Ref. Ares(2021)7499399 - 05/12/2021



ICT-56-2020 "Next Generation Internet of Things"
Grant Agreement number: 957218

IntellioT

Deliverable D3.2
Distributed AI algorithms & methods (first version)

Deliverable release date	30/11/2021
Authors	1. UOULU: Sumudu Samarakoon, Mohamed Abdelaziz 2. PHILIPS: Anca Bucur, Nancy Irisarri Méndez, Lex Joosten 3. SIEMENS: Arne Broering
Editor	Sumudu Samarakoon (UOULU)
Reviewer	Nancy Irisarri Méndez (PHILIPS), Federico Chiariotti (AAU)
Approved by	PTC Members: (Vivek Kulkarni, Konstantinos Fysarakis, Sumudu Samarakoon, Beatriz Soret, Arne Bröring, Maren Lesche) PCC Members: (Vivek Kulkarni, Jérôme Härri, Beatriz Soret, Mehdi Bennis, Martijn Rooker, Sotiris Ioannidis, Anca Bucur, Georgios Spanoudakis, Simon Mayer, Filippo Leddi, Harshitha Chandregowda, Maren Lesche, Georgios Kochiadakis)
Status of the Document	Final
Version	1.0
Dissemination level	Public

Table of Contents

1 Introduction 3
1.1 Overview of IntellIoT framework 3
1.2 Distributed AI and IoT device constraints 5
1.3 Challenges and research questions 7
1.4 IntellIoT Requirements 8
2 Distributed AI under constrained resources 10
2.1 Federated Learning Framework 10
2.2 AI for Agriculture 11
2.3 AI for Healthcare 13
2.3.1 Envisioned Patient Monitoring model 13
2.3.2 Envisioned Fitness Recommendation model 14
2.4 AI for Manufacturing 15
2.5 Preliminary Results & Summary 17
2.5.1 Client scheduling towards high FL accuracy 17
2.5.2 Energy-Efficient Model Compression and Splitting 19
3 Training and Retraining Towards Robust and Personalized AI 21
3.1 Monitoring and Out of Distribution Detection 21
3.2 AI Model Personalization 21
3.3 AI Model Retraining 21
3.4 Summary 22
4 Conclusion & Future Direction 23
5 Annex 24
6 References 25

1 INTRODUCTION

1.1 Overview of IntellIoT framework

IntellIoT aims to develop an architecture for the next generation (NG) Internet of Things (IoT) based on de-centralized hypermedia multi-agent systems that semi autonomously configure and instantiate IoT applications based on user-defined goals. A high-level view of IntellIoT's logical architecture was derived, as shown in Figure 1. The three pillars of the project (i.e., Collaborative IoT, Human-in-the-loop and Trustworthiness) are prominently featured in the architecture, as they are key to the IntellIoT concept.

In total, five core component groups have been identified, with individual components falling into one of the following groups:

Collaborative IoT enablers: This group contains the components that realize IntellIoT's Collaborative IoT pillar, focusing on the cooperation of various semi-autonomous entities (tractors, robots, healthcare devices, etc.) to execute multiple IoT applications.

Human-in-the-Loop enablers: This group contains the components involved in IntellIoT's Human-in-the-Loop (HITL) pillar, which focuses on involving the human in the process, when necessary, to solve complex situations that the system does not yet know how to handle.

Trust enablers: This group involves components that are part of IntellIoT's Trust pillar. This pillar focuses on privacy, security, and ultimately building trust into the IntellIoT framework.

Infrastructure management: This group is comprised of the computation and communication infrastructure and its management capabilities, which enable the deployment and management of edge applications.

Use-Case deployment: This group involves all components which are Use-Case specific, (i.e., pertaining to the use case environment deployment), such as edge devices and their hardware, edge apps, and edge AI models.

Artificial Intelligence (AI)/Machine Learning (ML) is integrated with constrained devices to enable semi-autonomous IoT applications and systems (e.g., self-driving tractors). This is done by leveraging distributed AI algorithms under compute, storage, mobility, and bandwidth constraints and by integrating the human-in-the-loop for safety, assistance, and continuous improvement of AI.

The high-stake IoT applications of IntellIoT (including autonomous tractors, robot arms, AR/VR, or telemonitoring of patients and personalized healthcare) require a novel paradigm change from classic ML calling for distributed, low-latency and reliable ML at the wireless network edge [1]. The key benefits of distributed ML at the edge are reduced latency and reduced cost of sharing training data, improved privacy, and higher inference accuracy due to the ability to training over data including privacy sensitive information.

Distributed IoT devices operate autonomously and perform decentralized or distributed training of their ML models subject to bandwidth, latency and privacy constraints of time-varying data, sparse and noisy feedback, and intermittent connectivity (e.g., channel blockage, network topology changes). Developed mechanisms will enable devices to make local, reliable, and fast inference, while accounting for network dynamics and other system-level aspects. The distributed data management and model training will ensure efficient implementation of all required data protection and privacy regulations.

