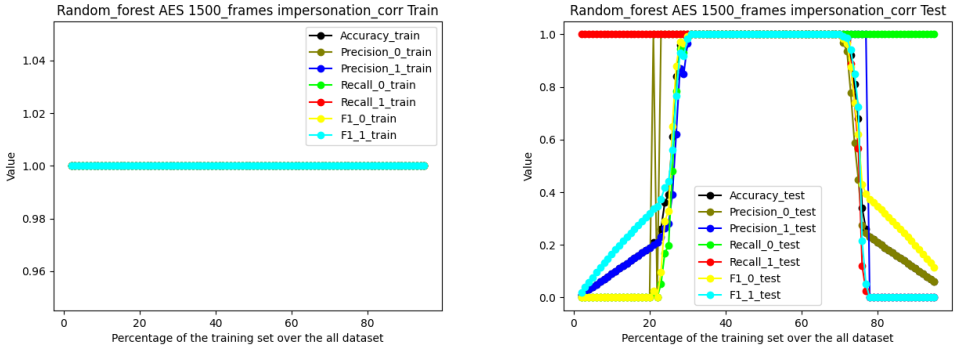

Figure B.106: Impersonation correlation reduced dataset

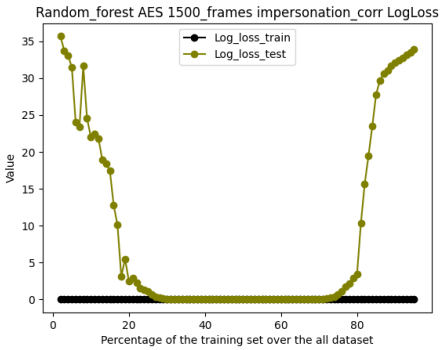

Figure B.107: Impersonation Log loss correlation reduced dataset

Reduced dataset

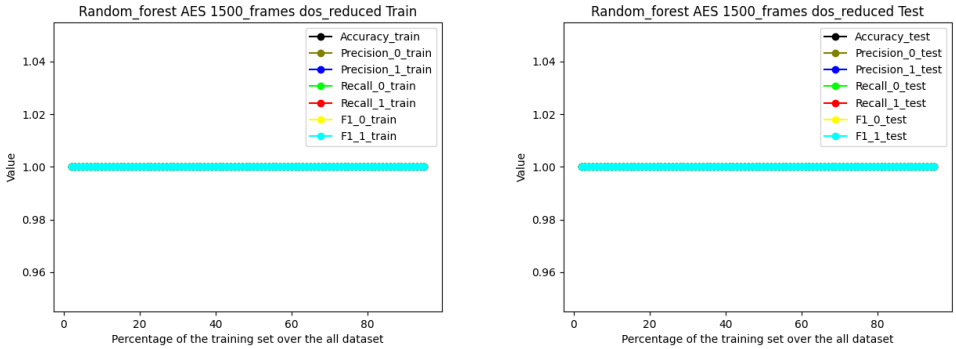

Figure B.108: DoS reduced dataset

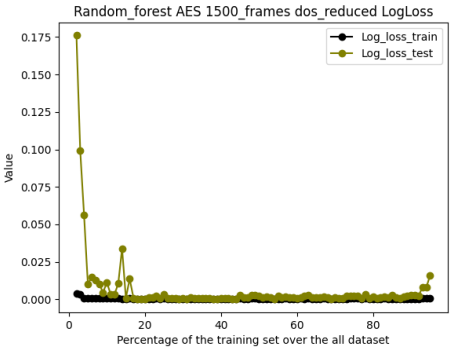

Figure B.109: DoS Log loss reduced dataset

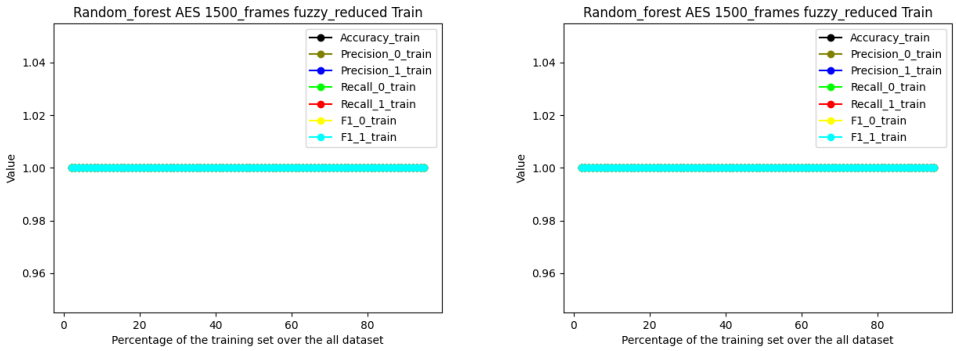

Figure B.110: Fuzzy reduced dataset

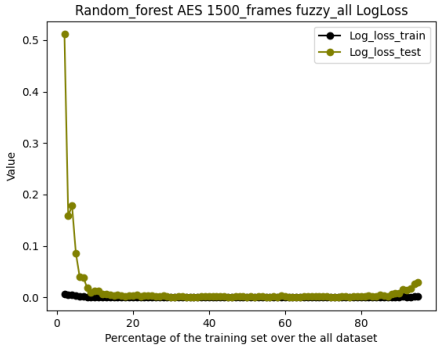

Figure B.111: Fuzzy Log loss reduced dataset

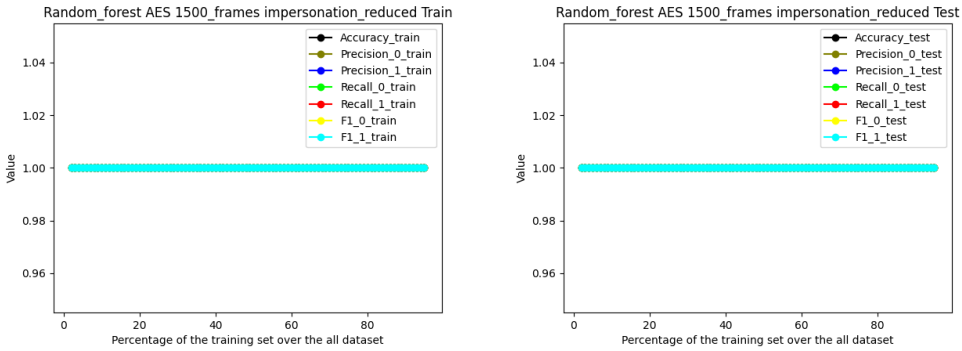

Figure B.112: Impersonation reduced dataset

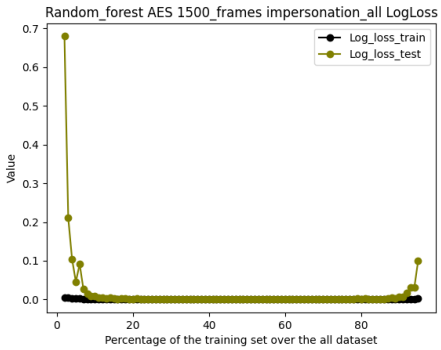

Figure B.113: Impersonation Log loss reduced dataset

B.3.2 Convolution

100 frames dataset

REDUCED DATASET

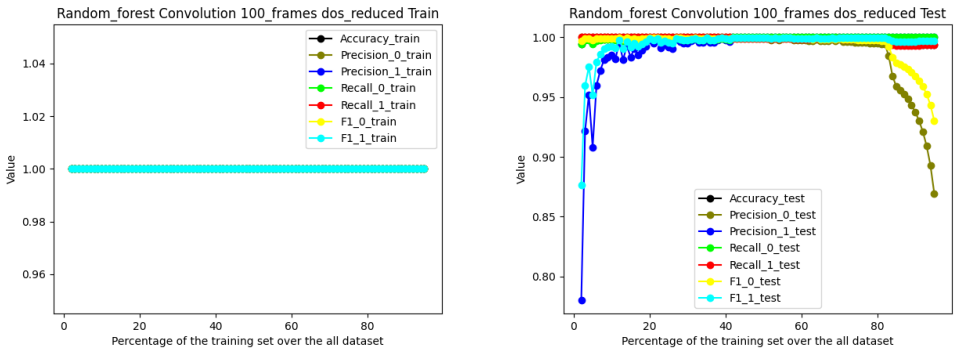

Figure B.114: DoS reduced dataset

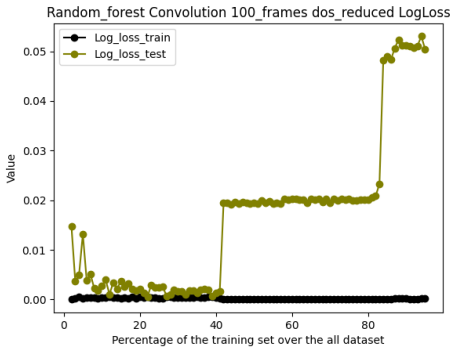

Figure B.115: DoS Log loss reduced dataset

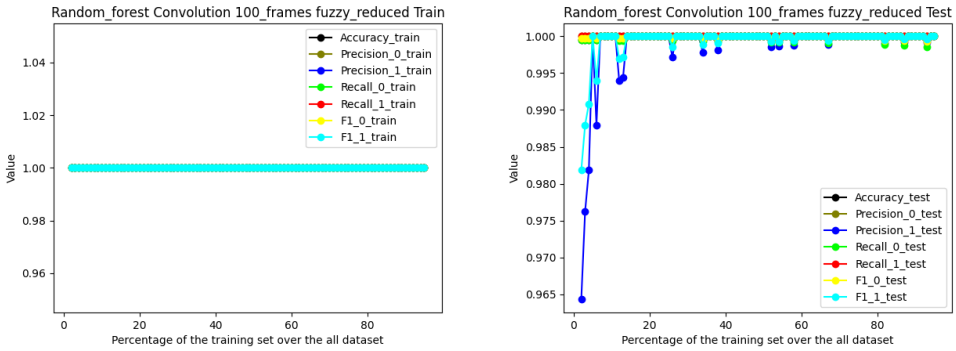

Figure B.116: Fuzzy reduced dataset

