MACHINE INTELLIGENCE FOR IOT AND EDGE

D2.1: Review on Existing Machine Learning and Data Mining Techniques, and Their Applicability

Work Package	WP2
Dissemination level	Public
Status	Final
Date	11/05/2021
Deliverable leader	Barış Bulut (Enforma)
Potential Contributors	Macq, Shayp, Sirris, NOS, UPORTO, BTH

Contributors

Name	Organization
Nicolás González-Deleito	Sirris
Sarah Klein	Sirris
Barış Bulut	Enforma
Burak Ketmen	Enforma
Sencer Sultanoglu	Eliar
Anna Hristoskova	Sirris
Sreeraj Rajendran	Sirris
Pedro Santos	ISEP
Veselka Boeva	BTH

Reviewers

Name	Organization
Sencer Sultanoğlu	Eliar
Anna Hristoskova	Sirris

Document History

Date	Main changes	Name
11-05-2021	Proposal for table of contents and initial list of areas for the machine learning and data mining techniques review based on the requirements from the BE partners	Nicolás González-Deleito and Sarah Klein
16-07-2021	BE input for the summary of distributed intelligence requirements	Nicolás González-Deleito, Sarah Klein, Anna Hristoskova, and Sreeraj Rajendran
04-08-2021	Review of the state of the art of data compression of time series, restructuring Section 4 to be aligned with Section 3.6	Sarah Klein, Nicolás González-Deleito
10-08-2021	Review Section 3.6 consolidation of requirements and the state of the art of data compression of time series	Anna Hristoskova
30-09-2021	Added section about “Anomaly Detection at the Edge” (S.4.3.2)	Pedro Santos
09-11-2021	Added section about “Privacy-preserving learning techniques" (S.4.4.1)	Pedro Santos
17-11-2021	Reviewed contributions to Section 4 after first contribution round	Nicolás González-Deleito
02-12-2021	Reviewed contributions to Section 4 after second contribution round and consolidated references	Nicolás González-Deleito
22-12-2021	First review of document	Anna Hristoskova

Table of Contents

List of Tables

Abbreviations

CPE	Customer-Premises Equipment
MES	Manufacturing Execution System
MFBB	MIRAI Framework Building Block
ML	Machine Learning
PLC	Programmable Logic Controllers
UC	Use Case
UC1	Use case 1: Distributed renewable energy systems (UC owner: 3E)
UC2	Use case 2: Secure Internet provisioning (UC owner: NOS)
UC3	Use case 3: Traffic management (UC owner: Macq)
UC4	Use case 4: Water management (UC owner: Shayp)
UC5	Use case 5: Continuous auto configuration of industrial controllers at edge (UC owner: Eliar & Enforma)

1. Executive Summary

Work package 2 is the first of the work packages to deal with the distributed AI concepts. Of the 4 tasks comprising the work package, Task 2.1 focuses on the investigation and design of machine learning and data mining techniques applicable in IoT/edge computing.

The significance of Task 2.1 is several fold: First, T3.1 on the reference architecture takes as input the identified machine learning techniques in T2.1. Next, also based on the result of Task 2.1, Task 2.2 will later aim to develop unsupervised and semi-supervised methods in order to automate knowledge extraction and learning in data stream scenarios. Also, Task 2.3, distributed/composable data mining models will be developed based on the state-of-the-art of Task 2.1. Furthermore, Task 2.4 will attempt to provide preliminary results of the implementation of the algorithms emerging also from Task 2.1. Finally, Task 3.1 on the reference architecture takes as input the identified machine learning techniques in Task 2.1.

Hence, the fast paced reader who wants to get a good grip on the project concepts is therefore advised to consider reading Deliverable 2.1 (and possibly also Deliverable 1.1) before moving on to further deliverables.

The deliverable starts off with a summary of the distributed intelligence use case requirements for each of the 5 use case scenarios in the project. It then puts the current techniques into perspective, conveying to the reader the state-of-the-art in all its nicety.

2. Introduction

The scope of the second work package to consider “Distributed AI Toolbox”. The work is expected to contribute to the establishment of AI techniques that model knowledge from IoT/edge information links and that will support the deployment of distributed AI applications. Its main activities consist of

(i) adaptation of machine learning and data mining methods to dataflows in IoT/edge computing,

(ii) design of AI algorithms and techniques for continual and evolving learning, and

(iii) design of distributed data mining and machine learning models and collaborative decision analysis methods.

Thus, the work is expected to provide the AI and ML building blocks for deployment in the designed WP3 MIRAI reference framework and MFBB.

As the first task under the work package, Task 2.1 attempts to capture the investigation and design of machine learning and data mining techniques applicable in IoT/edge computing. Here the work takes the requirements results of Task 1.1 and the application domains defined in work package 4. It then attempts to study the distributed AI techniques. While doing do, a summary of the distributed AI use case requirements are first studied (Section 3). The selected techniques will further be adapted to distributed data stream environments. Guidelines on how to fit the algorithms specifically to the problem domain will be developed and specified in detail. The result of the study is such that the next section, Section 4, can build on further by providing an up-to-date information on the technological state-of-the-art. The result here is a classification and analysis of the state of the art, specification and high-level design of the new techniques and algorithms.

MIRAI, despite its moderately sized consortium, is relatively use case rich. Hence, it is important to adequately identify and refer to each of the 5 use cases. Based off of the outcome of D1.1, the following numbering and naming convention is adopted throughout the text:

Numbering and naming convention:

- Use case 1: Distributed renewable energy systems (UC owner: 3E)
- Use case 2: Secure Internet provisioning (UC owner: NOS)
- Use case 3: Traffic management (UC owner: Macq)
- Use case 4: Water management (UC owner: Shayp)
- Use case 5: Continuous auto configuration of industrial controllers at edge (UC owner: Eliar & Enforma)

3. Summary of the distributed intelligence use case requirements

In this section we first summarize, per use case, the relevant distributed intelligence requirements, as they will have an impact on the machine learning and data mining techniques to be considered in WP2. These requirements are subsequently consolidated in view of identifying the relevant state-of-the-art areas (or themes) for which specific techniques will be described in Section 4.

3.1. UC1: Distributed renewable energy systems (UC owner: 3E)

3E provides consultancy and software solutions for monitoring and improving the performance of sustainable energy installations (such as photovoltaic and windmill plants) and for optimizing energy consumption. Through MIRAI, 3E aims at providing real-time status updates of plant assets and at offering asset control features (such as switching off plants when energy prices are negative, and charging and discharging battery systems to help stabilize the grid).

3E would like to execute (part of) the grid optimization service locally, on a prosumer's infrastructure. For this, the production and consumption need to be forecasted at the edge, while leveraging the knowledge from other edge devices and customer infrastructure. Being able to distribute the service in that way would enable a faster response to changes in the grid status, as well as to changes in the local energy production and consumption, and hence reduce latency. In addition, 3E would like to minimize bandwidth and communication costs when transferring data to a cloud backend.

The corresponding distributed intelligence requirements for this use case are:

- Forecast the future energy production, consumption, and market prices at the edge and execute (part of) the grid optimization service locally
- Leverage knowledge from other edge devices for improving forecasts
- Compress real-time (time series) data locally, with minimal information loss
- Provide a secure data sharing solution between edge devices and edge and cloud in view of competitive prosumer information

3.2. UC2: Secure Internet provisioning (UC owner: NOS)

NOS wants to provide a security solution allowing the detection and mitigation of DDoS attacks involving customer IoT devices. In addition, the solution will provide various levels of defence against such threats. To detect potential attacks, the incoming and outgoing network traffic of the customer's network will be analysed. Therefore, the solution should integrate with the Internet access points deployed at the customer's premises, the Customer-Premises Equipment (CPE), the edge node in this UC. The solution should also integrate intelligence in the cloud, for the training phase and to complement the inference phase, as the CPE's computational power is insufficient in sight of typical requirements for model training. The trained ML models will capture the typical legitimate traffic in the customer's network, so care must be taken to protect the customer's privacy both during the training phase and during run-time. Other elements besides the CPE and a cloud component may host components of this solution (e.g., middleboxes), leading to potential distribution of the inference process. The use of a transfer learning approach can be considered in case synergies between data/models produced by/for different customers can be explored to produce a global model (at the cloud) that is then transferred to the CPEs.

The intelligent system requirements for this use case are:

- Detect anomalous network traffic, i.e., traffic that does not match typical traffic behavior observed at the customer's network
- Protect customer identity and data, during training and operation

- Carry out inference in a decentralized fashion, i.e., ability to run a lightweight model at the edge and a more elaborate one in the cloud or an ISP middlebox
- (Train primarily at the cloud, due to the CPE’s limited computational power)

Other requirements that may be considered in the design of a solution:

- Data and models of many customers available at the cloud can be trained into an aggregated model, that is then transferred to CPE (transfer learning)
- Pre-process raw network traffic datasets at CPE to extract features, depending on the features selected as inputs to the developed ML mechanism

3.3. UC3: Traffic management (UC owner: Macq)

In the domain of smart mobility, Macq offers products ranging from sensor solutions for traffic monitoring, including advanced Smart Mobility cameras (able to operate on multiple lanes, distinguish between different types of vehicles, detect the number of passengers, estimate driving speed, etc.), to controllers for traffic light management at road intersections, and software packages for managing mobility-related data and extracting valuable insights for different types of decision makers (such as police and road authorities). Through MIRAI, Macq mainly aims at extracting valuable road safety analytics at the edge (specifically from their Smart Mobility cameras) and at reacting quickly when a potentially dangerous situation is detected.

Macq would like to extract relevant features from the image data (e.g. object identification) locally in order to avoid transferring bandwidth-intensive data and hence reduce latency, and to preserve road users' privacy. Macq would also like to leverage the computational power of other cameras potentially available at a given location and be able to not only reconstruct a full situational picture at that location (note that a camera might only capture a limited part of the location) but also to distribute computations horizontally at the edge based on the processing capabilities, current load, and connection link to the other neighboring cameras available. Finally, the cameras being deployed outdoors, the image preprocessing needs to be able to deal with noisy data (due to poor or unfavorable lighting conditions) and with missing data (as in the event of high image processing loads, cameras drop frames).

The corresponding distributed intelligence requirements for this use case are:

- Fuse locally data from different cameras capturing the same scene using different spectra and technologies (black and white, near infrared, color, thermal, time-of-flight) and from additional non-visual technologies (radar, sound)
- Extract optimal features locally from the fused data for further processing and distribution to other edge devices
- Build locally an accurate situational picture of a road location
- Leverage knowledge from other edge devices for dealing with noise and missing data, improving feature extraction, and building of an accurate situational picture
- Distribute computations among edge devices to reduce load on the single devices
- Minimize the sharing of road users' private data outside edge devices and, when needed, preserve their privacy

3.4. UC4: Water management (UC owner: Shayp)

Shayp provides solutions for the monitoring and management of water leakages for different types of buildings, ranging from small buildings such as individual homes to larger buildings such as schools, hospitals and office and administration buildings. Through MIRAI, Shayp aims at (1) reducing and optimizing communication bandwidth, as message transmission drains most of the battery power, (2) reducing the time it currently takes to detect a leakage and hence reacting quicker when a potentially damaging leakage is detected, and (3) adding remote control features to its meters, enabling future updates and remote calibration.