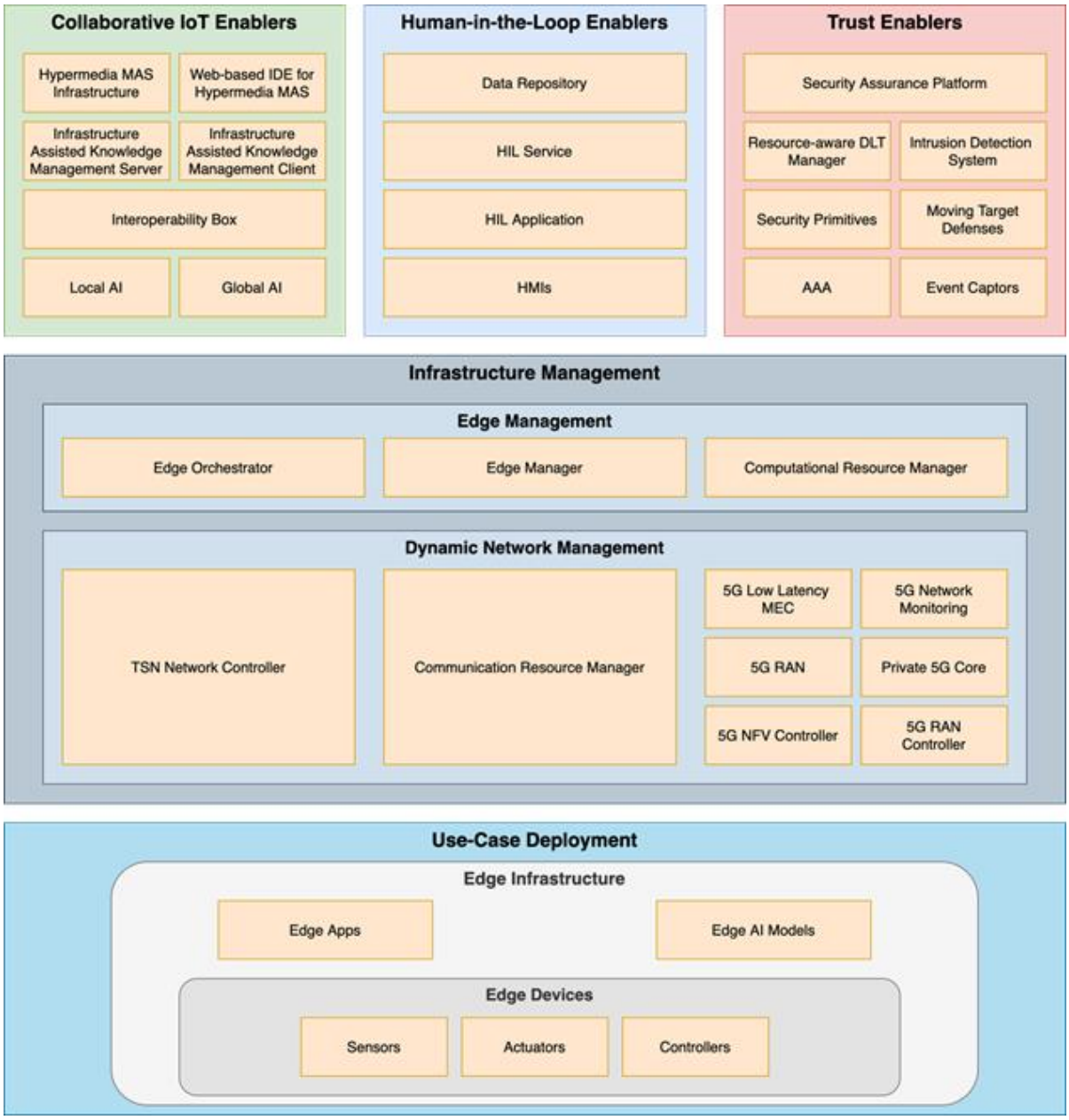

Figure 1. High-level view of IntelIIoT's logical architecture

Communication is instrumental in enabling scalable and distributed ML. Therefore, an ML-Communication co-design is followed to enable these new breeds of intelligent IoT applications. Distributed AI model execution and refinement will lead to increased system reliability and availability, correct operation, and efficient data collection, being resilient to potential constraints in connectivity. This is especially important in the healthcare use case where ensuring the safety of the human subjects at all times is essential.

1.2 Distributed AI and IoT device constraints

The classical concept of ML focuses mainly on offline and centralized ML [2][3]. In this case, the entire data set is given a priori and is used for the training process. To obtain accurate and reliable inference, a central controller divides the training data set into minibatches and allocates them to multiple processing devices, e.g., via a message passing interface (MPI) [4], running local training operations. The central controller iteratively collects and aggregates their local training results until the training loss converges. The said training process is separated from inference, and hence, the training cost and latency are commonly neglected.

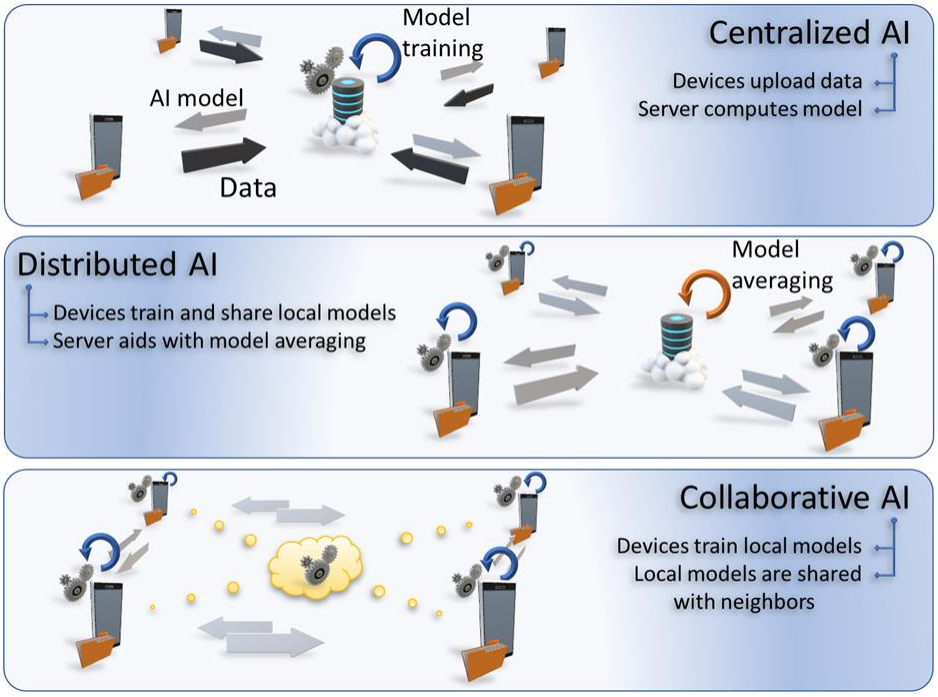

Figure 2. Different AI architectures

Unfortunately, such a one-time training process is vulnerable to initially unmodeled phenomena, as it is practically impossible to enumerate all preconditions and ensuing consequences, referred to as qualification and ramification problems, respectively [5]. In fact, the trained model using centralized ML is biased toward the initially fixed training data set, which fails to capture user-generated and the time-varying nature of data, yielding less reliable inference results. Besides, a user-generated data set can be privacy-sensitive, and the owners may not allow the central controller to directly access the data. Distributed ML can address these problems. Indeed, online decentralized training can preserve privacy by exchanging not the data set but the model parameters with (or without) a simple parameter aggregator [6], thereby reflecting a huge volume of user-generated data samples in real time. In so doing, the trained models are immediately obtained at the edge devices, enabling low-latency inference.

There are clear advantages using distributed AI methods (edge ML) [1][7][3].

- Performing inference locally on connected devices reduces latency and cost of sending device-generated data to the cloud for prediction.
- Rather than sending all data to the cloud for performing ML inference, inference is run directly on the device, and data are sent to the cloud only when additional processing is required.
- Getting inference results with very low latency is important in making mission-critical IoT applications that respond quickly to local events.

- Unlike cloud-based ML, edge ML is privacy preserving in which the training data are not stored at the cloud but are kept locally on every device, and the globally shared model is learned by aggregating locally computed updates, denoted as model state information (MSI), in a peer-to-peer manner or via a coordinating (federating) server.
- Higher inference accuracy can be achieved by training with a wealth of user-generated data samples that may even include privacy-sensitive information, such as healthcare records, factory/network operational status, and personal location history.

Edge ML is a nascent research field whose system design is entangled with communication and on-device resource constraints (i.e., energy, memory, chipset architecture, and computing power). In fact, the size of a neural network (NN) and its energy consumption may exceed the memory size and battery level of a device, hampering decentralized inference. Moreover, the process of decentralized training involves many devices that are interconnected over wireless links, hindering the training convergence due to the stale MSI exchange under poor wireless channel conditions. As such, enabling ML at the network edge introduces novel fundamental research problems in terms of jointly optimizing training, communication, and control under end-to-end (E2E) latency, reliability, privacy, as well as devices' hardware requirements.

Training ML at the network edge over wireless networks while considering latency and reliability opens up a novel research direction. In this respect, edge ML architectures and their operations should be optimized by accounting for communication overhead and channel dynamics while coping with several problems, such as straggling devices in the training process and generalization to unmodeled phenomena under limited local training data. In addition, all these aspects need to factor in on-device constraints, not to mention privacy guarantees. An interesting example of edge ML training architecture is federated learning (FL) [7] [6] in which mobile devices periodically exchange their NN weights and gradients during local training. FL has been shown to improve communication efficiency by MSI quantization [8], adjusting the MSI update period [7], and optimizing devices' scheduling policy [9]. These methods are still in their infancy and need to address a myriad of fundamental challenges, including the ML-communication codesign.