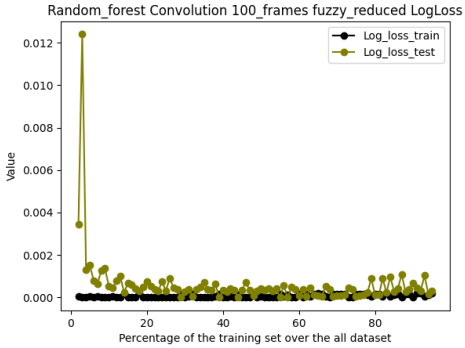

Figure B.117: Fuzzy Log loss reduced dataset

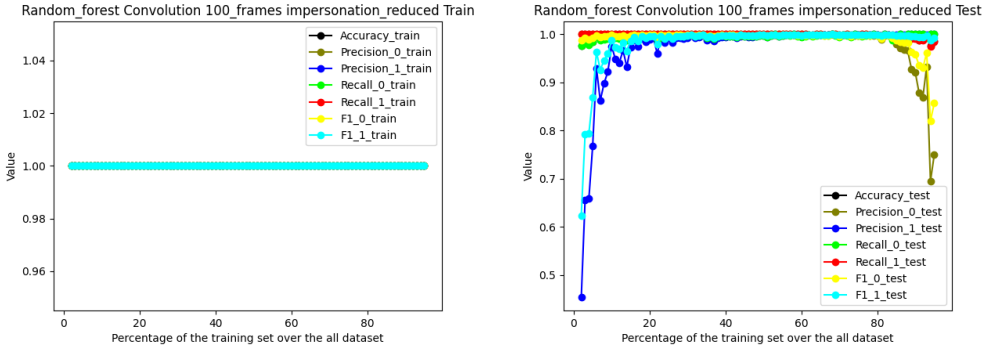

Figure B.118: Impersonation reduced dataset

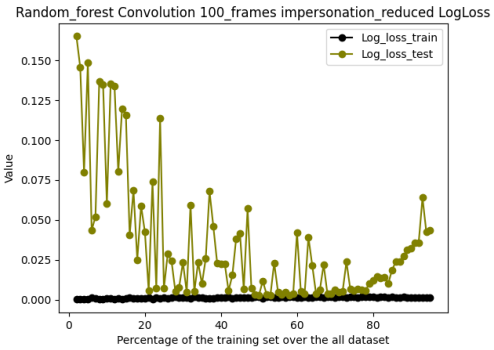

Figure B.119: Impersonation Log loss reduced dataset

2000 frames dataset

Full dataset

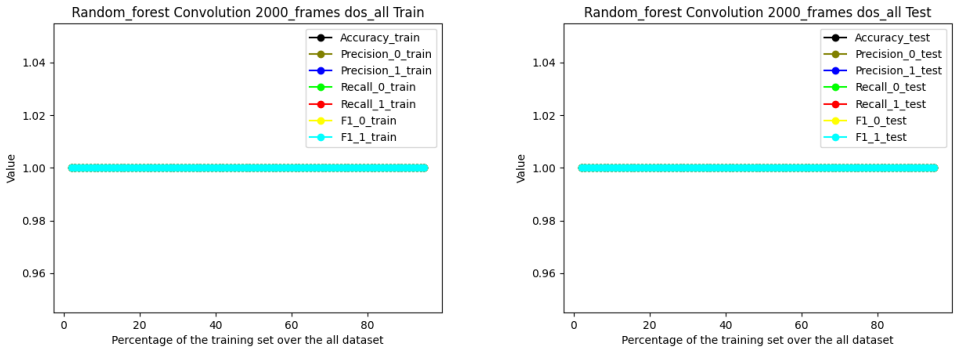

Figure B.120: DoS full dataset

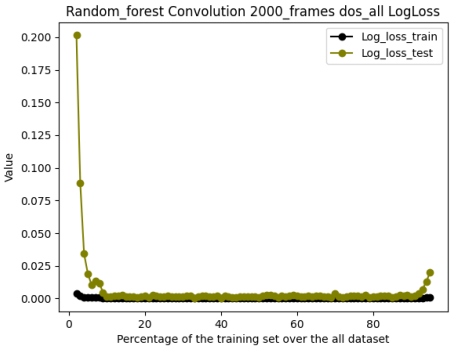

Figure B.121: DoS Log loss full dataset

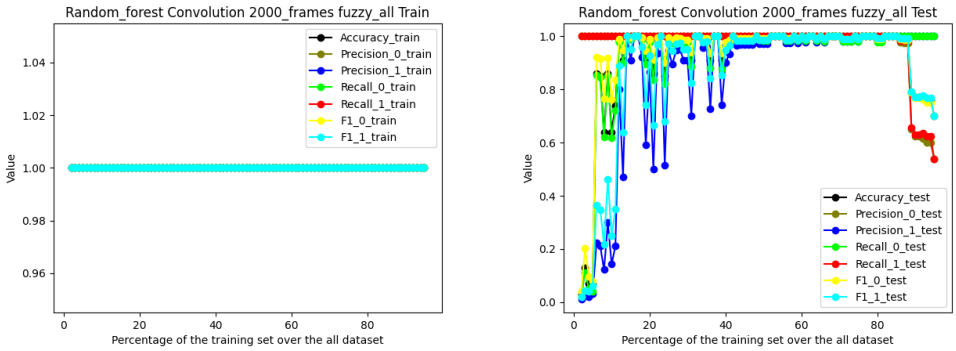

Figure B.122: Fuzzy full dataset

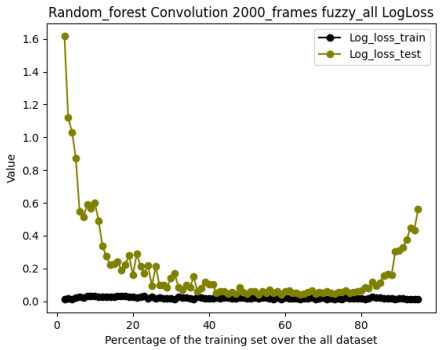

Figure B.123: Fuzzy Log loss full dataset

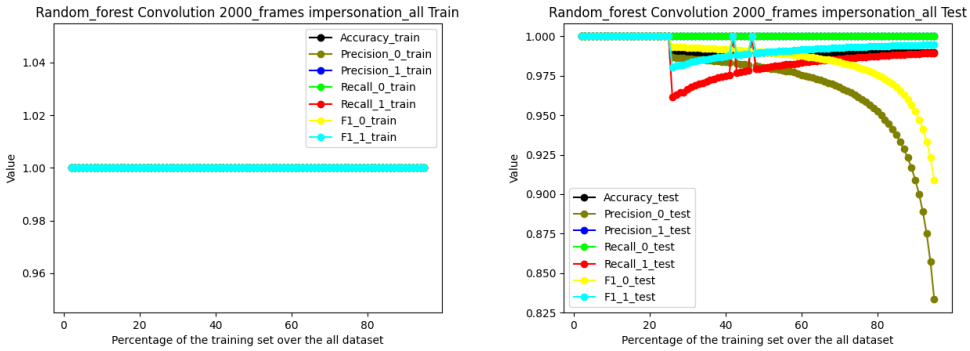

Figure B.124: Impersonation full dataset

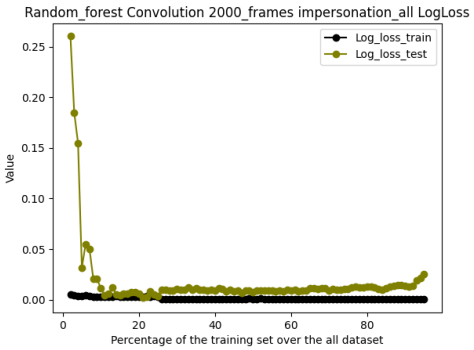

Figure B.125: Impersonation Log loss full dataset

Mean and scale dataset

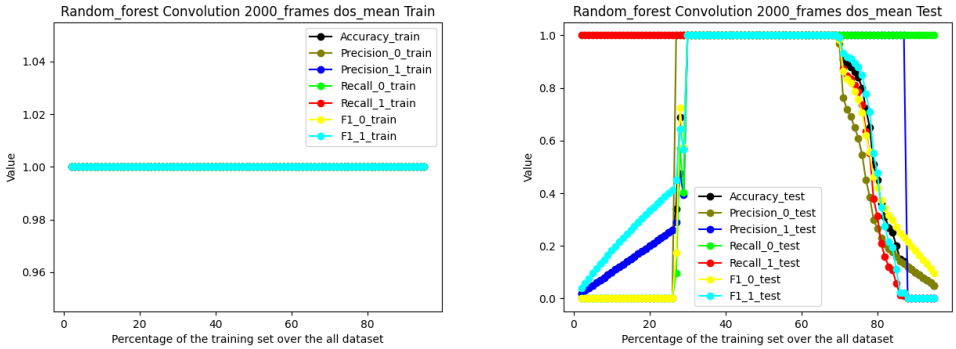

Figure B.126: DoS mean and scale dataset

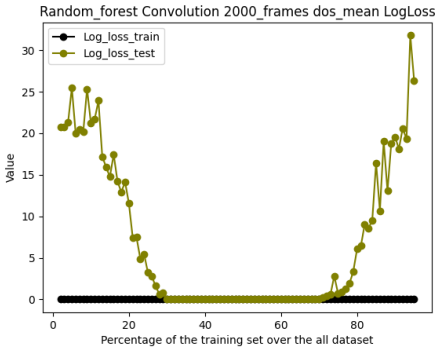

Figure B.127: DoS Log loss mean and scale dataset

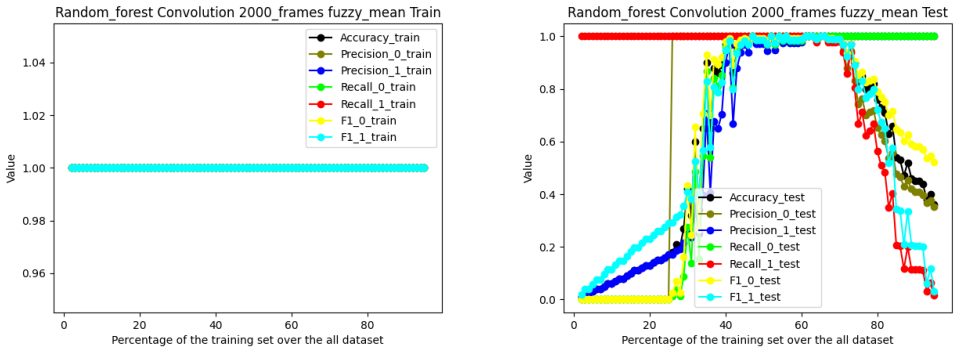