Shayp would like to perform part of the leakage detection reasoning locally on the smart water meters, and at least detect anomalous data water consumption at the edge. Full (or most) water consumption data should still be transferred to a cloud backend in order to confirm the occurrence of a leakage and determine its severity. As water consumption data is privacy-sensitive, computations and data transfers should be performed in a privacy-preserving fashion.

The corresponding distributed intelligence requirements for this use case are:

- Compress (time series) data locally, with minimal information loss and computation power
- Detect anomalies in water consumption data locally, by leveraging location/building specific information and in a battery-preserving fashion
- Preserve customers' privacy when manipulating and transferring water consumption data

3.5. UC5: Continuous auto configuration of industrial controllers at edge (UC owner: Eliar & Enforma)

Eliar produces textile machine process control devices and PLCs (Programmable Logic Controllers), which control the machines according to the desired recipe and process steps. Enforma provides data analysis in a plethora of areas from telecommunications records to fleet GPS data, from IoT, health and financial data to data emerging from IP networks and energy grids. Through MIRAI, the goal is to tune PID parameters adaptively with the output of the AI algorithm working on the process controllers and PLCs, which are IoT devices operating at the edge.

The textile dyeing process is a batch process which takes 5-12 hours depending on various process parameters such as fabric to be dyed, desired color, chemicals and dye. In the textile dyeing process, one of the important criteria that will ensure "right first time" is the correct temperature control of the machine. Currently, PID parameters are tuned by technicians according to their personal experience during installation of the dyeing machine. This may cause inconsistency in the process control and sustainability issues due to inefficient use of resources such as energy, steam, water, chemical, dye, and time.

The corresponding distributed intelligence requirements for this use case are:

- Generating time series data set, pre-processing (data cleaning, normalization, standardization, etc.) and labelling operations on time series in real time at edge
- Rule-based determination of process priorities that change according to the stage in the process /in MES
- Auto-tuning of PID parameters, namely k_p and k_i parameters (which refer to the coefficients of the so called proportional gain and integral gain, respectively) using Control Engineering approaches as well as ML techniques at edge, at certain intervals, whilst ensuring the PID continues to minimise process errors in real-time
- Ensuring that both the PID control within the machine itself and the steam usage on the basis of the enterprise can be given to the priority processes with the Federated Learning techniques
- Performing some of computations within MES to reduce the load on the edge devices

3.6. Consolidation of distributed AI requirements into relevant state-of-the-art areas

The following table provides a consolidation of the distributed AI requirements identified individually for each use case in the previous subsections, split into relevant state-of-the-art areas to be further explored in Section 4. Related use case specific requirements are grouped together in the third column of this table. The second column provides the state-of-the-art area linked to these requirements. Finally, to ease readability, these areas are further grouped together, when relevant, into higher-level domains.

Domain	**State-of-the-art area**	**Relevant individual UC requirements**
Data preprocessing	Distributed and multi-modal data fusion at the edge	• (UC2) Pre-process raw network traffic datasets at CPE to extract features, depending on the features selected as inputs to the developed ML mechanism • (UC3) Fuse locally data from different cameras capturing the same scene using different spectra and technologies (black and white, near infrared, color, thermal, time-of-flight) and from additional non-visual technologies (radar, sound) • (UC3) Leverage knowledge from other/nearby edge devices (cameras) for dealing with noise and missing data, improving feature extraction, and building of an accurate situational picture • (UC5) Generating time series data set, pre-processing (data cleaning, normalization, standardization, etc.) and labelling operations on time series in real time , at the edge
Computations at the edge	Prediction at the edge	• (UC1) Forecast the future energy production, consumption, and market prices at the edge and execute (part of) the grid optimization service locally
	(Distributed) feature extraction at the edge	• (UC3) Extract optimal features locally from the fused data for further processing and distribution to other edge devices (cameras) • (UC5) Rule-based determination of process priorities that change according to the stage in the process
	Anomaly detection at the edge	• (UC2) Detect anomalous network traffic, i.e., traffic that does not match typical traffic behavior observed at the customer's network • (UC4) Detect anomalies in water consumption data locally, by leveraging

Domain	**State-of-the-art area**	**Relevant individual UC requirements**
		location/building specific information and in a battery-preserving fashion
	Context understanding at the edge	• (UC3) Build locally an accurate situational picture of a road location
	Compression of time series data at the edge	• (UC1) Compress real-time (time series) data locally, with minimal information loss • (UC4) Compress (time series) data locally, with minimal information loss and computation power
Distributed AI	Federated learning	• (UC1) Leverage knowledge from other edge devices for improving forecasts • (UC2) Carry out inference in a decentralized fashion, i.e., ability to run a lightweight model at the edge and a more elaborate one in the cloud or an ISP middlebox • (UC3) Leverage knowledge from other edge devices for dealing with noise and missing data, improving feature extraction, and building of an accurate situational picture • (UC5) Ensuring that both the PID control within the machine itself and the steam usage on the basis of the enterprise can be given to the priority processes
	Transfer learning	• (UC2) Data and models of many customers available at the cloud can be trained into an aggregated model, that is then transferred to the CPE (transfer learning)
	Distributing computations among edge nodes	• (UC3) Distribute computations among edge devices to reduce load on the single devices • (UC5) Performing some of the computations within MES to reduce the load on the edge devices
Security and privacy	Privacy-preserving learning techniques	• (UC2) Protect customer identity and data, both when training and during operation • (UC3) Minimize the sharing of road users' private data outside edge devices and, when needed, preserve their privacy • (UC4) Preserve customers' privacy when manipulating and transferring water consumption data
	Secure data sharing	• (UC1) Provide a secure data sharing solution between edge devices and edge and cloud in view of competitive prosumer information

4. Review of the relevant state-of-the-art

This section provides an overview of the relevant state-of-the-art areas identified in Section 3.6, where subsections and subsubsections correspond respectively to the higher-level domains and state-of-the-art areas identified previously. The references are listed in Section 6.

4.1. Data preprocessing

In ML we can distinguish two processes: training (learning) and inference. The goal of the first process is to develop a model that will be later used to perform inference, i.e. to make predictions given input data.

Training of a model is driven by data. During the training process, data is fed to the model, and, based on the model output, and possibly other information, the model parameters are adjusted. Preprocessing of data is important for model training because the data collected from the application domain is often incomplete or is not encoded in a way to make training efficient or accurate.

Data is also fed to an ML model during the inference process. Because the ML model may take as input data that is derived from the collected raw data, data preprocessing may also take place during the inference process.

We can identify four steps in data preprocessing: data cleaning, integration, transformation and reduction.

The main purpose of data cleaning is to handle missing, noisy, inconsistent and redundant data, all of which reduce the quality of the training data and consequently of the model.

Data integration is needed to merge data from multiple sources into a single data set. Issues that may arise in this process are the use of different formats or attributes by the different sources and redundant data.

Data transformation allows to improve the performance and the accuracy of the training of the ML models. Indeed, the raw cleaned data may not be encoded properly for an efficient and accurate training. Commonly used data transformation techniques include normalization of numerical attributes and attribute selection.

If the data set is very large, it may be helpful to use data reduction to improve the performance of the learning process without affecting its accuracy. Some techniques commonly used are attribute subset selection, discretization and dimensionality reduction.

4.1.1. Distributed and multi-modal data fusion at the edge

Distributed and multi-modal data fusion at the edge level can significantly improve the data quality in a distributed environment. On the one hand, information from different homogenous edge devices can be leveraged, especially when it comes to missing or noisy data. On the other hand, it is possible to combine information from heterogeneous sources. For example, in UC3, it is possible to merge information from the traffic video cameras with the information received from non-visual sensors like a speed measurement via radar [DF1]. Additionally, data fusion can happen at different levels: at the level of raw data, at feature level or decision level [DF2].

One question to answer for both cases is on the actual exchange of data: is the solution *cloud-centric*, meaning that the raw data is uploaded to a cloud environment and from there pushed towards other edge devices, or is there direct communication between the single edge devices. The cloud-centric model can lead to several problems, like latency in communication especially if a huge amount of data is exchanged. In [DF3], a hierarchical automated data fusion architecture was introduced. With this hierarchical fusion process, the importance of different layers of information is considered such that results are available accurately and in a short time. This is especially important if decisions must be taken within a very short time, as for UC3. Further, for UC3, most of the state-of-the-art models are computation-intensive deep learning models. The training efficiency of these models on single-modality data has made great progress. However, reducing the number of used weights in data fusion deep learning models along with training efficiency improvements in multi-modal setup is still an active research direction [DF4].

Data from multiple sensors come with different representations and structures (e.g. traffic video cameras and speed sensors). The information from one sensor (one mode) can act as a soft label for another (intermodality). This information from multiple sensors might have different statistical properties making relational discovery challenging. However, such interrelationships are useful while dealing with noisy and missing data as they act as supplementary information. In addition, there will be complementary information available from these different modes (crossmodality) that should also be exploited for decision making. Intermodal and crossmodal information are abundant in various use cases (e.g. UC3, same road junction monitored at various angles/using different camera types). Current models capture features specific to each modality and further use a combination of these features during the fusion step to improve the final goal, which might not help in capturing the entire semantic relationships properly. A few semantic fusion strategies such as multiview fusion and transfer learning fusion have made some progress in this direction [DF5, DF6]. However, pushing the state-of-the-art in this direction will be critical for UC3. Data fusion methods for both homogeneous and heterogeneous devices should be revisited to take these problems into account.

Data fusion for homogeneous devices: In the case of homogeneous edge devices (e.g. two infrared cameras that cover the same area but from two different points of view) missing data in one device can be replaced by information from the other. This can be very helpful, especially when the chance of incomplete data is high, e.g. something was blocking the camera or due to the angle of sunlight. But also here, it has to be decided on how to fuse data in case of contradictions.

Data fusion for heterogeneous devices: One of the major problems when it comes to the fusion of information from different sensors is synchronization: Is the time exactly configured in each device? At what times do the different sensors provide information? And if the temporal resolution of one sensor is much higher than for another one, it must be decided whether data between two time events should be linearly or constantly interpolated. Questions like these must be answered together with the domain experts of the single use cases.