Energy limitation: In a large-scale ML system, each device is likely to be mobile, equipped with a capacity limited battery. Thus, the finite energy needs to be efficiently utilized for the computation process during training and inference operations, as well as the communication process. In this respect, one may prefer to slightly decrease the inference accuracy to save the training energy consumption that is proportional to the model size [10]. Unfortunately, traditional ML architectures cannot dynamically change the model size in real time and are thus unable to flexibly optimize the energy consumption. Next, with the helper–device split, the device may want to offload its computation to the helper with higher computation power without energy limitation. The bottleneck is the offloading communication overhead and optimizing such a computation–communication tradeoff is a large field of research in mobile edge computing (MEC) [11] [12].

Memory limitation: The optimal NN model size for inference is determined by the devices' memory sizes of static random-access memory (SRAM) and dynamic random-access memory (DRAM). This may conflict with the optimal model size for the training operations, which is mainly determined by the communication overhead induced by the MSI exchanges. Furthermore, the limited size of the devices' storages, such as hard disk drive (HDD) and solid-state drive (SSD), constrains the input data sample sizes as well as the number of intermediate calculation values stored during the training phase. Therefore, the training and inference operations need to be jointly optimized under the constraints of communication links, as well as of various types of devices' memories.

Privacy: Some of data samples owned by the devices may be privacy-sensitive, e.g., medical records. Exchanging MSIs instead of data samples can partly preserve privacy yet is still vulnerable to being reversely traced by eavesdroppers. Extra coding, such as homomorphic encryption [13], may guarantee privacy, but its processing delay may exceed the low-latency deadline. Exchanging redundant information may hide the private information but unfortunately results in extra communication delay. Inserting noise has a potential to protect privacy while playing a role as regularizer, as long as the noise level is properly adjusted; otherwise, it decreases the inference accuracy significantly.

1.3 Challenges and research questions

Enabling ML at the network edge hinges on investigating several fundamental questions, some of which are briefly summarized next.

- Q1. How do edge devices train a high-quality centralized model in a decentralized manner, under communication/on-device resource constraints and different NN architectures?

Classical ML has been based on the precept of a single central entity having full access to the global data set over ultrafast wired connections, e.g., a local inter-chipset controller manipulating multiple graphics processing units (GPUs) through NVLink intra-computer connections (supporting up to 600 Gb/s) [14], InfiniBand intercomputer links (up to 100 Gb/s) [15]; or a cloud server commanding multiple computing devices via Ethernet communication (up to 25 Gb/s) [16]. Using a deep NN, the central controller sifts through this global data for training and inference by exploiting the massive amount of storage and computing power: e.g., 11.5 Petaflops processing power supported by 256 tensor processing units (TPUs) and 4-TB high-bandwidth memory (HBM) [17], which is sufficient for operating the Inception V4 NN model consuming 44.3 GB [18].

These figures are in stark contrast with the capability of devices under edge ML. While 5G peak rates achieve 20 Gb/s [19] that is comparable only with Ethernet connections, the instantaneous rate may frequently fluctuate due to poor wireless channel conditions, hindering inter-device communication. Moreover, the computing power of mobile devices is a million times less powerful (e.g., Qualcomm Snapdragon 845's 16.6 Gflops [20]). Their memory size is also ten times smaller (Apple iPhone XS Max's 4 GB [21]) than a deep NN model size. Besides, the energy consumption (e.g., Google Pixel 3 XL's 2.15 W [22]), delimited by the battery capacity, is half million times smaller than a powerful centralized ML architecture (e.g., AlphaGo's 1 MW [23]). Since computation and communication consume battery power, the energy consumption of edge ML operations should be flexibly optimized over time, which is not feasible under the classical ML architecture, in which an NN model size is fixed.

Last but certainly not the least, the privacy guarantee of each device is crucial in edge ML, particularly when on-device data sets are associated with privacy-sensitive information. Perturbing the information exchange can ensure privacy, which may be at odds with the goal of high accuracy and reliability. Adding redundant information can provide a solution, at the cost of extra communication latency.

- Q2. How to enable reliable edge ML as opposed to best-effort cloud-based ML, subject to nonconvex loss functions and unevenly dispersed and unseen data samples?

A requisite for reliable edge intelligence is high inference accuracy, i.e., low loss, under not only the training data set but also unseen data samples. Therefore, average inference accuracy is insufficient, and its credibility interval for a given training data set ought to be considered. Credibility can be measured by the loss difference under training and the entire samples, referred to as a generalization error, in which the credibility interval is an achievable target generalization error that can be decreased by utilizing more training samples and a proper NN model architecture. Calculating the generalization error is relatively easy in centralized ML where the central controller feeds independent and identically distributed (IID) training data samples into each device. By contrast, it becomes more challenging in edge ML where the training data samples may become non-IID across devices.

Next, decentralized training dynamics in edge ML become more complicated even under a simple gradient descent algorithm. In fact, it is difficult to characterize the convergence behaviour of a decentralized training process, especially under non-IID training data set, as well as the limited communication/computation resource budget fluctuating over time. The situation is aggravated when a single NN model is split and shared by multiple devices due to the limited on-device memory size. Besides, most training algorithms intentionally insert noise, i.e., regularizes, to cope with nonconvex loss functions. However, it is challenging to optimize the regularization in edge ML, which is intertwined with wireless communication and privacy-preserving methods generating noise.

- Q3. How do the theoretical and technical enablers of edge ML impact E2E latency, reliability, and scalability throughout the training and inference processes, under both communication for ML (CML) and ML for communication (MLC) frameworks?

From the standpoint of MLC, stringent ultra-reliable low latency communication (URLLC) applications can be empowered using edge ML, whereas in CML, edge ML is enhanced via wireless connectivity under on device constraints. In this respect, edge ML design not only enhances CML but also MLC, calling for optimizing E2E latency, reliability, and scalability.

Specifically, the worst case E2E latency of a reference device is given by "training + inference + application" latency. MLC can reduce the application delays that are proportional to wireless communication latency, given as "payload/[bandwidth × spectral efficiency]." For instance, this is viable by improving spectral efficiency (SE) via enabling real-time joint source–channel coding [24], channel-agnostic E2E communications [25], and interference management [26]. The effective amount of bandwidth can also be increased using low-complexity dynamic spectrum access [27] and proactive resource management [28]. Furthermore, the payload size can be minimized by exchanging semantic information rather than raw data [29].

Achieving these benefits in MLC entails extra training and inference delays of ML operations. Here, training latency is captured by a loss or weight convergence delay [30] whereas inference latency refers to computation and memory access delays [10]. Since both the training and inference processes are performed at the network edge, their computation and communication need to be jointly optimized subject to limited energy, memory, computing power, and radio resource constraints. To further reduce latency, as done in [31], the training process can partly be offloaded to other proximal edge nodes with higher computational power.

E2E reliability of a reference device is the probability that the E2E latency does not exceed a target latency deadline and is determined by the reliability of both training and inference. For training reliability, the staleness of exchanged MSIs is the key bottleneck, as it disrupts the training convergence, and may lead to an unbounded training latency, emphasizing the importance of communication techniques to synchronize edge nodes. For inference reliability, high inference accuracy is mandatory for reducing the application latency and needs to be ensured under both training and unseen samples, necessitating suitable mathematical tools to quantify the generalization error.

