Scaling Up secure Processing, Anonymization and generation of Health Data for EU cross border collaborative research and Innovation

D3.1 — Interim report on Scalable Secure Multiparty Computation, Federated Learning and Unbiased AI techniques and tools

Funded by
the European Union

Grant Agreement Nr. 10109571

Project Information

Project Title	Scaling Up Secure Processing, Anonymization and Generation of Health Data for EU Cross Border Collaborative Research and Innovation		
Project Acronym	SECURED	**Project No.**	10109571
Start Date	01 January 2023	**Project Duration**	36 months
Project Website	https://secured-project.eu/		

Project Partners

Num.	Partner Name	Short Name	Country
1 (C)	Universiteit van Amsterdam	UvA	NL
2	Erasmus Universitair Medisch Centrum Rotterdam	EMC	NL
3	Budapesti Muszaki Es Gazdasagtudomanyi Egyetem	BME	HU
4	ATOS Spain SA	ATOS	ES
5	NXP Semiconductors Belgium NV	NXP	BE
6	THALES SIX GTS France SAS	THALES	FR
7	Barcelona Supercomputing Center Centro Nacional De Supercomputacion	BSC CNS	ES
8	Fundacion Para La Investigacion Biomedica Hospital Infantil Universitario Nino Jesus	HNJ	ES
9	Katholieke Universiteit Leuven	KUL	BE
10	Erevnitiko Panepistimiako Institouto Systimaton Epikoinonion Kai Ypolgiston-emp	ICCS	EL
11	Athina-Erevnitiko Kentro Kainotomias Stis Technologies Tis Pliroforias, Ton Epikoinonion Kai Tis Gnosis	ISI	EL
12	University College Cork - National University of Ireland, Cork	UCC	IE
13	Università Degli Studi di Sassari	UNISS	IT
14	Semmelweis Egyetem	SEM	HU
15	Fundacio Institut De Recerca Contra La Leucemia Josep Carreras	JCLRI	ES
16	Catalink Limited	CTL	CY
17	Circular Economy Foundation	CEF	BE

Project Coordinator: Francesco Regazzoni - University of Amsterdam - Amsterdam, The Netherlands

Copyright

Disclaimer

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the Health and Digital Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

Deliverable Information

Workpackage	WP3
Workpackage Leader	Semmelweis University
Deliverable No.	D3.1
Deliverable Title	Interim report on Scalable Secure Multiparty Computation, Federated Learning and Unbiased AI techniques and tools
Lead Beneficiary	SEM
Type of Deliverable	Report
Dissemination Level	Public
Due Date	29/02/2024

Document Information

Delivery Date	31/03/2024
No. pages	62
Version | Status	1 | Final
Deliverable Leader	Albert Aszalos (SEM)
Internal Reviewer #1	Apostolos Fournaris (ISI)
Internal Reviewer #2	Christos Strydis (EMC), Lennart Landsmeer (EMC)

Quality Control

Approved by Internal Reviewer #1	12/03/2024
Approved by Internal Reviewer #2	23/03/2024
Approved by Workpackage Leader	25/03/2024
Approved by Quality Manager	27/03/2024
Approved by Project Coordinator	27/03/2024

List of Authors

Name(s)	Partner
Francesco Regazzoni, Georgios Tasopoulos, Kyrian Maat, Marco Brohet	UvA
Paolo Palmieri	UCC
Albert Aszalos, Péter Pollner	SEM
Gergely Ács, Balázs Pejó	BME
Joppe Bos, Gareth T. Davies, SeoJeong Moon	NXP
Vincent Thouvenot, Stephane Lorin	THALES
Konstantina Karagianni, Alexander El-Kady, Vassilis Paliouras, Apostolos Fournaris	ISI
Christos Strydis	EMC

The list of authors reflects the major contributors to the activity described in the document. The list of authors does not imply any claim of ownership on the Intellectual Properties described in this document. The authors and the publishers make no expressed or implied warranty of any kind and assume no responsibilities for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information contained in this document.

Revision History

Date	Ver.	Author(s)	Summary of main changes
22.11.2023	0.1	Péter Pollner (SEM), Francesco Regazzoni (UvA)	Created the document and the initial version of its content
30.11.2023	0.2	Albert Aszalos (SEM), Paolo Palmieri (UCC)	Rewritten according to the partners' suggestions
13.12.2023	0.3	Albert Aszalos (SEM)	Rewritten according to the partners' suggestions
30.12.2023	0.4	all partners in D3.1	Detailed content in sections
10.01.2023	0.5	Albert Aszalos (SEM)	Rewritten according to the partners' suggestions
25.01.2024	0.6	Albert Aszalos (SEM)	Rewritten according to the partners' suggestions during the personal meetings in Athens
15.02.2024	0.7	all partners in D3.1	Detailed content in sections
29.02.2024	0.8	Vincent Thouvenot	Corrections requested by the coordinator
24.03.2024	0.9	Apostolos Fournaris (ISI), Christos Strydis (EMC)	Notes and suggestions as internal reviewers
27.03.2024	1.0	Francesco Regazzoni (UvA)	Final version

Table of Contents

Acronyms and Abbreviations

AI Artificial Intelligence. 11, 43

API Application Programming Interface. 33

ASIC Application-Specific Integrated Circuit. 17

BFV Brakerski-Fan-Vercauteren Homomorphic Encryption Scheme. 15, 16, 18

BGV Brakerski-Gentry-Vaikuntanathan Homomorphic Encryption Scheme. 15, 18

CKKS Cheon-Kim-Kim-Song Homomorphic Encryption Scheme. 15, 16, 18

CTG Cardiotocograph. 53

D Deliverable. 9

DL Deep Learning. 32, 33, 43

DNN Dense Neural Network. 24–26

ECG Electrocardiogram. 23, 53

EHR Electronic Health Record. 41, 53

FedAdagrad Federated Adaptative Gradient. 33

FedAdam Federated Adaptive Moment Estimation. 33

FedAvg Federated Averaging. 33

FedOpt Federated Optimisation. 33

FedProx Federated Learning with Proximal Aggregation. 33

FedYogi Federated Adaptive Optimization Algorithm. 33

FHE Fully Homomorphic Encryption. 15, 18, 26

FL Federated Learning. 9–13, 31–42, 54

FPGA Field Programmable Gate Array. 17

GDPR General Data Protection Regulation. 39

GPU Graphical Processing Unit. 17

HE Homomorphic Encryption. 9–16, 19, 20, 30, 32, 54

ISE Instruction Set Extensions. 17

KPI Key Performance Indicator. 6, 16–19, 30

ML Machine Learning. 12–16, 18, 30–37, 43, 45, 48, 54

MLP Multi-Layer Perceptron. 23, 24, 26

MR Magnetic Resonance. 37

ONNX Open Neural Network Exchange. 14, 15

PATE Private Aggregation of Teacher Ensembles. 38, 40

PETs Privacy-Enhancing Technologies. 9, 12

PTQ Post-Training Quantization. 13–15, 26, 54

QAT Quantization-Aware Training. 13, 15, 54

SDK Software Development Kit. 32

SIMD Single Instruction Multiple Data. 16, 17

SMPC Secure Multi-Party Computation. 9–16, 30, 40, 54

T Task. 43

TCN Temporal Convolution Network. 34

TFHE Fully Homomorphic Encryption over the Torus. 13–15

TLS Transport Layer Security. 32

UB Unbiased Artificial Intelligence. 9–11, 53, 54

UC Use Case. 6, 10, 11, 18, 19, 23, 34, 37–41, 43–45, 52, 53

WP Work Package. 54

1 Executive Summary

The goal of SECURED is aimed at realizing a hub that provides tools and services for Anonymization, Deanonymization, Secure Multiparty computation and Unbiasing of Health data that can be utilized by a broad range of users to build privacy preserving health application in a collaborative manner. Central to this initiative is the development of a library comprising tools designed to facilitate secure processing of data coming from several sources.

In essence, SECURED is committed to fostering innovation through two interconnected workflows. The first workflow, termed Data Flow, deals with anonymization of data, making possible the secure sharing and accessibility of data through anonymization and by generating synthetic versions of datasets. Simultaneously, the Processing Flow workflow focuses on securely processing and analyzing data within the healthcare ecosystem. This report provides a summary of the initial progress made in the research and development activities related to the latter Processing Flow. This work was preceded by Deliverable 4.1, which involved a comprehensive review of the current state of the art regarding available tools and concepts, and that served as starting point for the activities described here.

WP3 focuses on four primary research areas for data processing: Federated Learning (FL), Unbiased Artificial Intelligence (UB), Secure Multi-Party Computation (SMPC) and Homomorphic Encryption (HE). This deliverable presents:

- A brief introduction of the WP3 work package and of the relation between the technologies developed in this work package and the SECURED Use Cases.
- The description of the initial development of the Privacy-Enhancing Technologies (PETs) used in the SECURED project, that lead to the following initial achievements:
  - SMPC/HE: A preliminary analysis of available SMPC and HE libraries has been performed, considering computational tasks relevant for the SECURED use cases, such as encrypted inference in neural networks. Furthermore, selected HE libraries have been assessed using specific KPIs and benchmarked with relevant micro-benchmarks.
  - FL: A thorough Risk Analysis evaluated how to develop federated learning solutions that are both versatile and lightweight, scalable to the project use cases. The first selection of available libraries was done accordingly.
  - UB: First analysis had been made about Fairness for generative AI, which emerged as a highly relevant property for SECURED during the development of the project and was thus not covered in Deliverable D4.1.
- The conclusions that we draw from the current analysis and the planned immediate next steps.

The activities within this work package will culminate into the processing flow part of the SECURED library. In parallel, WP2 is developing the data flow components of the SECURED library. Both libraries will be integrated into the SECURED Innohub. This deliverable depicts the status of WP3 at M14 of the project, and reports current and future directions of the work package activities.

Related Documents

- SECURED Deliverable D1.2 - GDPR and Ethics Project Guidelines
- SECURED Deliverable D1.6 - Data Management Plan
- SECURED Deliverable D2.1 Interim report on data anonymization, deanonymization and synthetic data generation techniques, tools and services
- SECURED Deliverable D4.1 — State of the Art and initial technical requirements

2 Introduction

2.1 The role of WP3 tasks in the SECURED project

In the context of the SECURED project, WP3 focuses on the processing flow, and in particular on scaling up secure and private processing technologies. These include Secure Multi-Party Computation and Homomorphic Encryption, secure Federated Learning, and Unbiased Artificial Intelligence techniques. The outputs of WP3 will converge into the Innohub in WP4 and will be validated through the use cases in WP5.

Figure 1 depicts the interconnections between the components contributing to the work package. Research and development activities cover various methods for processing distributed data and the secure and unbiased training of AI models. The project arranges these activities into three main categories as follows.

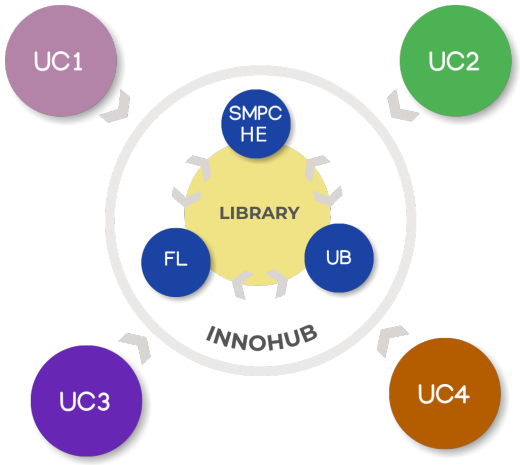


Figure 1 – The overall structure of information flow in WP3: three technical tasks build up the library in cooperation. The development will converge in the SECURED Innohub, which includes also components developed in other work packages. The SECURED Use Cases will "loosely" drive the selection of components and the optimization criteria.

1. Research and development of SMPC and HE schemes and associated primitives: This includes optimizing and accelerating SMPC and HE schemes for large-scale computation, as well as their application to specific health data computation scenarios.
2. Design of privacy-preserving and robust Federated Learning algorithms: This includes threat analysis, mitigation of privacy leakage, detection and elimination of misbehaving clients, and investigations on benchmark attacks with the secure federated learning framework.
3. Development of methods and tools to detect and mitigate biases in AI models: This includes establishing a baseline model for evaluating data bias, assessing bias on classical criteria like disparate impact and differential fairness, and developing mitigation strategies using preprocessing, in-processing, post-processing, or synthetic data.

The output of this research will converge into a library that implements the schemes, the optimizations, and the techniques developed. As part of the validation of the results, the techniques and the library will be used in the scenarios identified by the use cases (including the ones selected through the open call). The Use Cases will provide useful insight in the development of the technologies, but our ambition is to develop general solutions that can be applied in a wider number of applications. As such, the Use Cases do not impose strict requirements on the activities, but rather establish a baseline and offer metrics and feedback on the implementations. Ultimately, the project's end-product, the Innohub, will integrate the final results.

UC 1 provides an example of a scenario where performance requirements for the algorithms, primarily time complexity, are crucial to the successful processing. To ensure the feasibility of near "real-time" ultrasound-guided neurosurgery in practical scenarios, tools provided by SECURED must aim at producing results compatible with such scenario.

UC 2 stresses privacy in distributed data collection scenarios and sets a baseline for the robustness of AI models. The telemonitoring application for children deals with sensitive data from a vulnerable population, where technical issues can lead to unsuitable data for training and improving AI models. Filtering out biases and ensuring fairness, especially for minorities or rare diseases, are key aspects of the processing methods.

UC 3 addresses the issues of preventing the leakage of training data and protecting the characteristics of generative AI models. In the context of synthetic data generation for education, it is crucial that the generated data does not disclose enough information to identify any of the original data providers. Additionally, the model's features should not enable reverse-engineering to maintain fairness in examinations. The generation of rare yet significant cases, as opposed to commonplace and frequent examples, presents a challenge for bias-processing methods.

UC 4 outlines measures for securely processing large-volume datasets. When accessing genomic data for research and healthcare, maintaining the highest level of privacy is essential. This confidentiality is critical not only for the data provider, but also for all their biological relatives who may be affected.

2.2 Structure of the document

The remaining part of this deliverable presents results in the following sections:

section 3 summarizes the progress made in Secure Multi-Party Computation (SMPC) and Homomorphic Encryption (HE) methods. It provides an overview of the selection criteria for software libraries to be utilized in the SECURED project and indicates some of the initial challenges related to privacy-enhancing computation that will be addressed in the upcoming phases. Further, it delves into the analysis of selected Homomorphic Encryption (HE), by means of performance indicators defined for the purpose and by means of micro-benchmarks. Results of the profiling are provided in comprehensive tables.

section 4 offers a comprehensive rationale for the choice of solutions in the Federated Learning (FL) implementation. This section begins with the selection of appropriate types, followed by a detailed description of two software libraries. The selection criteria are presented alongside a thorough risk analysis, both in general terms and for each use case.

section 5 introduces early-stage fundamental research on Unbiased Artificial Intelligence (UB) and fairness, exploring relevant mathematical concepts. The connection with the project is illustrated through a questionnaire concerning the use cases.

section 6 concludes the report.

3 Secure Multi-Party Computation and Homomorphic Encryption

In this section, we concentrate on the Privacy-Enhancing Technologies (PETs) that will be used in the SECURED, namely Secure Multi-Party Computation (SMPC) and Homomorphic Encryption (HE). A more thorough exposition can be found in SECURED deliverable D4.1, or in topic-specific textbooks [1, 2]. We report here the basic concepts of these technologies for completeness and discusses them with particular focus on encrypted inference. Then, we focus on libraries implementing HE, and we compare them using several KPIs definied for the purpose, and experimentally with a relevant micro-benchmarks.

3.1 Technology Overview

SMPC allows a group of parties to jointly perform some computation without revealing any of the parties' inputs, while HE allows an evaluator to compute some function on encrypted data. Both techniques can be trivially realized with a trusted third party that handles all data, and the purpose of Secure Multi-Party Computation (SMPC)/Homomorphic Encryption (HE) tools is to emulate this trusted party (among the communicating parties) by means of cryptography.

A simple example task for SMPC is privacy-preserving Federated Learning (FL) (see Section 4 for a more thorough overview of FL): in each round, a group of parties collectively train a Machine Learning model by providing parameters, in a way that does not leak these parameters to the other training parties or the aggregator (the eventual holder of the model). An example for HE is privacy-preserving Machine Learning inference: a client wants to make a query to a proprietary Machine Learning model without revealing the input query or the output result, while the server that holds the model wants to keep their intellectual property private. The client encrypts the input and sends it to the server, which then homomorphically evaluates the model and sends the result back to the client, who can decrypt to get the correct output. More details on how to apply HE to ML inference can be found in Section 3.2.2.

A wide range of candidate schemes[1] exist for both SMPC and HE that collectively provide a wide variety of functionalities, security properties and performance tradeoffs. Very generally speaking, SMPC schemes will often have simple computational operations in a large number of communication rounds, while HE schemes will be much more computationally complex but operate in one round. For some tasks, such as inference of neural networks with non-linear activation functions [3, 4, 5], combining both SMPC and HE can provide considerable performance gains. So choosing one and then choosing a single software library is not as simple in general.

3.2 Investigations

3.2.1 Scaling up SMPC and HE

The main goal of this component of the SECURED project is to scale up the performance of Secure Multi-Party Computation (SMPC) (and Homomorphic Encryption (HE)) tools, and this can be realized via multiple concurrent efforts.

Primitives that underpin modern HE libraries require huge overheads (relative to plaintext computation) for memory, computation time and communication bandwidth. The first workstream of the scaling-up task is to investigate the potential of hardware acceleration, meaning purpose-built hardware for the computational tasks present in existing HE schemes and libraries. This workstream will also attempt to identify where hardware acceleration can be used to provide more performant SMPC, however this task has lower priority than the

[1] A note on nomenclature: constructions for SMPC are usually regarded as interactive protocols between a group of parties that contain a number of primitives as sub-routines, while HE can be cast as a primitive to be used in an interactive protocol. As a result, we use *schemes* to refer to the eventual instantiations for either SMPC or HE (or a combination of the two), meaning the technical subcomponents plus the necessary communication logic.

component looking at HE.

The second workstream is to target algorithmic optimization for SMPC and HE for the processing tasks that are relevant to the SECURED project.

Thirdly, the project aims to provide more performant privacy-preserving federated learning, and in particular the task of secure aggregation. More details on Federated Learning (FL) and its role in the SECURED project can be found in Section 4, and much of the discussion is deferred to that part of this document. It is important to note that there are many avenues for improving secure aggregation (and other related sub-tasks of FL) in scenarios that require various different security properties, and the tasks related to SMPC and FL are therefore closely intertwined.

3.2.2 Encrypted ML Inference

This section briefly describes the steps involved in performing ML inference using HE, meaning that the party providing input wishes to keep that input secret from the holder of the ML model. Note that this section is geared towards ML but the lessons learned and techniques can also be applied to other scenarios requiring homomorphic evaluation.

As a reminder, a homomorphic encryption scheme allows party A to encrypt some message m to get a ciphertext C, send this ciphertext to party B who applies a homomorphic evaluation operation that takes as input C and a function f and outputs a ciphertext C' that party A can decrypt to $f(m)$.

The central challenge of encrypted ML inference comes from the fact that HE schemes struggle to (homomorphically) evaluate certain non-linear functions on ciphertexts. If the larger function contains some of these 'difficult' functions under the hood, then HE computation requires extra measures to prevent lower accuracy of the function. For instance, convolutional neural networks (CNN) compose of certain activation layers with non-linear functions. Various methods have been proposed to approximate such functions while retaining accuracy. More information about the 'HE-friendliness' of various neural network operations can be found in a study by Obla et al. [6]. The workflow therefore becomes:

1. Train (or receive) a ML model, i.e., fix its structure and parameters/weights and test the model's accuracy;
2. Identify the "difficult" functions in the model, e.g., Rectified Linear Units(ReLU) activation function;
3. Replace these functions with HE friendly alternatives or with polynomial approximations of the original functions, e.g., replace ReLU with Swish or approximated ReLU;
4. Test the new model's accuracy and accept if it is sufficiently close to that of the original model, else adjust accuracy of approximations.

This process needs to work in tandem with the constraint of HE schemes, for example TFHE [7] as used in Concrete-ML is restricted to 16-bit integers, meaning that **quantization** is required. This process inherently affects the accuracy of a model, however the bit-width of the entire system then becomes a tunable parameter that provides a trade-off between accuracy and inference speed in the encrypted domain. There are two approaches for performing quantization:

- Post-Training Quantization (PTQ) simply adjusts all model weights to be within a certain bit-width;
- Quantization-Aware Training (QAT) performs the training process in a way that incorporates quantization throughout.

Quantization in the context of HE has been described in many works, including [8] (where the eventual models were referred to as *discretized neural networks*), and [9].

A goal of the next stage of the project is to further investigate the accuracy loss in both PTQ and QAT for the neural networks that will be of interest to the SECURED use cases (and later, the open call), in addition

to investigating the possibility of more efficient workflows when applying privacy-preserving techniques to ML inference. Preliminary results demonstrating PTQ accuracy and execution time with encrypted inference can be found in Section 3.4.2.