Further challenges exist in terms of the nature of the data collected from dynamic environments. Non-stationary nature in single modal data is currently solved using transfer learning techniques, which reduces training time requirements of these complex models. However, addressing distribution changes to multi-modal inputs is quite challenging. Online and incremental multi-modal fusion models should be explored to detect input distribution changes and adapt to the new environmental conditions [DF7, DF8].

In textiles process control, currently, very small amount of descriptive analytics (such as for generation of basic operational reports) is performed at the factory MES level. No analytics is taking place at the edge level. What is more, most of the locally generated data is dismissed without ever being analyzed even for basic statistical purposes. PID controllers are implemented on the edge device and their tuning at the moment is performed 100% manually and during installation stages of the factory production line. This situation is also representative of the industry's state-of-the-art level.

4.2. Computations at the edge

4.2.1. Prediction at the edge

Tuning describes the steps taken to ensure that for an input signal the output signal (response) of a process plant is as close to a desired response as possible (this closeness is measured by minimizing a cost function possibly subject to constraints) in order to determine the parameters of a controller that usually sits in the feedback loop of a process control schema [PE5, PE6]. Thus, the plant (i.e. process) response has to be calculated into the future (hence the predictions) for a number of time units (could be counted in seconds or discrete time steps), and the best predicted response will then be back propagated into the estimation of the controller parameters [PE3, PE4]. While this is called optimal controller tuning in the process industries, its steps resemble AI prediction steps in certain ways. In the textiles industry worldwide, controller tuning is performed during the first installation manually. Edge nodes from there on remain as dummy controllers with no capability to self-adapt [PE1, PE5]. What could be considered as "lightweight" data analytics (such as descriptive analytics, basic alarms) then takes place at the MES level. All calculations taking place at MES and practically no calculations taking place at the edge is the current situation in the industry worldwide. This limits the process to optimally adapt to changing conditions, which could be due to mistuning in the first place, drifting process parameters, or changing operating points [PE2]. Potential improvements are likely to improve optimality, process visibility, as well as reduce energy use, depending on the critically of process.

4.2.2. (Distributed) feature extraction at the edge

Optimal feature extraction is unavoidable for vision-based traffic analysis systems, operating at the edge, for two main reasons: privacy preservation of sensitive image data, and bandwidth reduction for collaborative situational awareness and decision making. The best performing vehicle detection and classification models make use of state-of-the-art deep learning models that are both computationally expensive and extract higher-dimensional features. The normal approach to address this problem is to enable edge devices with quick initial feature extraction and classification if the edge model is confident and fall back to the large neural network (NN) model in the cloud to perform final classification. This kind of distributed approach is challenging mainly due to: (i) issues in fitting high accuracy deep models on the edge devices, (ii) large communication costs because of partitioning of deep learning models and (iii) difficulty in coordinated decision making between edge devices and the cloud.

In [FE1] the authors proposed a novel distributed deep neural network (DDNN) with a joint training that minimizes communication overhead and maximizes the usefulness of the extracted features. They make use of an early exit strategy to distribute the load between the edge devices and the cloud. Further, the authors also considered various feature aggregation techniques to select the optimal features and proposed a confidence measure at various exit points based on a normalized entropy measure. A brief survey of the early exit strategies can be found in [FE2]. Even with these available techniques, it is quite challenging to come up with an optimal distributed feature extraction algorithm due to the lack of maturity of the available techniques.

The applicability of the available algorithms for optimal feature extraction will be investigated during the initial stages of MIRAI. Further improvements over the state-of-the-art will be made to enhance the capabilities of multi-exit neural networks taking the network architecture, communication costs and accuracy into consideration. In addition, optimal training, and inference strategies for a subset of edge devices (for example, a couple of cameras monitoring an intersection) will be explored. Adaptive neural network exit strategies based on the inference confidence measure for an edge device subset is also an interesting direction to be investigated.

Within the textiles industry, it is not uncommon to see edge devices holding sufficient amounts of memory and computational power to perform certain ‘advanced’ work, if this is what was expected of them. However, in the current state-of-the-art, local devices are usually not tasked with performing various stages of data analytics such as data preprocessing (including feature extraction), modelling, and etc. UC5’s devices are also on par with the state-of-the-art, meaning that the local devices, although having ‘some’ capabilities to perform some of the advanced analytics, do not perform any feature extraction in the sense of true artificial intelligence.

4.2.3. Anomaly detection at the edge

Anomalies can be caused by errors in the data but can also be indicative of a new, previously unknown, underlying process [AD1]. [AD2] defines an outlier as an observation that deviates so significantly from other observations as to arouse suspicion that it was generated by a different mechanism. Outliers are both rare and unusual: rarity suggests that they have a low frequency relative to non-outlier data, and unusual suggests that they do not fit neatly into the data distribution. In turn, **novelties** are newly occurring unobserved patterns in the data, that were not considered as anomalous data points. A novelty score may be assigned for these previously unseen data points, using a decision threshold score. The points which significantly deviate from this decision threshold may be considered as anomalies or outliers. Techniques used for anomaly detection are often used for novelty detection and vice versa.

Anomalies can take different forms [AD3]:

- **Point anomalies:** Point anomalies often represent an irregularity or deviation that happens randomly and may have no particular interpretation. For example, a big credit transaction which differs from other transactions is a point anomaly.
- **Contextual or Conditional Anomalies:** Some points can be normal in a certain context, while detected as anomaly in another context. Having a daily temperature of 25°C in summer in Germany is normal, while the same temperature in winter is regarded as an anomaly.
- **Collective or Group Anomalies:** There are cases where individual points are not anomalous, but a sequence of points is labeled as an anomaly. For example, a bank customer withdraws $500 from her bank account every day of a week. Although withdrawing $500 occasionally is normal for the customer, a sequence of withdrawals is an anomalous behavior.

Labeled data availability conditions the type of training that can be done. Anomalies are rare occurrences, so having labeled datasets is often hard. Also, anomalous behavior may change over time, for instance, the nature of the anomaly can change so significantly and that it remains unnoticed. We discuss the various options of labeled data availability in the perspective of learning modes; Table 1 provides an overview of the trade-offs.

Supervised: Presumes existence of annotated dataset with labels of both normal and anomalous data instances. Problem maps into 'traditional' binary / multi-class classification problem; as such, well-established classification methods can be used.

Semi-supervised: In practice, datasets of (labeled) normal instances can be obtained with some ease, whereas it is the anomalous data points that are harder to obtain. These techniques leverage existing labels of single (normally positive class) to separate outliers. Semi-supervised or one-class classification anomaly detection (AD) techniques assume that all training instances have only one class label.

- Operating Principle: Semi-supervised techniques learn a discriminative boundary around the normal instances, and test instances that do not belong to the majority class are flagged as anomalous.
- Assumptions: (i) proximity and continuity: points which are close to each other both in input space and learned feature space are more likely to share the same label; (ii) robust features can be learned to separating normal from outlier data points.

Unsupervised: Labeled data is hard to obtain in the first place. Unsupervised deep anomaly detection techniques detect outliers solely based on intrinsic properties of the data instances.

- Operating Principle: Unsupervised anomaly detection algorithm produces an outlier score of the data instances based on intrinsic properties of the dataset, such as distances or densities.
- Assumptions: (i) "normal" regions in the original or latent feature space can be distinguished from "anomalous" regions in the original or latent feature space; (ii) the majority of the data instances are normal compared to the remainder of the data set.

Table 1. Trade-offs of semi-supervised and unsupervised techniques for anomaly detection.

	Advantages	*Disadvantages*

Semi-Supervised	Use of labeled data (usually of one class) can produce **considerable performance improvement over unsupervised techniques**.	The features extracted may not be representative of fewer anomalous instances and hence **prone to the over-fitting problem**.
Unsupervised	• **Learns the inherent data characteristics** to separate normal from an anomalous data point. • **Cost-effective technique** to find the anomalies since it does not require annotated data for training the algorithms.	• Often challenging to learn commonalities within data in a complex and high dimensional space. • **Sensitive to noise and data corruptions**. • Less accurate than supervised or semi-supervised techniques.

Anomaly detection is often also viewed as an instance of the one-class classification (OCC) problem [AD4]. OCC tries to identify objects of a specific class amongst all objects, by primarily learning from a training set containing only the objects of that class. This is different from and more difficult than the traditional classification problem, which tries to distinguish between two or more classes with the training set containing objects from all the classes.

We will now discuss five selected techniques:

Density-Based Clustering [AD5] is an *unsupervised* clustering technique, often called as DBSCAN that can double to identify outliers. Given a set of points over a space, DBSCAN creates clusters by assessing if a point *p* is a cluster point (in DBSCAN, a 'core' point) if at least *minPts* points are within distance *d* of it. Unclustered points can be interpreted as anomalies.

Autoencoders [AD6] are an architecture of neural networks that implement *semi-supervised* learning. Autoencoders represent data within multiple hidden layers by reconstructing the input data, effectively learning an identity function. The autoencoder learns a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore insignificant data ("noise"). Autoencoders can be used for anomaly detection as, when trained solely on normal data instances (which are the majority in anomaly detection tasks), fail to reconstruct the anomalous data samples. The data samples which produce high residual errors are considered outliers. The choice of autoencoder architecture depends on the nature of the data. Convolution networks are preferred for image datasets. Long short-term memory (LSTM) based models tend to produce good results for sequential data. The choice of right degree of compression, i.e., dimensionality reduction is often a hyper-parameter that requires tuning for optimal results.

Sometimes, NN can be concatenated. The second NN is an autoencoder, to learn a representation of the input features (for dimensionality reduction or anomaly detection). The first NN is dedicated to feature extraction, being selected as a function of the nature of the data. Examples: Convolution networks are preferred for image datasets. LSTM based models tend to produce good results for sequential data. An example can be found in [AD7] for the goal of anomaly traffic detection mechanism named D-PACK. As the authors describe, it "consists of a convolutional neural network (CNN) and an unsupervised deep learning model (e.g., Autoencoder) for auto-profiling the traffic patterns and filtering abnormal traffic (…) Notably, D-PACK inspects only the first few bytes of the first few packets in each ow for early detection."

Principal Component Analysis (PCA) [AD8] is an *unsupervised* mechanism used most often for dimensionality reduction. Assumption: data lies on or near a low d-dimensional linear subspace. Axes of this subspace are an effective representation of the data. Identifying those axes is known as Principal Component Analysis, and can be obtained by Eigen or Singular Value Decomposition. In other words, PCA finds the directions of maximal variance in the training data.

One-Class Support Vector Machines are boundary-based mechanisms for *semi-supervised* learning. Traditional Support Vector Machines (SVM) select a decision boundary for which the margin between data points of different classes is maximized. Other interpretation is that SVMs maximize the distance between the convex hulls of points belonging to each class. One-Class SVM (OC-SVM) [AD9] uses a hypersphere to encompass all of the instances (as opposed to using a hyperplane to separate two classes of instances). Data points outside the hypersphere are classified as anomalies.