Finally, E2E scalability is specified by the number of supported devices, NN model size, and communication throughput to ensure a target E2E reliability. In this respect, the major obstacles are on-device constraints, whereby the number of federating devices is limited by preserving their privacy. The range of federation is also determined by memory and NN model sizes, as FL requires an identical model size for all federating devices. Moreover, the federation range should consider tasks' correlations and channel conditions, so as not to exchange redundant MSIs and to avoid straggling devices under limited wireless capacity, respectively.

1.4 IntelIIoT Requirements

Requirement 1: Enable semi-autonomous IoT applications by leveraging distributed AI algorithms under compute, storage, mobility, and bandwidth constraints and by integrating the human-in-the-loop for safety, assistance, and continuous improvement of AI.

Description:

Distributed IoT devices need to operate autonomously and perform decentralized or distributed training of their ML models subject to bandwidth, latency and privacy constraints of time-varying data, sparse and noisy feedback, and intermittent connectivity. Additionally, it is essential to enable devices to make local, reliable, and fast inference, while accounting for network dynamics and other system-level aspects. ML-communication co-design will enable inference on IoT devices with constrained memory and computational power resulting in increased system reliability and availability, correct operation, and efficient data collection, being resilient to potential constraints in connectivity. The success of achieving these goals is summarized in Table 1.

Table 1. Measure of Success

Tool	Description	Success Indicators	Current Status
Federated learning	With a centralized training model as benchmark for training accuracy, enable distributed solutions achieving an accuracy beyond 95% of benchmark within 50 inter-device communication rounds.	KPI-3.1	Completed
Knowledge distillation	At least 10 times ML model size reduction during knowledge distillation for constrained IoT devices.	KPI-3.2	Not started
Split learning	At least 10% of energy reduction during distributed training while ensuring high training accuracy compared to centralized training methods.	KPI-3.4	Completed

2 DISTRIBUTED AI UNDER CONSTRAINED RESOURCES

Training ML at the network edge over wireless networks while considering latency and reliability paves the way for the edge ML. In this view, edge ML architectures and their operations should be optimized by accounting for communication overhead and channel dynamics, while coping with several problems such as straggling devices in the training process and generalization to unmodeled phenomena under limited local training data. Not only that, but all these aspects also need to factor in on-device constraints. In this regard, it is crucial to manage the limited resources in IoT networks via improving the communication, computation, and energy efficiency within edge ML architecture while preserving the data privacy.

2.1 Federated Learning Framework

One prominent edge ML training architecture is FL [7], [6] wherein edge devices periodically exchange their NN weights and gradients during local training. The collaborative learning of a shared prediction model in FL does not require that the local training data on devices to be transferred to the cloud.

The fundamental problem behind FL is to minimize a loss $F(w)$ parameterized by model weights w over a given dataset $\mathcal{D}$ of D samples as follows:

$$\text{minimize}_w \; F(w) = \frac{1}{D}\sum\nolimits_{d\in\mathcal{D}} \ell(d),$$

where $\ell(d)$ is the loss defined over the data sample d. As the data is generated at K edge devices (a.k.a. clients), the global dataset is the composition of local datasets $\mathcal{D}_k$, i.e., $\mathcal{D} = \cup_k \mathcal{D}_k$. Then, defining $F_k(w) = \frac{1}{D_k}\sum_{d\in\mathcal{D}_k} \ell(d)$, the aforementioned objective can be recast as a distributed formulation that is given by,

$$\text{minimize}_w \; F(w) = \frac{1}{D}\sum\nolimits_{d\in\mathcal{D}_k,k} \ell(d) = \sum\nolimits_k \frac{D_k}{D} F_k(w).$$

This decoupling over clients enables local loss minimization at each client while reaching consensus over clients through a central helper (parameter server, PS), which is the concept behind the FL.

While FL has been introduced as a distributed training over homogeneous clients towards obtaining a homogeneous model, the state-of-the-art of FL has been evolved to address data and model heterogeneity, model personalization, and improving communication and computation efficiencies under resource constraints. Due to the nature of three use cases investigated under IntelliIoT with distributed devices requiring autonomy via ML to carry out similar tasks, which need to collaborate over resource limited wireless connectivity, FL is a matured distributed ML architecture that is the best candidate for the IntelliIoT framework.

Within IntelliIoT architecture, we define the main two components of FL framework as Local AI Component and Global AI Component, respectively.

Local AI Component: This component exists on the edge device and has access to the local data. Using the local datasets at the robot, patient, or tractor devices, the Local AI Component is responsible of training AI models on the client/device. Such training allows local AI models to capture the correlations between locally observed data and corresponding control decisions, as well as running inference on control decisions at runtime. The Local AI Component iteratively communicates with the Global AI Component during FL (i) to obtain the Global AI model prior to train over the local dataset and (ii) to share the updated Local AI model after the local training as depicted in Figure 3.

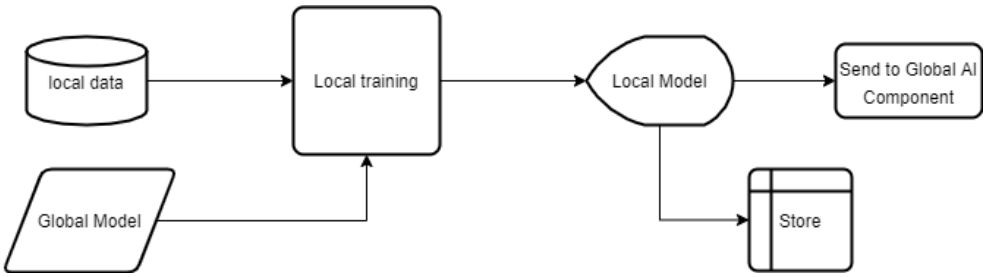

Figure 3. FL in the Local AI Component

Global AI Component: This is the centralized entity that supports individual devices to share the training results of their local Manufacturing, Healthcare, and Agriculture AI Models with one another while at the same time maintaining local data private. The Global AI Component is located on a central server, and it manages the entire federated learning process. This Component communicates with local devices and plays two major roles. First, as model aggregator: AI model updates from local devices are sent to the Global AI Component, which then aggregates these updates to create an updated model as illustrated in Figure 4. The updated global AI model is evaluated and, pending improved model performance and considering model personalization, is shared with local devices. This enables sharing the local knowledge from each device with one another. Second, as resource-aware retraining takes into account available computation and communication resources to schedule local devices that need to contribute for re-training tasks. The Global AI Component must be configurable in terms of number of local devices that can participate, AI model parameters and aggregation methodologies.

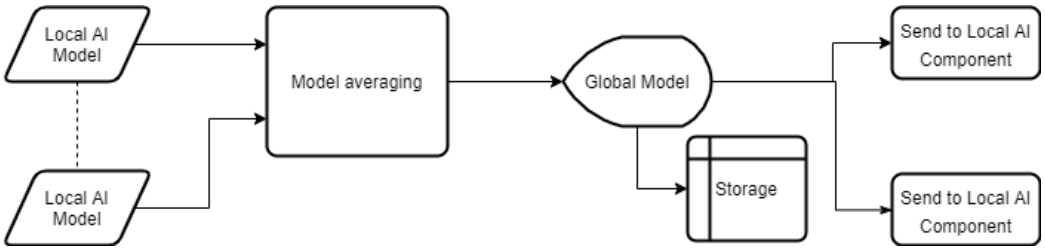

Figure 4. FL in the Global AI Component

Although the training architecture is common for the three use cases, the role of AI is defined in terms of the use case specific tasks as discussed next.