3.2.3 Libraries for SMPC and HE

Schemes for SMPC and HE are normally very general, meaning that they can support computing *arbitrary functions* on encrypted data. In practice, the function being computed has a major impact on the viability of a given scheme, and therefore it is useful to categorize schemes based on their usefulness for particular processing tasks. In order to narrow the focus, a first step is to assess which schemes are of potential interest for the computation tasks being considered in the SECURED project, and this step then feeds into the task of choosing suitable libraries.

For the task of hardware acceleration, it is often challenging to assess the suitability for incorporating acceleration techniques. The libraries for SMPC and HE in question often support multiple schemes, which in turn use multiple lower-level libraries, and acceleration can see major gains when performed on the time-consuming operations in these lower-level libraries. It is also necessary to take into account which of these operations are present in multiple of the schemes, and whether the field/ring these operations act in is consistent among the different schemes.

The following text indicates some libraries for SMPC/HE that were considered but eventually not pursued as part of the SECURED project. `Lattigo` [10] is a HE library written in the Go language, developed with distributed systems in mind. Integrating Go with the other components of the SECURED infrastructure such as dedicated hardware units would likely have resulted in many challenges[2]. `SCALE-MAMBA` [11] library is a general-purpose SMPC library, but active development stopped over two years ago. The `FANNG-MPC` framework [12] is a successor to `SCALE-MAMBA` targeted at Machine Learning applications, however, it is only very recently introduced and source code is at the time of writing unavailable. However, some forked versions are under active development [13]. The `swanky` [14] library was regarded as not being mature enough for deployment in SECURED.

At the time of writing, we have identified the `EzPC` [15, 16] and `CrypTen` [17, 18] libraries as being promising for use with SMPC in the context of Machine Learning if the need arises in later stages of the SECURED project. `EzPC` is particularly appealing for its CryptFlow component [19, 20] that converts TensorFlow or ONNX models into SMPC protocols: if any of the participants in the open call require that the ML model be kept private, then this could add value to the software solution. `CrypTen` essentially acts as a SMPC wrapper around PyTorch, meaning that regular operations of PyTorch can be performed but in a way that is by default done using an appropriate SMPC mechanism. At the time of writing, however, neither `EzPC` nor `CrypTen` appear to provide any benefits that are directly applicable to the use cases of SECURED that are not already provided by our chosen libraries (below).

After these considerations have been taken into account, the following three libraries appear to be the most promising for meeting the goals of the project for the use cases and the open call.

OpenFHE. The OpenFHE library [21] is a general-purpose HE framework supporting a large number of schemes and a large degree of configurability.

Concrete-ML. Concrete-ML [22] is a privacy-preserving Machine Learning library that enables running Machine Learning inference in the HE setting. It uses the TFHE scheme [7] for encryption operations. The closely related Concrete [23] library is built on the same TFHE instantiation, and can be regarded as the more general version of Concrete-ML.

MP-SPDZ. MP-SPDZ [24] is a general-purpose SMPC library. It has been used as a tool in many academic projects that enable privacy-preserving options for workflows similar to the ones considered in SECURED.

[2] In essence, the main reason that one would choose to use `Lattigo` is if the rest of the system uses the Go language. The library does not offer any additional HE schemes compared to other libraries, and furthermore it is maintained by a private company (TuneInsight SA) who may choose to stop open-source development at any time.

3.2.4 Interfaces of SMPC and HE Libraries

This section will describe the steps that a user needs to take in preparing to use the three libraries that are provisionally decided to be the most promising for use in the SECURED project.

OpenFHE. OpenFHE [21] supports a wide-range of FHE schemes, combining multiple design decisions made from previous FHE projects such as PALISADE [25] or HEAAN [26]. OpenFHE focuses on not only supporting multiple back-ends to facilitate hardware acceleration, but also providing a user with a configuration that removes most parameter selection and generally improves the ease of use to start with FHE. In practice, this means one can use OpenFHE for common FHE schemes, including but not limited to BGV, BFV or CKKS, by creating a `CCParams` struct in C++ and specifying the `CryptoContext` (i.e. scheme) to be used. Making the decision of what scheme to pick would be the first area of parameters for a user to think about, but once this has been selected, the library will automatically provide default configurations for many values regarding the scheme to be used. However, a user might still need to input a multiplicative depth and can additionally tweak parameters related to the required security level of 128 bits, 100 bits, 192 bits, 256 bits or non-standard security level, which will require the user to estimate the ring dimension for the desired amount of bits. All these options are to be set before the encrypted communication in the `CryptoContext`, after which the OpenFHE HE operations can be used.

Concrete and **Concrete-ML.** Concrete [23] is built on top of the TFHE scheme [7], and therefore can be seen as a compiler that transforms Python functions into a domain that is suitable for TFHE: in the library documentation [27] these resulting functions are referred to as Concrete *circuits*. The Concrete documentation lists which Python operators and which NumPy functions are natively supported, though it is not possible to define functions that have floating point inputs or outputs (intermediate floating point values may be possible, but only if the function can map them to an integer table lookup). This process requires the user to define the input set (bit width and shape) and also to gain an understanding of whether the operations make sense in the encryption domain, for example the evaluation function cannot have an `if` statement that uses an encrypted value.
Concrete-ML [22], which is built on top of Concrete, is used for privacy-preserving inference in Machine Learning models. As discussed in Section 3.2.2 there are two approaches for encrypted inference, QAT and PTQ. There are largely four areas of inputs to develop a model in a QAT approach. The first area is the training and evaluation data. The second area is the parameters for the Neural Network structure, such as number of layers and the types of activation functions. The third area of input is the quantization parameters including number of bits for weights, number of bits for activation and inputs, and maximum accumulator bit-width. The fourth area is the choice between `scikit-learn` and `PyTorch`. For applying PTQ to already-trained models, the library needs a number of bits as a quantization parameter and the model itself: this can either be a customized Concrete-ML model, a `PyTorch` model, or an ONNX model.

MP-SPDZ. Since MP-SPDZ [24] is a general-purpose SMPC library, it is required to know beforehand (i) the number of parties participating in the computation; (ii) which of the participating parties will do computations, which will provide inputs, and which will receive outputs[3]; (iii) the function being computed; (iv) which function inputs are public (if any) and the format of the public and private inputs; (v) which function outputs are public (if any) and which parties should receive which function outputs; (vi) the security model in which the computation will be conducted. It must also be decided if the system setup defined by these parameters implies that a (data-independent) pre-processing phase is required.
Once these decisions are made, the function being computed must be converted to a suitable format for MP-SPDZ. Many Machine Learning algorithms are natively supported by MP-SPDZ, including logistic and linear regression, decision trees; it is also possible to import pre-trained models directly from `PyTorch` and to use a `Keras` interface for training.

[3] Note that additional configurations exist, for example the so-called *dealer model* where one party provides correlated randomness to other parties, more details can be found in the MP-SPDZ documentation [28].

3.3 Comparison of Homomorphic Encryption Libraries

In the following, we describe the initial results in evaluating the tools and libraries that are of interest to the SECURED project. At this stage, the main focus has been on tools related to HE, both in terms of assessing which libraries are the best options for applying a given scheme and also which schemes are the best options for given tasks. We decided to focusing initially on Homomorphic Encryption (HE) (rather than SMPC) for two main reasons. Firstly, the performance of HE schemes varies hugely depending on the task being performed and the parameters of the scheme, such as the choice between BFV or CKKS. Secondly, the HE component of this work package is the one that is most promising for acceleration with dedicated hardware. As a consequence, an in depth analysis is necessary to identify as early as possible which schemes and sub-operations of those schemes are the more relevant targets for the being accelerated during the next stages of the project.

As discussed in Subsection 3.2, well-established open-source libraries for HE have already been developed by both academia and industry. Here, a more in-depth discussion is given on how the libraries that were selected. Their fitness with the SECURED project was assessed with a number Key Performance Indicators (KPIs). Subsection 3.3 describes the KPIs that have been identified as relevant for the SECURED project. Based on our assessment, we selected Concrete-ML [22] for specific ML-related applications and statistics operations and OpenFHE [21] for any other application. We thus proceed testing these libraries. Subsection 3.4 investigates the main bottlenecks in these libraries and will serve as base for the acceleration approaches.

The Fig.2 provides a structural overview of the relationships among the libraries and how they are connected through specific interfaces.

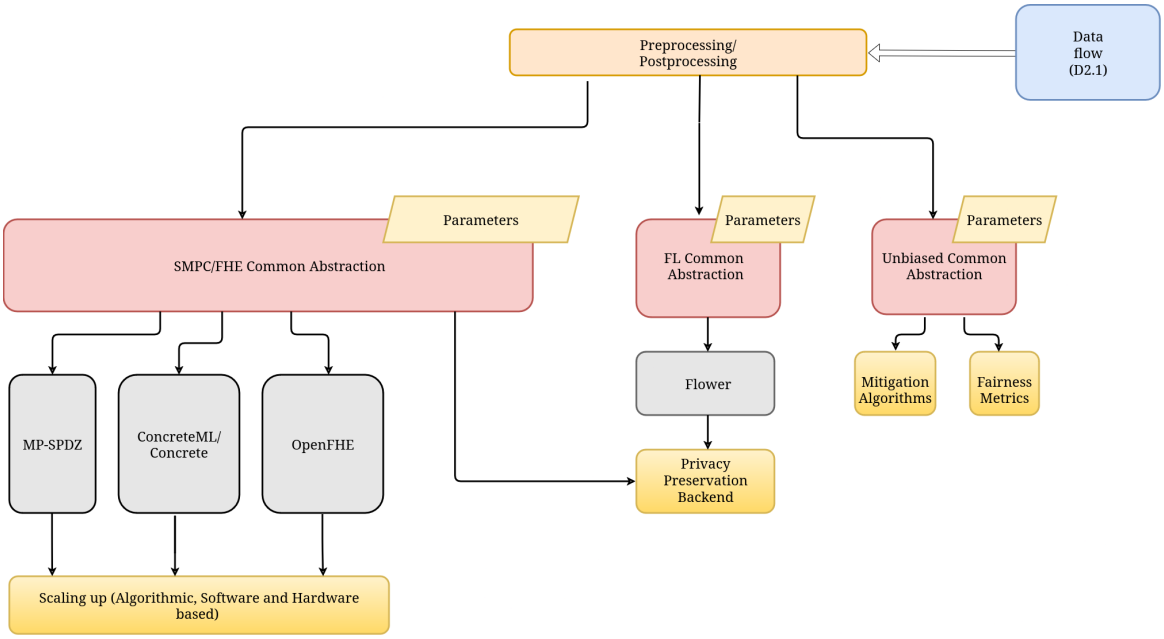


Figure 2 – High level overview of the relationship among libraries.

The HE libraries are assessed using a set of KPIs that we first define in Subsubsection 3.3.1. We provide a definition, motivate the relevance, and explain how the KPI is measured. Then, we present and discuss the results for each library in Subsubsection 3.3.2.

3.3.1 Key Performance Indicators (KPIs)

SIMD support

Purpose: Modern processors generally support performing operations on data in a *vectorized* manner, i.e. applying the same operation on multiple data. This paradigm is known as Single Instruction Multiple Data

(SIMD). SIMD instructions can offer significant speed-ups when a program is compiled with them enabled.

Definition: We define SIMD support as the out-of-the-box support, i.e. the library offers support for SIMD by itself, for compiling and running the schemes with SIMD instructions of a specific library.

How to measure: Given a specific library, SIMD support is classified in these 2 levels:

- 0 - The library does not offer any out-of-the-box support for running the schemes with SIMD instructions.
- 1 - The library does offer out-of-the-box support for running the schemes with SIMD instructions or a connection to SIMD backend (e.g. Intel HEXL).

Parallelization support

Purpose: As modern processors are built with multiple hardware threads, it is crucial for performance to optimally utilize them in parallel. Preference will be given to libraries that do not assume single-threaded execution.

Definition: We define parallelization support as the out-of-the-box support for compiling and running the schemes with multiple threads of a specific library.

How to measure: Given a specific library, parallelization support is classified in these 2 levels:

- 0 - The library does not offer any out-of-the-box support for running the schemes with multiple threads. It provides only single-threaded execution.
- 1 - The library does offer out-of-the-box support for running the schemes with multiple threads. It provides support for a multi-threading library (e.g. OpenMP).

GPU acceleration support

Purpose: As some libraries that implement some schemes are written in a way to enable parallel execution, some libraries offer the ability to use GPU to execute these parallel-friendly tasks. Indeed, some schemes may benefit from the vast number of cores in a GPU and report a speed-up on the execution time.

Definition: We define GPU acceleration support as the out-of-the-box support for the use of GPUs for the underlying scheme computations of a specific library.

How to measure: Given a specific library, support for acceleration using GPUs is classified in these 2 levels:

- 0 - The library does not offer any out-of-the-box support for a backend that is using GPUs for accelerating the scheme computations.
- 1 - The library does offer out-of-the-box support for a backend that can make use of GPUs for accelerating the scheme computations.

Dedicated Hardware Acceleration Support

Purpose: Hardware acceleration can lead to large performance increases in specialized environments. If certain devices, for example GPUs, or techniques, for example parallelization, are not available, hardware acceleration can offer another option for speed-up. Therefore, we see it as an upside to have the flexibility of designing for hardware acceleration in different domains such as intrinsic, Instruction Set Extensionss (ISEs) or outright support for specific devices such as FPGA, ASIC, etc. This KPI encapsulates the upside of hardware acceleration support.

Definition: We define Dedicated Hardware Acceleration Support as the support for the use of dedicated hardware of a specific library.

How to measure: Given a specific library, Dedicated Hardware Acceleration Support is classified in these 2 levels:

- 0 - The library does not offer any out-of-the-box support for acceleration using Dedicated Hardware.
- 1 - The library does offer out-of-the-box support for acceleration using Dedicated Hardware by providing the ability to use a custom backend for speeding-up the scheme operations.

Compatibility with time series data processing

Purpose: For specifically UC 2 mentioned in D4.1 section 7.3.2.1, there is a need for FHE of time-series data. To assess the viability of the FHE library for this use-case, we should consider whether the library implements schemes that can be used in conjunction with time series data. If a library does not support any schemes that can be tied to time-series data, then the library would not be deemed suitable for UC 2, which means this KPI is especially important with regard to this specific Use Case, and any future Use Cases involving time-series data and FHE.

Definition: We define compatibility with time series data processing, the out-of-the-box support of a library for processing time series, by supporting the appropriate FHE scheme. Since we assume that time series data consists of integers, we look for support for BFV and BGV.

How to measure: Given a specific library, compatibility with time series data processing is classified in these 2 levels:

- 0 - The library does not offer any out-of-the-box support for BFV or BGV.
- 1 - The library does offer out-of-the-box support for BFV or BGV.

Compatibility with the ML models used

Purpose: Many of the Use Cases in SECURED rely on Machine Learning. Data that has been encrypted via FHE has to be used as input for a model, which means that support for specifically ML-encrypted data would be a good addition to the library. Additionally, ML models work mostly with floating point number representations, which means that libraries that support this would be more beneficial for these Use Cases.

Definition: We define the "compatibility with ML" as the out-of-the-box support of a library for processing ML models, by supporting the appropriate scheme. To not constrain ourselves on the input data, we define the scheme to be used, either one that naturally supports floating-point operations such as CKKS or one that provides an API that includes ML model integration via quantization.

How to measure: Given a specific library, compatibility with the ML models used is classified in these 2 levels:

- 0 - The library does not offer any out-of-the-box support for CKKS.
- 1 - The library does offer out-of-the-box support for CKKS.

Project Activity

Purpose: Libraries used by the SECURED project should be mature and well-maintained. This KPI tries to encapsulate how well the library is currently being maintained. Although we cannot make any predictions about the future, we believe that libraries that currently have a lot of activity will have a higher chance of being active in the future than libraries that are no longer active anymore.

Definition: We define project activity based on the number of days between the date of the last commit to the project and February 8, 2024.

How to measure: Given a specific library, project activity is classified in these 3 levels:

- 2 - The library's last commit was less than 12 months ago.
- 1 - The library's last commit was between 12 and 36 months ago.
- 0 - The library's last commit was more than 36 months ago.

3.3.2 Results

Table 4 – Summary of HE libraries comparison using the selected KPIs.

	SEAL	HElib	OpenFHE	Lattigo	Concrete	ConcreteML
SIMD	1	1	1		1	1
Parallelization	1	1	1		1	1
GPU			0		1	1
Dedicated HW		0	1		0	0
Time series	1	1	1		0	0
ML	1	1	1		0	1
Project Activity	1	2	2	2	2	2

Table 4 lists the comparison on all KPIs for the candidate libraries. Values were omitted where the property assessed by the KPI was not explicitly reported in literature. We additionally do not place TFHE or TFHE-rs in this table, as this is sufficiently covered in Concrete and ConcreteML with their TFHE basis. The table showcases that most libraries are very close together in their properties, but we can read of that OpenFHE has the largest amount of desirable KPI's we have set. In terms of GPU acceleration support, only Concrete and ConcreteML showcase direct desirable properties with SIMD and parallelizability as added options, on top of the ML support from ConcreteM. We therefore finally select OpenFHE and ConcreteML as prime candidates for further investigation.

3.4 Benchmarking of Homomorphic Encryption Libraries

This subsection presents an in depth benchmarking of the OpenFHE and ConcreteML libraries. This analysis will drive the acceleration phase that will be carried out in the upcoming months of the project. We begin this evaluation defining the type of micro-benchmarks that will be used in the evaluation, we then report the result obtained.

3.4.1 Micro-benchmarks type

In the library-specific evaluation, we use two main types of benchmarks. Both types of micro-benchmarks measures execution time, but one aims at identify the bottlenecks within the functions of the libraries, the second aims at identify the overhead caused by the addition of privacy preserving technologies to the computation.

More in details, the main purpose of the first type was to identify the functions in the libraries that cause the largest overhead. This analysis is done to identify the most suitable candidates for acceleration. Practically, we measure this by profiling the target libraries using the micro-benchmarks provided with the library themselves. Each of the library under analysis, in fact, provides a set of micro-benchmarks. We execute these benchmarks on our platform and report the runtime measured for each micro-benchmark, i.e. for each function. The measured time is the time needed for a certain function in the library to complete the execution of the given micro-bench.

The main purpose of the second type of benchmark was to analyze the runtime overhead of HE on plaintext processing. The set of benchmarks should be meaningful in practice and stand for a representative workload, within the SECURED UCs and beyond. To generate these benchmarks, we started from the T2 benchmark

suite, which was extended to also perform plaintext processing. The encrypted and the plain-text processing have been then executed and the difference in execution time has been compared.

3.4.2 Results

In this subsubsection, we present initial measurements on the metrics that are associated with the above described micro-benchmarks. We perform specific measurements separately for OpenFHE, ConcreteML and its underlying library Concrete.

The experiments that were carried on in the next sections were executed on several platforms, configuration details of which are summarized on Table 5.

	Comp1	**Comp2**	**Comp3**	**Comp4**	**Comp5**
OS	WSL Ubuntu 22.04	Ubuntu 20.04	Centos 8.5	Almalinux 8.7	Debian 10
Hardware Architecture	x86_64	x86_64	x86_64	x86_64	x86_64
CPU op-mode(s)	32-bit, 64-bit	32-bit, 64-bit	32-bit, 64-bit	32-bit, 64-bit	32-bit, 64-bit
CPU(s)	8	12	48	40	256
Model name	11th Gen Intel® Core™ i7-1165G7 @ 2.80 GHz,	12th Gen Intel® Core™ i7-12700	Intel® Xeon® Silver 4214CPU @ 2.20 GHZ	Intel® Xeon® Silver 4210R CPU @ 2.40 GHz	AMD EPYC 7662 CPU @ 2.0 GHz
CPU freq. (MHz)	2800	2100	2200	3200	2000
Installed memory (GB)	16	16	128	128	508

Table 5 – Computer platforms used in the following sections as platforms for the benchmarks

OpenFHE

To evaluate OpenFHE, we extracted selected macro-benchmarks from the T2 compiler [29]. These benchmarks have been designed to exhaustively assess an HE library's compatibility in a wide variety of domains. These include ML operations (i.e. neural network), operations usually used in genomics (i.e. chi-squared), operations used in information theory (i.e. Hamming distance, Manhattan distance etc) and other computationally intensive applications (e.g. Fibonacci and matrix multiplication). The results of these benchmarks are displayed on Table 6, where we can see the performance of the benchmarks when executed normally (in "plaintext") and when executed under Homomorphic Encryption with OpenFHE. The variants of the benchmarks can be seen in row 2 and we split the OpenFHE performance according to the underlying scheme (BFV and CKKS) and the domain that the data are represented. Specifically the BFV scheme is used in either integer domain (notated as BFV(int)) or in binary domain (notated as BFV(bin)).

The configurations for these tests were based upon the PALISADE alternatives that were provided by the T2 compiler, which we converted to corresponding OpenFHE code. We made a custom configuration to the t2 compiler to be able to support compilation for OpenFHE (previously only PALISADE, an earlier version of OpenFHE, was supported). The experiments were run 3 different platforms (Comp1, Comp3 and Comp5). We selected these 3 platforms as representatives of different categories of machines. Comp1 is a commercial laptop with a state of the art commercial CPU with 12 cores and limited amount of available memory (16 GB), Comp3 is a server-grade machine with a server-grade CPU with 40 cores and larger amount of available memory (128 GB) and finally Comp5 is a powerful server-grade machine equipped with 256 cores and significantly more available memory. The benchmarks from Table 6 were run using the default build options of OpenFHE, meaning with OpenMP multi-threading support on and with machine-specific optimizations off. We used the gcc and g++ compiler version 13.2.0.