Isolation forests [AD10] is an *unsupervised* mechanism that, as presented by the authors, while "existing model-based approaches to anomaly detection construct a profile of normal instances, then identify instances that do not conform to the normal profile as anomalies", isolation forests "explicitly isolate anomalies instead of profiles normal points." The core assumption is that fewer partitioning steps are required in order to isolate an anomalous sample in a dedicated partition. Recursively generate partitions on the sample by randomly selecting an attribute Randomly selecting a split value for the attribute, between the minimum and maximum values allowed for that attribute. Anomalies will usually require less partitions to be isolated, i.e., have smaller path lengths.

Table 2 offers a summary comparison of the above five methods and their suitability for deployment on edge devices. *Semi-supervised* methods typically demand *training to be done outside the edge* device.

Table 2. Comparison of selected anomaly detection techniques.

	Type	***Advantages***	***Disadvantages***	***Suitability for edge devices***
Density-based Clustering	Unsupervised	Low computational cost	Selecting best parameter values (and minPts) for normal-class membership	Appropriate for edge devices, including search for best parameter values
Auto-encoder	Semi-supervised	Architectures for different types of data (images, time-series, etc.)	Computationally expensive (both training and inference)	Possible if suited libraries/compiling strategies are used
PCA	Unsupervised	Reasonable computational cost on training	Selecting best threshold value (distance to lower-d hyperplane)	Trained model can be easily implemented on edge
OC-SVM	Semi-supervised	Minimizes convex set of normal data	Computationally expensive on training	Trained model can be easily implemented on edge
Isolation Forests	Unsupervised	Low computational cost	Selecting best threshold value for path length value (used to classify as normal or anomalous)	Appropriate for edge devices

Anomaly Detection in Univariate Time Series

The main assumption about spatial data is that the data points are independent from each other [AD3]. Therefore, anomaly detection happens by either: (i) measuring the deviation of the abnormal points to the rest of the data; (ii) clustering the whole dataset and mark all points as anomalies that lie in less dense regions. In time-series data, it is presumed that data points are not completely independent. It is assumed that the latest data points in the sequence influence their following timestamps. Following this, values of the sequence change smoothly or show a regular pattern. Sudden changes in the sequence can be regarded as an anomaly.

There are a number of time-series patterns to take into consideration: (i) trend: if its mean is not constant but increases or decreases over time; (ii) seasonality: periodic recurrence of fluctuations. Stationarity: stationary time-series is a time-series having the same characteristics over every time interval. A stationary time-series will have: (i) constant mean (thus no trend exists in the time-series); constant variance; (iii) constant autocorrelation over time; and (iv) no seasonality, i.e., no periodic fluctuations. Anomaly detection methods for time-series can be broken down into two main categories [AD11]: (i) anomaly detection based on prediction of the time series and (ii) anomaly detection based on unusual shapes of the time series.

Machine Learning on Edge Devices

The execution of ML models in resource-constrained IoT/edge devices, specifically on microcontrollers, finds use in a variety of applications such as image processing [AD12], gesture-based interaction for cane users [AD13], temperature forecasting [AD14], activity classification [AD15] and room occupancy estimation [AD16]. It is challenging due to the reduced RAM and flash memory available and reduced computing frequency that is typical of such devices. Specific solutions have been proposed, such as: ProtoNN [AD17], a compressed and accurate k-nearest neighbors (kNN) for resource-scarce devices; CMSIS [AD12], optimized software kernels for deploying NNs on Arm Cortex-M CPUs; and SeeDot [AD18], a domain-specific language to express ML algorithms and a compiler of fixed-point code that can efficiently run on constrained IoT devices. Reviewing application-driven works, neural networks tend to be the most targeted ML model for deployment [AD12, AD14, AD16] and kNN [AD13, AD15]. Most works in the field tackle ARM Cortex processors ([AD12, AD13, AD15, AD16]), whereas other works target even smaller microcontrollers such as the MCU 8051 [AD14].

4.2.4. Context understanding at the edge

Context understanding is a general problem in most industrial AI use cases. If the context is unknown or not modelled, this can lead to misleading results. As an example, think about the estimation of risk at a road intersection. Imagine, a bike approaches the intersection from east and a car from south. And now, the risk whether the bike rider and the car driver see each other, and brake early enough, is of interest. This risk varies drastically in case the sun is shining compared to when it is raining. First, the view is worse when it is raining, and, second, on a wet street the braking distance is much longer. Hence, in this example the humidity changes the context for a machine learning model.

On the edge, different approaches can lead to a context-awareness. In the example above, the easiest solution is to add a humidity sensor that provides information to the edge device and define a Boolean variable whether it is either raining or not. This Boolean variable can be considered in the machine learning model that estimates the risk.

Often though, the context cannot be expressed in this binary way but is a continuously and slowly changing variable. In the literature, this refers to concept drift [CU1]. While there are several well-established algorithms for unconstrained devices, on the edge, it is harder to detect concept drift as usually less historical data is available. In the study of [CU2], an evaluation of different well-known online – hence for a continuous data stream – drift detection techniques were performed for different kinds of context changes (abrupt and gradual) and their influence on different machine learning models for time series data. They mainly focused on Page-Hinkley Test [CU3], the adaptive windowing approach (ADWIN) [CU4], Drift Detection Method [CU5] and Early Drift Detection Method [CU6]. They showed that taking the context into account when forecasting electricity flow between different states in Australia, especially using the Page-Hinkley Test, drastically improved the results.

Other points to consider when detecting context at the edge level are missing, noisy, and unbalanced data and further, the level of complexity of the context. In the publication of [CU7], an adversarial autoencoder was implemented that was able to detect the context of human activities. The model the authors introduce can further synthesize and restore missing sensory data to facilitate user context detection. Note though that (re-)training of such a model at the edge can lead to computational problems as data memory and computational power are usually limited.

4.2.5. Compression of time series data

With more and more sensors being installed and collecting data with high temporal resolution, a huge amount of raw data would need to be transferred to a central device to be collected. In industrial use cases, this is challenging but most of the time doable. As soon as devices are in the field though, sending all data in real-time via a mobile connection becomes at least very challenging and expensive.

One way to facilitate data transfer from the edge to a central device is to compress the data before sending. Recently, several algorithms were proposed to handle compression of time series on constrained devices [CTS1, CTS2, CTS3, CTS4, CTS5]. In general, we can distinguish whether:

- all information should be retrieved during decoding (in which case a **lossless** compression is needed) or not (**lossy** compression would be acceptable).
- a training phase on existing data is needed (and the compression should be **adaptive**) or not (**non-adaptive** compression).
- the same algorithm is used for encoding and decoding (**symmetric** compression) or not (**asymmetric** compression).
- the data is compressed using a fixed or variable number of bits/bytes (**bit-/byte-level** compression).

The decision on which compression algorithm is the best therewith strongly depends on the use case. Furthermore, especially in constrained devices, the best balance between different quality measures has to be found:

- **Is a higher compression ratio more important than the computational cost?** In this case, if the field device is connected to the electricity network and has sufficient storage this is probably the case. In contrast, if the device is stationary, such that a stable connection bandwidth can be guaranteed, but runs on a battery, the computational cost or limited storage capacity would be rather more decisive.
- **Is the accuracy (or distortion) more important than the compression speed?** In this case, if the decision has to be taken very quickly or in near real-time, it might be preferable to lose some information during a lossy compression but therefore have a high speed, while the contrary is true for data where small deviations are important, but the decision has some more time to be taken, resulting in lossless compression as the preferred option.

For time series data compression on constrained devices, the most important classes of compression algorithms are:

- **Dictionary-based algorithms**, where common sequences in the data are encoded using a dictionary. This dictionary is stored in the edge device as well as in the central device. The streaming data on the edge will be encoded by matching pre-defined sequences in the dictionary values, e.g. the subsequence 056010 can be a dictionary value with key “a”. In this case, only the dictionary key “a” is sent from the edge to the central device where it can be decoded again into the sequence. The compression rate depends on the length of the defined sequences and on the dictionary size, as with more dictionary entries, also the dictionary keys increase in size. Note, that with increasing dictionary size, also the computation cost increases on the edge device to find the corresponding sequence. Well-known dictionary-based algorithms are TRISTAN [CTS6] and CONRAD [CTS7].
- **Function Approximation algorithms**, where the idea is to approximate the data stream by a function. As this is usually not feasible over the full data stream, the data is segmented in time and a piece-wise approximation is performed. The functions in use are often linear functions [CTS8], polynomials [CTS9] and discrete wavelets [CTS10].
- **Sequential algorithms**, which consist of easy compression techniques used sequentially. Often, they use delta or run-length encoding. The first encodes a time series by storing the difference between the current and the last value instead of sending the actual value, which can be beneficial for large values that only slightly change over time. The second stores how often a single value occurs and saves the value and the number of repetitions, which is beneficial for long sequences of constant values. Additionally, Huffman encoding is often used

to further decrease the size of stored data. The encoder creates a dictionary that associates each symbol to a binary representation and replace each symbol of the original data with the corresponding representation [CTS1]. One sequential algorithm that is designed for IoT scenarios is the so-called Sprintz algorithm [CTS11]. It focuses on energy-efficiency and speed by combining forecasting, bit packing, run-length encoding and entropy coding. In case forecasting is too expensive, either due to computational or memory cost, Run-Length Binary Encoding (RLBE) [CTS5] is a lossless technique that combines different encodings in five consecutive steps. It was first used for smart meter data and therewith is designed for constrained devices. Another way to reduce the size of a single value is given by transforming them to binary code words. One example for this is the so-called Fibonacci encoding that can be used as one step in the sequential algorithms with the nice advantage that single values do not need an explicit separator but a series of values can be encoded as one long string [CTS12].

In the context of MIRAI, compression of time series data is considered in two use cases, namely in UC1, where real-time data is collected locally and should be transferred with minimal information loss, and in UC4, where the time series data is also being compressed locally with minimal information loss and minimal computation power.

In UC4 data compression should lead to fewer messages to be sent in order to increase battery lifetime of the field devices. As the devices are stationary, we can assume that the bandwidth is stable. Further, we know that small deviations in the data can make a big difference to detect water leakages. Therefore, **Function Approximation** algorithms are not suitable for UC4 as they often smooth minor changes.

Dictionary-based algorithms could be used and trained on a central device. This could lead to the identification of common patterns either for a single device or for groups of devices. Keep in mind though, that in this case the dictionary (and possibly dictionary updates) would need to be sent to the edge device, either once or periodically. This could lead to high energy consumption and storage requirements on the device, which is counter-productive for increasing the battery lifetime in UC4.