2.2 AI for Agriculture

The agricultural domain faces many dangerous situations and unfortunately, many deadly accidents occur where farming vehicles are involved [32] be it either in conjunction with other vehicles or caused by the human operator of the device. Hence, it is essential to develop an intelligent IoT environment of a semi-autonomous agricultural vehicle in collaboration with other devices, infrastructure, and human operators to improve safety, reliability, and security.

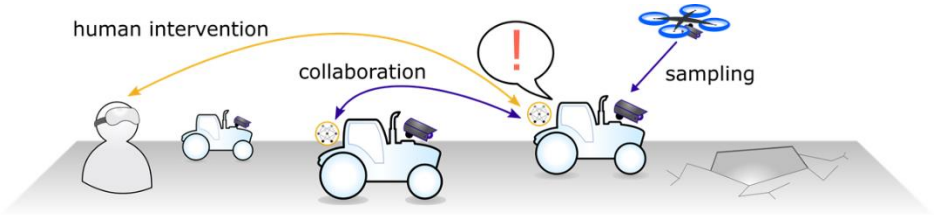

Figure 5. Abstract view of agriculture use case

Within the scope of the project, the role of AI in agriculture is to aid tractors to navigate in the presence of obstacles. The tractors are programmed to follow a sequence of waypoints and their on-board camera sensors can detect obstacles on their path. In the presence of obstacles, tractor controller requires a command sequence that corresponds to tractor manoeuvre without collisions.

Training of the agriculture AI models are carried out in a supervised learning fashion. Here, sequences of control commands issued by a human operator at the presence of obstacles are recorded along the corresponding sensory information, i.e., video feed from onboard camera, to generate the training dataset. Using the training data collected at the tractor is then used to train a 3D convolutional NN (CNN), which receives sensory data as the input and produces the control commands as the output. The choice of the NN architecture is to extract the higher order features from the time-dependent visual data and then to regress the linear and angular velocity components that are normalized into a scale from 0 to 100.

It is worth highlighting that the individual tractors are unlikely to collect a sufficiently large dataset to train an efficient AI model. Hence, it is essential for tractors to collaborate with one another to access a rich dataset. Due to the data privacy and the limitations in communications among tractors and storage within tractors, sharing raw sensory data is not a viable option. In this regard, the FL framework is adopted to enable collaborative training among the tractors.

The inference initiates with a manoeuvre request. Using the camera feed and the information about the obstacle, AI model generates a sequence of linear and angular velocities that allow tractor to navigate through the obstacles. Along the control decisions, AI model computes a confidence of the inferred decisions, which is compared with a predefined confidence target. This acts as a safety measure whereas a low confidence is used to request the human aid, i.e., HITL. This procedure is summarized in Figure 6.

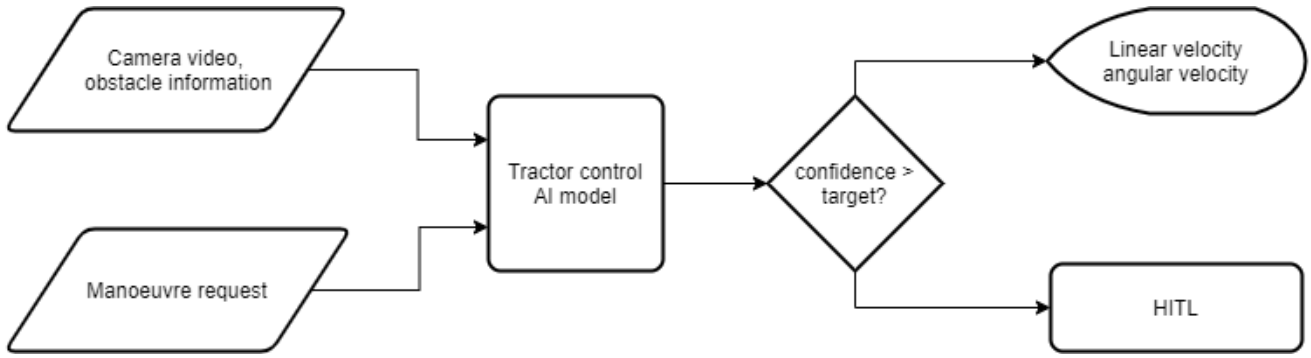

Figure 6. Functionality of agriculture AI model: Manoeuvring around obstacles

Table 2. Key components of AI for Agriculture use case

Property	Description
Input	Camera video feed Obstacle information
Output	Linear velocity Angular velocity Confidence of the decision
NN architecture	CNN
Triggered by	Manoeuvre request from tractor controller

Triggers	Human takeover
Sensor	https://www.stereolabs.com/zed-2/
Actuator	AVL eTractor prototype The tractor is controlled by a vehicle control unit (VCU) that commands several underlying control systems each including an embedded controller that is connected via different CANbusses to the VCU

2.3 AI for Healthcare

In AI for healthcare, we deal with sensitive data and cross-silo (inter-hospital) or cross-device (inter-patient) communication. As mentioned, due to the intrinsically distributed nature of the system, we will implement a federated learning solution [7]. Continuous and active learning components will be implemented as well to enable personalization to individual users and their specific needs. Furthermore, the IntellIoT implementation will explore solutions to some of the problems caused by the unpredictable behaviour of Edge devices in a federated learning setup [33].

The framework will be used to develop models for prediction and early detection of potential health degradation or negative events when clinical intervention is needed, for recommendations, and for fulfilling the information needs of the users. We will as well assess the effectiveness of the interventions. The solution will collect the information provided by devices and sensors, presenting it to the healthcare providers, and triggering requests when the intervention of the clinician is needed for a particular patient (implementing the concept of human-in-the-loop). Wearables and other sensors will gather health related data (e.g., vital signs, physical activity, sleep quality) and will transmit this data to the Local AI Component on an Edge device.

Due to the sensitive content of the interventions produced by the AI models, these will have to be approved by a physician before they are sent to a patient. Also, in some cases the AI models will produce inconclusive interventions at which point a separate component (HyperMAS) will handle escalation to a service (HIL), which in turn will involve human assistance from a physician. The human-in-the-loop and the IntellIoT framework make it possible to use the selected intervention to train the AI models further with this new labelled dataset, which is otherwise extremely scarce and difficult to obtain.

Possible AI model architectures that will be implemented within IntellIoT include neural-backed decision trees (NBDT) [34], which address the issue of model interpretability, and Long Short-Term Memory (LSTM) [35] networks, which are suitable for measurements taken in a time sequence. Traditional probability models will also be employed as appropriate. The local training of these models will take place periodically, with training moments at the Local AI Component in the Edge device determined by device restrictions (idle and charging) or determined by the amount of new data collected, and at the Global AI Component in a resource-aware manner determined by the number of Local AI Components that have registered their model updates.