The security level throughout these tests was set at 128-bit security, as provided by default in OpenFHE. To make sure all T2 PALISADE equivalent benchmarks ran 128-bit security in OpenFHE, we removed the specific *Ring Dimension* set by the PALISADE benchmark and we let OpenFHE decide the *Ring Dimension* based on our security level. The *Plaintext Modulus* also needed to be changed to allow the 128-bit security, and was

		Plaintext Execution Time (s)			Encrypted Execution Time (s)								
	Machine	Comp1	Comp3	Comp5	Comp1			Comp3			Comp5		
Benchmarks	Mode				BFV (int)	BFV (bin)	CKKS	BFV (int)	BFV (bin)	CKKS	BFV (int)	BFV (bin)	CKKS
Chi squared		4×10^{-9}	15×10^{-9}	14×10^{-9}	0.045	-	0.113	0.12	-	0.222	0.104	-	0.213
Matrix Multiplication	4×4	168×10^{-9}	252×10^{-9}	217×10^{-9}	1.04	-	0.471	0.911	-	1.141	0.836	-	0.979
	8×8	1215×10^{-9}	1835×10^{-9}	1617×10^{-9}	3.683	-	4.366	8.043	-	9.071	6.775	-	7.659
	16×16	9817×10^{-9}	13929×10^{-9}	12723×10^{-9}	28.861	-	36.721	62.972	-	72.303	53.928	-	59.203
Batched Private Information Retrieval	4-bit	-	-	-	-	26.923	-	-	39.263	-	-	31.849	-
	8-bit	-	-	-	-	98.447	-	-	101.155	-	-	87.105	-
	64-bit	146×10^{-9}	201×10^{-9}	136×10^{-9}	0.007	-	0.032	0.017	-	0.009	0.015	-	0.009
	128-bit	294×10^{-9}	411×10^{-9}	272×10^{-9}	0.008	-	0.004	0.017	-	0.009	0.014	-	0.011
	256-bit	583×10^{-9}	806×10^{-9}	525×10^{-9}	0.005	-	0.003	0.026	-	0.009	0.014	-	0.011
Squared Euclidean Distance	64-bit	123×10^{-9}	175×10^{-9}	148×10^{-9}	0.385	-	0.521	1.015	-	1.328	0.868	-	1.001
	128-bit	242×10^{-9}	341×10^{-9}	288×10^{-9}	0.839	-	1.275	2.072	-	2.412	1.729	-	2.122
CRC	8-bit	1.75×10^{-9}	2.01×10^{-9}	0.68×10^{-9}	-	51.61	-	-	75.347	-	-	64.916	-
	32-bit	1.78×10^{-9}	2×10^{-9}	0.64×10^{-9}	-	-	-	-	-	-	-	-	-
Hamming Distance	4-bit	-	-	13×10^{-9}	6.755	18.729	-	9.378	22.519	-	7.874	19.953	-
	8-bit	-	-	22×10^{-9}	16.216	-	-	17.862	-	-	16.009	-	-
Relational Manhattan Distance	4-bit	7×10^{-9}	22×10^{-9}	18×10^{-9}	-	267.328	-	-	201.878	-	-	156.658	-
	8-bit	15×10^{-9}	35×10^{-9}	31×10^{-9}	-	508.605	-	-	399.663	-	-	309.201	-
Fibonacci	20	26×10^{-9}	70×10^{-9}	65×10^{-9}	10.377	-	-	18.924	-	-	17.724	-	-
	30	72×10^{-9}	103×10^{-9}	91×10^{-9}	10.727	-	-	19.366	-	-	18.149	-	-
	40	75×10^{-9}	-	116×10^{-9}	-	-	-	-	-	-	-	-	-
Neural Network	50	751.1×10^{-9}	1048.3×10^{-9}	1070×10^{-9}	21.747	-	252.657	30.323	-	299.124	25.11	-	246.784
	100	1613.4×10^{-9}	2103×10^{-9}	2138×10^{-9}	44.986	-	514.449	60.972	-	628.726	50.166	-	501.776
	150	$2462{,}2 \times 10^{-9}$	3152.6×10^{-9}	3207×10^{-9}	66.953	-	769.189	91.59	-	890.668	74.844	-	733.678

Table 6 – Results of plaintext versus OpenFHE-protected execution from selected macro-benchmarks from T2 [29]. All results are expressed in seconds (s) and benchmarks are run on three different machines: Comp1, Comp3 and Comp5

set at 65537). The value was chosen to be consistent with other benchmarks that needed a higher *Plaintext Modulus*.

An interesting observation that was made while testing is that when the openMP multi-threading support is enabled in OpenFHE, it automatically spawns as many threads as the available cores of the system. In the case of Comp3 and Comp5, that is 128 and 256 respectively. We observed that when that many threads are spawned and the underlying problem does not provide enough workload, which is the case for every benchmark we are using the overhead of the creation and possibly the synchronization of these threads result in a much slower execution time. For that reason and as a preliminary decision, we fixed the number of threads to a lower number, which we selected to be 16. OpenFHE gives you the ability to control the number of openMP threads spawned by setting the environmental variable *OMP_NUM_THREADS* to the desired number. All the results in table 6 are reporting the execution time when *OMP_NUM_THREADS* is set to 16 or lower (to the max system available cores if it is lower than 16). For the scope of this deliverable we did not delve deeper into exploring the optimal selection of number of threads, but we see that there is value on fine-tuning this parameter, on a per-application as well as a per-machine basis, as the speedup that can be obtained by parallelization depends both on the workload provided by the application and on the machine's characteristics. It should be noted here, that the parallelization is offered by OpenFHE and it used when computing the underlying primitive operations and not on the application level. For example, one could imagine that the matrix multiplication could be parallelized at a per multiplication manner, but this is not the case on our benchmark. In this example, the parallelization is used when computing each multiplication and multiple threads are spawned to compute this single multiplication.

From the results in Table 6, we can see that linear transformations that are important for the medical focus of SECURED can have higher execution times as we move to larger problems. The scaling factor with regard to the dimension is high, which makes it a prime target for acceleration when working with medical imaging that will require more pixels. It also indicates a preference on using CKKS if available, as this performs better on all chosen benchmarks except the neural network inference. The overall communication cost of running FHE compared to plaintext is large, but the execution times for most of the small applications are reasonable.

OpenFHE provides a series of benchmarks that are used to evaluate the execution time of the primitive opera-

	Standard Build			Built with Intel HEXL backend			
Benchmark	**Time (us)**	**CPU (us)**	**Iterations**	**Time (us)**	**CPU (us)**	**Iterations**	**Speedup**
NTTTransform1024	15.1	15.1	46192	7.74	7.73	90582	96.10%
INTTTransform1024	14.7	14.7	47352	8.49	8.49	82463	74.15%
NTTTransform4096	71.9	71.9	9738	36	36	19346	98.67%
INTTTransform4096	69.4	69.4	10085	39.4	39.4	17775	76.25%
NTTTransformInPlace1024	14.8	14.8	46880	7.72	7.72	90550	93.15%
INTTTransformInPlace1024	14.5	14.5	48154	8.4	8.4	83400	73.19%
NTTTransformInPlace4096	70.7	70.7	9893	36	36	19456	96.66%
INTTTransformInPlace4096	68.6	68.6	10187	38.8	38.8	18043	77.12%
BFVrns_KeyGen	3672	3671	191	3487	3486	203	6.28%
BFVrns_MultKeyGen	6118	6117	114	5935	5933	117	2.63%
BFVrns_EvalAtIndexKeyGen	6142	6141	113	5911	5909	118	4.42%
BFVrns_Encryption	4860	4859	144	4052	4051	172	19.44%
BFVrns_Decryption	2623	2124	413	2722	2638	490	18.64%
BFVrns_Add	95.4	95.3	30001	137	137	4949	-83.50%
BFVrns_AddInPlace	15.7	15.7	44588	10.8	10.8	61942	38.92%
BFVrns_MultNoRelin	15984	13015	59	23832	12806	52	-11.86%
BFVrns_MultRelin	20554	19779	44	22592	11330	60	36.36%
BFVrns_EvalAtIndex	4827	2413	293	2571	2028	366	24.91%
CKKSrns_KeyGen	5106	4980	105	5173	5171	104	-0.95%
CKKSrns_MultKeyGen	8385	8383	80	7984	7982	87	8.75%
CKKSrns_EvalAtIndexKeyGen	8238	8236	85	7958	7957	86	1.18%
CKKSrns_Encryption	4295	4294	162	3710	3709	185	14.20%
CKKSrns_Decryption	226	226	3091	135	135	5169	67.23%
CKKSrns_Add	101	101	9015	98.8	98.8	7092	-21.33%
CKKSrns_AddInPlace	64.9	64.9	10730	65.5	65.5	10757	0.25%
CKKSrns_MultNoRelin	214	214	3267	211	211	3282	0.46%
CKKSrns_MultRelin	8985	6906	102	9686	9682	128	25.49%
CKKSrns_Relin	9733	6445	117	8988	8984	82	-29.91%
CKKSrns_RelinInPlace	8906	6201	99	6533	6530	100	1.01%
CKKSrns_Rescale	812	811	688	503	503	1109	61.19%
CKKSrns_RescaleInPlace	726	726	962	454	454	1537	59.77%
CKKSrns_EvalAtIndex	12108	6052	108	6329	6326	149	37.96%
BGVrns_KeyGen	5215	5113	104	4884	4702	111	6.73%
BGVrns_MultKeyGen	8642	8641	81	7948	7947	86	6.17%
BGVrns_EvalAtIndexKeyGen	8387	8385	83	7866	7864	89	7.23%
BGVrns_Encryption	4563	4562	152	3810	3809	182	19.74%
BGVrns_Decryption	448	448	1595	376	376	1863	16.80%
BGVrns_Add	115	115	7239	99.3	99.2	7091	-2.04%
BGVrns_AddInPlace	120	120	6825	48.6	48.6	14826	117.23%
BGVrns_MultNoRelin	288	288	2237	166	166	4240	89.54%
BGVrns_MultRelin	9018	8782	124	7995	7987	126	1.61%
BGVrns_Relin	7035	7032	95	8347	8344	87	-8.42%
BGVrns_RelinInPlace	6794	6791	98	7018	7015	78	-20.41%
BGVrns_ModSwitch	846	845	670	486	486	1424	112.54%
BGVrns_ModSwitchInPlace	831	831	684	491	491	1450	111.99%
BGVrns_EvalAtIndex	6246	6243	114	6864	6860	100	-12.28%

Table 7 – Runtime of micro-benchmarks of OpenFHE when built without and with the Intel HEXL backend

tions of the schemes that OpenFHE supports (BFV, CKKS and BGV) when used in different domains (integer or binary) and on the execution time of some underlying primitives that are being used by these schemes (e.g NTT domain operations). Since these benchmarks match our definition of micro-benchmarks, we run them too. The results of these micro-benchmarks can be found in Table 7. For these benchmarks we selected as our target platform only Comp5. The standard OpenFHE micro-benchmarks were built with the default built options, meaning with openMP multi-threading support on and with machine-specific optimizations off. We used clang and clang-++ version 13.0.1-6 deb10u4.

Additionally, OpenFHE provides support for specialized hardware backends, meaning that one can build the library and link it with a specialized backend that provides optimized implementations for the primitives that OpenFHE uses. Intel HEXL is a library that accelerates modular arithmetic operations used in homomorphic encryption, especially taking advantage of vector instructions on modern CPUs and OpenFHE offer support on building it as a backend. We followed the instructions given in the OpenFHE documentation on HEXL [30] and we benchmarked the performance specifically on Comp5. The results for the Intel HEXL compared to a baseline build can be seen in the 2nd and 3rd column of Table 7. The default build option were used, meaning with openMP multi-threading support on and with machine-specific optimizations off and compiler clang and clang-++ version 13.0.1-6 deb10u4.

ConcreteML

To evaluate ConcreteML library we designed a relative simple model, structured as a binary classification Multi-Layer Perceptron, to process ECG segments for arrhythmia detection. This relates to UC-2. The model proves highly effective in identifying arrhythmic patterns within ECG data, achieving an accuracy of 97.53%.

Dataset: The dataset employed to develop an arrhythmia detection model was derived from MIT-BIH Arrhythmia database [31]. This dataset comprises 48 ECG recordings, each lasting 30 minutes and captured from two channels. These recordings were collected from 47 individuals and were analyzed by the BIH Arrhythmia Laboratory between 1975 and 1979. The MIT-BIH Arrhythmia database is a subset of a larger collection of 4,000 ambulatory ECG recordings, each with a duration of 24 hours, gathered from a diverse group of patients at Boston's Beth Israel Hospital. This patient group consisted of approximately 60% inpatients and 40% outpatients. To ensure the inclusion of rare, yet clinically significant arrhythmias that might be overlooked in a small, random sample, 25 recordings were deliberately selected. Additionally, 23 recordings were chosen at random from the same collection to ensure a comprehensive representation of the patient demographic and the spectrum of cardiac activities encountered in clinical settings. The recordings were digitized at a sampling rate of 360 Hz per channel, with each analog value quantized to 11 bits resolution across a 10 mV range. Annotations for each recording were provided by at least two cardiologists, ensuring the accuracy and reliability of the data for arrhythmia detection research.

Preprocessing: The preprocessing and feature extraction stages are pivotal steps that follow the collection of samples. This is due to the fact that continuous raw data often contain redundant information which, without explicit utilization, does not contribute directly to building a compact and efficient model for arrhythmia detection. The detailed methodologies for data extraction and preprocessing are elaborated by Sakib et al. [32]. Initially, the data undergo a crucial filtering process to remove noise. The first step involves applying a Butterworth high-pass filter with a cutoff frequency of 1Hz to eliminate the DC component and mitigate baseline fluctuations. Following this, a Butterworth band-rejectfilter is employed to minimize the 60 Hz AC interference. Lastly, to remove high-frequency disturbances, a Butterworth low-pass filter with a cutoff frequency of 25 Hz is utilized. In the MIT-BIH arrhythmia dataset, which includes more than 15 distinct heartbeat annotations, these are classified into five main categories: N (Normal beat), S (Supraventricular ectopic beat), V (Ventricular ectopic beat), F (Fusion beat), and Q (Unknown beat). Since a vast majority of the heartbeats are categorized as N (comprising almost 90%), the rest are grouped into class A (abnormal beat) to convert the task into a binary classification problem. To ensure dataset balance, the N class was downsampled to equal the number of beats in class A. The dataset utilized for the development of the model does not comprise raw data. Instead, it features a set of derived characteristics, detailed in Table 8. These features are calculated based on the R-peaks annotated for each heartbeat in the original MIT-BIH dataset.

Feature Group	Feature Name
RR Intervals	Pre-RR
RR Intervals	Post-RR
Heartbeat Intervals features	PQ Interval
Heartbeat Intervals features	QT Interval
Heartbeat Intervals features	ST Interval
Heartbeat Intervals features	QRS Duration
Heart beats amplitude features	P peak
Heart beats amplitude features	Q peak
Heart beats amplitude features	R peak
Heart beats amplitude features	S peak
Heart beats amplitude features	T peak
Morphology Features	Morphology Feature 1
Morphology Features	Morphology Feature 2
Morphology Features	Morphology Feature 3
Morphology Features	Morphology Feature 4
Morphology Features	Morphology Feature 5

Table 8 – Features.

The Pre-RR and Post-RR intervals represent the time duration from the previous and next R peak, measured in milliseconds, respectively. The P, Q, R, and T peaks correspond to the amplitudes of each data point in the ECG recording, measured in millivolts. The PQ, QT, ST, and QRS complex intervals denote the duration between the corresponding peaks. Additionally, the QRS morphology features consist of five samples within the QRS complex.

Concluded dataset: The final dataset comprises 24,148 samples, each containing 32 features (16 features per channel). Each sample is labeled as either N (normal, 0) or A (abnormal, 1). After feature extraction, the data standardization technique was employed to scale the data into a specific range. Standardization is applied to introduce a Gaussian distribution with unit standard deviation to the features.

Model design: The selection criteria for the ideal model to identify arrhythmias in heartbeats focus on two key aspects, performance and size. For performance, the model needs to display high accuracy and effectively handle new, unseen data. In terms of size, the model's compactness is crucial, enabling its deployment on edge devices like FPGAs for efficient, real-time processing. These factors were taken into account to ensure the chosen model is reliable, accurate, and suitable for deployment in resource-constrained environments. Based on the these considerations and in accordance with the dataset format, we employed a compact architecture using a small MLP, more specific a Dense Neural Network. The operation of the dense layer is described by the formula in Eq. 1. The output vector $\mathbf{y}$ of the dense layer is computed using the following equation:

$$\mathbf{y} = \sigma(\mathbf{W} \cdot \mathbf{x} + \mathbf{b}) \tag{1}$$

where $\mathbf{x}$ represents the input vector of dimension n, $\mathbf{W}$ is the weight matrix of dimension $m \times n$, $\mathbf{b}$ is the bias vector of dimension m, and σ denotes the activation function.

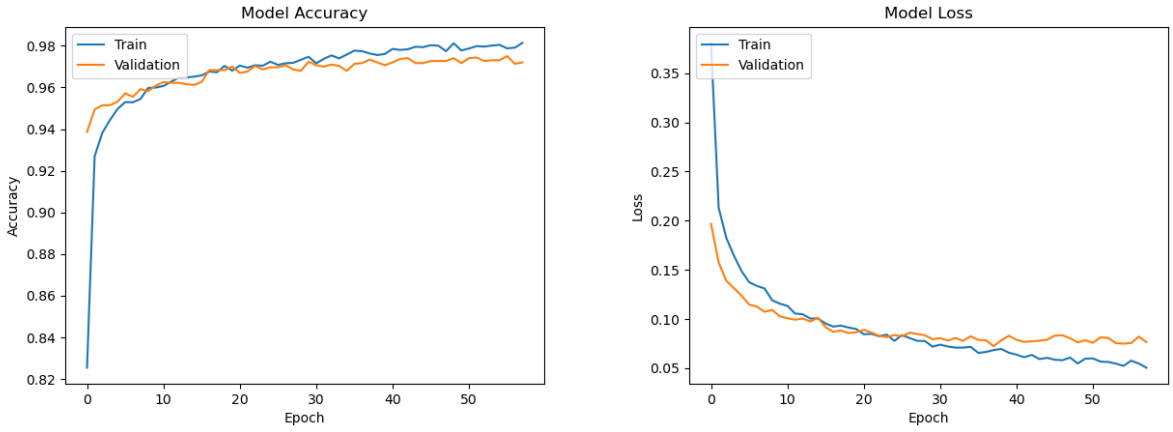
Architecture: The architecture of the DNN depicted in Figure 3 consists of five fully connected layers. The first and second layers each have 64 neurons, followed by a third layer with 32 neurons, a fourth layer with 16 neurons, and concluding with a final layer that has a single neuron. The overall model contains 8,897 parameters. In the first four layers, the ReLU activation function was employed, while the sigmoid activation function was used in the final layer, as it is a binary classification problem. Dropout layers of 20% were inserted

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	2112
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 64)	4160
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2080
dropout_2 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 16)	528
dense_4 (Dense)	(None, 1)	17

Total params: 8,897
Trainable params: 8,897
Non-trainable params: 0

Figure 3 – DNN structure.

after each of the first three layers to mitigate overfitting. A dropout layer randomly sets a percentage of the outputs of the previous layer to zero during training. Additionally, the Adam optimizer was selected as the optimization algorithm to minimize the loss function. The learning rate was set to 0.001, and the batch size was chosen at 32 samples based on experimentation.



(a) Accuracy. **(b)** Loss.

Figure 4 – Training and Validation Phases.

Training/Evaluation: The training and validation phases are depicted in Figure 4. Throughout these phases, the number of epochs could extend to a maximum of 100. However, if there was no improvement in validation accuracy after 20 consecutive epochs, the training process was halted, and the best model identified up to that point was selected. Various performance metrics were employed to provide a comprehensive evaluation of the developed model for arrhythmia detection. Each metric offers a unique perspective on the assessment of the neural network's effectiveness. The outcomes of these evaluations are summarized in Table 9, proving the model's performance

Results: The DNN demonstrates a high level of accuracy, achieving 97.56%. Accuracy is a measure that calculates the proportion of correctly predicted instances out of the total instances, providing a general assessment of the model's performance. Furthermore, the model exhibits a precision of 97.79%. Precision is the ratio of true positive predictions to the total number of positive predictions made by the model, indicating the reliability of the model's positive class predictions. This metric is crucial, especially when the implications of false positive errors are significant. Recall, or Sensitivity, is another vital performance metric, with the model achieving 97.32%. Recall measures the proportion of true positive predictions out of all actual positive instances within the dataset, assessing the model's ability to identify positive instances accurately. This metric is particularly

Metric	Result(%)
Accuracy	97.56
Precision	97.79
Recall	97.32
F1 Score	97.55

Table 9 – Performance metrics of the NN.

important in medical contexts, such as distinguishing arrhythmic heartbeats from normal ones, where failing to detect a positive instance (a false negative) could have severe consequences. Lastly, the F1 score, which is the harmonic mean of precision and recall, serves as a balance between the two, emphasizing their equal significance. It is especially useful in situations with an uneven class distribution. The DNN achieved an F1 score of 97.55%, indicating a strong balance between precision and recall in its predictions.