Most suitable for UC4 are **Sequential algorithms**. They are usually very efficient in terms of computation power and memory, and are lossless. Furthermore, in case of small but persistent value changes in the data, which often indicate a leakage in UC4, more messages would be sent, which directly leads to improvements in the detection time.

4.3. Distributed AI

4.3.1. Federated learning

Federated Learning (FL) [FL1, FL2] aims at the development of a high-quality centralized model by aggregating updates provided by multiple edge nodes. While some learning takes place at the edge nodes, the requirement on the quality of training at the edge is alleviated, allowing to benefit of techniques for operation in resource-constrained platforms. In turn, and leveraging a transfer learning approach, the high-quality centralized model can be transferred back to the edge nodes for inference operation. However, this option requires identifying and developing an ML algorithm that can learn a shared model from local updates. Also, edge-specific aspects captured in the edge updates may be eroded when computing the global model.

The iterative nature of FL requires massive communication between the central server and edge devices to train a global model [FL2]. The communication overhead at each iteration is not negligible, especially for complex models, large scale applications, and high frequency updates, and becomes a challenge to be addressed [FL1, FL2, FL3]. Many studies that aim at reducing communication costs have been recently proposed. For example, [FL4] use models of different sizes to address heterogeneous clients equipped with different computation and communication capabilities, while the work in [FL5] uses collaborative decentralized learning in combination with the master-slave model. Majority solutions that address the problem of reducing network overhead in FL can be classified in two main categories. The first category incorporates works that *reduce the total number of bits* transferred for each local update, by means of data compression. Studies that aim at *reducing the number of local updates* during the training process are included in the second category.

FL promises to deliver model results developed and updated at the edge nodes. In UC5the goal is to improve the capability of the edge nodes to provide self-tuning of the PID controllers. Furthermore, the current device specs may at time prohibit the deployment of computationally demanding FL techniques. Therefore, currently the FL aspect of the developments for the textiles process control use case is deferred until the second half of the project, approximately when we expect to get the initial results of the automated PID tuning first.

Approaches that reduce the total number of bits

[FL6] propose an enhanced FL technique by introducing an asynchronous learning strategy on the clients and a temporally weighted aggregation of the local models on the server. Different layers of the deep neural networks are categorized into shallow and deep layers and the parameters of the deep layers are updated less frequently than those of the shallow layers. In addition, a temporally weighted aggregation strategy is applied on the server to make use of the previously trained local models, thereby enhancing the accuracy and convergence of the central model. [FL7] design two novel strategies to reduce communication costs. The first is relied on the use of lossy compression on the global model sent server-to-client. The second strategy uses Federated Dropout, which allows users to efficiently train locally on smaller subsets of the global model and also provides a reduction in both client-to-server communication and local computation. [FL8] propose Deep Gradient Compression (DGC) to greatly reduce the communication bandwidth. The authors of [FL9] introduce a new compression framework, entitled Sparse Ternary Compression, that is specifically designed to meet the requirements of the FL environment. [FL10] implement a Federated Optimisation (FedOpt) approach by designing a novel compression algorithm, entitled Sparse Compression Algorithm (SCA) for efficient communication, and then integrate the additively homomorphic encryption with differential privacy to prevent data from being leaked. [FL11] develop a novel framework, that significantly decreases the size of updates while transferring weights from the deep learning model between clients and their servers. A novel algorithm, namely FetchSGD, that compresses model updates using a Count Sketch, and takes advantage of the mergeability of sketches to combine model updates from many workers, is proposed in [FL12]. [FL13] present a federated trained ternary quantization (FTTQ) algorithm, which optimizes the quantized networks on the clients through a self-learning quantization factor.

Approaches that reduce the local updates

A novel FedMed method with adaptive aggregation using topK strategy to select the top workers who have lower losses to update model parameters in each round is proposed in [FL14]. [FL15] have provided a novel filtering procedure on each local update and transfer only the important gradients to the server. The study proposed by Wang et al. [FL16] identifies the relevant updates of participants and upload them only to the server. In particular, at each round, the participants receive the global tendency and checks the relevancy of their local updates with the global model and upload only if they align. An FL protocol of two-step client selection based on their resource constraints instead of the random client selection is proposed in [FL17]. In addition, a global model update algorithm, namely FedPSO proposed to transmit the model weights only for the client that has provided the best score (such as accuracy or loss) to the cloud server [FL18]. However, using conventional approaches such as quantization and sparsification is less helpful. It is vital to find out more efficient FL schemes other than FedAvg which converge with the same speed as FedAvg and apply to any FL applications [FL19]. For example, the studies in [FL20, FL21] have explored an approach that applies clustering optimization to bring efficiency and robustness in FL's communication. The most representative updates are only uploaded to the central server for reducing network communication costs.

4.3.2. Transfer learning

In turn, Transfer Learning [TL1] aims to improve the performance of learners on target domains by transferring the knowledge contained in different but related source domains. In this manner, the dependence on a large number of target-domain data can be reduced for constructing target learners. In the context of distributed edge/cloud learning, transfer learning can refer to process of leveraging a centralized collection of datasets of similar source processes to create generic models, that are later fine-tuned to each edge node by complementing training at the node. This modular training that some ML algorithms can perform allows to ease the training requirements on the edge node. For example, in UC2 neural networks can be trained by the operator using large datasets from multiple households with similar traffic patterns [TL2]. At each household, the generic model can be refined by just retraining the last layer of neurons.

4.3.3. Distributing computations among edge nodes

In textiles process control, local controllers control the assigned processes with knowledge of their setpoints and output errors, as well as an initial prescription from the MES system. However, they have no ongoing AI computation at the edges nor do they incorporate important external data such as steam availability or the batch priority. This leads to each node currently acting totally independently of one another, leading to suboptimal control consequences especially in the case of insufficient steam resources or poorly tuned PID controller parameters. A decentralized control structure where thermal regulation in buildings is controlled in a predictive manner is presented in [DC1]. How PID control can benefit from smart actuator and fieldbus technologies for a distributed system is explored in [DC2]. A real-time decentralized and distributed control schemes for Heating Ventilation and Air Conditioning (HVAC) systems in energy efficient buildings where a thermal dynamic model of building systems and a steady-state resource allocation problem are also introduced is designed in [DC3]. A scalable control method for multizone HVAC systems with the objective to reduce energy cost while satisfying thermal comfort and indoor air quality simultaneously is studied in [DC4].

4.4. Security and privacy

4.4.1. Privacy-preserving learning techniques

Privacy Threat Model and Attack Taxonomy

Looking concretely at privacy threat models, the following actors and their scope of operation are involved [PP1]: (i) data owners, whose data may be sensitive, (ii) model owners, which may or may not own the data and may or may not want to share information about their models, (iii) model consumers, that use the services that the model owner exposes, usually via some sort of programming or user interface, and (iv) adversaries, that may also have access to the model's interfaces as a normal consumer does, e.g. feeding new input data for inference. If the model owner allows, they may have access to the model itself.

The following types of attacks on data privacy can be identified [PP1]: Membership inference attacks and related Shadow models attacks, Reconstruction attacks, Property inference attacks, andModel extraction attacks.

Membership Inference Attacks: Membership inference tries to determine whether an input sample ***x*** was used as part of the training set *D*. This attack only assumes knowledge of the model's output prediction vector (typically a black-box attack), and targets supervised machine learning models (and generative models such as GANs and VAEs). In the case of a white-box attack, i.e., if attacker has access to the model parameters and gradients, results may be more accurate.

Implementation of Membership inference attacks is done through Shadow Model Attacks, explained in the next section. Finally, the following defenses can be explored: (i) Differential Privacy: if two databases differ only by one record and are used by the same algorithm (or mechanism), the output of that algorithm should be similar. Differential privacy offers a trade-off between privacy protection and utility or model accuracy; (ii) Regularization techniques: aim to reduce overfitting and increase model generalization performance; (iii) Prediction vector tampering: as many models assume access to the prediction vector during inference, one can restrict the output to the top k classes or predictions of a model.

Shadow Models Attacks: Some supervised learning attacks use shadow models and meta-models to infer membership from target datasets / target models [PP2]. The main intuition behind such attacks is that models behave differently when they see data that does not belong to the training dataset. The objective of the attacker is to construct an attack model that can recognize such differences in the target model's behavior, and use them to distinguish members from non-members of the target model's training dataset (based solely on the target model's output). The main idea is as follows:

1. The adversary trains shadow models (one per output class of the target model) using shadow datasets $D_{shadow} = \{\boldsymbol{x}_{shadow,i}, \boldsymbol{y}_{shadow,i}\}_{i=1}^{n}$, that are of the same format and distribution as the target dataset.
2. After training the shadow models, the adversary constructs an attack dataset $D_{attack} = \{f_i(x_{shadow,i}), y_{shadow,i}\}_{i=1}^{n}$, where fi is the respective shadow model.
    a. For all records in the training dataset of a shadow model, the model is queried and obtain the output. These output vectors are labeled “in” and added to the attack model's training dataset.
    b. The shadow model is also queried with a test dataset disjoint from its training dataset. The outputs on this set are labeled “out” and also added to the attack model's training dataset.
3. The attack dataset is used to train the meta-model. As the attack dataset reflects the black-box behavior of the shadow models on their training and test datasets, the meta-model will be essentially performing inference based on the outputs of the shadow models. In fact, a collection of c_{target} meta-models is trained, one per each output class of the target model.
4. The trained meta-model is used for testing using the outputs of the target model.

The results in (Shokri et al., 2017) show that learning how to infer membership in shadow models' training datasets (for which we know the ground truth and can easily compute the cost function during supervised training) produces an attack model that successfully infers membership in the target model's training dataset, too.

Reconstruction Attacks: Reconstruction attacks try to recreate one or more training samples and/or their respective training labels. These are also referred to as attribute inference or model inversion: given output labels and partial knowledge of some features, the attacker tries to recover sensitive features or the full data sample.

There are two types of such attacks: those that aim at an actual reconstruction of the data, and those that aim to create class representatives or probable values of sensitive features that do not necessarily belong to the training dataset (e.g., faces of a person).

The implementation of reconstruction attacks assumes that the adversary has access to the model f, the priors of the sensitive and non-sensitive features, and the output of the model for a specific input x. The attack is based on estimating the values of sensitive features, given the values of non-sensitive features and the output label. This method uses a maximum a posteriori (MAP) estimate of the attribute that maximizes the probability of observing the known parameters.

Finally, the defense against such attacks leverages the fact reconstruction attacks often require access to the loss gradients during training. Hence, most proposed defenses against these attacks propose techniques that affect the information retrieved from these gradients.