In healthcare scenarios, it is nearly impossible to have open connections between components. The most favoured approach for communication is thus a polling approach wherein the workers continuously poll for tasks from the coordinator. This approach uses REST based APIs exposed by the coordinator for workers to consume. For each training round, workers publish model weights along with other information to the coordinator using the API. Also, workers can publish the size of the local dataset, which can be used to determine the most valued minimum number of workers and to determine the bias during aggregation.

The overall solution developed in IntellIoT and demonstrated in the domain of cardiovascular diseases has applicability in other patient populations. The infrastructure can be applied for the development and deployment of new algorithms, with customized apps and sensors.

2.3.1 ENVISIONED PATIENT MONITORING MODEL

The general structure of a patient monitoring model would take sequences (e.g., blood pressure (BP), heart rate (HR), and similar) as well as embedded attributes (e.g., the heart-rate classification and the New York Heart Association classification [36] as input, and depending on the patient condition, the model will need to determine whether the patient is doing fine, or whether the doctor needs to take a look at the patient. For this model it is perhaps not entirely necessary to use a machine learning method, and perhaps a more standard threshold system may be able to perform the job just as well. However, it may also be the case that the machine learning system will be able to detect more subtle patterns thereby reducing unnecessary warnings and alarms, while still being able to monitor the patient adequately and safely.

The sequences would be put through a multi-layered LSTM architecture to end up with a sequence of vectors. This sequence would then be put through a final linear layer and a sigmoid function which will be the threshold inputs to a binary soft NBDT. The threshold inputs need to be binary as this is required for the sigmoid function to apply proper node weighting. Then the number of outputs of the final linear layer will be the number of nodes in the NBDT. The reason for using a NBDT is that the patient monitoring software needs to be interpretable and transparent, as both patients as well as physicians need to be able to understand how and why the model works in order to trust it, have the quality that they are more easily understood than more abstract neural networks.

The main issue with this model would be data collection, as it is very hard to find a sufficiently large publicly available dataset of people about to suffer from an adverse cardiovascular health event or suffering from other acute cardiovascular ailments. One solution would be data synthesisation. The idea being quite broadly that one may be able to generate reasonably accurate data to pre-train the model on, thereby reducing the need for real data. This data synthesizing could be based on a randomized generation of data, and reliance on the threshold systems currently in use for most patient monitoring systems (e.g., if HR > 200 bpm then alarm) for labelling. Then after pre-training, federated learning might be applied with cardiologists in the loop in order to train the algorithm afterwards.

2.3.2 ENVISIONED FITNESS RECOMMENDATION MODEL

A fitness recommendation model would predict the patients HR, BP, etc. based on the patient's speed, historical values, and relevant attributes. The purpose of such a model is to predict the patient's performance under different demands of the patient's speed. Then a certain target speed can be prescribed. This target speed will depend on the patient's personal characteristics, the sport they perform, the target HR and BP, and other attributes.

As shown in [37], the model uses firstly the contextual sequences and attributes as embeddings. This is done in a way such that the model can learn to take both previous exercise sessions into account, as well as taking the user's attributes into account. The attribute embedding is a lookup table which generates a sequence based on the value of various attributes (i.e., gender, sport type, and user identification number).

The input sequence would be the speed of the patient. This input sequence is then concatenated with the contextual vectors and is passed through a 2-layer LSTM. These are then passed through a linear layer and a SELU activation function to result in the final sequences as the predictions of the model.

Table 3. Key components of AI for Healthcare use case

Property	Description
Input	Measurements of steps taken, electrocardiogram, blood pressure, body temperature, blood oxygenation, and body weight. Past measurements of patient heart rate (HR), blood pressure (BP), and workout speed. Patient's attributes such as gender, heart failure class, and use of medications.
Output	Notifications for clinicians. Future HR and BP values and patterns.
Algorithm and NN architectures	Domain rules NBDT

	LSTM
Triggered by	New measurements exceed or fail corresponding thresholds. Results of training round indicate a change.
Triggers	Escalate to clinicians
Sensors	Smartwatch, blood pressure monitor, thermometer, pulse oximeter, and weight scale.

2.4 AI for Manufacturing

The manufacturing domain is currently undergoing a tremendous change towards more and more digitalization and automation behaviour on the manufacturing shop floor. It is expected that machines will be interacting more and more autonomously with each other. To cope with desire of the customers for more individualized products, there is a trend to move away from production lines towards flexible production cells. These production cells can be realized different tasks and can be combined with each other in varying order and purposes. This flexibility will be facilitated using autonomous guided vehicles (AGVs), which bring work pieces and pre-products from one production cell to another. This intelligence and autonomy will be realized through an increased usage of AI.

Within the scope of IntellIoT, the role of AI in manufacturing is twofold.

Usable area detection: It is essential to identify usable work pieces from a stock of candidate items. Based on the end goal of engraving text/drawing in a circular area on a wooden piece, which is then extracted out using a milling machine, the objective of AI is to identify the largest circular area that can be placed on top of any given wood piece. Here, the visual data is obtained from a camera deployed on top of the robot arm that handles workpieces throughout the whole procedure. The input image of the workpiece is subjected to perspective distortions, in which, such distortions are rectified by pre-processing the input utilizing some visual references placed on the workpiece storage, engraver and milling machine. Next the processed input is used to identify the workpiece edges and then, to compute the centre and the radius of the largest circle (and then the area), which can be placed within the workpiece surface (for an example, see Figure 7). When AI fails to identify the usable area, the request is placed for human intervention as illustrated in Figure 8.