Figure B.128: Fuzzy mean and scale dataset

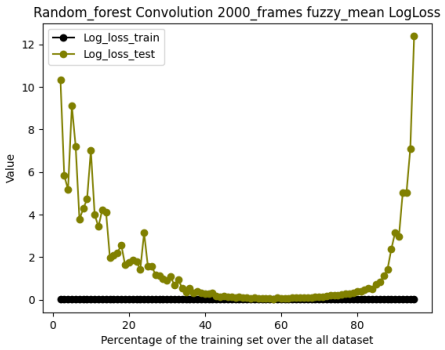

Figure B.129: Fuzzy Log loss mean and scale dataset

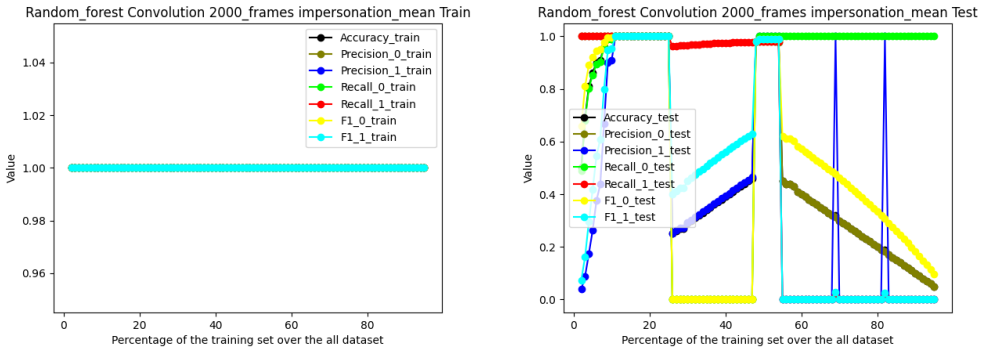

Figure B.130: Impersonation mean and scale dataset

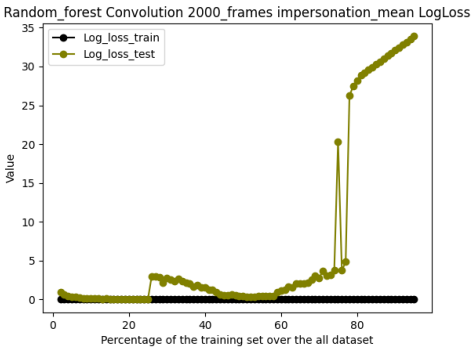

Figure B.131: Impersonation Log loss mean and scale dataset

Correlation reduced dataset

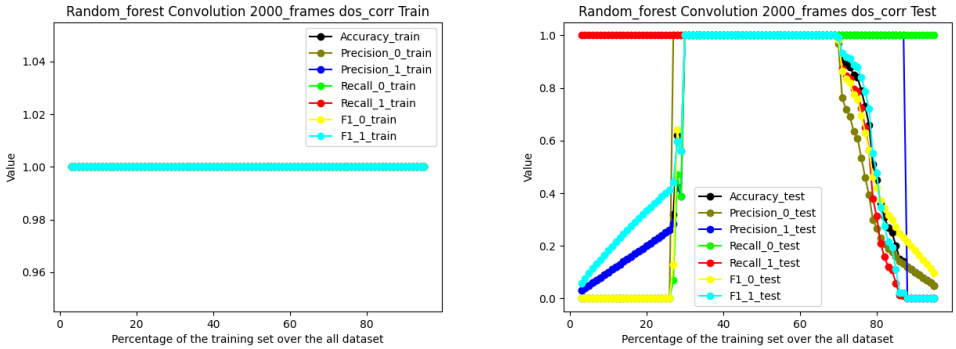

Figure B.132: DoS correlation reduced dataset

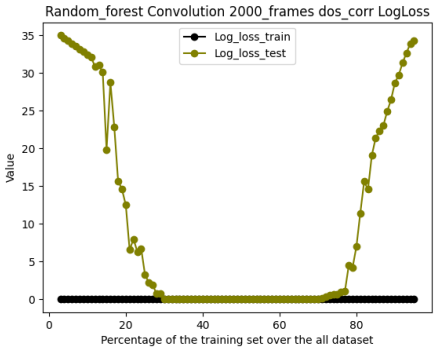

Figure B.133: DoS Log loss correlation reduced dataset

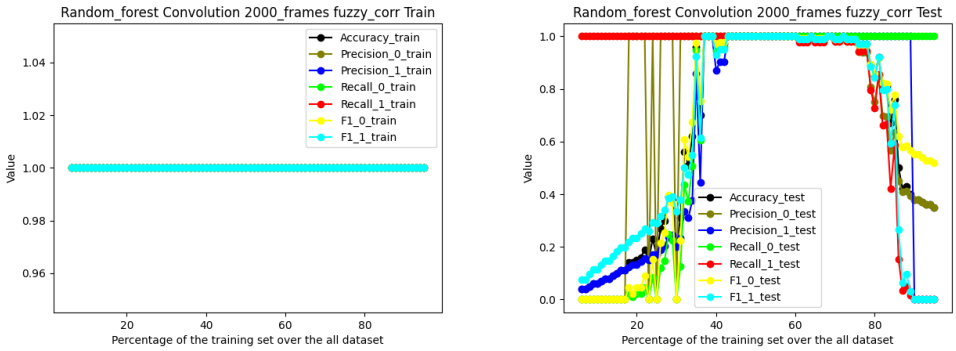

Figure B.134: Fuzzy correlation reduced dataset

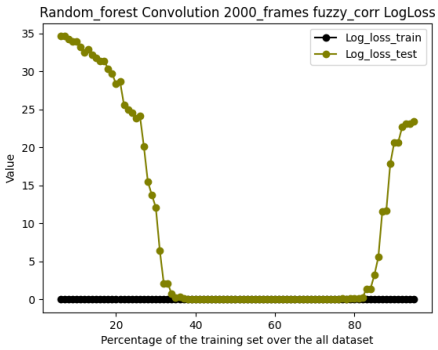

Figure B.135: Fuzzy Log loss correlation reduced dataset

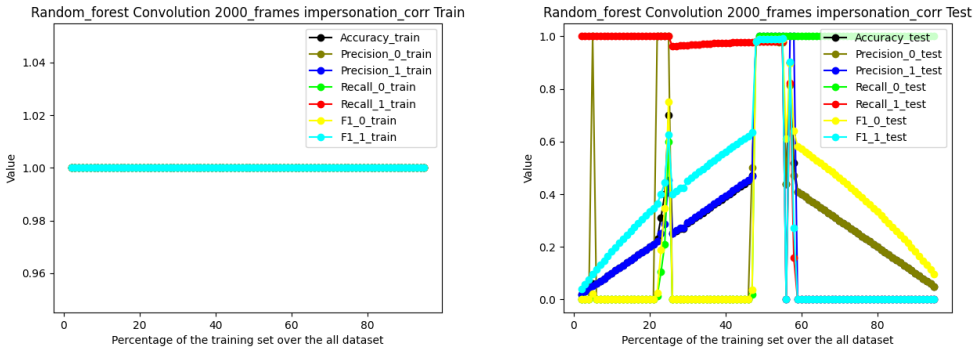

Figure B.136: Impersonation correlation reduced dataset

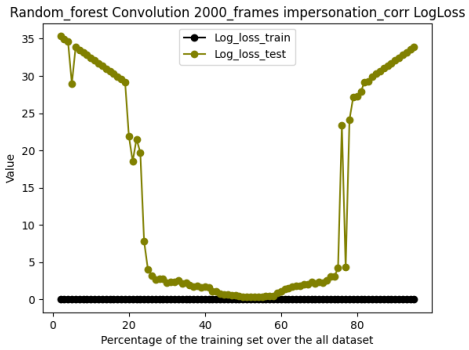

Figure B.137: Impersonation Log loss correlation reduced dataset

Reduced dataset

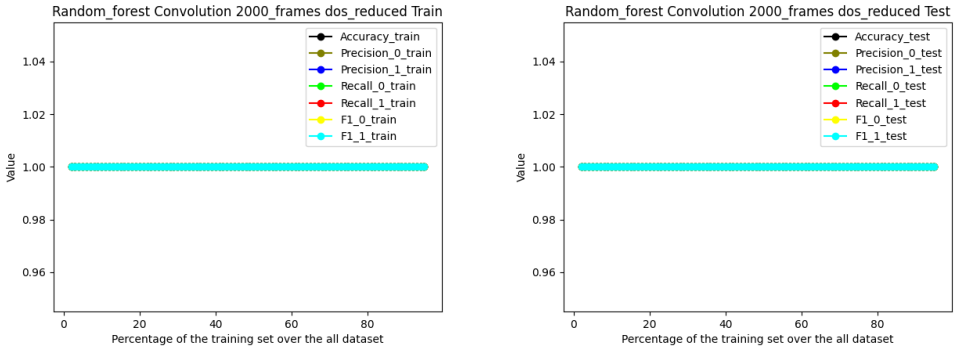

Figure B.138: DoS reduced dataset