FHE implementation: In order to run the model in FHE we quantized our model and then produced the FHE equivalent, based on the quantized one (for more details on Post-Training Quantization (PTQ), see Section 3.2.2). We calculated the accuracy of these models for different quantization-bits. ConcreteML provides a quantization function, in order to translate the initial model to a quantized one. It allows quantization of the model up to 8-bits. In our case, the maximum number of allowed quantization is 7-bits, as the library could not handle 8-bit precision of some NP-library functions. In Table 10 we have a comparison between the produced quantized model and the corresponding FHE equivalent, by using ConcreteML library running on the same dataset used on our initial non-quantized model. We can see that the maximum accuracy we achieve is 94.2% for 7 quantization-bits, which is close to the reference accuracy of the non-quantized model but not as good. Also, we observe that the FHE execution times are high in comparison to the execution time of the MLP quantized or non-quantized model, in order to achieve accuracy relative to the reference.

Quantization bits	Execution time Quantized Model (s)	Execution time FHE Model (s)	Quantization Time (s)	Final Accuracy
7	8×10^{-6}	524	83	94.2%
6	7×10^{-6}	447	36	92.3%
5	1×10^{-5}	259	22	82.6%
4	9×10^{-6}	285	18	67.4%
3	9×10^{-6}	22	17	58.7%
2	9×10^{-6}	0.8	17	48.6%

Table 10 – Execution time comparison between the MLP model and the FHE-MLP model. Reference accuracy of non-quantized model is 97.53% and execution time is 10^{-4} sec. 64-bit hardware architecture and Intel® Core™ i7-6700HQ CPU @ 2.60GHz × 8.

Concrete

The potential of the Concrete (`concrete-python`) [23] library to be used as a means of the basic software toolset for the implementation of homomorphically encrypted operations involved in signal–and image–processing, hence relating to UC-1, is explored in the following text.

The benchmarks tested using Concrete are the following:

- Matrix-matrix multiplication. This is very often the dominant basic operation in signal– and image– processing algorithms. The cases where both matrices are encrypted, or where one of the matrices only is encrypted and the other is in plaintext (clear or not encrypted), have been examined;
- Basic operations on images: Operations on pixels (Colour Inversion), Resizing, Interpolation and Regridding.

The basic goal is to quantify the impact of the size of the operands, namely, the sizes of the matrices and the word-length of their elements, and the sizes of the images and the word-length required to express pixels, on the performance of `concrete`.

Table 11 reports the performances obtained for several cases of matrix-matrix multiplication, both for the encrypted-by-encrypted and the clear-by-encrypted cases. The notation MM_XY_$(c\times d)\times(e\times f)$_$b$ denotes the multiplication of two matrices of sizes $(c\times d)$ and $(e\times f)$, respectively, while b is the number of bits required to represent the integer elements. In this notation, the values of X, Y can be either C or E, denoting that the corresponding matrix is clear (C) or encrypted (E). For example, MM_EE_$(16\times16)\times(16\times16)$_$4$ performs HE matrix multiplication of two encrypted matrices of dimensions 16×16, with elements of 4 bits $(0-15)$. OL means Off-Limits, i.e., operation cannot be executed on the specific system, due to memory limitations. Maximum memory usage includes system memory usage.

The benchmark programs were developed, relying on a number of operations which belong into the category of operations supported by Concrete. Such operations include the highly optimized `Numpy` function `numpy.matmul` for matrix-by-matrix multiplication, or the property `nd.array.shape` which returns the dimensions of an array. In addition, the Concrete compiler is also designed to offer highly optimized performances of the Encrypt, Run, and Decrypt functions in the context of the corresponding compatible operations and structures in the HE plane.

Operation	Comp1	Comp2		Comp3		Comp4		Comp5
	Exec. time (s)	Exec. time (s)	Max. mem (GB)	Exec. time (s)	Max. mem (GB)	Exec. time (s)	Max. mem (GB)	Exec. time (s)
MM_EE_$(2\times2)\times(2\times2)$_$2$	1.454	0.959	1.25	0.634		0.48	30	0.59
MM_EE_$(2\times2)\times(2\times2)$_$3$	14.862	1.438	1.34	0.865		0.81	30	0.80
MM_EE_$(2\times2)\times(2\times2)$_$4$	OL	1.603	1.37	0.869		1.03	30	0.855
MM_EE_$(2\times2)\times(2\times2)$_$4$_Enc		0.003		0.004		0.004		0.005
MM_EE_$(2\times2)\times(2\times2)$_$4$_Run		0.589		0.538		0.479		0.30
MM_EE_$(2\times2)\times(2\times2)$_$4$_Dec		0.002		0.015		0.002		0.002
MM_EE_$(2\times2)\times(2\times2)$_$8$		OL		95.029		113.86	50	45.8
MM_EE_$(8\times8)\times(8\times8)$_$4$		34.669		7.622	42.70	9.921	31	5.313
MM_EE_$(8\times8)\times(8\times8)$_$6$		269.353	4.77	99.072	46.99	79.049		84.441
MM_EE_$(16\times16)\times(16\times16)$_$2$		50.711	1.25	5.647		9.727	31	1.96
MM_EE_$(16\times16)\times(16\times16)$_$3$		110.618		16.732	3.46	27.899	31.51	16.648
MM_EE_$(16\times16)\times(16\times16)$_$3$_Key		0.557		0.059				0.412
MM_EE_$(16\times16)\times(16\times16)$_$3$_Enc		0.058		0.059		0.066		0.047
MM_EE_$(16\times16)\times(16\times16)$_$3$_Run		108.708		16.009		26.978		16.454
MM_EE_$(16\times16)\times(16\times16)$_$3$_Dec		0.022		0.019		0.022		0.012
MM_EE_$(16\times16)\times(16\times16)$_$4$		261.578		38.943	3.91	58.602		38.102
MM_EE_$(16\times16)\times(16\times16)$_$6$		2521.781		759.477	7.79	501.904	34.5	281.207
MM_EE_$(16\times16)\times(16\times16)$_$7$		6500.225		4368.284		2909.258		886.094
MM_EE_$(16\times16)\times(16\times16)$_$8$				27 266.867	97	10 346.090	97	5868.156
MM_CE_$(16\times16)\times(16\times16)$_$8$				0.268	14.11	0.239		0.248
MM_CE_$(256\times256)\times(256\times256)$_$8$				32.511	29.21	52.601	38.61	28.55
MM_EE_$(1\times8)\times(8\times1)$_$4$		2.240	1.32	1.370	14.36	1.597	30	1.160
MM_EE_$(1\times8)\times(8\times1)$_$6$		19.152	3.10	10.054	16.34	10.832	30.5	7.425
MM_EE_$(1\times8)\times(8\times1)$_$8$		OL		203.584	42.18	239.522	58.81	95.895

Table 11 – HE matrix multiplications, (`concrete`)

Table 12 reports the performances of some basic operations on images, for $2-$D images of typical sizes. More specifically, color inversion, resizing, interpolation and regridding are given as example cases in the table.

Operation				Execution Time (s)					
				plaintext			HE		
	Input image size	Output image size	Bits per pixel	Comp2	Comp3	Comp5	Comp2	Comp3	Comp5
ImageColourInversion	100×100	100×100	8	1.268×10^{-3}	0.146×10^{-3}	0.110×10^{-3}	2.152	1.178	0.723
DownSize $(1:2)$	100×100	50×50	8	1.417×10^{-3}	0.806×10^{-3}	0.598×10^{-3}	2.427	1.361	1.087
InterpImage	100×100	199×199	8	10.588×10^{-3}	4.707×10^{-3}	4.596×10^{-3}	9.97	4.791	3.452
regrid_x_ax_512	512×512	512×512	8	591×10^{-3}	427.39×10^{-3}	874.93×10^{-3}	133.468	85.782	77.202
regrid_x_ax_512div4	256×256	256×256	8	24.777×10^{-3}	36.6×10^{-3}	38.262×10^{-3}	25.329	12.032	10.828

Table 12 – Basic Image Operations

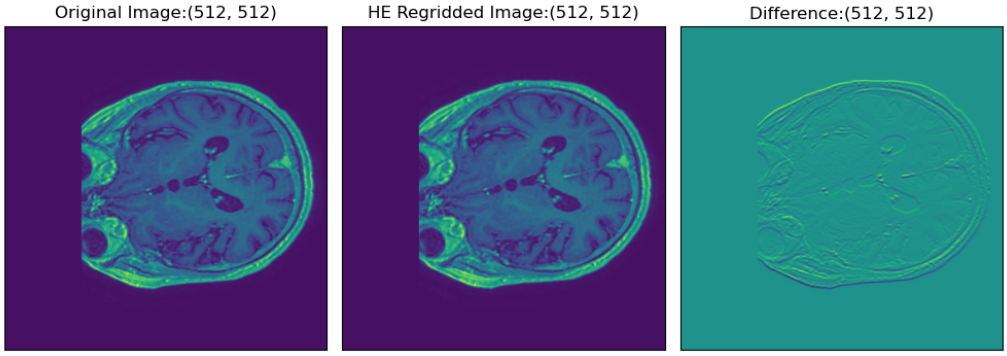


Figure 5 – HE Regridded 512×512 Image

Execution time is given for the plaintext execution and for the HE execution.

The regridding procedure required in UC-1 is initially expressed as a translation of a given image in a 2D space by a vector comprising not necessarily integer elements. Therefore, new pixel values need to be computed, at non-integer grid points.

If regridding along one axis is assumed, this translation operation, which includes computation of new pixel values at the target grid points, can be expressed as a transformation

$$Y = CX,$$

i.e., a matrix multiplication in which the operand C is a suitably constructed transformation matrix and X is the original image to be translated. Matrix C does not depend on the image data, but only on the initial and target grid. In order to perform the operation in `concrete`, C is mapped to a fixed-point representation, subsequently expressed in integer format. The resolution achieved by the translation, is defined by the number of the bits allocated for the representation of the integers in the modified C, in this way affecting the complexity of the operation. The potential of breaking the image matrix to a set of smaller matrices, offers a trade-off parameter towards obtaining smaller execution times. This can be seen in the last two rows of Table 12, where regrid_x_ax_512 corresponds to the regridding of a 512×512 image as a unit (Fig. 5), while regrid_x_ax_512div4 corresponds to the regridding of each of four partitions of the same image in parallel.

Discussion

Concrete implements a variant of TFHE and has been developed by zama. Concrete-python is user-friendly; the "programmable bootstrapping" feature of the framework offers a useful alternative option to overcome certain implementation difficulties in HE processing, such as the accumulation of errors after successive multiplications or the implementation of operations other than addition and multiplication, with the use of look-up tables. However, Concrete has the disadvantage of small precision of the encoded data; 16 bits is reported as the upper limit. To enlighten on the information in Table4 about Concrete, the following hold: Regarding hardware acceleration features, in 2022 zama announced their intention to develop FH accelerators, including FPGAs, expected in 2024 (https://www.zama.ai/post/introducing-the-concrete-framework). Furthermore, ConcreteML has evolved as a Privacy preserving ML framework built on top of Concrete.

Based on the experiments performed on the aforementioned computers, a number of conclusions regarding Concrete HE matrix operations are derived, and are described in the following. When both operands are encrypted (i.e., Encrypted $\times$ Encrypted) the computational load of the Concrete execution becomes very heavy, in terms of required computation and memory resources leading to excessively long execution times compared to plaintext counterparts.

The word-length of the elements of the matrices in the plaintext field, is the most critical parameter that defines the execution time and required resources of the HE operation on a specific computer. In accordance with Concrete documentation, 16 bits has been found to be the limit; therefore, in the available computers, only operations which result in a maximum of 16-bit products of the elements can be performed (i.e., 8×8, 9×7). It was observed that the larger the memory of the computer which is used for the processing, the higher in the range 2 to 16 bits, the performance can go. Also, as the number of bits rises, the execution time rises exponentially. Figure 6 depicts the impact of the number of bits of the elements of the matrices being multiplied, for a 16×16 multiplication in Concrete.

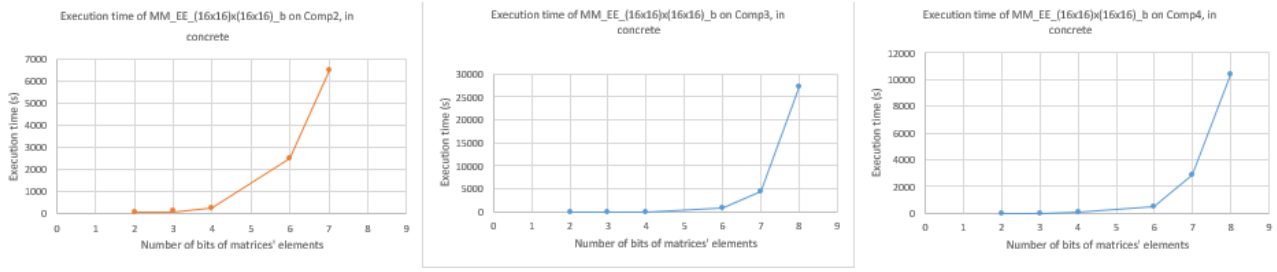


Figure 6 – Execution time (in seconds) of the operation MM_EE_$(16\times16)\times(16\times16)$_$b$ on various computers, as a function of the number of bits of the elements of the matrix operands.

As the dimensions of the matrices rise, execution time also rises exponentially. Figure 7 illustrates three separate plots of the execution times achieved on each one of three different computers, respectively, for 4-bit element matrix multiplications of dimensions from 2×2 to 16×16. Testing reached a limit of matrices with 16 rows/columns. Because matrix multiplication is a dense operation which can benefit by tight pipe-lining, it was obvious from the tests that multiple cores and parallel threads greatly improve the execution time. For the runs on Comp2 and Comp3, all cores of the system were 100% active for the largest portion of the execution (Figure 8). However, in the run on Comp4, which is characterized by a higher processor frequency, for certain cases the assignment scheme resulted in a faster execution time, via smaller effort (utilization) of the cores.

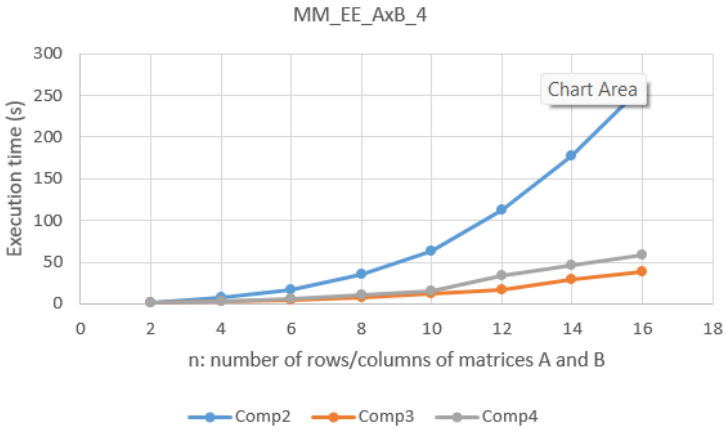


Figure 7 – Execution time (in seconds) of the operation of HE matrix multiplication in Concrete, for matrices with 4-bit elements, as a function of the dimensions of the matrices. Test performed on three different computers are shown.

When one of the operands only is encrypted and the other is clear, operations become feasible and manageable; matrices of dimensions of up to 512×512 were used in the experiments. Exploitation of the parallelism within the Concrete HE operations and mapping on parallel computer systems (multiple cores, parallel threads), and on massively parallel hardware is expected to further decrease execution time.

From linear algebra it is well known that many matrix operation algorithms, such as matrix multiplication, can be realized by schemes which use partitions of the matrices. If such an option exists, when encrypting the data, it seems that the execution time can be improved dramatically.

A serious issue to consider when it comes to evaluating a HE operation with Concrete, is the size of the input set. The input set must be carefully constructed so that it is large enough to avoid error in result, yet not excessively large, to avoid unnecessary memory and delay, and it must provide sufficient information to model input sample distribution.

Figure 8 – CPU history for MM_EE_$(16\times16)\times(16\times16)$_7, by gnome-system-monitor.

Finally, regarding the security level: The concrete-rust main programs on the lower level of the compiler are reported to use $\lambda = 128$ as the default level of security. The Lattice Estimator [33] has been used to define parameters for security. However, the parameter sets for the Concrete HE executions performed during the testings presented herein, don't always seem to meet this standard.

Figure 9 shows a print-out of the show_statistics configuration option available in Concrete, to enlighten on the data sizes involved.

```
Statistics
--------------------------------------------------------------------------
size_of_secret_keys: 6808
size_of_bootstrap_keys: 0
size_of_keyswitch_keys: 0
size_of_inputs: 1786773504
size_of_outputs: 1786773504
p_error: 3.73079332527805e-11
global_p_error: 9.780003030257878e-06
complexity: 851.0
programmable_bootstrap_count: 0
key_switch_count: 0
packing_key_switch_count: 0
clear_addition_count: 0
encrypted_addition_count: 134217728
encrypted_addition_count_per_parameter: {
    LweSecretKeyParam(dimension=851): 134217728
}
clear_multiplication_count: 134217728
clear_multiplication_count_per_parameter: {
    LweSecretKeyParam(dimension=851): 134217728
}
encrypted_negation_count: 0
--------------------------------------------------------------------------
```

Figure 9 – The Concrete show_statistics = True print-out.

3.5 Conclusions

This section presents an overview of work undertaken up to the point of this deliverable on SMPC and HE. It discusses these technologies, with a particular focus on encrypted ML inference and summarizes the main features of the selected libraries.

Further, as starting point for accelerations, we concentrated on libraries implementing HE, and we compared them at high level, defining and using a number of KPIs, and at low level using a number of micro-benchmarks that allowed us to identify the bottlenecks within the components of the libraries and the overhead caused by the encrypted computation. These results will drive the acceleration task that will be carried out in the following months of the SECURED project.

4 Federated Learning

Federated Learning (FL) [34] is a Machine Learning approach that enables training models across multiple decentralized devices or servers containing data, without exchanging that data directly. In traditional Machine Learning, data is typically collected, centralized, and then used to train a model. However, in federated learning, the model is sent to where the data resides, allowing training to happen on the devices or servers holding the data.

In more detail, in every training round (i.e., epoch) data owners (i.e., clients) train a common model locally and share the corresponding model updates (i.e., gradients) which are aggregated into a single global model for the following round. The process is aided by a trusted server (i.e., aggregator) as illustrated in Figure 10.

4.1 Types of Federated Learning

There are various Federated Learning (FL) [35] systems (detailed in D4.1), tackling different scenarios. The two most notable directions are Vertical or Horizontal, and Cross-silo or Cross-device. The first angle is concerned with the feature space of the underlying datasets, the second depends on the number and nature of clients.

- Vertical/Horizontal: Client datasets in Vertical setting have different feature space, while in Horizontal setting, the datasets have the same feature space across all clients. For example, the clients know different information about the same patients (e.g., one client knows the patients' cholesterol levels, while another knows the patients' blood sugar levels), or the same information about different patients (e.g., the clients know the platelets counts of different patients), respectively.
- Cross-Silo/Cross-Device: the former refers to the case when the number of clients is limited, but they are reliable, controlling significant computational power and sophisticated technical capabilities. The latter encompass scenarios with a virtually unlimited number of clients, but with limited computational resources and small datasets. For example, collaboration between a few hospitals with appropriate resources or between millions of mobile devices with limited battery life and bandwidth.

A high-level description of the Federated Learning protocol is provided below.

1. The aggregator server initializes the model and determines the hyperparameters.
2. In order to converge the model, the following are necessary:

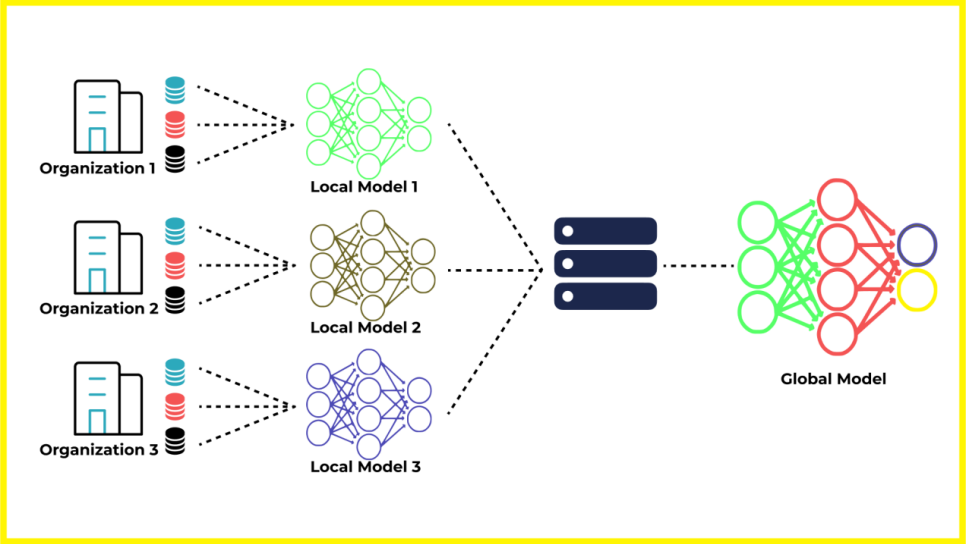


Figure 10 – Illustration of Federated Learning. Source: *Enhancing Data Security: Unleashing the Power of Federated Learning*, by Marouane AOUFI (https://www.linkedin.com/pulse/enhancing-data-security-unleashing-power-federated-learning-aoufi/)

   (a) The aggregator broadcasts the model to *some* clients.
   (b) Those clients *train* that model on their local dataset and share the result with the aggregator.
   (c) The aggregator *aggregates* the received model updates into the new global model.
3. The final model is broadcasted to all participating clients.