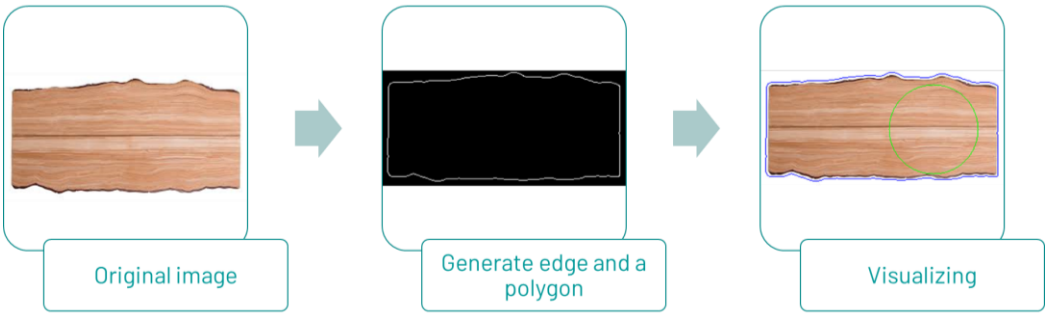

Figure 7. Key steps of usable area detection

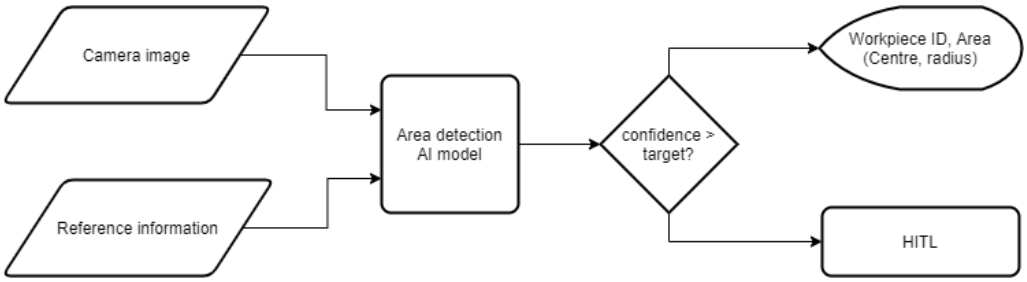

Figure 8. Inference architecture of usable area detection AI

Grabbing and placing workpieces: The next task where AI is involved is with the workpiece transportation. Depending on the shape of the workpiece and the orientation of its placement, AI needs to provide the optimal grabbing point to the robot controller to place the gripper and steadily pick the workpiece. Similar to the prior scenario, the grabbing point in terms of the 2D coordinates (x,y) and the gripper angle is inferred from the visual input of the camera placed at the robot arm. After grabbing the workpiece, utilizing the prior knowledge on the locations of other machines, the destination coordinates are calculated and fed to the robot arm controller via the HyperMAS. Failures of grabbing and placing initiate requests for HITL. The functionalities behind grabbing and placing AI model are illustrated in Figure 9.

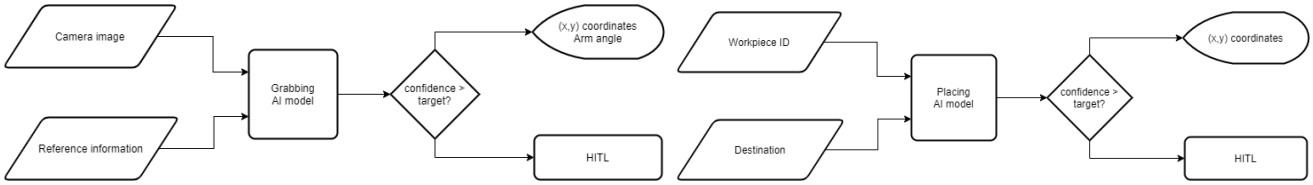

Figure 9. Functionalities of grabbing (left) and placing (right) tasks

Training of the manufacturing AI models is carried out in an unsupervised manner for usable area detection task and in a supervised learning fashion for grabbing and placing tasks. Here, a set of images of workpieces that are placed on the storage, engraver, and milling machine are collected first. For the area detection, the Otsu's method [38] to distinct the foreground and background to identify the edges and geometry to obtain the largest incircle are used with OpenCV toolbox [39]. Therein, the choices of several filter sizes act as training hyperparameters. For grabbing and placing tasks, the images are captured along corresponding labels of grabbing point including (x,y) coordinates and the gripper angle to generate the training dataset. In the supervised training setting, the images are fed into 2D CNN, which is trained to minimize a regression loss in terms of the difference between the predicted grabbing/placing tuples and the label.

Note that the individual robots are unlikely to collect a sufficiently large dataset to train an efficient AI model. Hence, it is essential for multiple robots to collaborate with one another to access a rich dataset. Due to the data privacy and the limitations in communications among robots, sharing raw sensory data is not a viable option. In this regard, the FL framework is adopted to enable collaborative training among the individual manufacturing AI models.

The inference initiates with a manoeuvre request. Using the camera feed and the information about the obstacle, AI model generates a sequence of linear and angular velocities that allow tractor to navigate through the obstacles. Along the control decisions, AI model computes a confidence of the inferred decisions, which is compared with a predefined confidence target. This acts as a safety measure whereas a low confidence is used to request the human aid (HITL).

Table 4. Key components of AI for Manufacturing use case

Property	Description
Input	Workpiece image
Output	Usable area of the workpiece Workpiece grab-spot
NN architecture	CNN
Triggered by	Requests for workpiece details and workpiece transportation Manoeuvre request from tractor controller
Triggers	Human takeover
Sensor	Camera IDS GV-5040FA-C-HQ
Actuator	Robot Arm UR5

2.5 Preliminary Results & Summary

Towards enabling distributed AI under resource constraints, within the scope of IntellIoT objectives, two key research work has been carried out.

2.5.1 CLIENT SCHEDULING TOWARDS HIGH FL ACCURACY

Aligning with the KPI 3.1, the focus of this research is to ensure high FL accuracy compared to a centralized training design while the clients/devices are scheduled under the limited computation and communication resources. Due to the limited number of communication resource block (RB) availability, it is essential to identify the clients with good connectivity conditions that are significantly contributing to the training process. Especially, in absence of perfect knowledge on channel state information (CSI), this becomes a challenging task. Furthermore, the limited processing capabilities at clients are likely to become computation stragglers slowing down the entire training process. In this view, the proposed client scheduling algorithm in [40] [41] address two key aspects: (i) prevent scheduling both communication and computation stragglers and (ii) schedule the clients that have the highest significance in terms of improving the trained model accuracy. Additionally, the impact of CSI prediction along the client scheduling to improve overall training process is accounted therein. The proposes client scheduling algorithms in [40] [41] and baselines and the benchmark is summarized in Table 5.

Table 5. Proposed algorithms, baselines, and benchmark algorithm

	Model	Description
Proposed methods	QAW-GRP	PS uses dataset sizes to prioritize clients and gaussian process regression (GPR)-based CSI predictions for scheduling.
	QAW	PS uses pilot-based CSI estimation and dataset sizes to prioritize clients.
	QUNAW	PS uses pilot-based CSI estimation without accounting dataset size of clients.
Baselines	RANDOM	Clients are scheduled randomly.
	PF	Clients are scheduled with fairness in terms of successful model uploading.
Ideal setup	IDEAL	All clients are scheduled assuming no communication or computation constraints.

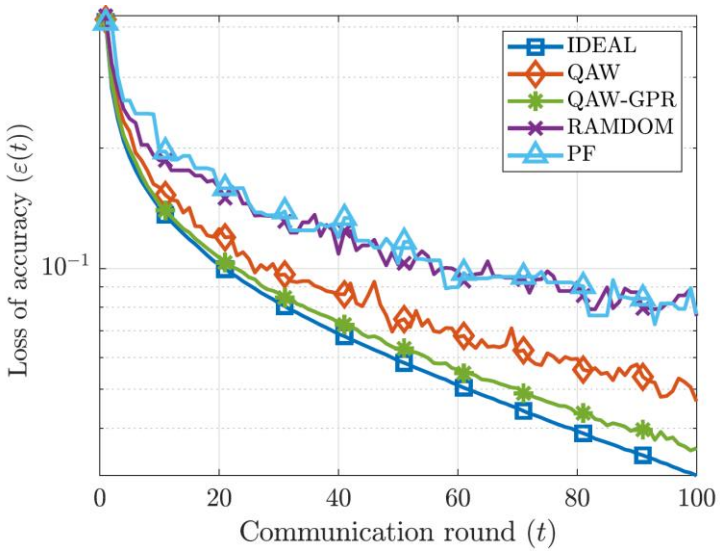

Figure 10. Comparison of the loss of accuracy in FL methods

Figure 10 compares the loss of accuracy $\varepsilon(t)$ in FL methods with different client scheduling at each model aggregation round t with respect to the centralized model training. It can be noted that IDEAL has the lowest loss of accuracy $\varepsilon(100) = 0.03$ due to the unconstrained communication and computation resources. The state-of-the-art scheduling baselines diverge from the centralized models yielding higher losses over each training round. In contrast, the proposed client scheduling algorithms exhibit close to IDEAL performance even under the limited resources.