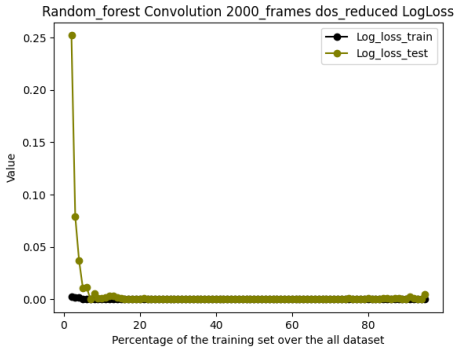

Figure B.139: DoS Log loss reduced dataset

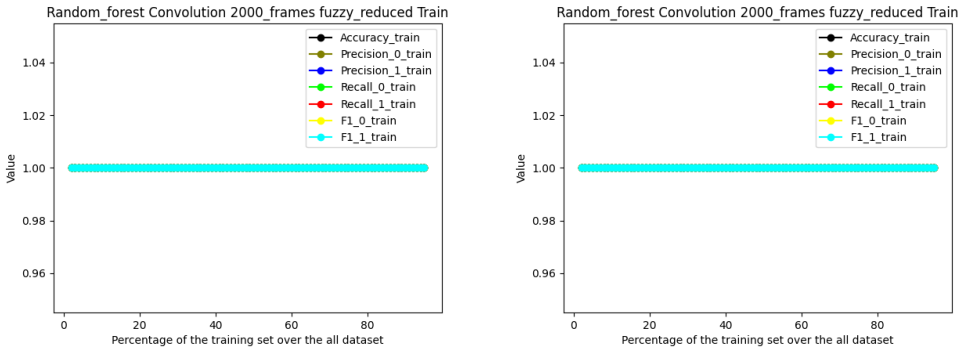

Figure B.140: Fuzzy reduced dataset

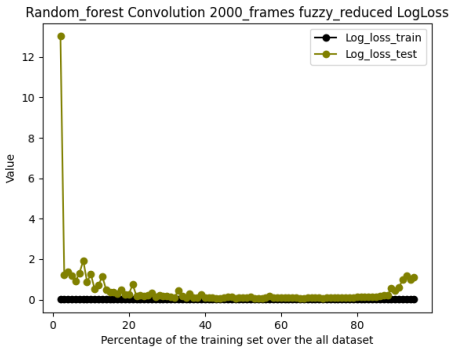

Figure B.141: Fuzzy Log loss reduced dataset

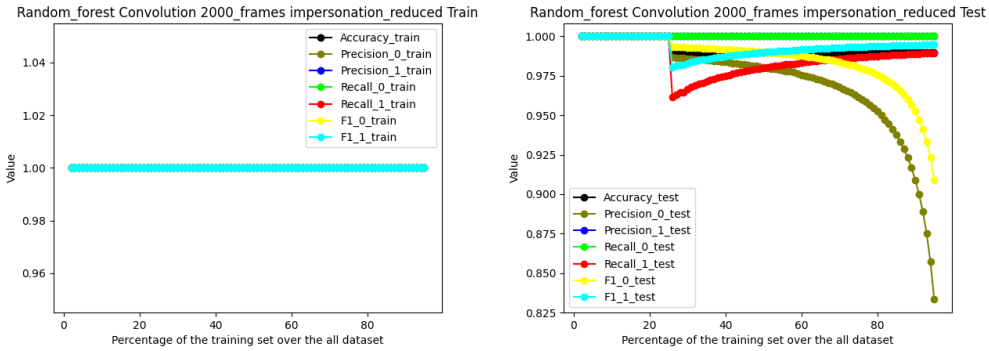

Figure B.142: Impersonation reduced dataset

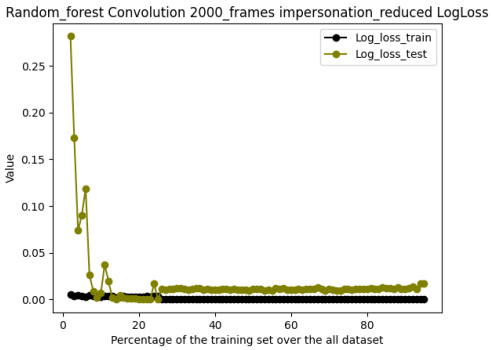

Figure B.143: Impersonation Log loss reduced dataset

B.4 Multiclass SVC

B.4.1 AES

75 frames dataset

Reduced dataset

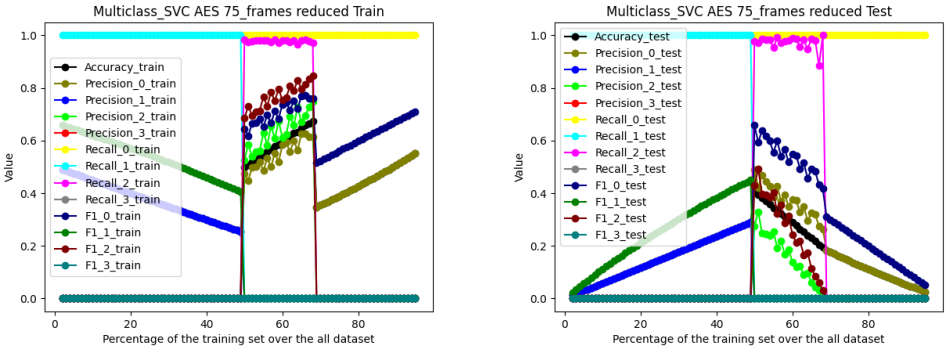

Figure B.144: Reduced dataset

1500 frames dataset

Full dataset

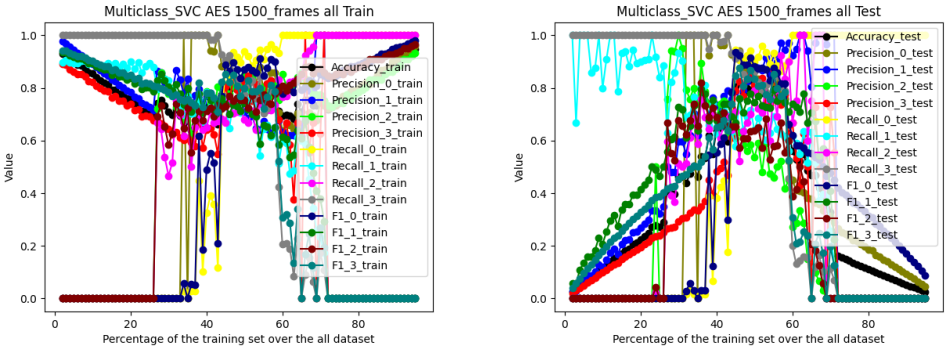

Figure B.145: Full dataset

Mean and scale dataset

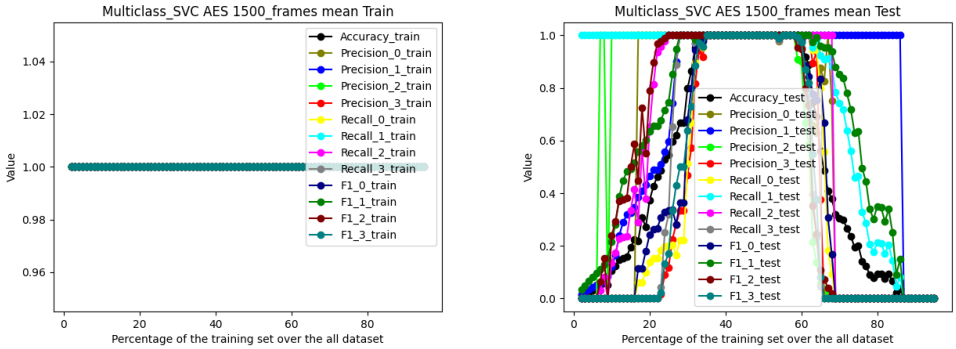

Figure B.146: Mean and scale dataset

Correlation reduced dataset

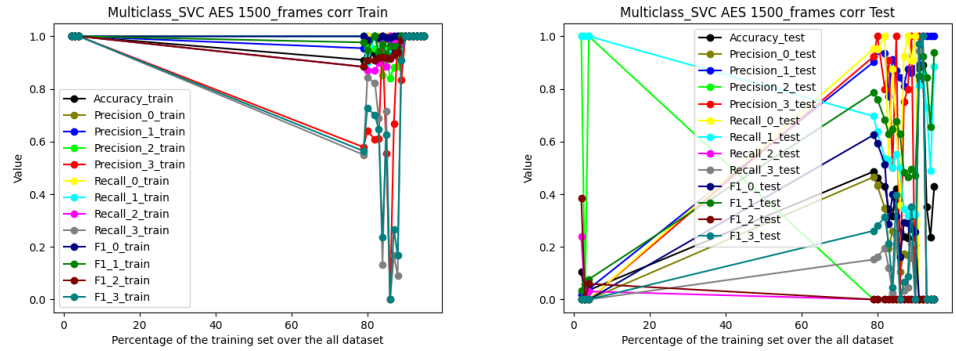

Figure B.147: Correlation reduced dataset

Reduced dataset

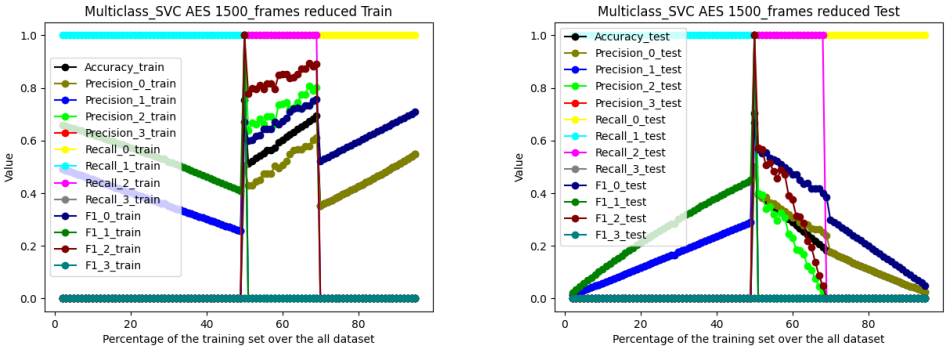

Figure B.148: Reduced dataset

B.4.2 Convolution

100 frames dataset