Federated Learning, with its distributed and collaborative nature, brings forth several privacy and security concerns that need to be carefully addressed. These concerns arise due to the decentralized processing of sensitive data and the potential vulnerabilities associated with sharing models and aggregating information across multiple participants. Understanding and mitigating these concerns is crucial for ensuring the privacy and security of Federated Learning systems [36].

4.2 Selected Frameworks

In recent years, various efforts, both open-source and commercial, have aimed to incorporate Federated Learning (FL) technology into diverse sectors, including healthcare and other industries. Notable frameworks in this space include TensorFlow Federated, Nvidia Flare, PySyft, FedML, FATE, Flower, OpenFL, Fed-BioMed, IBM Federated Learning, HP Swarm Learning, FederatedScope, FLUTE, and more. Among these, Flower and Nvidia Flare stand out as the most widespread [37] [38]. Therefore, we systematically compare Flower and Nvidia Flare in the following sections.

Nvidia Flare

Nvidia Flare [39] (NVIDIA Federated Learning Application Runtime Environment) is a domain-agnostic, open-source, and extensible SDK for Federated Learning. It empowers researchers and data scientists to adapt their existing ML/DL workflows to a federated paradigm and enables platform developers to build a secure, privacy-preserving solution for distributed multi-party collaboration. The framework offers both simulated and real-world Federated Learning capabilities.

- *Security/Privacy:* Flare addresses security and privacy through a range of privacy-preserving algorithms. These include techniques such as excluding variables, truncating weights by percentile, applying sparse vector methods, utilizing Homomorphic Encryption. Flare ensures the identities of communicating peers with the use of mutual Transport Layer Security (TLS).
- *Documentation:* The quality of the documentation is one of the areas where Nvidia Flare excels the most among other software solutions. The documentation comprises key functionalities, example applications, and separate guides for regular usage, along with programming guides for developers interested in building experiments using the available tools. Notably, it provides more example codes than Flower.
- *User-Friendly Interface*: Flare offers an end-to-end operational environment for various user roles. It features a comprehensive provisioning system that generates security credentials for secure communications, facilitating the secure deployment of Federated Learning (FL) applications in real-world scenarios. Additionally, Flare provides an FL Simulator for running proof-of-concept studies locally. In production mode, users can conduct FL studies by submitting jobs through admin commands, either on a Notebook or using the Flare Console — an interactive command tool. Through various commands, users can initiate and halt specific clients or the entire system, submit new jobs, check job statuses, clone existing jobs to create new ones, and perform other essential tasks. Flare implements a role-based user authorization system, dictating what actions a user can or cannot perform
- *Compatibility:* The SDK is a lightweight, flexible, and scalable Python package, and allows using any training libraries (PyTorch, TensorFlow, XGBoost, or even NumPy) and apply them in real-world FL settings.

- *Learning curve:* Nvidia Flare offers a shallower learning curve compared to Flower. Users have reported that it requires more time to become familiar with Flare, and the installation process is also considered more cumbersome [37] [38].
- *Customization:* Flare supports several server aggregation strategies including Scaffold, FedProx, Fed Average, and FedOpt. Flare allows users to experiment with different algorithms and data splits using different levels of heterogeneity based on a Dirichlet sampling strategy.
- *Community Support:* Flare is actively maintained and developed by Nvidia. However, the general research community seems to favor Flower over Flare probably due to is faster learning time and simplicity.
- *Scalability:* Flare is scalable to tens of clients [39], or even more, which is sufficient for a typical cross-silo federated learning scenario in SECURED. However, there are no public evaluations for hundreds of clients.

Flower

Flower [40] is an open-source Python library designed for Federated Learning. It offers a flexible and extensible architecture for constructing Federated Learning systems, supporting various Deep Learning frameworks such as TensorFlow, PyTorch and Keras. Flower simplifies the development and coordination of Federated Learning systems by providing high-level abstractions and tools for client-server communication, model aggregation, and coordination among participating clients.

- *Security/Privacy:* Flower supports secure aggregation and differential privacy (DP-FedAvg). Clients can establish SSL connection to the server just like in Flare.
- *Documentation*: The Flower documentation is extensive and exhibits a smooth transition from simpler concepts to more complex ones. It is well-organized and excels at assisting users of various expertise levels in onboarding to Flower. Moreover, a comprehensive API reference is available online.
- *User-Friendly Interface*: Flower provides a user-friendly interface that streamlines the setup and configuration of Federated Learning (FL) experiments.
- *Compatibility*: It is framework-agnostic and supports a wide range of Machine Learning frameworks (PyTorch, TensorFlow, Hugging Face Transformers, PyTorch Lightning, MXNet, scikit-learn, JAX, TFLite, fastai, Pandas and Numpy).
- *Learning curve*: Flower has a steeper learning curve compared to Flare. It provides a complete setup guide with well-explained tutorials and examples. The framework is simple and easy to extend with new server-side and client-side functionalities.
- *Customization*: Flower supports several aggregation strategies including FedAvg, FedOpt, FedProx, FedAdagrad, FedAdam and FedYogi. Federated Averaging is the default aggregation strategy. Custom aggregation strategies can also be implemented for specific use cases. The client API allows the developer to customize how each client in the federated network behaves.
- *Community Support*: Flower has active developer communities and several research papers in FL. They publish their code that can be seamlessly integrated into Flower.
- *Scalability*: Flower is scalable to several hundred clients, significantly more than in typical cross-silo Federated Learning settings used in SECURED.

Pybiscus [41], built on top of Flower, is a simple tool to perform Federated Learning on various models and datasets. It aims at automating as much as possible the FL pipeline, and allows adding virtually any kind of dataset and model.

Verdict

In summary, Flower is easier to start working with and adapts to a wide range of regular Machine Learning workflows. It offers several built-in security and privacy features, including Differential Privacy and Secure Aggregation. Flower is also easier to extend with new and customized functionalities, though it is less user-friendly than Flare. Both frameworks are open source and support necessary security and privacy functionalities, with sufficiently detailed documentation. The main decision criteria were to favor simplicity and faster learning with Flower, or a more comprehensive and production-ready user interface with Flare. Since SECURED, being a research and innovation project, is expected to develop and test several new security and privacy primitives that have not been integrated into any frameworks, the potentially faster and easier integration with Flower makes it the preferred choice for SECURED.

In more details, Flower [40] offers various aggregation strategies for the server. From accuracy point-of-view, FedAdagrad, FedAdam, FedAvg, FedXgbNnAvg, FedXgbBagging, FedXgbCyclic, FedAvgAndroid, FedAvgM, FedOpt, FedProx, FedYogi, and QFedAvg are supported. From robustness point-of-view, FedMedian, FedTrimmedAvg, Krum, and Bulyan are supported. From a privacy point-of-view (besides secure aggregation as in FaultTolerantFedAvg), DPFedAvgAdaptive and DPFedAvgFixed are supported. The overhead of these advanced mechanisms are benchmarked on the corresponding website and paper.

An initial implementation of some aggregation methods for outlier detection using Flower corresponding to UC 2 (telemetry) can be found in the internal project repository (`https://uva-hva.gitlab.host/secured/wp3/t3.2/telemetry.git`). We designed and implemented a robust scheme to detect sudden changes in the status of patients. A single Temporal Convolution Network (TCN) predicts the readings of five different sensors deployed on the patient's body. Anomaly is triggered when the differences between the predicated and actual values exceed a certain threshold. The training is also simulated in a federated setting with different number of clients having identically distributed training data. Results show that Federated Learning has negligible performance loss compared to the centralized training in this particular setting.

4.3 Risk Analysis

Risk analysis is a systematic process used to identify, assess, and prioritize potential risks or uncertainties that could negatively impact the success, objectives, or operations of the collaboration within Federated Learning. It enhances decision-making and facilitating communication and transparency among stakeholders. It also helps participants proactively address uncertainties, make informed decisions, and increase the likelihood of the collaborations' success.

Risk analysis involves evaluating the likelihood of risks occurring and their potential consequences to make informed decisions about how to manage or mitigate these risks effectively. In general, the risk analysis is composed of the following main steps, repeated until all Risks become acceptable:

1. Define the **context** of the analysis: the aim of ML and work flow of data collection
2. Define the perimeter of the analysis, i.e., the actors (**Risk Sources**) and the data involved (**Assets**).
3. Define the controls already in place and assess their mitigation level.
4. Define the associated Risks for each Risk Source (**Adversary goals**).
5. Define possible **Threats** for each Risk.
6. Apply further controls to mitigate Risks (**Mitigations**).

Within this document, for all relevant federated SECURED use cases, we rigorously go through the Assets, the possible corresponding Risks and their Sources, and detail the related feasibility and likelihoods. We also mention a couple of Threats that might realize the Risks, and finally suggest several mitigation strategies to provide a comprehensive overall picture. Before presenting the results of the risk analyses, we review the main concepts that are important for the analysis.

4.3.1 Attack Categories

In the realm of cybersecurity and Machine Learning, attacks can be broadly categorized along four edges:

i) who is the attacker (Attacker Identity),

ii) what are its capabilities (Attacker Capability),

iii) what information is available to the attacker (Attacker Information),

iv) when does the attack take place (Attack Time).

4.3.1.1 Attacker Identity

An attack can be carried out by **insiders** (e.g., the aggregator server or any client) or **outsiders** (e.g., an eavesdropper on the communication channel between clients and the server or among users of the final model). The insider attacker is more capable, especially when executed by the server. Relative to the clients, the source of an attack can be a **single client** or **multiple clients** launching a coordinated attack.

4.3.1.2 Attacker Capability

Concerning the access to modifications each user has, the two most common threat models in Federated Learning are **honest but curious** (or semi-honest) and **malicious**. The former is a passive, while the latter is an active attack, depending on their engagement level and objectives in exploiting vulnerabilities or attacking systems.

Passive attacker is one who monitors or observes the system or network without directly altering or affecting its operation. They may eavesdrop, collect data, or analyze network traffic to gather information, gain insights, or identify potential weaknesses for future exploitation. Its objective is to gather sensitive information, such as credentials, personal data, or system configurations, without being detected. They typically try to remain undetected while focusing on reconnaissance and information gathering.

Active attacker takes more direct and intrusive actions within the system or network. They may attempt to manipulate, disrupt, or interfere with the normal functioning of systems or data. Their aim is to cause damage, steal sensitive information, disrupt operations, or compromise the integrity of systems.

4.3.1.3 Attacker Information

Another aspect to classify the attacker is the so-called background knowledge, which could refer to their level of understanding, expertise, and familiarity with various aspects of systems, networks, vulnerabilities, and security measures. This knowledge significantly influences an attacker's capability to launch successful attacks and evade detection. Concerning Machine Learning, the attacker may need information about the training data, and depending on its extent, there are many categories [42]. For example, the attacker could access to a portion of the training data with or without labels, or know only about the statistical properties (e.g., its distribution).

The different access types that are available to the attacker also falls here. The terms **white box** and **black box** can be used to describe different levels of access, knowledge, or control that attackers may have over the Federated Learning system and its components.

- A **white box** attacker has detailed knowledge or access to the internal workings, algorithms, or models used in the Federated Learning process. This level of access allows the attacker to fully understand the architecture, parameters, or even the training data used in the distributed model. Such an attacker may perform targeted attacks, potentially compromising the model's integrity and harming the overall learning

process. They may also attempt to reverse-engineer the model or uncover sensitive information from the Federated Learning setup.

- A **black box** attacker lacks detailed knowledge or access to the internal workings of the federated learning process. They only interact with the system externally, without insights into the model's architecture, individual device data, or model updates. Black box attackers aim to disrupt or compromise the Federated Learning process without detailed knowledge of the model or the individual device data. Their focus is on attacking the communication or transmission aspects of the Federated Learning setup.

4.3.1.4 Attack Time

Considering which stage of the process the attacker has access to, the attack can happen at **training time** and at **inference time**. In the former, the attacker attempts to learn, influence, or corrupt the FL model itself, for instance by running an active data or model poisoning attacks. In the latter, the attacker targets the gradients (individual or aggregated) to uncover sensitive details about the underlying datasets of other clients.

4.3.2 Threats and defense techniques: Security and Privacy aspects

It is important to assess and address threats (detailed in D4.1) in the design and implementation of Federated Learning systems to ensure robust security protections.

Threats

Concerning **security**, these threats include malicious behaviors [43], such as Sybil Attack [44] (an adversary creates multiple fake participants to disrupt or manipulate the collaborative learning process), Byzantine Attack [45] (involves participants behaving maliciously by providing incorrect or misleading updates to the federated learning system), Adversarial Attack [46] (specially crafted input data that are intentionally designed to cause misclassification or incorrect behavior in Machine Learning models), Poisoning Attack [47] (malicious participants intentionally inject malicious updates into the learning process), and Backdoor Attack [48] (hidden vulnerability or malicious behavior intentionally inserted into a model by an adversary).

On the **privacy** side, Federated Learning may expose sensitive data [49]. Unauthorized access to participant data or the leakage of private information during the model training or aggregation process can lead to privacy breaches and compromises. Some corresponding attacks are Model inversion [50] (exploits the outputs of a Machine Learning model to infer sensitive information about individuals by reconstructing attributes of the datasets used for training), Membership inference [51] (infer membership information by analyzing the model's outputs for the target data samples), Property inference [52] (aims to infer sensitive or confidential properties of the training data), Reconstruction attacks [53] (reconstruct the original training data by leveraging the trained model and its outputs, parameters, or gradients), and (Hyper)parameter inference [54] (aims to infer sensitive or confidential information about the parameters of a Machine Learning model).

Defense Techniques

Ensuring the integrity of the Federated Learning system and implementing robust defense mechanisms to detect and mitigate **security** attacks is crucial for maintaining the security and reliability of the collaborative learning process. Several mitigation strategies can be employed to counter these threats and maintain the integrity and privacy of the Federated Learning process [55]. Such mechanisms are Byzantine Resilience [56] (the ability of the system to withstand and mitigate malicious behaviors or attacks from participants), Adversarial Robustness [57] (the ability of the system to withstand and mitigate adversarial attacks, particularly those involving adversarial examples), and Certified Defenses [58] (provides a formal guarantee or certification of the model's robustness).

Concerning **privacy** safeguards, the sharing, and aggregation of FL data can pose privacy risks. There are a handful of privacy-preserving techniques that either partly mitigate or entirely prevent the above-listed attacks, such as Differential Privacy [59] (inject noise to the training process to prevent the identification of individual data samples), Secure Aggregation [60] (allow participants to aggregate their local model updates without revealing their individual contributions).

4.3.3 Results for the Use Cases

In this subsection we scrutinize each SECURED Use Case, i.e., we highlight the actors, the assets, and the adversarial risks and threats considering the previously mentioned attacker dimensions. Furthermore, we list possible mitigation strategies. At the end of this section, we provide a table where risks are summarized for all UCs with their specifications.

4.3.3.1 Use Case 1: Real-Time tumor classification Development

This pilot aims to achieve high-accuracy, real-time, and low-latency performance for hospital tumor classification. In more detail, it uses real-time ultrasound data of the brain to enhance an earlier Magnetic Resonance scan record. As a corollary, the data can be used for identifying the patient (as one can infer the face of the patient) from Magnetic Resonance scan data, so besides its medical nature, there is a second reason why it must be protected. On the other hand, the adjustment process (3D transformation) involves only classical algorithms (no Machine Learning is involved). Since Federated Learning is not involved in this use case, **no risk analysis is necessary** for Federated Learning aspects.

4.3.3.2 Use Case 2: Telemonitoring for children Development

Context. This pilot aims to achieve live automated real-time remote monitoring for children by taking advantage of all the clinical data received from patients, such as blood pressure, heart rate, oxygen saturation, and more. After being passed through the anonymization processes and made certain that privacy is ensured by the benchmarking tools, they are fed to the federated system to train ML algorithms.

Risk Sources. The actors are healthcare institutions such as hospitals and the aggregator server (in case not a decentralized serverless mechanism is utilized [61]). The data originates from children by remote sensing medical devices (which is more trustworthy than an average IoT device). The device relays the measurements in an encrypted form to the hospital, which trains a local model, which are aggregated by the InnoHub.

Assets. The assets for this Use Case are the medical data used for training and the trained model. The types of data are basic medical measurements, such as blood pressure. The sensors generating the data are equipped on children, who enjoy elevated protection in the eyes of the law, including the privacy point of view. Therefore, this data is highly sensitive and personal.

Besides its intellectual property, training the model costs time, resources, and money, so to avoid harming the institutions participating in the training, the federated model should be protected. In turn, the model might contain information about exact training samples.

Finally, during training, the intermediate models are accessed by both the hospitals and the aggregator server (if exists). The reason for the former is to adjust the learning parameters of the model if it performs suboptimally. Yet, having access to the model updates (or gradients) opens the door to a more powerful family of privacy attacks, especially if those values are not aggregated, but correspond to specific participants.

Adversary goals. We do not consider intentional data poisoning among the threats, as the data subjects have neither motivation nor capability to tamper with the safety-hardened medical devices. Similarly, the medical institutes have no incentives to intentionally manipulate the collected data, as the corresponding reputation loss (if noticed) would make such an act catastrophically undesired. On the contrary, the data could unintentionally be corrupted (e.g., remote medical device malfunction, incorrect human usage or placement, etc.) or plainly of

low quality (e.g., missing features, non-representative samples, etc.).

Evasion attacks (i.e., modifying the input of the model at inference time to cause misclassification) are also not a real concern. Firstly, there is not much motivation of the patients (or the user in general) to fool the system, as it serves his/her own interest. Secondly, infiltrating the remote sensing medical device is not an easy task, so the effort might not worth the desired result.

The most probable attacks concerning this Use Case are passive (i.e., the attackers are honest but curious or semi-honest). The attacker aims to infer as much information as possible about the assets. For instance, the medical institutions participating in federated learning could infer sensitive details about the other participants' data. This risk is even more realistic if the attacker has access to the intermediate models of the training (such as the aggregator server, if applicable). Another attacker might be third parties with various access to the trained model (e.g., white or black box). Such malicious actors could execute data-stealing and model-stealing attacks.

Threats. As a summary of the above risks, the possible threat scenarios are enlisted below.

- During federated learning training, participants aim to deduce information about each other's datasets. This could occur through various means, such as a reconstruction attack [53], membership inference attack [62], or property inference attack [63].
- In the training phase, a malicious attacker, equipped with access to the aggregator server, might attempt to extract information about datasets through the same techniques mentioned above, i.e., reconstruction, membership inference, and property inference attacks.
- A third party possessing white box access to the trained model, for example, having purchased it, seeks to infer details about the data used for training. This inference can be achieved through methods like the reconstruction attack proposed in [64].
- Another third party, with black box access to the trained model, aims to deduce information such as weights, hyperparameters, etc., about the model. This objective can be realized through a model stealing attack proposed in [65].
- The presence of bad data from certain federated learning participants can distort and diminish the performance of the shared model. This situation is particularly relevant in scenarios where the quality of datasets is questionable [66], significantly impacting the final models [67].

Mitigations. There are several countermeasures to tackle any of these threats. However, most existing techniques affect other aspects of the Federated Learning model: for example, privacy-preserving techniques could make the data quality measurements harder or even decrease the model's performance (training time & accuracy). Consequently, the methods enlisted below should be applied with care.

- Data Privacy Attacks using the model updates: Secure Aggregation [60] plays a crucial role by concealing only the gradients. This ensures that even if an attacker can glean sensitive information from the aggregated model, the connection to specific participants remains obscured. Additionally, Differential Privacy [59] provides a means to restrict the extent to which leaked information can be realized, preventing undue attribution.
- Dishonest Server: While Secure Aggregation (possibly enhanced with shared noise to hide the aggregated model from the server) and Differential Privacy are the predominant privacy-enhancing technologies to thwart such privacy attacks, other methods, albeit without formal guarantees, exist. These include techniques such as regularization [68] or compression [69].
- Data Privacy Attacks using the final model: Differential Privacy serves as a mitigating factor. However, other applicable techniques increase the gap between the training data and the trained final model. Examples include Model Distillation [70] and PATE [71].
- Model stealing: Differential Privacy also diminishes the success of such attempts. However, alternative techniques can complement it by complicating the relationship between the model and predictions.

Ensemble Models [72], Model Stacking [73], and similar approaches are among these effective countermeasures.

- Bad data: Evaluating the quality of datasets in Federated Learning involves various methods. The Shapley value [74] and its approximations stand out as flagship definitions, but several other methods exist. Care must be taken though, as these alternatives vary significantly in terms of complexity, applicability, and inherent properties [75].