Property Inference Attacks: Property inference is the ability to extract dataset properties which were not explicitly encoded as features or were not correlated to the learning task. Property inference aims to extract information that was learned from the model unintentionally – even well generalized models may learn properties that are relevant to the whole input data distribution and sometimes this is unavoidable or even necessary for the learning process – or that may be used to gain insights about the training dataset.

Some examples are extraction of information about the ratio of women and men in a patient dataset, when this information was not an encoded attribute or a label of the dataset, or a neural network that performs gender classification and can be used to infer if people in the training dataset wear glasses or not. Property inference target either dataset-wide properties or emergence of properties within a batch of data (on the collaborative training of a model).

Model Extraction Attacks: Model extraction is a class of black-box attacks where the adversary tries to extract information and potentially fully reconstruct a model by creating a substitute model f' that behaves very similarly to the model under attack f. There are two types of model extraction attacks:

- Task accuracy extraction: Create models that match the accuracy of target model f in a test set that is drawn from the input data distribution and related to the learning task. The adversary is interested in creating a substitute that learns the same task as the target model equally well.
- Fidelity extraction: Create a substitute model f' that matches f at a set of input points that are not necessarily related to the learning task. The adversary aims to create a substitute that replicates the decision boundary of f as faithfully as possible.

In both cases, it is assumed that the adversary wants to use as few queries as possible. Knowledge of the target model architecture is not strictly necessary, if the adversary's substitute model of same or higher complexity than model under attack. Some works focus on recovering information from the target model, e.g., hyper-parameters in the objective function; in neural networks: activation types, optimization algorithm, number of layers, etc.

Privacy-preserving learning techniques

Ensemble of Teacher Models [PP3]: The motivation for this proposal is that models may inadvertently or implicitly store some of its training data; careful analysis of the model may reveal sensitive data. Enter Private Aggregation of Teacher Ensembles (PATE), an approach to providing strong privacy guarantees for training data. PATE combines, in a black-box fashion, multiple models trained with disjoint datasets (e.g., records of different subsets of users). As the models rely directly on sensitive data, they are not published, but instead used as "teachers" for a "student" model. The student learns to predict an output chosen by noisy voting among all of the teachers, and cannot directly access an individual teacher or the underlying data or parameters.

The student's privacy properties can be understood both intuitively (since no single teacher and thus no single dataset dictates the student's training) and formally, in terms of differential privacy. These properties hold even if an adversary can not only query the student but also inspect its internal workings. Compared with previous work, the approach imposes only weak assumptions on how teachers are trained: it applies to any model, including non-convex models like DNNs.

Differential Privacy [PP4]: Differential privacy constitutes a strong standard for privacy guarantees for algorithms on aggregate databases. It is defined in terms of the application-specific concept of adjacent databases. Consider each training dataset is a set of image-label pairs: two of these sets are adjacent if they differ in a single entry, i.e., if one image-label pair is present in one set and absent in the other.

Definition: A randomized mechanism M: D → R with domain D and range R satisfies (ε, δ)-differential privacy if for any two adjacent inputs d, d0 ∈ D and for any subset of outputs S ⊆ R it holds that

$$\Pr\big[M(d) \in S\big] \leq e^{\varepsilon} \Pr\big[M(d') \in S\big] + \delta$$

A common paradigm for approximating a deterministic real-valued function f: D → R with a differentially private mechanism is by adding noise calibrated to f's sensitivity S_f, which is defined as the maximum of the absolute distance $|f(d) - f(d')|$ where d and d' are adjacent inputs. Consider the Gaussian noise mechanism defined by $M(d) \triangleq f(d) + N(0, S_f^2\sigma^2)$. A single application of the Gaussian mechanism to function f of sensitivity S_f satisfies (ε,δ)-differential privacy if $\delta \geq \frac{4}{5}\exp(-(\sigma\varepsilon)^2/2)$ and $\varepsilon > 1$.

Differential privacy for repeated applications of additive noise mechanisms follows from composition theorems. The task of keeping track of the accumulated privacy loss in the course of execution of a composite mechanism, and enforcing the applicable privacy policy, can be performed by the privacy accountant.

As a concrete example, consider the Differentially Private Stochastic Gradient Descent Algorithm. A method for training a model with parameters by minimizing empirical loss function $\mathcal{L}(\theta)$. At each step of the SGD: (i) compute the gradient $\nabla_\theta \mathcal{L}(\theta, x_i)$ for a random subset of examples; (ii) clip the ℓ_2 norm of each gradient; (iii) compute the average; (iv) add noise in order to protect privacy; (v) take a step in the opposite direction of this average noisy gradient; and (vi) compute privacy loss of the mechanism *(based on the information maintained by the privacy accountant)*.

Distributed Learning [PP5]: In distributed or federated learning: (i) nodes train with local data and produce model updates; (ii) model updates are leveraged to train a global model; and (iii) nodes download globally-improved models or parameters. Distributed/federated learning offers an inherent level of privacy, as local data does not leave the respective node. The work of [PP5] leverages and applies this architecture to enable multiple parties to jointly learn an accurate neural network model for a given objective, without sharing their input datasets. The training of neural networks relies to a great extent on Gradient Descent; the authors propose Selective Stochastic Gradient Descent (SSGD). The main intuition behind selective parameter update is that during SGD, some parameters contribute much more to the neural network's objective function and thus undergo much bigger updates during a given iteration of training.

Classification over Encrypted Data [PP6]: Computations can be performed over encrypted data – this is referred to as Homomorphic computing. Authors design and present three private classifiers that satisfy this privacy constraint: hyperplane decision, Naïve Bayes, and decision trees. To enable their operation, the authors define a library of building blocks with which the classifiers are implemented: (i) comparison with unencrypted inputs; (ii) comparison with encrypted inputs; (iii) reversed comparison over encrypted data; (iv) negative integers comparison and sign determination; (v) argmax (an operation that finds the argument that gives the maximum value from a target function) over encrypted data; (vi) computing dot products; and (vii) dealing with floating point numbers. After describing the building blocks, the authors present protocols to implement the 3 classifiers:

(i) Private hyperplane decision: this classifier computes. The protocol proposed by the authors requires *k - 1* encrypted comparisons of *L* bits integers, *7(k - 1)* homomorphic operations (refreshes, multiplications, subtractions), and *k - 1* roundtrips to the comparison protocol.

(ii) Secure Naïve Bayes classifier: also based on argmax.

(iii) Private decision trees: decision trees can be described by polynomials, and the building blocks can be used to create a protocol to execute decision trees.

Trusted Execution [PP7]: In Machine Learning-as-a-Service (MLaaS) contexts, the service provider is free to choose the type of the model to train, how to configure and train it, and what transformations, if any, to apply to the inputs into the model. These choices can adaptively depend on the user's data and ML task. The user obtains API access to the trained model but no other information about it.

Chiron is a proposal of a privacy-preserving architecture for MLaaS. It conceals the training data from the service operator, as:

- Service provider provides enclaves to user. Code in an enclave can safely operate on secret data without fear of unintentional disclosure to the platform. A commercial example of enclaves is Intel Software Guard Extensions (SGX).
- User establishes secure communication channels with enclaves to transfer data directly. Chiron reveals neither the training algorithm nor the model structure to the user, providing only black-box access to the trained model.

The operation of the training enclave is the following:

1. Untrusted service provider code examines data, generates model spec and passes it to the ML toolchain.
2. ML toolchain uses the spec to generate model-training code.
3. Service provider code transforms data and breaks it into batches for training.
4. Model-training code is invoked for each batch, updating the model.

To enforce data confidentiality while allowing the provider to select, configure, and train a model any way they want, Chiron employs a Ryoan sandbox (itself based on the hardware-protected enclave). An enclave alone is insufficient because it only protects trusted code executing on an untrusted platform. Code can only be trusted if it is public and thus can be checked by users. In Chiron, however, the ML service provider's code is untrusted, thus users must be assured that this code is not stealing their data even though they cannot inspect it. Ryoan [PP8] enables service providers to keep proprietary code secret while simultaneously ensuring users that the confined code cannot leak their data. Instead of asking users to trust the provider's code, Ryoan asks them to trust the sandbox that confines this code. Users can audit Ryoan to gain confidence in its correctness.

4.4.2. Secure data sharing

To be able to provide end-to-end secure data sharing, three important points need to be considered: (1) an end-to-end secure communication channel, (2) protection of data in transit, and (3) access control to data at rest by users and/or applications.

There are many existing protocols and mechanisms available to be used for assuring secure end-to-end communication channel, such as TLS/SSL running on TCP protocol and DTLS running on UDP protocol. However, these transport layer security protocols are designed to be used in general purpose systems with unconstrained devices in mind. When it comes to constrained devices, they become inadequate. This is because these traditional protocols require too many processing steps and rely on advanced encryption algorithms, involving significant computing power and memory. On a device where resources such as power (e.g., battery) are scarce and need to be optimized, such protocols cannot be used. Given that, there is a need for alternative lightweight protocols adapted to constrained environments such as the one proposed in [DS1], iTLS. However, the key challenge is to combine different security algorithms and make them suitable for constrained environments in the scope of the MIRAI project, taking into account the energy (battery), computing power and memory scarcity as the evaluation criteria.

Having a secure communication channel is only a first step in realizing secure data sharing, as data needs also to be protected before sending. Many existing encryption algorithms/protocols such as AES, triple DES, RSA, HMAC [DS2] and homomorphic encryption [DS3] allowing encrypted data to be processed without decrypting it, are not designed to be used in constrained devices. They target the general-purpose systems where there is no limitation on computing power, memory, and energy. Reduced versions of these protocols, such as power-efficient AES [DS4] and some other security algorithms in [DS5, DS6] designed for constrained devices, exist. However, the challenge is to combine a security algorithm with the communication security protocol and tailor these to fit with the requirements in MIRAI.

The final step in the process of secure data sharing is to ensure that only authorised entities (physical users or apps) are allowed to access the data both in transit and at rest. To do so, fine-grained data access control for data sharing, reliable data access control and enforcement solutions are required. The envisioned access control mechanism must be lightweight and able to run on very resource-constrained edge devices, while also being able to meet the corresponding security requirements. As of today, most access control systems, such as OAuth [DS7], OpenID [DS8], run on high-performance devices with complex processing steps. There are many attempts to extend Oauth for use on resource constrained devices, such as in a IoT system [DS9, DS10]. However, looking into their processing steps that involve heavy encryption algorithms, they are still considered as heavy schemes. Our goal is to investigate new lightweight access control and enforcement mechanisms for data sharing in distributed edge analytics for very resource-constrained devices, devices with limited power and memory, and that require long battery lifespan. We will take the following requirements into account when designing these mechanisms: (1) fine-grained and privacy-aware access control, (2) low computing power, and (3) few processing steps.