Reduced dataset

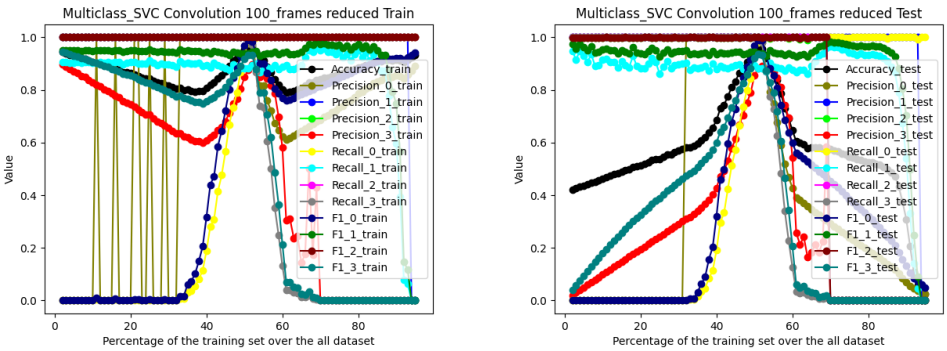

Figure B.149: Reduced dataset

2000 frames dataset

Full dataset

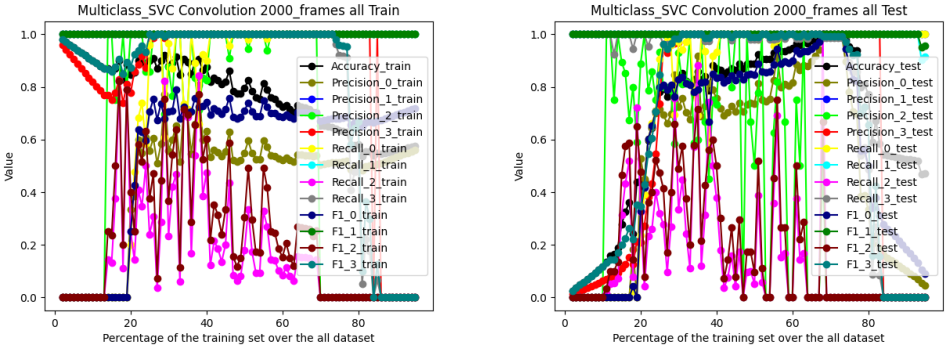

Figure B.150: Full dataset

Mean and scale dataset

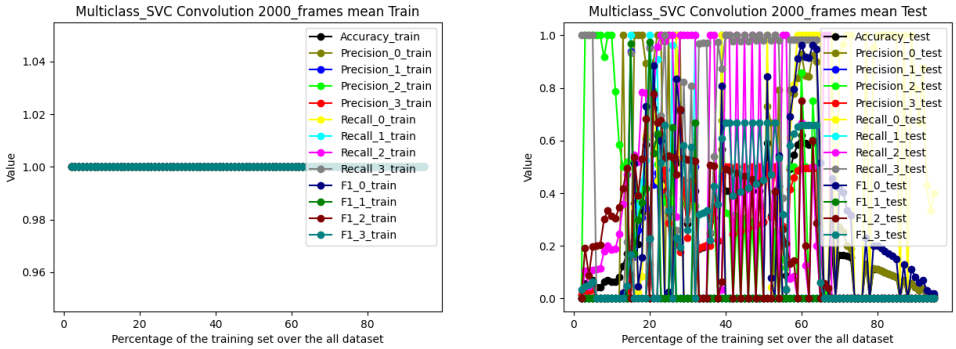

Figure B.151: Mean and scale dataset

Correlation reduced dataset

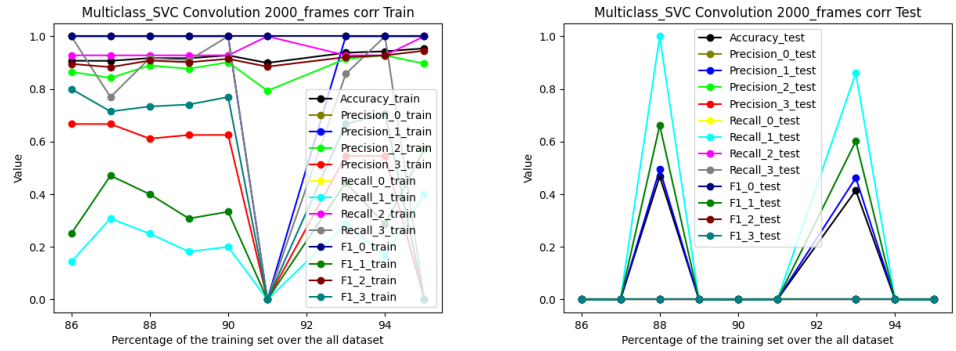

Figure B.152: Correlation reduced dataset

Reduced dataset

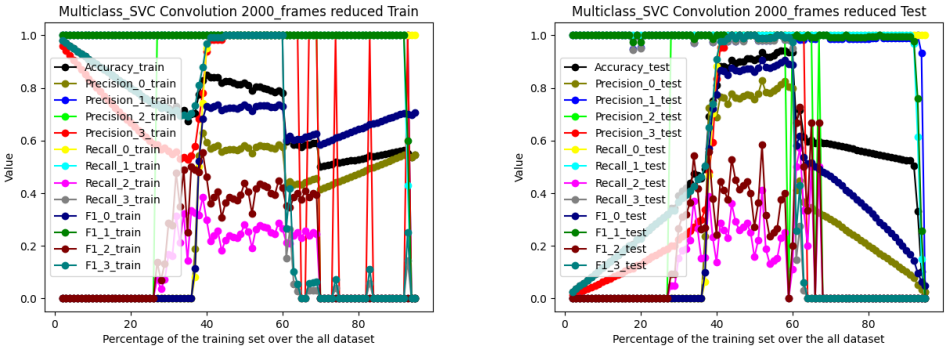

Figure B.153: Reduced dataset

B.5 Multiclass Random forest

B.5.1 AES

75 frames dataset

Reduced dataset

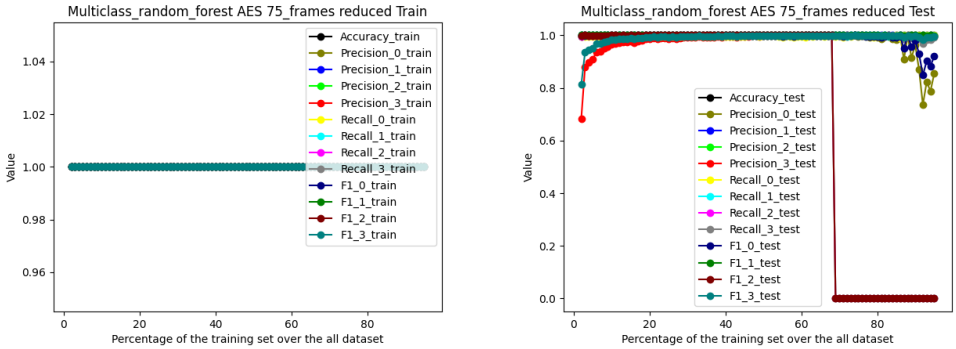

Figure B.154: Reduced dataset

1500 frames dataset

Full dataset

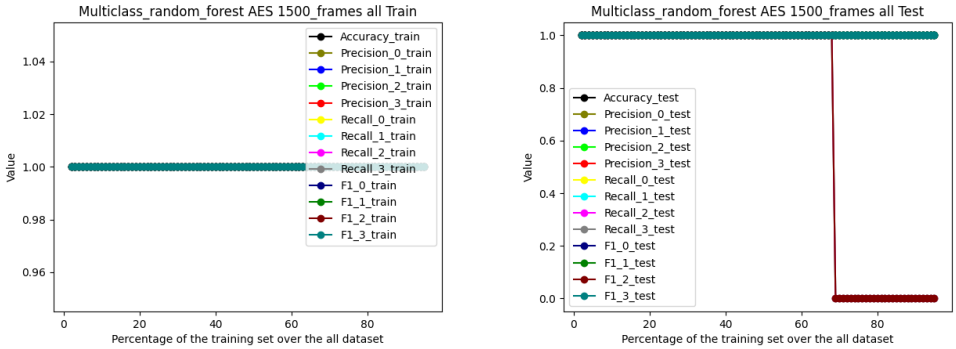

Figure B.155: Full dataset

Mean and scale dataset

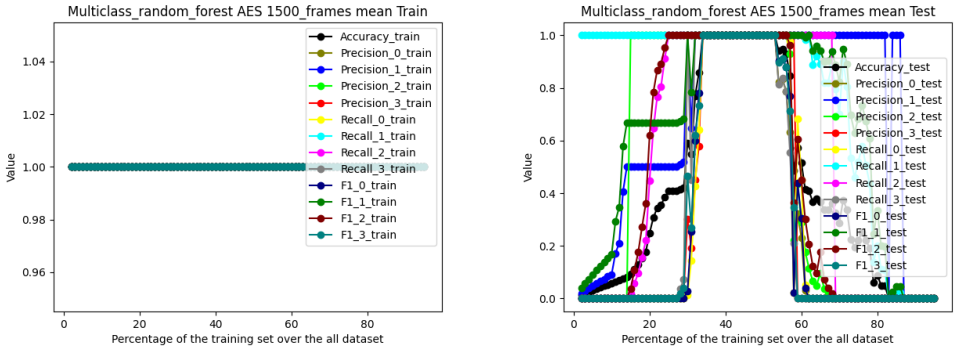

Figure B.156: Mean and scale dataset

Correlation reduced dataset

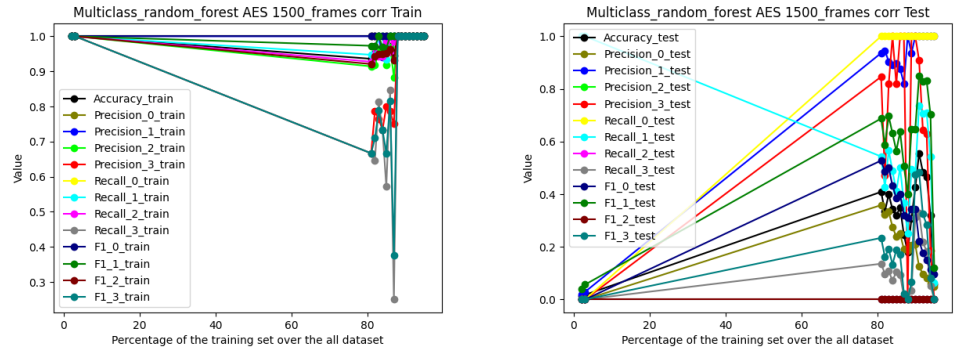

Figure B.157: Correlation reduced dataset

Reduced dataset

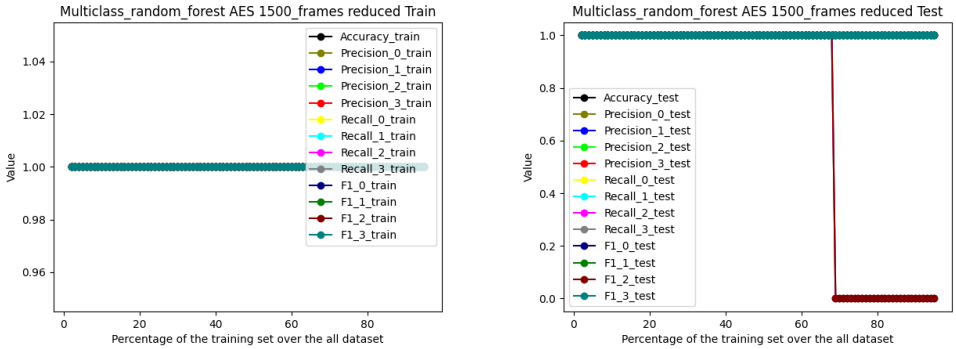