4.3.3.3 Use Case 3: Synthetic data generation for education

Context. Data for training health personnel is limited by access due to privacy rights and the size of available data sets from actual cases. Using the SECURED architecture, this task expects to generate a large amount of GDPR-compliant synthetic data from accurate statistical data from the consortium health institutes. Such a process must protect individual data privacy, which will be realized via the SECURED anonymization tools. Artificial data will be used in the education of medical doctors to show the progression of a particular disease or lesion (e.g., a tumor), demonstrate intervention cases (e.g., change in fetal heartbeat signal) or teach statistical methods (e.g., for work flow management plans).

Risk Sources. The actors are healthcare institutions such as hospitals (with access to the medical data) and educational institutes such as medical universities (with access to the medical data as well and to the final trained model).

Assets. The corresponding assets for this Use Case are the medical data used for training and the trained generative model. The utilized data types are cardiotocographic, pathologic, and radiologic data. The cardiotocographic data consists of two time series measuring the fetal heartbeat and uterine contractions during pregnancy and labor. Concerning pathologic data (often tissue samples from a biopsy), cell-level images of the (cancerous) colon are utilized. Finally, the exact radiologic data are mammographic, i.e., x-ray images focusing on (cancerous) breast. All data are accompanied by clinical metadata (e.g., age, time of intervention, symptom classification code). Moreover, besides its medical nature, all of these data could be considered sensitive and personally identifiable information, as they could be unique to the patients. Hence, it must be protected.

The trained generative model should also be handled with care due to its confidentiality. Training such models need significant effort in terms of time and computational resources (which ultimately translates into invested money), so model stealing or model extraction would harm the medical institutes participating in the training. Such a model carries intellectual property as well as information about the training samples, which should be protected.

Adversary goals. Medical institutes have no incentives to intentionally tamper with the collected genuine data, so data poisoning poses no threat. The main reason is the inevitable reputation splash back and the corresponding public shame resulting from such an act if it comes to light. Hence, such misdeeds are not likely. Although an insider attacker within the institution could launch such attacks, however, this analysis does not consider such attackers, as are not specific to Federated Learning, and they pose an unacceptable risk for the patients anyway. On the other hand, the data could unintentionally be corrupted (e.g., medical recording device malfunction, human annotation error, etc.) or plainly of low quality (e.g., missing features, non-representative samples, etc.).

Evasion attacks are also not applicable in relation with generative models. The reasonable attacks concerning this use case are passive (i.e., the attackers are honest but curious), i.e., aims to infer as much information as possible about the assets. For instance, the medical institutions participating in federated learning could infer sensitive details about the other participants' data. A similar attack can be executed by accessing the aggregator server (if employed). Standard privacy attacks are acquiring additional information from the model about the underlying training data. Besides the participants, third parties with access to the trained model (e.g., the black box model is sold to them) could also execute such a data-stealing attack. Additionally, model stealing attacks are also plausible, as these third parties with access to the model as a service might reverse-engineer the model (i.e., its parameters) from the outputs.

Threats. As a summary of the above, the possible risk scenarios are enlisted below. Additionally, we included

some relevant threats which could realize these risks.

- During training, the FL participants wish to infer information about other participant's datasets. This can be realized for instance via the reconstruction attack proposed in [53], or via the membership inference attack proposed in [62], or even via the property inference attack proposed in [63].
- During training, a malicious attacker with access to the aggregator server could try to infer information about the datasets via the previously mentioned reconstruction, membership inference, or property inference attack.
- A third party with a white box access to the trained model (e.g., bought it) wish to infer information about the data used for training. This can be realized for instance via the reconstruction attack proposed in [64].
- A third party with a black box access to the trained model wish to infer information (e.g., its weights, hyperparameters, etc.) about the model. This can be realized for instance via the model stealing attack proposed in [65].
- The bad data of some FL participants distort and decrease the performance of the shared model. This can happen, as for many datasets in the wild the quality is questionable [76], which would affect the final models severely [67].

Mitigations. There are various countermeasures to tackle any of these threats. However, most existing techniques affect other aspects of the FL model. For instance, applying a privacy-preserving mechanism could make the data quality determination harder, increase (or create) the fairness gap, and negatively affect the model's performance (longer training time & poorer accuracy) too. Consequently, they should be applied with care, and a balancing exercise must be done. A non-comprehensive mitigation list for the highlighted threats is below.

- Data Privacy Attacks using the gradients: Secure Aggregation [60] hides the individual model updates, so even if a participant could infer some sensitive information from the aggregated model, the attacker would still not be able to connect that information to any of the participants. Besides preventing the attribution of the leaked information, via Differential Privacy [59] it is also possible to limit to what extent can the leakage be materialized.
- Dishonest Server: Secure Aggregation hides individual model updates, but the aggregated gradients still could leak information. Masking the aggregated gradient from the aggregator is also possible via SMPC. Besides, Differential Privacy applies here as well, just as compression [69] and regularization [68].
- Data Privacy Attacks using the final model: Differential Privacy would also decrease this attack's success, but other techniques are also applicable which increase the distance between the training data and the trained final model, e.g., Model Distillation [70], PATE architecture [71], etc.
- Model stealing: Differential Privacy would also decrease this attack's success, but other techniques are also applicable which complicate the relationship between the model and the prediction, e.g., Ensemble Models [72], Model Stacking [73], etc.
- Bad Data: A handful of methods exists to evaluate the quality of datasets in FL. The flagship definition is the Shapley value [74] and its approximations, but others exists as well, and they heavily vary based on their complexity, applicability, and properties [75].

4.3.3.4 Use Case 4: Access to genomic data Development

Context. The goal of this task is to test the use of the SECURED framework to analyze protected genetic data from approximately a million cancer patients, opening up the opportunity of doing biomedical research at an unprecedented scale. However, there are two major bottlenecks to exploiting this data, the actual access to it from external partners due to privacy and legal issues and the time required to perform the analysis. This task

will utilize Federated Learning to train models on all existing genetic data as well as synthetic data, produced by the initial genomic data as basis, maintaining patient privacy while increasing accuracy and reducing time.

Risk Sources. The identified actors in this Use Case are the biobanks, the participating healthcare institutes, and the aggregator server. The biobanks have a strict access policy, meaning only aggregated query results could leave the institution's hardware. At the same time, individual records (i.e., genomic sequences or EHR data) must remain within the premises.

Assets. The corresponding assets for this use case are the genomic data used for training and the trained model. The utilized genetic data is tabular, and either the entire dataset (with slightly over 2000 features) or a compressed version (e.g., a scalar value representing a risk score calculated by some function) is used. Furthermore, this data with genetic origin might be complemented with environmental data (e.g., details contained in the EHR, represented in around 20 features) to enhance further the final prediction, which is a risk factor (i.e., chance) for approximately ten cancer types. Genetic data is interdependent, meaning that your genetic information — which is sensitive personal data due to its medical nature — contains details about others (e.g., your family) as well. If leaked, such data could compromise the privacy of the corresponding data subject and many generations (even in the future).

Moreover, the trained model could be stolen or extracted, harming the health institute that financed the training, as accessing the biobank's data is costly. Training a computationally not demanding model would already require heavy investment. Moreover, the model could carry sensitive information of an entire nation. Hence, unauthorized access to the model would lead to sensitive data disclosure. The model's prediction must also encompass an explanation of the final decision to ease the medical experts' use of this supplementary system. The envisioned architecture of the model is linear regression or random forest, which enables such explanations by default without much intervention. On the other hand, such extra information easily enables stronger privacy attacks.

Since model sharing will be studied in UC 2, at the moment it is not considered for this use case. Because of this, in this use case, besides the server, only the participants have access to the model. These parties either already have access to the biobank's data or could access it with less effort than reverse engineering the trained model. Consequently, no motivation exists to execute any of the mentioned model-related attacks.

Adversary goals. We can assume the biobanks are trusted, so the genetic data is clean, and no malicious manipulation can happen. As such, data poisoning poses no threat. However, not all data is adequate for all tasks: there might be specific risk calculations where the data from a particular biobank is more suitable than others or vice versa. Besides, evasion attacks might be feasible once the model is shared with third parties in some form (e.g., inference/prediction-as-a-service). However, as mentioned already, this is out of scope.

A reasonable attack concerning this use case is passive (i.e., the attackers are semi-honest, and the attack aims to infer as much information as possible about the assets). For instance, a malicious actor accessing the aggregator server could infer sensitive details about the biobanks' genetic datasets or the complementary EHR data.

Threats. The possible risk scenarios are listed below as a summary of the above. Additionally, we included some relevant threats that could realize these risks.

- The inadequate data of some biobanks (concerning particular tasks) or corrupted EHR data (due to human input, etc.) distort and decrease the performance of the shared model. If this happens, it will affect the final models severely [67].
- During training, a malicious attacker with access to the aggregated model could try to infer information about the datasets via the reconstruction [53], membership inference [62], or property inference [63] attack.

Mitigations. There are various countermeasures to tackle any of these threats. However, they should be applied with care. A non-comprehensive mitigation list for the highlighted threats is below.

- Bad Data: A handful of methods exist to evaluate the quality of datasets with a particular task in mind

for the participants of FL. The flagship definitions are the Shapley value [74] and its approximations, but others exist as well, and they heavily vary based on their complexity, applicability, and properties [75].

- Model reverses engineering: the defense techniques (such as Secure Aggregation and Differential Privacy) mentioned in the other use cases also apply here. Moreover, due to the nature of the data, the training takes place at the biobanks. Furthermore, due to the nature of the model (ensemble), it could be sufficient to aggregate the predictions instead of the models, providing a privacy boost. Finally, due to the characteristics of the shared data, it is feasible to rely on cryptography instead of an aggregator server, which further limits the attack surface.

4.4 Conclusions

In the context of the SECURED project, Federated Learning enables privacy preservation of health data by maintaining distributed data used in the training of a centralized model locally. The selection criteria for the choice of the FL library, Flower, are derived from performing risk analysis. A compact overview of the results is in Table 13. This approach will help in selecting the FL parameters to be used in the SECURED Library as well.

Use Case (Federated Learning)	Risk Source	Access				Privacy		Robustness		Fairness	
		Final model (query access)	Final model (full access)	Model updates	Data	Data extraction	Model extraction	Training time	Inference time	Data bias	Data quality
Telemetry	Remote Sensor				X			X		?	X
	Hospitals			X	X	X		X			
	Aggregator			X		X	?	X			
	Third party	X	X			X			X		
	Third party	X					X		X		
Synthetic data generation	Hospitals				X			X		?	X
	University			X	X	X		X			
	Aggregator			X		X	?	X			
	Third party	X	X			X			X		
	Third party	X					X		X		
Genomic	Biobank			X	X			X		?	X
	Health Institute			X	X	X		X			
	Aggregator			X		X	?	X			

Table 13 – Identified risks for the use cases.

5 Handling Bias and Fairness in AI

5.1 Overview of Concepts

AI and ML tend to reproduce and amplify bias from training data. Since we need to mitigate this problem, we use Fairness approaches to insure unbiased AI. State of the art regarding unbiased techniques is presented in deliverable D4.1. For completeness, in this section we summarize the most important concepts described there. In the ML process, bias may come from different origins:

- Bias coming from data: two biases are particularly important: representation and sampling biases due to non-representative samples in the different entities participating in the federation;
- Bias coming from the algorithms: evaluation bias occurring during the algorithm evaluation with a bias present in dataset used for evaluation ;
- Bias coming from the users: population bias arises when statistics, demographics, representatives, and user characteristics are different in the user population of the platform compared to the original target population.

The objective of Task 3.3 is to automatically detect these biases and to develop methods for mitigation of such biases, during the three phases of a ML process:

- During pre-processing the objective is to enhance the fairness of models by rectifying training data;
- During in-processing, we customize ML/DL algorithms to directly train fair models, with the help of an adversarial network.
- During post-processing, approaches will be developed to revise the prediction scores of a Machine Learning model after training to make predictions fairer.

We propose fairness techniques methods for two settings:

- In-processing methods for classification and regression tasks;
- Fairness for generative models.

For the first setting, relevant metrics are extensively described in Deliverable D4.1. Compared to that, here we extended the metrics for regression tasks and we selected an in-processing method that is suitable for applications in the SECURED project.

The second setting comes from the needs of Use Case 3. We describe fairness metrics for generative models and describe two methods for such problems: one based on importance weighting and the other based on transfer learning. Since this topic was not previously address in Deliverable D4.1, we explain it here with the due level of details.

5.2 Background libraries

Table 14 summarizes the selected libraries dedicated to fairness, discussed in depth in Deliverable D4.1. Documentation and tutorials existing for these libraries include examples on their use in classification task and regression task, but, at the moment of the writing, the documentation available for generative model is quite scarce. Because of this lack of available documentation, these libraries are currently not the best choice for UC 3. The libraries characteristics are summarized in Table 14.

For classification, existing libraries implementing in-processing approaches are not suitable for complex use case such as the ones in the SECURED project and in the medical domain in general, even if the target use cases are simplified removing the need of fairness.

Considering the pre-processing approaches instead, the reweighing technique, implemented e.g. in AIF360, will be used as benchmark for complex Use Cases in SECURED. Furthermore, enhancing the state, there are other and more promising algorithms for the regression and classification tasks will be implemented and tested within the SECURED use case.

Module	Language	Methods	Data	ML Task	License
AIF360[77]	Python/R	State-of-the Art metrics (Disparate Impact, Equal Opportunity Difference, etc.). Ten state-of-the-art mitigation algorithms (reweighing, reject option classification, etc.)	Tabular	Classification	Apache 2.0
fairlearn[78]	Python	Classical metrics (Demographic Parity, equalized odds, etc.). Adversarial, reduction, preprocessing (correlation remover) and post-processing (threshold optimize)	Tabular	Classification and Regression	MIT
fairness[79]	Python	State-of-the Art mitigation methods by Calders[80], Feldman[81]	Tabular	Classification	Apache 2.0
fairness-compass[82]	Java	Classical fairness metrics (Equalized odds, calibration, etc.)	Tabular	Classification	Apache 2.0

Table 14 – Existing libraries for Fairness

5.2.1 Testing metrics for regression tasks

Metrics for classification have been described in Deliverable 4.1, therefore here we focus on the progresses made in the last months. Regression tasks need specific metrics to measure fairness. [83] proposes a method to measure fairness defined by Equation (2).

$$\mathbb{E}(f(x_i)|x_i \in A) - \mathbb{E}(f(x_i)|x_i \in B), \tag{2}$$

where:

- f is the regressor. For instance, in the use case 2, it is a tool to predict one quantitative value of the monitored children;
- $\mathbb{E}$ is the notation of the mathematical expected value;
- A and B the two groups, e.g. in use case 2 children over and under an age.

There are several metrics to measure Fairness for regression tasks. One option available for us is to extend the metric to measure fairness of the model performance, as in Equation (3):

$$\mathbb{E}(f(x_i)|x_i \in A, Y) - \mathbb{E}(f(x_i)|x_i \in B, Y), \tag{3}$$

where Y is the ground truth for the quantitative value. Here we are measuring the ability of the AI model to predict with the same level of performance, regardless of the target category (for instance, in Use Case 2, this metric will allow measuring the ability of the AI model when applied to the children).

Fairness can also be calculated according to the AI model calibration, e.g., by considering the Equation (4):

$$\mathbb{E}(Y|x_i \in A, f(x_i)) - \mathbb{E}(Y|x_i \in B, f(x_i)). \tag{4}$$

Similar extensions are applicable to the individual fairness metrics proposed in D4.1 for the regression task.

Steinberg et al. [84] propose to measure fairness of a regressor by using probabilistic classification. The authors introduce approximations of independence, separation and sufficiency criteria by observing that they factorize

as ratios of different conditional probabilities of the protected attributes.They train Machine Learning classifiers, distinct from the predictor, as a mechanism to estimate these probabilities from the data.

5.2.2 Fair classifier based on triplet loss

Metric learning [85] has been used to achieve individual fairness [86], whose objective is that two individuals differing only by the sensitive attribute are treated in the same way by the model. Metric learning aims at automatically constructing task-specific distance metrics from (potentially weak) supervised data, in a Machine Learning manner. Triplet learning [87] can be used in this context. Triplet loss have been used for fairness, e.g., by [88] and [89]. Since they have been successfully used in applications with large amount of data, they appear to be suitable also for the use cases of SECURED.

Consider a binary supervised classification task. $x = (x_1, \ldots, x_K, s)$ is a feature vector and y a binary label to predict. $s \in \{0, 1\}$ a sensitive attribute. The objective is to learn an embedder E into a d-dimensional fair embedding space. $z = E(x)$ denotes the embedding representation. Given a triplet $((x^a, y^a), (x^+, y^+), (x^-, y^-))$ with an anchor, a positive and a negative. (z^a, z^+, z^-) is the corresponding embedding. Let $\alpha \in \mathcal{R}^+$ the triplet margin and assuming the use of the Euclidean distance. The Triplet loss is given by Equation (5).

$$l_{triplet} = \max(||z^a - z^+||_2^2 - ||z^a - z^-||_2^2 + \alpha, 0). \tag{5}$$

Triplet Loss encourages dissimilar pairs to be distant from any similar pair by at least a certain margin value.

The two followed triplet selection methods are candidate for potential bias mitigation selection:

- the positive is a copy of the anchor but with the sensitive feature flipped, thus $x^+ = (x_1, \ldots, x_K, \bar{s})$ and x^- is selected randomly amongst samples for which $y^- \neq y^a$
- the positive is a copy of the anchor but with the sensitive feature flipped, thus $x^+ = (x_1, \ldots, x_K, \bar{s})$ and x^- is selected randomly among all samples except the anchor

Gornet et al. [88] uses the triplet loss to train a Convolutional neural network dedicated to facial recognition. Triplet loss has proved its utility in the case of unbalanced dataset (e.g., [90]), which are relevant for SECURED, for instance, in UC 3, where image generation can be used with unbalanced dataset. In that case, to evaluate both the quality of the images generated and the fairness of the generator, we need a classifier as fair as possible (see Section 5.3.1).

5.3 Fairness for generative models

5.3.1 Metrics

For a Generative model, we define fairness by the ability of the model to generate data while preserving the equal repartition according to the sensitive attribute. A generative model is fair for a sensitive attribute if, for each different value of the sensitive attribute, the model has the same probability of generating data. For instance, an image generator is fair for gender if the generator has the same probability of generating an image of each gender. For the medical domain, it is problematic when considering cardiovascular disease, where there is some historical bias against women who are underrepresented in the studies [91].

In many cases, the sensitive attribute, denoted by s, is a latent attribute and can only be inferred from the data generated (e.g., the gender of someone in an image can be inferred, but is not clearly labeled). For this reason, we need metrics based on the ability to infer the sensitive attribute according to the data generated. For that, we need a classifier C able to infer the sensitive attribute according to the data generated. For an observation x, the classifier C produces a soft output $C(x)$. We can compute a discrepancy measure $D(.,.)$ between $\mathbb{E}_{x \sim q_\theta}(C(x))$, with q_θ the distribution imposed by the generator G_θ and $x \sim q_\theta$ an observation sample

from q_θ, and the uniform probability vector $\bar{p} = [1/k, \ldots, 1/k]$, with k the number of different classes of the sensitive attribute. The fairness discrepancy (FD) [92] is given by Equation (6). As the FD is dependent of a classifier C, we denote it in Equation (6) by FD_C.

$$FD_C(G_\theta) = ||\bar{p} - \mathbb{E}_{x \sim q_\theta}(C(x))||_2. \tag{6}$$

If $FD_C(G_\theta) = 0$, then G_θ is considered to be perfectly fair for the sensitive attribute. [93] proposes similar metrics while considering in addition whether the contextual features are preserved. However, both FD_C and their measure are highly dependent on the classifier C. The C can make errors in practice. Even with a perfectly fair G_θ, $FD_C(G_\theta)$ can be not equal to zero.

If we consider Use Case 3 about image generation for educational purpose, e.g., of X-ray lung, we can transpose the different element in the following way:

- C is a classifier that predicts that an X-ray lung image corresponds or not to a sick patient;
- q_θ is the data distribution learned by a generator G_θ trained to generated X-ray lung image;
- By default, in this case, $\bar{p} = [1/2, 1/2]$ there are two classes (sick and not sick). In this case, that means that we want to generate as many images of a sick patient as of a healthy person. We can adapt this value according to that we want to generate more data of one category versus the other.

As it can be seen, the sensitive attributes are to be known to compute these metrics. These metrics have to be defined together with the experimental protocol before the start of the experiments (see Figure 2 of Deliverable D4.1 for the lifecycle of an AI system with the stages where bias can occur). Using the $||.||_2$ is a common approach, but some other distances (e.g., $||.||_1$) can be used too.

5.3.2 Methods

5.3.2.1 Importance Weighting

[94] works on generative adversarial network [95] and considers a case where a large data set D_{bias} is available, but is biased against a sensitive attribute (i.e., one category of the sensitive attribute is underrepresented). It assumes that a small data set D_{ref} is equilibrated regarding the sensitive attribute. They propose to reweight the observation of D_{bias} by using the Algorithm 1. In the Algorithm, $Y = 0$ is used to indicate that the observation-sensitive attribute is in the favored category, $Y = 1$ is used otherwise. An example of application of the first stage of Algorithm 1 on the SECURED project is the classification of chest images, when we have access to two chest image datasets: one being a large chest image dataset that includes several people having a disease and only few healthy people and the other dataset being much smaller but balanced chest image dataset. The classifier given by Algorithm 1 is trained on the sick data from the large dataset and on the healthy data from the second dataset, ultimately learning to distinguish on the basis of a chest image whether the person is sick or not.