5. Conclusion

This deliverable was the outcome of the work carried out under Task 2.1 of work package 2. Being the first in a line of tasks in the work package, a consortium-wide contribution was placed in the preparation, and it is expected to help lay the groundwork for numerous tasks to be undertaken in the remainder of the project.

6. References

6.1. Distributed and multi-modal data fusion at the edge

[DF1] Munir, Arslan, Erik Blasch, Jisu Kwon, Joonho Kong, and Alexander Aved. "Artificial intelligence and data fusion at the edge." *IEEE Aerospace and Electronic Systems Magazine* 36, no. 7 (2021): 62-78.

[DF2] K. Kenda, B. Kažič, E. Novak and D. Mladenić, "Streaming data fusion for the internet of things," *Sensors*, vol. 19, no. 8, p. 1955ff, 2019.

[DF3] R. Dautov, S. Distefano and R. Buyya, "Hierarchical data fusion for Smart Healthcare," *Journal of Big Data*, vol. 6, no. 1, pp. 1-23, 2019.

[DF4] Gao, Jing, Peng Li, Zhikui Chen, and Jianing Zhang. "A survey on deep learning for multimodal data fusion." *Neural Computation* 32, no. 5 (2020): 829-864.

[DF5] Fadadu, Sudeep, Shreyash Pandey, Darshan Hegde, Yi Shi, Fang-Chieh Chou, Nemanja Djuric, and Carlos Vallespi-Gonzalez. "Multi-view fusion of sensor data for improved perception and prediction in autonomous driving." *arXiv preprint arXiv:2008.11901* (2020).

[DF6] Ghaith, Iyad H., Aseel Rawashdeh, and Shadi Al Zubi. "Transfer Learning in Data Fusion at Autonomous Driving." In *2021 International Conference on Information Technology (ICIT)*, pp. 714-718. IEEE, 2021.

[DF7] Sun, Fuchun, Huaping Liu, Chao Yang, and Bin Fang. "Multimodal Continual Learning Using Online Dictionary Updating." *IEEE Transactions on Cognitive and Developmental Systems* 13, no. 1 (2020): 171-178.

[DF8] Wang, Kai, Luis Herranz, and Joost van de Weijer. "Continual learning in cross-modal retrieval." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3628-3638. 2021.

6.2. Prediction at the edge

[PE1] Yang, Yu & Srinivasan, Seshadhri & Hu, Guoqiang & Spanos, Costas. (2020). "Distributed Control of Multi-zone HVAC Systems Considering Indoor Air Quality".

[PE2] Zhang, Xuan & Shi, Wenbo & Yan, Bin & Malkawi, Ali & Li, Na. (2017). "Decentralized and Distributed Temperature Control via HVAC Systems in Energy Efficient Buildings".

[PE3] D. Pamela, K. Gerard Joe Nigel, P. Kingston Stanley, Stephy Mol Mathew, "Monitoring and control of various process using distributed control system", Materials Today: Proceedings, 2021, ISSN 2214-7853, https://doi.org/10.1016/j.matpr.2020.11.976.

[PE4] Petru-Daniel Moroşan, Romain Bourdais, Didier Dumur, Jean Buisson, "Building temperature regulation using a distributed model predictive control", Energy and Buildings, Volume 42, Issue 9, 2010, Pages 1445-1452, ISSN 0378-7788, https://doi.org/10.1016/j.enbuild.2010.03.014.

[PE5] W. W. Shein, Y. Tan and A. O. Lim, "PID Controller for Temperature Control with Multiple Actuators in Cyber-Physical Home System", 2012 15th International Conference on Network-Based Information Systems, 2012, pp. 423-428, doi: 10.1109/NBiS.2012.118.

[PE6] Dongik Lee, Jeff Allan, Haydn A. Thompson, Stuart Bennett, "PID control for a distributed system with a smart actuator", Control Engineering Practice, Volume 9, Issue 11, 2001, Pages 1235-1244, ISSN 0967-0661, https://doi.org/10.1016/S0967-0661(01)00069-7.

6.3. (Distributed) feature extraction at the edge

[FE1] Teerapittayanon, Surat, Bradley McDanel, and Hsiang-Tsung Kung. "Distributed deep neural networks over the cloud, the edge and end devices." *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017.

[FE2] Scardapane, Simone, Michele Scarpiniti, Enzo Baccarelli, and Aurelio Uncini. "Why should we add early exits to neural networks?." *Cognitive Computation* 12, no. 5 (2020): 954-966.

6.4. Anomaly detection at the edge

[AD1] Chalapathy, R., & Chawla, S. (2019). Deep Learning for Anomaly Detection: A Survey. *ArXiv:1901.03407 [Cs, Stat]*. http://arxiv.org/abs/1901.03407

[AD2] Hawkins, D. (1980). *Identification of Outliers*. Chapman and Hall.

[AD3] Braei, M., & Wagner, S. (2020). Anomaly Detection in Univariate Time-series: A Survey on the State-of-the-Art. *ArXiv:2004.00433 [Cs, Stat]*. http://arxiv.org/abs/2004.00433

[AD4] Moya, M. M., & Hush, D. R. (1996). Network constraints and multi-objective optimization for one-class classification. *Neural Networks*, *9*(3), 463–474. https://doi.org/10.1016/0893-6080(95)00120-4

[AD5] Sander, J., Ester, M., Kriegel, H.-P., & Xu, X. (n.d.). Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications. 30.

[AD6] Kiran, B., Thomas, D., & Parakkal, R. (2018). An Overview of Deep Learning Based Methods for Unsupervised and Semi-Supervised Anomaly Detection in Videos. *Journal of Imaging*, *4*(2), 36. https://doi.org/10.3390/jimaging4020036

[AD7] Hwang, R.-H., Peng, M.-C., Huang, C.-W., Lin, P.-C., & Nguyen, V.-L. (2020). An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection. *IEEE Access*, *8*, 30387–30399. https://doi.org/10.1109/ACCESS.2020.2973023

[AD8] Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *2*(11), 559–572.

[AD9] Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., & Platt, J. C. (n.d.). *Support Vector Method for Novelty Detection*. 7.

[AD10] Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*, 413–422. https://doi.org/10.1109/ICDM.2008.17

[AD11] Aggarwal, C. C. (2016). *Outlier Analysis* (2nd edition). Springer Publishing Company, Incorporated.

[AD12] L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs," arXiv:1801.06601 [cs], Jan. 2018.

[AD13] S. G. Patil, D. K. Dennis, C. Pabbaraju, N. Shaheer, H. V. Simhadri, V. Seshadri, M. Varma, and P. Jain, "GesturePod: Enabling On-device Gesture-based Interaction for White Cane Users," in Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology. New Orleans LA USA: ACM, Oct. 2019, pp. 403–415.

[AD14] J. Pardo, F. Zamora-Martínez, and P. Botella-Rocamora, "Online Learning Algorithm for Time Series Forecasting Suitable for Low Cost Wireless Sensor Networks Nodes," Sensors, vol. 15, no. 4, pp.9277–9304, Apr. 2015.

[AD15] V. M. Suresh, R. Sidhu, P. Karkare, A. Patil, Z. Lei, and A. Basu, "Powering the IoT through embedded machine learning and LoRa," in 2018 IEEE 4th World Forum on Internet of Things (WF-IoT). Singapore: IEEE, Feb. 2018, pp. 349–354.

[AD16] C. Leech, Y. P. Raykov, E. Ozer, and G. V. Merrett, "Real-time room occupancy estimation with Bayesian machine learning using a single PIR sensor and microcontroller," in 2017 IEEE Sensors Applications Symposium (SAS). Glassboro, NJ, USA: IEEE, 2017, pp. 1–6.

[AD17] C. Gupta, A. S. Suggala, A. Goyal, H. V. Simhadri, B. Paranjape, A. Kumar, S. Goyal, R. Udupa, M. Varma, and P. Jain, "ProtoNN: Compressed and Accurate kNN for Resource-scarce Devices," p. 16.

[AD18] S. Gopinath, N. Ghanathe, V. Seshadri, and R. Sharma, "Compiling KB-sized machine learning models to tiny IoT devices," in Proceedings of the 40th ACM SIGPLAN Conference on

Programming Language Design and Implementation. Phoenix AZ USA: ACM, Jun. 2019, pp. 79–95.

6.5. Context understanding at the edge

[CU1] Ž. Indrė, M. Pechenizkiy and J. Gama, "An overview of concept drift applications," *Big data analysis: new algorithms for a new society*, pp. 91–114, 2016.

[CU2] H. Mehmood, P. Kostakos, M. Cortes, T. Anagnostopoulos, S. Pirttikangas and E. Gilman, "Concept drift adaptation techniques in distributed environment for real-world data streams," *Smart Cities*, vol. 4, no. 1, pp. 349–371, 2021.

[CU3] E. S. Page, "Continuous Inspection Schemes," *Biometrika*, vol. 41, no. 1/2 p. 100–115, 1954.

[CU4] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing.," in *Proceedings of the 2007 SIAM international conference on data mining*, 2007.

[CU5] J. Gama, P. Medas, G. Castillo and P. Pereira Rodrigues, "Learning with Drift Detection," *SBIA*, pp. 286-295, 2004.

[CU6] M. Baena-Garcia, J. Del Campo-Avila, R. Fidalgo, A. Bifet, R. Gavalda and R. Morales-Bueno, "Early Drift Detection Method," in *Fourth International Workshop on Knowledge Discovery from Data Streams*, 2006.

[CU7] A. Saeed, T. Ozcelebi and J. Lukkien, "Synthesizing and reconstructing missing sensory modalities in behavioral context recognition," *Sensors*, vol. 18, no. 9, p. 2967, 2018.

6.6. Compression of time series data

[CTS1] G. Chiarot and C. Silvestri, "Time series compression: a survey," in *arXiv preprint arXiv:2101.08784*, 2021.

[CTS2] A. L. Fitriya, T. W. Purboyo and A. L. Prasasti, "A review of data compression techniques," *International Journal of Applied Engineering Research*, vol. 12, no. 19, p. 8956–8963, 2017.

[CTS3] P. Kavitha, "A Survey on Lossless and Lossy Data Compression Methods," *Engineering Technology*, vol. 7, no. 03, p. 5, 2016.

[CTS4] T. Lu, W. Xia, X. Zou and Q. Xia, "Adaptively compressing IoT data on the resource-constrained edge," in *3rd {USENIX} Workshop on Hot Topics in Edge Computing (HotEdge 20)*, 2020.

[CTS5] J. Spiegel, P. Wira and G. Hermann, "A comparative experimental study of lossless compression algorithms for enhancing energy efficiency in smart meters," in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, 2018.

[CTS6] A. Marascu, P. Pompey, E. Bouillet, M. Wurst, O. Verscheure, M. Grund and P. Cudre-Mauroux, "TRISTAN: Real-time analytics on massive time series using sparse dictionary compression," in *2014 IEEE International Conference on Big Data (Big Data)*, 2014.