Figure B.158: Reduced dataset

B.5.2 Convolution

100 frames dataset

Reduced dataset

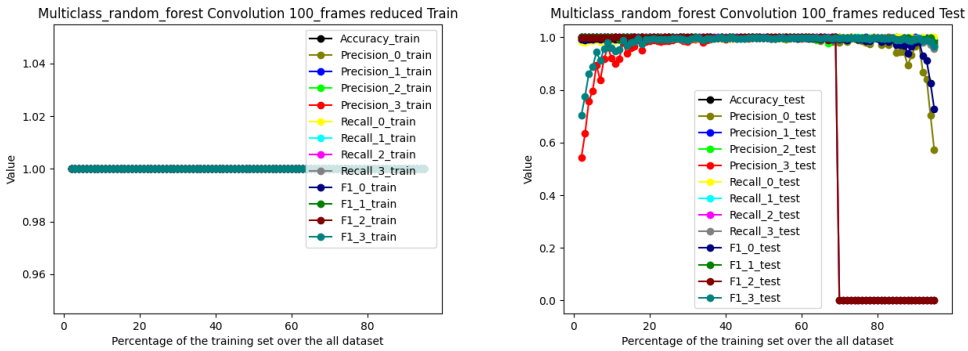

Figure B.159: Reduced dataset

2000 frames dataset

Full dataset

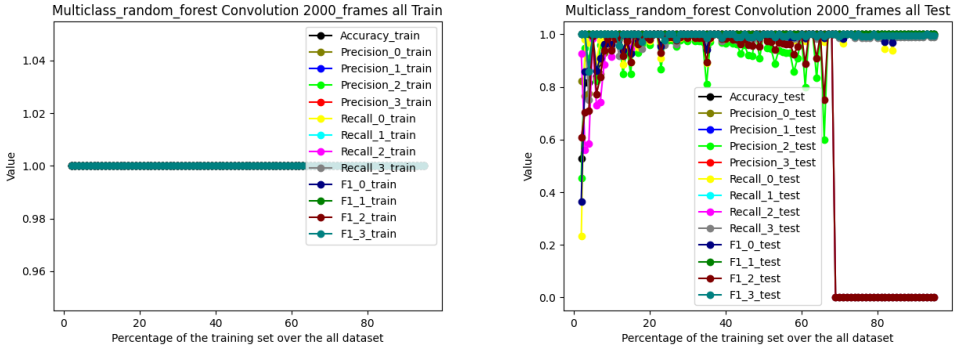

Figure B.160: Full dataset

Mean and scale dataset

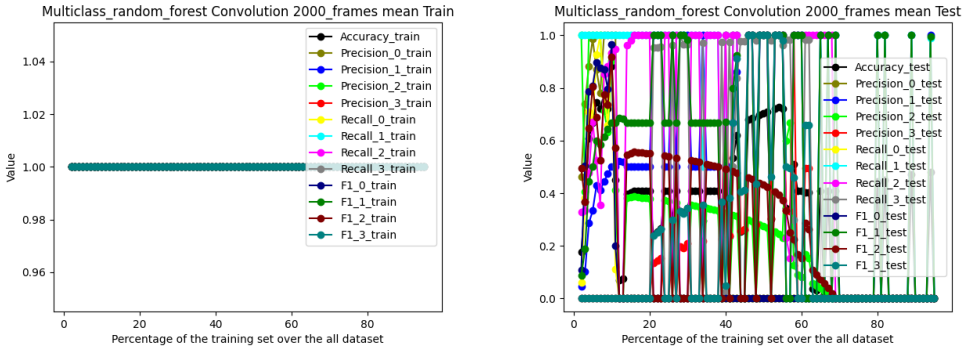

Figure B.161: Mean and scale dataset

Correlation reduced dataset

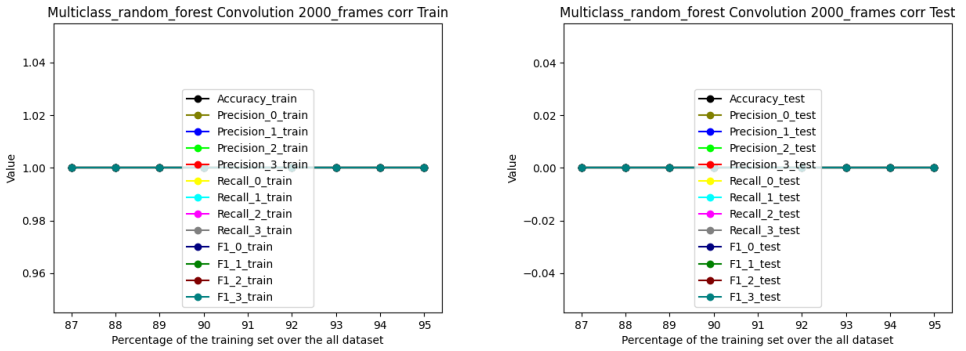

Figure B.162: Correlation reduced dataset

Reduced dataset

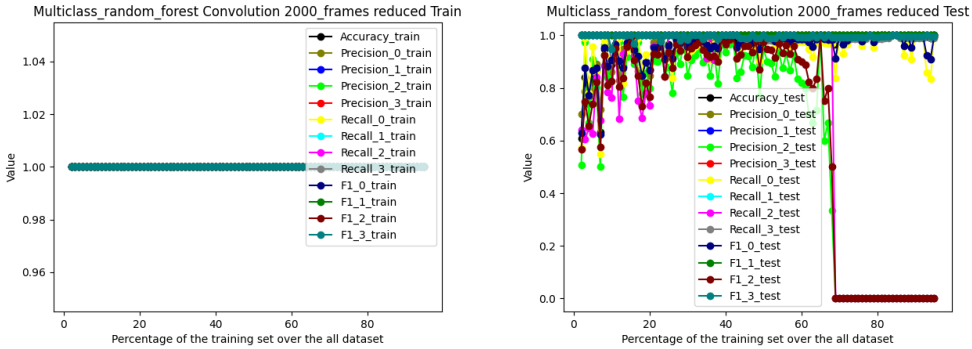

Figure B.163: Reduced dataset

Bibliography

[1] Robert Bosch GmbH (1991). *CAN Specification, Version 2.0*, Postfach 50, D-7000 Stuttgart 1.

[2] Sanjeev Das, Jan Werner (2019). *SoK: The Challenges, Pitfalls, and Perils of Using Hardware Performance Counters for Security*, University of North Carolina at Chapel Hill, Georgia Institute of Technology, Stony Brook University

[3] RISC-V SBI specification (2024). *Github RISV-V SBI Specification*, URL: `https://github.com/riscv-non-isa/riscv-sbi-doc`, RISC-V International.