This approach suffers from several drawbacks. First, we need to access both a large biased dataset D_{bias} and a reference without bias dataset D_{ref}. Second, the importance of weighting will be dependent on a classifier. The weights can potentially change a lot according to the classifier that is trained. Moreover, this classifier will be learning on an unbalanced dataset. However, this classifier is easy to be implemented and offers a benchmark for other classifiers.

When applied to use case 3 on lung X-ray image generation with an under-representation of healthy person versus sick patient, it is translated as following:

- D_{bias} is a large lung X-ray dataset where sick patients are strongly over-represented compared to healthy patients;

Algorithm 1 Importance weighting for generative model

1. **Estimation of weight importance:**
   (a) Learn binary classifier c for distinguishing $(D_{bias}, Y = 0)$ versus $(D_{ref}, Y = 1)$;
   (b) Compute the importance weight $\hat{w}(x) = \frac{c(Y=1|x)}{c(Y=0|x)}$ for all $x \in D_{bias}$;
   (c) For all $x \in D_{ref}$, assign $\hat{w}(x) = 1$.
2. **Weight GAN training:**
   (a) Initialize the model parameter θ at random and set the full dataset $D = D_{bias} \cup D_{ref}$;
   (b) while training do:
      i. Sample a batch of points B from D at random;
      ii. Set the loss $L(\theta; D) = \frac{1}{|B|} \sum_{x_i \in B} \hat{w}(x_i) l(x_i, \theta)$;
      iii. Estimate the gradients and update the model parameters according to the optimization strategy.

- D_{ref} is a small lung X-ray dataset where sick patients and healthy patients are more or less equally represented;
- c is a classifier that has been trained to distinguish between sick and healthy patients;
- The weight GAN is the final generator that will generate new lung X-ray images of sick and healthy patients. This generator built in this way is expected to generate a higher proportion of healthy people than a generator based on D_{bias} alone, and to generate data of better quality than one produced using a generator trained only on D_{ref}.

5.3.2.2 Fair Generative models based on Transfer Learning

Generative model are data costly. Working on the large dataset, even biased, allows pre-training a model able to extract important features from data. Transfer Learning allows transferring this information on model fine-tuned on a smaller unbiased dataset.

[96] proposes an approach based on Transfer Learning [97] to ensure fairness in adversarial generative models [95]. Transfer learning aims at improving the performance of target learners on target domains by transferring the knowledge contained in different but related source domains. Neural networks have achieved significant success in many areas including classification, regression and clustering. However, as their learning depends on data that has been seen during the training phase, these methods work well under a common assumption: the training and test data are drawn from the same feature space and the same distribution. When the distribution changes, most of the models have to be rebuilt using new training data. In many real-world applications, it is expensive or impossible to re-collect the needed training data and rebuild the models. Moreover, most models have to be trained again when a new task slightly differs from the original task, even though the knowledge and skills learned in previous tasks could be used for novel tasks. Transfer learning allows the domains, tasks, and distributions used in training and testing to be different (see Figure 11).

[96] proposes two approaches based on Transfer Learning to achieve Fair Generative Adversarial network, given by Algorithms 2 and 3.

Algorithm 2 Fair Transfer Learning (FairTL), reformulated from [96].

1. Pre-train a generative model using the large, biased dataset;
2. Fine-tune by updating all the layers of the generator and the discriminator.

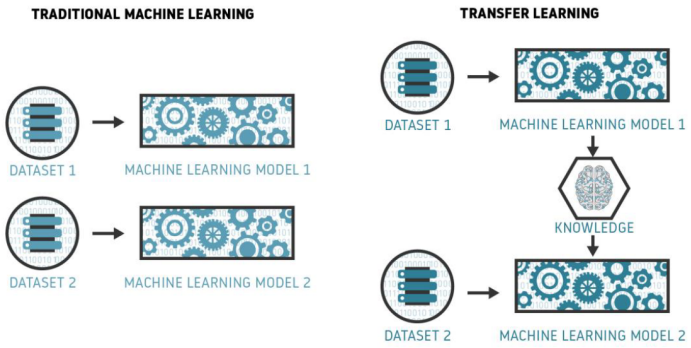


Figure 11 – Training process for traditional Machine Learning models and transfer learning.

Algorithm 3 Fair Transfer Learning ++ (FairTL++), a refinement of FairTL, reformulated from [96]

1. Pre-train a generator and a discriminator using the large biased dataset;
2. Transfer Learning on the reference dataset
   (a) First stage of Transfer Learning on reference dataset:
      - Frozen first layers of the discriminator and update the other layers of the discriminator. Update layers of the generator.
      - The network has two discriminators: the one that is currently updated and the discriminator of the pre-trained generative model whose layers are frozen;
      - The loss of the discriminator fine-tuned is regularized by the discriminator of the pre-trained models.
   (b) Second stage of Transfer Learning on the reference dataset :
      - Update all the layers of the discriminator and the generator
      - The networks have two discriminators: the one that is currently updated and the discriminator of the pre-trained generative model whose layers are frozen;
      - The loss of the discriminator fine-tuned is regularized by the discriminator of the pre-trained models.

For Algorithm 2, first a Generative Adversarial Network is trained on a biased large dataset, and then is fine-tuned on a reference dataset without bias. Due to the small size of the reference dataset, fine-tuning is susceptible to mode collapse. Mode collapse is a common problem when training GAN[98]. When training a GAN, there are two objectives:

- The generator can reliably generate data that fools the discriminator;
- The generator generates data samples that are as diverse as the distribution of real-world data.

Mode collapse happens when the generated samples are very similar or even identical. Algorithm 3 proposes two solutions to avoid mode collapse.

The first solution consists of using linear-probing before fine-tuning [99]. In linear-probing, a classifier head is updated while freezing lower layers. [99] demonstrates that when a classifier is adapted to a new task, it can be more efficient to first use linear-probing for a limited number of epochs and then use fine-tuning. Experimental results from [99] suggest that linear-probing before fine-tuning allows to better adapt task-specific parameters before Fine-Tuning, and generally works better for transfer learning. [96] adapts this approach for generative adversarial networks, by considering the discriminator as a classifier. This is achieved by the first stage of Transfer Learning of Algorithm 3 where the first layers of the pre-trained discriminator are frozen while all the layers of the generator and the last layers of the discriminator are updated. This first stage is useful because it is advantageous to first update a classifier head and frozen first layers for a limited number of epochs before fine-tune all the layers when a classifier is adapted to a new task. [96] proposes to adapt this characteristic to the

discriminator. [96] makes an empirical study to choose the number of layers to freeze. In real application, this empirical study should be impossible to perform. Indeed, to choose the number of layers to freeze, the authors executed Algorithm 2 on a large reference dataset, which will be not available in practice, and evaluated the mean layer weight change, assuming that a low change in their weight indicates that they are less associated with the sensitive attribute. In practice, we will not have access to the large reference dataset, but only to the small reference dataset. However, the need for a large dataset is to limit instability during training. By repeating Algorithm 2 on the small reference dataset, it is possible to also extract information about the standard deviation of the mean layer weight changes, and so limit the impact of the instability. However, this is a potentially costly approach. The second stage of Transfer Learning is achieved by fine-tune both the generator and the discriminators (partially) updated during the first stage while maintaining the second discriminator layers again frozen.

The second solution to avoid mode collapse is based on a multiple feedback approach. [100] and [101] use multiple pre-trained discriminators to improve generative adversarial network performance in case of small dataset. Based on this idea, [96] proposes to use the pre-trained discriminator on the large biased dataset during the training of the adversarial network on the reference dataset in a multiple feedback loop. They retain a frozen copy of the pre-train discriminator and train a new discriminator, ensuring double feedback on the data generated by the generator. The weight in the feedback of each discriminator is a hyperparameter. The loss associated to this double feedback approach is given by Equation (7).

$$\min_{G_t} \max_{D_t} = \mathbb{E}_{x \in D_{ref}}(\log D_t(x)) + \lambda \mathbb{E}_{z \sim p_z(z)}(\log(1 - D_t(G(z)))) + (1 - \lambda)\mathbb{E}_{z \sim p_z(z)}(\log(1 - D_S(G(z)))), \quad (7)$$

where:

- D_{ref} is the reference dataset;
- D_t is the discriminator that is updated;
- D_S is the pre-trained discriminator that is frozen;
- $p_z(z)$ is the noise distribution as input to the generator;
- G is the generator that is trained;
- λ a hyperparameter that controls the weighting given to D_t and D_S.

Both Algorithms 2 and 3 are illustrated by Figure 12, which comes from [96]. They can be used on a pre-trained generative adversarial network. Such property is interesting for SECURED as it means that the two methods are suitable to sanitize a generative adversarial network trained on a biased dataset.

As for the previous method, the different notation can be translated in the following way on use case 3 if we consider lung X-ray image generation with a small proportion of healthy persons versus sick patients:

- D_{bias} is a large lung X-ray dataset where sick patients are strongly over-represented compared to healthy patients;
- D_{ref} is a small lung X-ray dataset where sick patients and healthy patients are more or less equally represented;
- G is the final X-ray image generator. This generator is expected to generate a higher proportion of healthy people than a generator based on D_{bias} alone and of better quality than a generator trained only on D_{ref}. This is the output of the algorithm that is more relevant for SECURED.
- D_t and D_S are part of the training piping and are not crucial for the final output to SECURED. They are trained to distinguish between true images and the images generated by the generators. They are useful during the training because they force the generators to generate more realistic images as possible.
- $p_z(z)$ is a technical parameter. The generator takes in input a vector and from this vector generates an image. $p_z(z)$ corresponds to the distribution of the vector given as input of the generator.

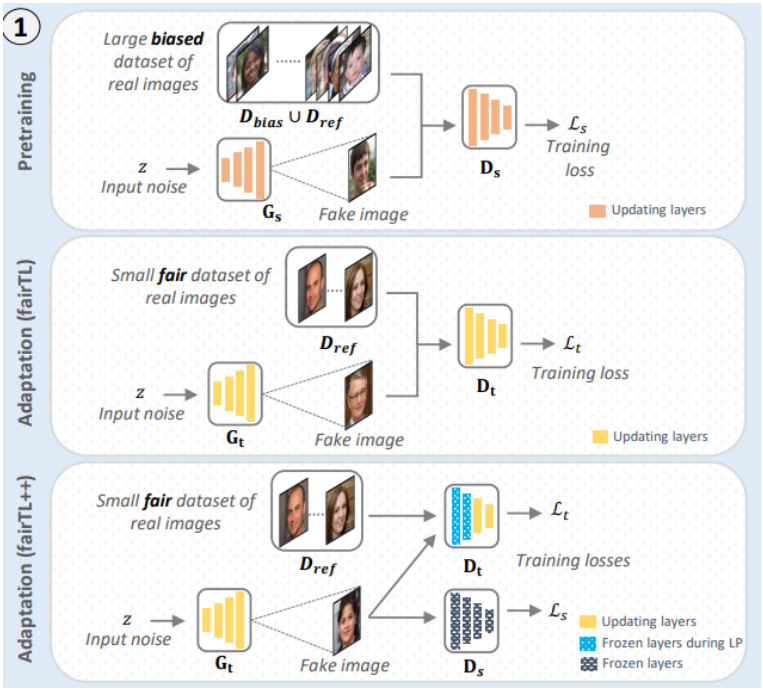


Figure 12 – Fair Transfer Learning and Fair Transfer Learning++ illustration from [96].

5.4 Plan for integration for Fair generative models

5.4.1 Deployment as micro-service

With the progress of the project and the more precise definition of the use cases, the need for handling bias have been identified, especially for medical image generation in an unbalanced context. These needs, not yet identified in Deliverable D4.1, have led to a review of the state of the art on this domain, which is still relatively unexplored in the literature. In this section, we present how we plan to integrate the reviewed methodologies within SECURED. This plan may be updated as the project progresses and new requirements are identified.

We plan to deploy fairness module as four docker [102] or podman [103]containers. Each container will contain Python scripts that are executable directly via command line or that can be called via an API. At the moment, we plan to include in the module the following:

- A classification script, used to train a classifier able to predict the image class (e.g., if it is a sick or healthy patient). This classifier will be used to compute evaluation metrics (as explained in previous sections) and will be used when the importance weighting approach will be used. It would take in input a configuration file;
- A training script, used to train the image generator, and taking as input a configuration file. The configuration parameter will provide at least the needed paths and the method used (importance weighting or Fair Transfer Learning, for instance);
- An evaluation script, used to compute Fairness Discrepancy and other classical metrics used for evaluation of the images generated;
- A generator script, that uses the generator trained to generate new images.

The non-exhaustive list of Python modules that will be used include the deep learning frameworks PyTorch [104] and PyTorch Lightning [105] , one visualization module (e.g. plotly [106] or matplotlib [107]), Python modules to improve the user experience (e.g., typer [108] and rich [109]) and a Python module dedicated to packaging and dependency management like poetry [110].

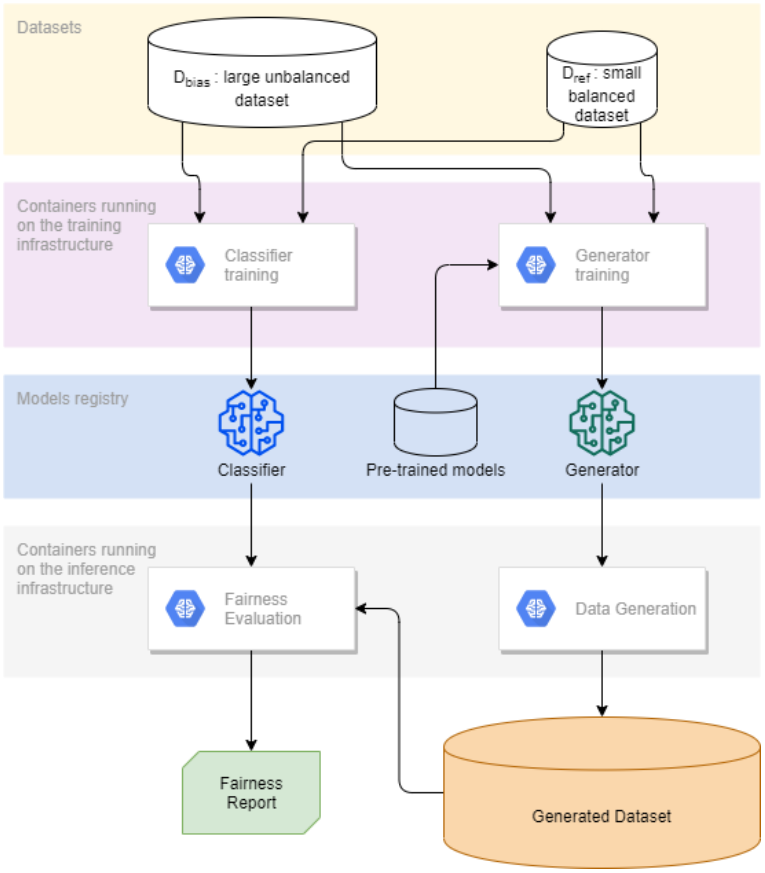


Figure 13 – Flow of the Fairness module.

Figure 13 illustrates the envisioned flow of the Fairness module. The module will be integrated into the SECURED framework, but each component will be accessible also standalone.

5.4.2 Development process

To develop, test and demonstrate the fairness module, we will use the datasets that are extensively discussed in Deliverable D2.1. We repeat here the main characteristics of these datasets that are relevant for explaining the activities of WP3 focused on services dedicated to handling bias. The way in which datasets are distributed could be different (e.g., for handling bias, datasets will be divided into a small reference dataset without bias and a larger biased dataset). As an example, the **ChestX-ray14** dataset includes metadata describing whether the image is of a healthy or diseased chest, as well as the type of disease. Most of the data concern images of healthy patients. We divide the dataset into two disjoint datasets: the first contains a large amount of data, with a large majority of images of healthy patients, and simulates the dataset denoted D_{biased}. The second dataset is smaller and balanced between the number of images of healthy patients and those with the disease under study, and simulates the D_{ref} dataset.

The work plan for the initial phase of the development is described below:

1. Implement and test the methods on Open Data, with the following three potentials data sources (the dataset descriptions are reported from Deliverable D2.1 below for completeness):
    - **ChestX-ray14** [111]: medical imaging dataset which comprises 112,120 frontal-view X-ray images of 30,805 (collected from the year 1992 to 2015) unique patients with the text-mined fourteen common disease labels, mined from the text radiological reports via NLP techniques;
    - **NODE21** [112]:consists of frontal chest radiographs with annotated bounding boxes around nodules. It consists of 4882 frontal chest radiographs where 1134 CXR images (1476 nodules) are annotated

with bounding boxes around nodules and the remaining 3748 images are free of nodules and hence represent the negative class;

- **CheXpert** [113]:a large dataset of chest X-rays and competition for automated chest X-ray interpretation, which features uncertainty labels and radiologist-labeled reference standard evaluation sets.

2. Development of micro-service derived from the code developed/used in these experiments
3. Demonstration of the developed micro-service

5.5 Relationship with SECURED Use Cases

5.5.1 Questionnaire about fairness requirements

To collect in an organized way all the information related to the need of fairness in the Use Case, a questionnaire has been prepared and shared with the Use Case providers. The questions included in the questionnaire were:

- **Questions regarding interests for the developed techniques:**
  - Do you suspect any source of bias relative to the data (real-life phenomenon, experimental protocol, data collection)?
  - What could be the sensitive attributes of your data?
  - What do you expect with the use of unbiased techniques in your use case?
- **Questions regarding the data:**
  - Are some of your datasets shareable?
  - If no
    - Do you know if public datasets (similar enough for our study) are available?
    - What kind of data are needed? In which format?
  - Could you describe the data pre-processing?
- **Questions regarding the models:**
  - What are the tasks of your models (classification, regression, forecasting, segmentation, detection, data generation, etc.)?
  - Are the models learned available for sharing with us?
  - If no, could you explain/provide the architecture and the training procedure?
  - What are the current performances of the model (which performance metrics are relevant and what are their scores)?
  - Could you provide information on the model usage? Is there post-processing? Is the inference performed in batch (on a group of several instances aggregated) or one by one?
  - How Fairness services could interact with your ML models?
  - Are there any specific requirements or limitations that we should be aware of when integrating with your AI model?

5.5.2 Use Case 3

Despite the collection and analysis of the answers to the questionnaire are still in progress, we can anticipate the results already analyzed from Use Case 3, since they are already available.

Use Case 3 is focused on the data generation for education. There are various potential data sources, such as images obtained from X-ray (e.g., mammography), with MRI (e.g., nervous systems), or with ultrasound (e.g., brain). Further data types are time series datasets, e.g., ECG or CTG. The third possible data type is textual data with Electronic Health Records. For imaginary and time series datasets, metadata are potentially accessible. There are numerous open datasets available for all data modalities.

Use Case 3 can utilize fairness algorithms, because most input datasets are unbalanced: some categories of population or symptoms are underrepresented. In the educational setting, a balanced sample is required for the symptoms that are the target of a given lecture.

E.g., for pathological whole slide images, one can suspect bias because healthy cases are typically missing from a real tissue bank. Due to the invasive nature of sample collection, healthy samples are never collected intentionally. Random samples are expected from accidents or death cases due to other diseases. For educational data generation, the sensitive attributes can be any data that is unique or rare. The Unbiased Artificial Intelligence solution is expected to create synthetic samples for outlier cases as well.

The Use Case (UC) 3 leader can provide data on their premises. In particular, it is possible to access the data with remote protocols that ensure no data transfer across country borders. The legal conditions for data access must be ensured by the technical partner which wants to experiment with the data.

Using these datasets require background knowledge, such as type of needed pre-processing, existing methods to generate data without fairness consideration or performance metrics and benchmark results. Two options can be followed during the development of the methods described: 1) Use classical GAN-variants architectures and adapt methods explained in this section; 2) Use architectures provided within SECURED adapt methods explained in this section.

In terms of evaluation, we will use Fairness discrepancy to measure the fairness of generated images. Metrics like Fréchet Inception Distance will be used to measure the quality of images. Results will be assessed by a comparative study of data generator without fairness and with fairness.

5.6 Conclusions

The preliminary activities of the task discussed in this section are summarized in Table 15. In WP3, further architectures (e.g., Wasserstein generative adversarial network [114]) will be tested.

Main techniques explored	Fair Classifier based on triplet loss
Existing or new method	Existing methods
Linked use case	Under definition via a Questionnaire being answered by use case providers

Table 15 – In-processing methods for classification and regression tasks status.

Based on the responses in a questionnaire, we are currently consider enriching and implement methods in centralized context; and to implement methods in Federated Learning context. In Table 16, we provide the

Techniques explored	Fair Transfer Learning, Importance Weighting (Existing methods)
Linked use case	• Potential use case: UC3, dedicated to data generation for education. • Various types of data (images, time series, textual data). • Current target: X-ray images

Table 16 – Fairness for generative models status.

status of work on the fairness applied to data generation.