[CTS7] A. Khelifati, M. Khayati and P. Cudrè-Mauroux, "CORAD: Correlation-Aware Compression of Massive Time Series using Sparse Dictionary Coding," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019.

[CTS8] E. Bristol, "Swinging Door Trending: Adaptive trend recording?," in *ISA National Conf. Proc., 1990*, 1990.

[CTS9] F. Eichinger, P. Efros, S. Karnouskos and K. Böhm, "A time-series compression technique and its application to the smart grid," *The VLDB Journal*, vol. 24, no. 2, pp. 193–218, 2015.

[CTS10] M. Abo-Zahhad, "ECG signal compression using discrete wavelet transform," in *Discrete wavelet transforms-theory and application*, 2011, pp. 143-168.

[CTS11] D. Blalock, S. Madden and J. Guttag, "Sprintz: Time series compression for the internet of things," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1-23, 2018.

[CTS12] A. S. Fraenkel and S. T. Klein, "Robust Universal Complete Codes for Transmission and Compression," *Discrete Applied Mathematics*, vol. 64, pp. 31-55, 1996.

6.7. Federated learning

[FL1] Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., & Bacon, D. (2017). Federated Learning: Strategies for Improving Communication Efficiency. *ArXiv:1610.05492 [Cs]*. http://arxiv.org/abs/1610.05492

[FL2] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. *ArXiv:1602.05629 [Cs]*. http://arxiv.org/abs/1602.05629

[FL3] Chang, W.-T., Tandon, R. (2020). Communication efficient federated learning over multiple access channels. *ArXiv preprint arxiv:2001.08737*

[FL4] Diao, E., Ding, J., Tarokh, V. (2021). Heterofl: Computation and communication efficient federated learning for heterogeneous clients. *ArXiv, vol. 2010.01264*

[FL5] Reisizadeh, A., Mokhtari, A., Hassani, H., Jadbabaie, A., Pedarsani, R. (2020). Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. *ArXiv, vol. 1909.13014*.

[FL6] Chen,Y., Sun, X., Jin, Y. (2020). Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," IEEE Transactions on Neural Networks and Learning Systems, vol. 31, pp. 4229–4238, 2020.

[FL7] Caldas, S., Konečný, J., McMahan,H.B., Talwalkar, A. S. (2018). Expanding the reach of federated learning by reducing client resource requirements, ArXiv, vol. 1812.07210, 2018.

[FL8] Lin, Y., Han, S., Mao, H., Wang, Y., Dally, W. (2018). Deep gradient compression: Reducing the communication bandwidth for distributed training, ICLR 2018, ArXiv, vol. 1712.01887.

[FL9] Sattler, F., Wiedemann, S., Müller, K., Samek,W. (2020). Robust and communication-efficient federated learning from non-i.i.d. data," IEEE Transactions on Neural Networks and Learning Systems, vol. 31, pp. 3400–3413, 2020.

[FL10] Asad, M., Moustafa, A., Ito, T. (2020). Fedopt: Towards communication efficiency and privacy preservation in federated learning, Applied Sciences, vol. 10, p. 2864, 2020.

[FL11] Malekijoo, A., Fadaeieslam, M. J., Malekijou, H., Homayounfar, M., Alizadeh-Shabdiz, F., Rawassizadeh, R. (2021). Fedzip: A compression framework for communication-efficient federated learning, ArXiv, vol. 2102.01593, 2021.

[FL12] Rothchild, D. et al. (2020). Fetchsgd: Communication-efficient federated learning with sketching, in ICML, 2020.

[FL13] Xu, J., Du, W., Cheng, R., He, W., Jin, Y. (2020) Ternary compression for communication-efficient federated learning, IEEE transactions on neural networks and learning systems, 2020.

[FL14] Wu, X., Liang, Z., Wang, J. (2020). Fedmed: A federated learning framework for language modeling. Sensors (Basel, Switzerland), vol. 20, 2020.

[FL15] Asad, M., Moustafa, A., Muhammad, A. (2021). Ceep-fl: A comprehensive approach for communication efficiency and enhanced privacy in federated learning. Appl. Soft Comput., vol. 104, p. 107235, 2021.

[FL16] Wang, L., Wang, W., Li, B. (2019). Cmfl: Mitigating communication overhead for federated learning, IEEE 39th International Conference on Distributed Computing Systems (ICDCS), pp. 954–964, 2019.

[FL17] Nishio, T. and Yonetani, R. (2019). Client selection for federated learning with heterogeneous resources in mobile edge. ICC 2019 – 2019 IEEE International Conference on Communications (ICC), pp. 1–7, 2019.

[FL18] Park, S., Suh, Y., Lee, J. (2021). Fedpso: Federated learning using particle swarm optimization to reduce communication costs, Sensors (Basel, Switzerland), vol. 21, 2021.

[FL19] Xia, Q., Ye, W., Tao, Z., Wu, J., Li, Q. (2021). A survey of federated learning for edge computing: Research problems and solutions, High-Confidence Computing, vol. 1, issue 1, 2021.

[FL20] Al-Saedi, A., Casalicchio, E., Boeva, V. (2021). An Energy-aware Multi-criteria Federated Learning Model for Edge Computing. FiCloud 2021.

[FL21] Al-Saedi, A., Boeva, V., Casalicchio, E. (2021a). Reducing Communication Overhead of Federated Learning through Clustering Analysis. ISCC 2021.

6.8. Transfer learning

[TL1] Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2021). A Comprehensive Survey on Transfer Learning. *Proceedings of the IEEE*, *109*(1), 43–76. https://doi.org/10.1109/JPROC.2020.3004555

[TL2] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A Survey on Deep Transfer Learning. *ArXiv:1808.01974 [Cs, Stat]*. http://arxiv.org/abs/1808.01974

6.9. Distributing computations among edge nodes

[DC1] Moroşan, P. D., Bourdais, R., Dumur, D., & Buisson, J. (2010). Building temperature regulation using a distributed model predictive control. Energy and Buildings, 42(9), 1445-1452.

[DC2] Lee, D., Allan, J., Thompson, H. A., & Bennett, S. (2001). PID control for a distributed system with a smart actuator. Control Engineering Practice, 9(11), 1235-1244.

[DC3] Zhang, X., Shi, W., Yan, B., Malkawi, A., & Li, N. (2017). Decentralized and distributed temperature control via HVAC systems in energy efficient buildings. arXiv preprint arXiv:1702.03308.

[DC4] Yang, Y., Srinivasan, S., Hu, G., & Spanos, C. J. (2021). Distributed Control of Multizone HVAC Systems Considering Indoor Air Quality. IEEE Transactions on Control Systems Technology.

6.10. Privacy-preserving learning techniques

[PP1] Rigaki, M., & Garcia, S. (2021). A Survey of Privacy Attacks in Machine Learning. *ArXiv:2007.07646 [Cs]*. http://arxiv.org/abs/2007.07646

[PP2] Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017). Membership Inference Attacks Against Machine Learning Models. *2017 IEEE Symposium on Security and Privacy (SP)*, 3–18. https://doi.org/10.1109/SP.2017.41

[PP3] Papernot, N., Abadi, M., Erlingsson, Ú., Goodfellow, I., & Talwar, K. (2017). Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. *ArXiv:1610.05755 [Cs, Stat]*. http://arxiv.org/abs/1610.05755

[PP4] Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep Learning with Differential Privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–318. https://doi.org/10.1145/2976749.2978318

[PP5] Shokri, R., & Shmatikov, V. (2015). Privacy-Preserving Deep Learning. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 1310–1321. https://doi.org/10.1145/2810103.2813687

[PP6] Bost, R., Popa, R. A., Tu, S., & Goldwasser, S. (2015). Machine Learning Classification over Encrypted Data. *Proceedings 2015 Network and Distributed System Security Symposium*. Network and Distributed System Security Symposium, San Diego, CA. https://doi.org/10.14722/ndss.2015.23241

[PP7] Hunt, T., Song, C., Shokri, R., Shmatikov, V., & Witchel, E. (2018). Chiron: Privacy-preserving Machine Learning as a Service. *ArXiv:1803.05961 [Cs]*. http://arxiv.org/abs/1803.05961

[PP8] Tyler Hunt, Zhiting Zhu, Yuanzhong Xu, Simon Peter, and Emmett Witchel. 2018. Ryoan: A Distributed Sandbox for Untrusted Computation on Secret Data. ACM Trans. Comput. Syst. 35, 4, Article 13 (December 2018), 32 pages. DOI: https://doi.org/10.1145/3231594

6.11. Secure data sharing

[DS1] P. Li, J. Su and X. Wang, "iTLS: Lightweight Transport-Layer Security Protocol for IoT With Minimal Latency and Perfect Forward Secrecy," in IEEE Internet of Things Journal, vol. 7, no. 8, pp. 6828-6841, Aug. 2020, doi: 10.1109/JIOT.2020.2988126.

[DS2] R. Yegireddi and R. K. Kumar, "A survey on conventional encryption algorithms of Cryptography," 2016 International Conference on ICT in Business Industry & Government (ICTBIG), 2016, pp. 1-4, doi: 10.1109/ICTBIG.2016.7892684.

[DS3] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, Mauro Conti. A Survey on Homomorphic Encryption Schemes: Theory and Implementation. ACM Computing Surveys, Vol. 51, No. 4, Article 79. Publication date: July 2018.

[DS4] S. Agwa, E. Yahya and Y. Ismail, "Power efficient AES core for IoT constrained devices implemented in 130nm CMOS," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1-4, doi: 10.1109/ISCAS.2017.8050361.

[DS5] Y. Al-Aali and S. Boussakta, "Lightweight block ciphers for resource-constrained devices," 2020 12th International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), 2020, pp. 1-6, doi: 10.1109/CSNDSP49049.2020.9249644.

[DS6] Sohel Rana, Saddam Hossain, Hasan Imam Shoun and Dr. Mohammod Abul Kashem, “An Effective Lightweight Cryptographic Algorithm to Secure Resource-Constrained Devices” International Journal of Advanced Computer Science and Applications (IJACSA), 9(11), 2018. http://dx.doi.org/10.14569/IJACSA.2018.091137

[DS7] OAuth. https://oauth.net/

[DS8] OpenID. https://openid.net/

[DS9] im, A., Adnan, M.A.: An OpenID based authentication service mechanisms for internet of things. In: 2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS), Singapore, pp. 687–692 (2019).

[DS10] Mahmood, K., Chaudhry, S.A., Naqvi, H., Shon, T., Ahmad, H.F.: A lightweight message authentication scheme for smart grid communications in power sector. Comput. Electr. Eng. 52, 114–124 (2016).