[4] RISC-V OpenSBI (2024). *RISC-V Open Source Supervisor Binary Interface (OpenSBI)*, URL: `https://github.com/riscv-software-src/opensbi`, RISC-V International.

[5] Scikit learn (2024). *Scikit learn documentation*, URL: `https://scikit-learn.org/stable/`

[6] FreeRTOS (2024). *FreeRTOS documentation*, URL: `https://www.freertos.org/Documentation/RTOS_book.html`

[7] Kendryte (2019). *Kendryte doc datasheet K210*, URL: `https://github.com/kendryte/kendryte-doc-datasheet/tree/master`

[8] Milk-V (2024). *Milk-V Duo documentation*, URL: `https://milkv.io/docs/duo/overview`

[9] Andrew Waterman, Krste Asanovi (2019). *The RISC-V Instruction Set Manual, Volume I: Unprivileged ISA*, RISC-V International

[10] Andrew Waterman, Krste Asanovi, John Hauser (2021). *The RISC-V Instruction Set Manual, Volume II: Privileged Architecture*, RISC-V International

[11] Hyunsung Lee, Seong Hoon Jeong and Huy Kang Kim (2017). *OTIDS: A Novel Intrusion Detection System for In-vehicle Network by using Remote Frame*, PST (Privacy, Security and Trust)

[12] Daemen, Joan, Rijmen, Vincent 2003. *AES Proposal: Rijndael*. National Institute of Standards and Technology

[13] Jason Lowe-Power (2024). *Gem5 documentation*, URL: `https://www.gem5.org/documentation/`. Gem5

[14] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna,

Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood (2011). *The gem5 simulator*. ACM SIGARCH Computer Architecture News

[15] SAE International (2018), *Serial Control and Communications Heavy Duty Vehicle Network - Top Level Document*. Truck Bus Control and Communications Network Committee

[16] Rafi Ud Daula Refat, Abdulrahman Abu Elkhail, Azeem Hafeez, and Hafiz Malik (2022), *Detecting CAN Bus Intrusion by Applying Machine Learning Method to Graph Based Features*. University of Michigan

[17] Raisa Abedin Disha and Sajjad Waheed (2022), *Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique*. Department of Information and Communication Technology, Bangladesh University of Professional