6 Conclusions

The first deliverable in Work Package 3 reports interim progress about the activities in the direction of the processing flow of the SECURED library. The SECURED processing flow runs in four parallel tasks: three dedicated to scaling up federated learning, unbiased AI and secure multi-party computation/homomorphic encryption respectively, and a fourth dedicated to the implementation of these technologies in a library.

The Secure Multi-Party Computation / Homomorphic Encryption line of work focuses on scaling up the tasks involved in computing on encrypted data. For Machine Learning applications, quantization is necessary and two possible solutions (PTQ and QAT) are in the early stages of testing for the libraries and use cases that are of interest to the SECURED project. Further results have been reached with the software selections, and the approach to be followed in **the common library** have been sketched. Further, to identify targets for the acceleration, we focused on Homomorphic Encryption and we explored the bottleneck of components of selected libraries and the overhead of the encryption computation over plaintext one for representative tasks.

Concerning Federated Learning, we have carried out a careful risk analysis that served us as base for defining the selection criteria for the choice of the FL library to be used in the SECURED project. The Flower library as been selected based on these criteria. In the next month of the project, the approach that we used to derive the selection criteria will also be the base for the selection of the FL parameters to be used in the SECURED Library.

Regarding Unbiased Artificial Intelligence, with the development of the project, we have identified fairness as an highly relevant property for the SECURED use case and applications. To this end, we carried out a first analysis on Fairness for generative AI and we have developed a plan for integrating fairness into SECURED. To have a better understanding about the need of Fairness in the use case, we have prepared a questionnaire that has been proposed to all the use case leader. The available answers to this questionnaire have been reported and discussed.

Acknowledgments

Funded in part by the European Union (Grant Agreement Nr. 101095717, SECURED Project). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the Health and Digital Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] D. Evans, V. Kolesnikov, and M. Rosulek, "A pragmatic introduction to secure multi-party computation," *Found. Trends Priv. Secur.*, vol. 2, no. 2-3, pp. 70–246, 2018. [Online]. Available: https://doi.org/10.1561/3300000019

[2] Z. Brakerski, "Fundamentals of fully homomorphic encryption," in *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, O. Goldreich, Ed. ACM, 2019, pp. 543–563. [Online]. Available: https://doi.org/10.1145/3335741.3335762

[3] C. Juvekar, V. Vaikuntanathan, and A. P. Chandrakasan, "GAZELLE: A low latency framework for secure neural network inference," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, W. Enck and A. P. Felt, Eds. USENIX Association, 2018, pp. 1651–1669. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/juvekar

[4] P. Mishra, R. Lehmkuhl, A. Srinivasan, W. Zheng, and R. A. Popa, "Delphi: A cryptographic inference service for neural networks," in *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, S. Capkun and F. Roesner, Eds. USENIX Association, 2020, pp. 2505–2522. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/mishra

[5] F. Boemer, R. Cammarota, D. Demmler, T. Schneider, and H. Yalame, "MP2ML: a mixed-protocol machine learning framework for private inference," in *ARES 2020: The 15th International Conference on Availability, Reliability and Security, Virtual Event, Ireland, August 25-28, 2020*, M. Volkamer and C. Wressnegger, Eds. ACM, 2020, pp. 14:1–14:10. [Online]. Available: https://doi.org/10.1145/3407023.3407045

[6] S. Obla, X. Gong, A. Aloufi, P. Hu, and D. Takabi, "Effective activation functions for homomorphic evaluation of deep neural networks," *IEEE Access*, vol. 8, pp. 153 098–153 112, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3017436

[7] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachène, "TFHE: fast fully homomorphic encryption over the torus," *J. Cryptol.*, vol. 33, no. 1, pp. 34–91, 2020. [Online]. Available: https://doi.org/10.1007/s00145-019-09319-x

[8] F. Bourse, M. Minelli, M. Minihold, and P. Paillier, "Fast homomorphic evaluation of deep discretized neural networks," in *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, ser. Lecture Notes in Computer Science, H. Shacham and A. Boldyreva, Eds., vol. 10993. Springer, 2018, pp. 483–512. [Online]. Available: https://doi.org/10.1007/978-3-319-96878-0_17

[9] A. Stoian, J. Fréry, R. Bredehoft, L. Montero, C. Kherfallah, and B. Chevallier-Mames, "Deep neural networks for encrypted inference with TFHE," in *Cyber Security, Cryptology, and Machine Learning - 7th International Symposium, CSCML 2023, Be'er Sheva, Israel, June 29-30, 2023, Proceedings*, ser. Lecture Notes in Computer Science, S. Dolev, E. Gudes, and P. Paillier, Eds., vol. 13914. Springer, 2023, pp. 493–500. [Online]. Available: https://doi.org/10.1007/978-3-031-34671-2_34

[10] "Lattigo v5," Online: https://github.com/tuneinsight/lattigo, Nov. 2023, ePFL-LDS, Tune Insight SA.

[11] A. Aly, B. Coenen, K. Cong, K. Koch, M. Keller, D. Rotaru, O. Scherer, P. Scholl, N. P. Smart, T. Tanguy, and T. Wood, "SCALE-MAMBA," https://github.com/KULeuven-COSIC/SCALE-MAMBA.

[12] N. Aaraj, A. Aly, T. Güneysu, C. Marcolla, J. Mono, R. Paludo, I. Santos-González, M. Scholz, E. Soria-Vazquez, V. Sucasas, and A. Suresh, "FANNG-MPC: framework for artificial neural networks and generic MPC," *IACR Cryptol. ePrint Arch.*, p. 1918, 2023. [Online]. Available: https://eprint.iacr.org/2023/1918

[13] "SCALE-MAMBA fork," https://github.com/Gugi264/SCALE-MAMBA.

[14] Galois, Inc., "swanky: A suite of rust libraries for secure computation," https://github.com/GaloisInc/swanky, 2019.

[15] N. Chandran, D. Gupta, A. Rastogi, R. Sharma, and S. Tripathi, "Ezpc: Programmable and efficient secure two-party computation for machine learning," in *IEEE European Symposium on Security and Privacy, EuroS&P 2019, Stockholm, Sweden, June 17-19, 2019*. IEEE, 2019, pp. 496–511. [Online]. Available: https://doi.org/10.1109/EuroSP.2019.00043

[16] "EzPC: Easy Secure Multiparty Computation," Online: https://github.com/mpc-msri/EzPC, Microsoft Research.

[17] B. Knott, S. Venkataraman, A. Y. Hannun, S. Sengupta, M. Ibrahim, and L. van der Maaten, "Crypten: Secure multi-party computation meets machine learning," in *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, M. Ranzato, A. Beygelzimer, Y. N. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021, pp. 4961–4973. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/2754518221cfbc8d25c13a06a4cb8421-Abstract.html

[18] "CrypTen," Online: https://github.com/facebookresearch/CrypTen, Facebook RResearch.

[19] N. Kumar, M. Rathee, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow: Secure tensorflow inference," in *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*. IEEE, 2020, pp. 336–353. [Online]. Available: https://doi.org/10.1109/SP40000.2020.00092

[20] D. Rathee, M. Rathee, N. Kumar, N. Chandran, D. Gupta, A. Rastogi, and R. Sharma, "Cryptflow2: Practical 2-party secure inference," in *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, J. Ligatti, X. Ou, J. Katz, and G. Vigna, Eds. ACM, 2020, pp. 325–342. [Online]. Available: https://doi.org/10.1145/3372297.3417274

[21] A. A. Badawi, J. Bates, F. Bergamaschi, D. B. Cousins, S. Erabelli, N. Genise, S. Halevi, H. Hunt, A. Kim, Y. Lee, Z. Liu, D. Micciancio, I. Quah, Y. Polyakov, R. V. Saraswathy, K. Rohloff, J. Saylor, D. Suponitsky, M. Triplett, V. Vaikuntanathan, and V. Zucca, "Openfhe: Open-source fully homomorphic encryption library," in *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography, Los Angeles, CA, USA, 7 November 2022*, M. Brenner, A. Costache, and K. Rohloff, Eds. ACM, 2022, pp. 53–63. [Online]. Available: https://doi.org/10.1145/3560827.3563379

[22] Zama, "Concrete ML: a privacy-preserving machine learning library using fully homomorphic encryption for data scientists," 2022, https://github.com/zama-ai/concrete-ml.

[23] ——, "Concrete: TFHE Compiler that converts python programs into FHE equivalent," 2022, https://github.com/zama-ai/concrete.

[24] M. Keller, "MP-SPDZ: A versatile framework for multi-party computation," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020. [Online]. Available: https://doi.org/10.1145/3372297.3417872

[25] K. Rohloff, D. Cousins, and Y. P. (project leads), "PALISADE homomorphic encryption software library," 2016. [Online]. Available: https://gitlab.com/palisade

[26] J. H. Cheon, A. Kim, M. Kim, and Y. S. Song, "HEAAN library," 2016. [Online]. Available: https://github.com/snucrypto/

[27] "Concrete documentation," Online: https://docs.zama.ai/concrete/getting-started/fhe_basics., 2022.

[28] M. Keller, "MP-SPDZ documentation," 2022, https://mp-spdz.readthedocs.io/en/latest/readme.html.

[29] C. Gouert, D. Mouris, and N. G. Tsoutsos, "Sok: New insights into fully homomorphic encryption libraries via standardized benchmarks," *Proc. Priv. Enhancing Technol.*, vol. 2023, no. 3, pp. 154–172, 2023. [Online]. Available: https://doi.org/10.56553/popets-2023-0075

[30] "Openfhe hexl acceleration," Online: https://github.com/openfheorg/openfhe-hexl, the OpenFHE Project.

[31] G. Moody and R. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[32] S. Sakib, M. M. Fouda, and Z. M. Fadlullah, "Harnessing artificial intelligence for secure ecg analytics at the edge for cardiac arrhythmia classification," p. 137–153, Jun. 2021. [Online]. Available: http://dx.doi.org/10.1201/9781003028635-11

[33] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *J. Math. Cryptol.*, vol. 9, no. 3, pp. 169–203, 2015. [Online]. Available: http://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml

[34] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[35] P. M. Mammen, "Federated learning: Opportunities and challenges," *arXiv preprint arXiv:2101.05428*, 2021.

[36] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[37] V. Cristea, "Federated learning: a comparison of methods: How do different federated learning frameworks compare?" 2023.

[38] R. Hamsath Mohammed Khan, "A comprehensive study on federated learning frameworks: Assessing performance, scalability, and benchmarking with deep learning model," 2023.

[39] H. R. Roth, Y. Cheng, Y. Wen, I. Yang, Z. Xu, Y.-T. Hsieh, K. Kersten, A. Harouni, C. Zhao, K. Lu *et al.*, "Nvidia flare: Federated learning from simulation to real-world," *arXiv preprint arXiv:2210.13291*, 2022.

[40] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão *et al.*, "Flower: A friendly federated learning framework.(2022)," *HAL Id: hal-03601230*, 2022.

[41] Thales, "Pybiscus: a flexible federated learning framework," 2023.

[42] A. Wainakh, E. Zimmer, S. Subedi, J. Keim, T. Grube, S. Karuppayah, A. Guinea, and M. Mühlhäuser, "Federated learning attacks revisited: A critical discussion of gaps," *Assumptions, and Evaluation Setups in IEEE Access [under review]*, 2022.

[43] R. Gosselin, L. Vieu, F. Loukil, and A. Benoit, "Privacy and security in federated learning: A survey," *Applied Sciences*, vol. 12, no. 19, p. 9901, 2022.

[44] C. Fung, C. J. Yoon, and I. Beschastnikh, "The limitations of federated learning in sybil settings." in *RAID*, 2020, pp. 301–316.

[45] J. Shi, W. Wan, S. Hu, J. Lu, and L. Y. Zhang, "Challenges and approaches for mitigating byzantine attacks in federated learning," in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022, pp. 139–146.

[46] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[47] C. Fung, C. J. Yoon, and I. Beschastnikh, "Mitigating sybils in federated learning poisoning," *arXiv preprint arXiv:1808.04866*, 2018.

[48] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 2938–2948.

[49] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[50] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, I. Ray, N. Li, and C. Kruegel, Eds. ACM, 2015, pp. 1322–1333. [Online]. Available: https://doi.org/10.1145/2810103.2813677

[51] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*. IEEE Computer Society, 2017, pp. 3–18. [Online]. Available: https://doi.org/10.1109/SP.2017.41

[52] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, "Property inference attacks on fully connected neural networks using permutation invariant representations," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[53] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.

[54] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016*, T. Holz and S. Savage, Eds. USENIX Association, 2016, pp. 601–618. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/tramer

[55] L. Lyu, H. Yu, and Q. Yang, "Threats to federated learning: A survey," *arXiv preprint arXiv:2003.02133*, 2020.

[56] J. So, B. Güler, and A. S. Avestimehr, "Byzantine-resilient secure federated learning," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2168–2181, 2020.

[57] N. Carlini, A. Athalye, N. Papernot, W. Brendel, J. Rauber, D. Tsipras, I. Goodfellow, A. Madry, and A. Kurakin, "On evaluating adversarial robustness," *arXiv preprint arXiv:1902.06705*, 2019.

[58] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 656–672.

[59] B. Pejó and D. Desfontaines, *Guide to Differential Privacy Modifications: A Taxonomy of Variants and Extensions*. Springer Nature, 2022.

[60] G. Ács and C. Castelluccia, "I have a dream!(differentially private smart metering)." in *Information hiding*, vol. 6958. Springer, 2011, pp. 118–132.

[61] E. T. M. Beltrán, M. Q. Pérez, P. M. S. Sánchez, S. L. Bernal, G. Bovet, M. G. Pérez, G. M. Pérez, and A. H. Celdrán, "Decentralized federated learning: Fundamentals, state of the art, frameworks, trends, and challenges," *IEEE Communications Surveys & Tutorials*, 2023.

[62] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning," in *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 1–15.

[63] L. Wang, S. Xu, X. Wang, and Q. Zhu, "Eavesdrop the composition proportion of training labels in federated learning," *arXiv preprint arXiv:1910.06044*, 2019.

[64] N. Haim, G. Vardi, G. Yehudai, O. Shamir, and M. Irani, "Reconstructing training data from trained neural networks," *Advances in Neural Information Processing Systems*, vol. 35, pp. 22 911–22 924, 2022.

[65] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction {APIs}," in *25th USENIX security symposium (USENIX Security 16)*, 2016, pp. 601–618.

[66] C. H. Lee and H.-J. Yoon, "Medical big data: promise and challenges," *Kidney research and clinical practice*, vol. 36, no. 1, p. 3, 2017.

[67] L. Budach, M. Feuerpfeil, N. Ihde, A. Nathansen, N. Noack, H. Patzlaff, F. Naumann, and H. Harmouch, "The effects of data quality on machine learning performance," *arXiv preprint arXiv:2207.14529*, 2022.

[68] F. Mireshghallah, H. A. Inan, M. Hasegawa, V. Rühle, T. Berg-Kirkpatrick, and R. Sim, "Privacy regularization: Joint privacy-utility optimization in language models," *arXiv preprint arXiv:2103.07567*, 2021.

[69] K. Chaudhuri, C. Guo, and M. Rabbat, "Privacy-aware compression for federated data analysis," in *Uncertainty in Artificial Intelligence*. PMLR, 2022, pp. 296–306.

[70] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, pp. 1789–1819, 2021.

[71] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson, "Scalable private learning with pate," *arXiv preprint arXiv:1802.08908*, 2018.

[72] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[73] B. Pavlyshenko, "Using stacking approaches for machine learning models," in *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*. IEEE, 2018, pp. 255–258.

[74] E. Winter, "The shapley value," *Handbook of game theory with economic applications*, vol. 3, pp. 2025–2054, 2002.

[75] V. Siomos and J. Passerat-Palmbach, "Contribution evaluation in federated learning: Examining current approaches," *arXiv preprint arXiv:2311.09856*, 2023.

[76] A. Paullada, I. D. Raji, E. M. Bender, E. Denton, and A. Hanna, "Data and its (dis) contents: A survey of dataset development and use in machine learning research," *Patterns*, vol. 2, no. 11, 2021.

[77] "Aif360," Online: https://aif360.res.ibm.com/, Feb. 2024.

[78] H. Weerts, M. Dudík, R. Edgar, A. Jalali, R. Lutz, and M. Madaio, "Fairlearn: Assessing and Improving Fairness of AI Systems," *Journal of Machine Learning Research*, vol. 24, 2023. [Online]. Available: http://jmlr.org/papers/v24/23-0389.html

[79] S. A. Friedler, C. Scheidegger, S. Venkatasubramanian, S. Choudhary, E. P. Hamilton, and D. Roth, "A comparative study of fairness-enhancing interventions in machine learning," 2018. [Online]. Available: https://pypi.org/project/fairness/

[80] T. Calders and S. Verwer, "Three naive bayes approaches for discrimination-free classification," *Data Mining and Knowledge Discovery*, vol. 21, pp. 277–292, 2010. [Online]. Available: http://www.cs.ru.nl/~sicco/papers/dmkd10.pdf

[81] "Blackboxauditing," 2019. [Online]. Available: https://github.com/algofairness/BlackBoxAuditing/

[82] "Fairness compass," 2021. [Online]. Available: https://github.com/axa-rev-research/fairness-compass

[83] J. Fitzsimons, A. Al Ali, M. Osborne, and S. Roberts, "A general framework for fair regression," *Entropy*, vol. 21, no. 8, p. 741, Jul. 2019. [Online]. Available: http://dx.doi.org/10.3390/e21080741

[84] D. Steinberg, A. Reid, and S. O'Callaghan, "Fairness measures for regression via probabilistic classification," 2020.

[85] M. Kaya and H. S. Bilge, "Deep metric learning: A survey," *Symmetry*, vol. 11, p. 1066, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:201710040

[86] C. Ilvento, "Metric learning for individual fairness," 2020.

[87] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Jun. 2015. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2015.7298682

[88] M. Gornet, C. Kirchner, and C. Tessier, "Operational fairness for facial authentication systems," *ERCIM News*, vol. 2022, no. 131, 2022. [Online]. Available: https://ercim-news.ercim.eu/en131/special/operational-fairness-for-facial-authentication-systems

[89] A. Martzloff, N. Posocco, and Q. Ferré, "A fair classifier embracing triplet collapse," *CoRR*, vol. abs/2306.04400, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2306.04400

[90] W. Lei, R. Zhang, Y. Yang, R. Wang, and W.-S. Zheng, "Class-center involved triplet loss for skin disease classification on imbalanced data," in *2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI)*, 2020, pp. 1–5.

[91] S. Desai, A. Munshi, and D. Munshi, "Gender bias in cardiovascular disease prevention, detection, and management, with specific reference to coronary artery disease," *Journal of mid-life health*, vol. 12, no. 1, p. 8, 2021.

[92] C. T. H. Teo and N.-M. Cheung, "Measuring fairness in generative models," 2021.

[93] S. Tan, Y. Shen, and B. Zhou, "Improving the fairness of deep generative models without retraining," 2021.

[94] K. Choi, A. Grover, T. Singh, R. Shu, and S. Ermon, "Fair generative modeling via weak supervision," 2020.

[95] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[96] C. T. Teo, M. Abdollahzadeh, and N.-M. Cheung, "Fair generative models via transfer learning," 2022.

[97] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," 2020.

[98] Y. Kossale, M. Airaj, and A. Darouichi, "Mode collapse in generative adversarial networks: An overview," in *2022 8th International Conference on Optimization and Applications (ICOA)*, 2022, pp. 1–6.

[99] A. Kumar, A. Raghunathan, R. Jones, T. Ma, and P. Liang, "Fine-tuning can distort pretrained features and underperform out-of-distribution," 2022.

[100] S. Um and C. Suh, "A fair generative model using total variation distance," 2022. [Online]. Available: https://openreview.net/forum?id=F1Z3QH-VjZE

[101] N. Kumari, R. Zhang, E. Shechtman, and J. Zhu, "Ensembling off-the-shelf models for GAN training," *CoRR*, vol. abs/2112.09130, 2021. [Online]. Available: https://arxiv.org/abs/2112.09130

[102] "Docker," https://www.docker.com/.

[103] "Podman," https://podman.io/.

[104] "Pytorch," https://pytorch.org/.

[105] "Pytorch lighning," https://lightning.ai/docs/pytorch/stable/.

[106] "plotly," https://plotly.com/python/.

[107] "Matplotlib," matplotlib.org/.

[108] "Typer," pypi.org/project/typer/.

[109] "Rich," https://rich.readthedocs.io/en/stable/introduction.html.

[110] “Poetry,” https://python-poetry.org/.

[111] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, “Chestx-ray14,” 2018, https://paperswithcode.com/dataset/chestx-ray14.

[112] E. Sogancioglu, K. Murphy, and B. van Ginneken., “Node21,” 2022, https://node21.grand-challenge.org/Data/.

[113] eremy Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, J. Seekins, D. A. Mong, S. S. Halabi, J. K. Sandberg, R. Jones, D. B. Larson, C. P. Langlotz, B. N. Patel, M. P. Lungren, and A. Y. Ng, “Chexpert,” 2019, https://stanfordmlgroup.github.io/competitions/chexpert/.

[114] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN,” *CoRR*, vol. abs/1701.07875, 2017. [Online]. Available: http://arxiv.org/abs/1701.07875