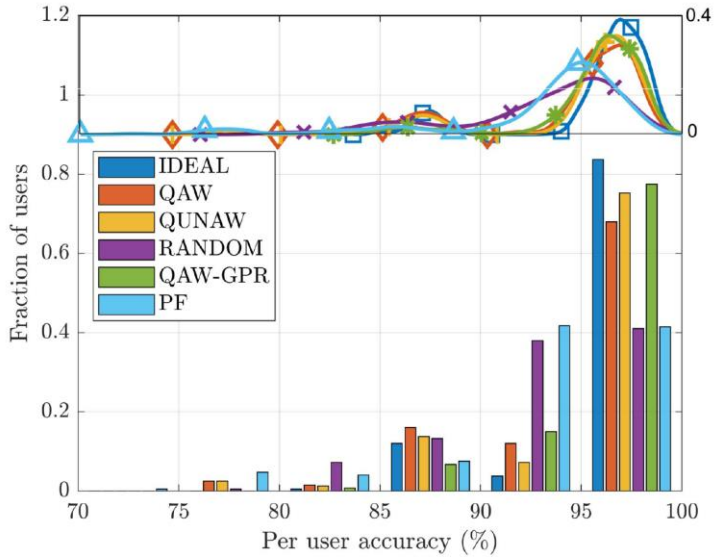

Figure 11. Fairness comparison in terms of the training accuracy among clients

Figure 11 indicates how well the global model generalizes for individual clients. Therein, the global model is used at the client side for inference, and the per client histogram of model accuracy is presented. The global model in IDEAL generalizes well over the clients yielding the highest accuracy on average (96.8%) over all clients and the lowest variance with 5.6. The proposed algorithms achieve 95.2%-96.1% average accuracies with variance about 7.8 while CSI-and-processing power-agnostic RANDOM and PF yield an average accuracy of 93.4%and 94.1% respectively with

the highest variance of 18.2 and 17.6. This indicates that client scheduling without any insight on data size distribution and CSI fails to provide high training accuracy or fairness under communication constraints.

2.5.2 ENERGY-EFFICIENT MODEL COMPRESSION AND SPLITTING

The goal of this research is to address the KPI 3.4 by enabling higher energy efficiency at the edge. In this view, the goal is to minimize the total energy at the client-side by performing model compression in order to reduce the model size and time-varying model splitting between the client and the server to account for the trade-off between local computation and transmission energies over time-varying wireless channels. By defining the output of each layer in the NN as a representation for the input data, the optimal design in terms the minimal energy consumption for the generation and communication of a representation under dynamic channel conditions is derived [42]. To further improve communication efficiency, the representation is compressed prior to transmission.

Figure 12 shows the processing energy and the output size for every layer in the NN before and after compression. It can be noticed that the compression step significantly reduces the processing energy for more than 98% for first-most fully connected (fc1 and fc2) layers which comprise around 60% of the NN parameters. Moreover, the decrease in the output size of the layers is reflected by eliminating entire neurons after pruning.

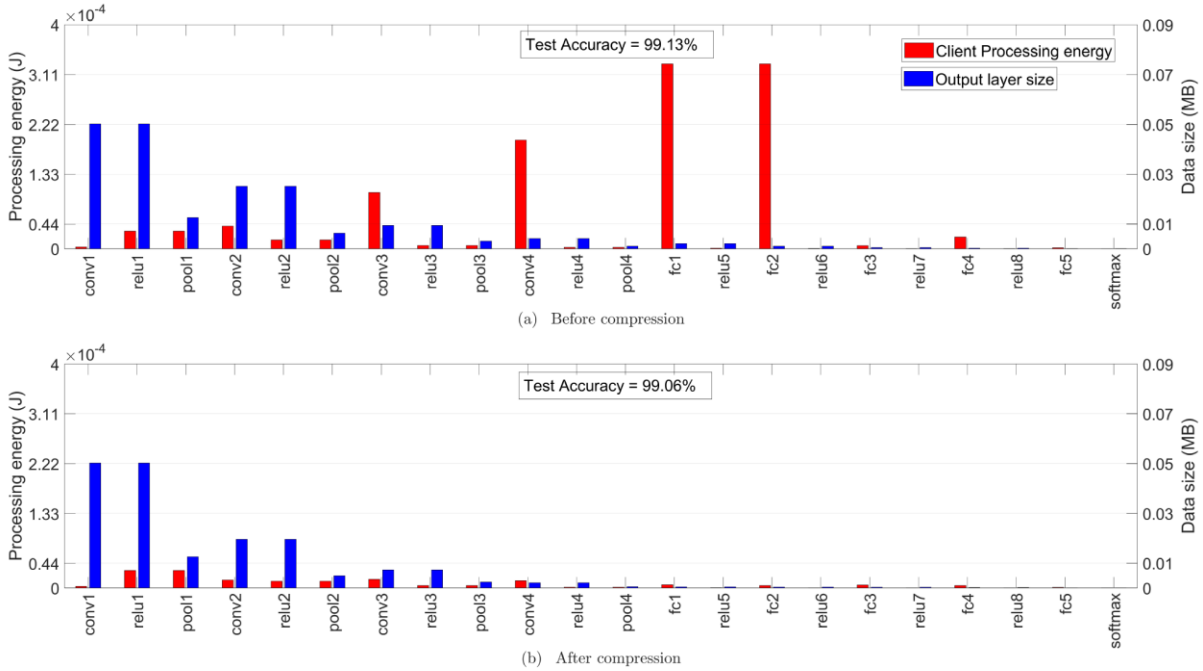

Figure 12. Per layer processing energy and output size before and after compression

The energy efficiency aspects of the proposed layer selection (PS) solution are then compared with three baselines namely (i) *Pool-1*: The client processes the data up to the first pooling layer and shares with the server, (ii) *softmax*: The client runs over all the model layers and transmits the labels, and (iii) *PSNC*: The client selects the optimal layer for a given channel observation employing the original design and transmits it without any compression. Table 6 compares the energy consumption (E: total, E_p: processing at client, E_{tr}: transmission) and CO_2 emission [43] results for different values of the bandwidth W. The proposed design PS outperforms all the other baselines in terms of total energy consumption and consequently CO_2 emission. For W = 15KHz, softmax performs slightly similar to PS. This is consequent to the high transmission energy at the low bandwidth regime. For W = 15MHz, the abundance of bandwidth enables the client to process few layers and transmit larger output sizes to save processing energy. This is observed by the remarkable performance of PSNC since most NN parameters are positioned in the fully connected layers.

Table 6. Comparison of energy consumption and CO_2 emission

	POOL-1	SOFTMAX	PS	PSNC
		$W = 15KHz$		
$E\ (J)$	8.66×10^{-4}	1.69×10^{-4}	1.69×10^{-4}	4.36×10^{-4}
$E_p\ (J)$	0.66×10^{-4}	1.67×10^{-4}	1.64×10^{-4}	2.60×10^{-4}
$E_{tr}\ (J)$	7.98×10^{-4}	0.03×10^{-4}	0.05×10^{-4}	1.76×10^{-4}
$EU\ CO_2e\ (kg)$	0.66×10^{-10}	0.13×10^{-10}	0.13×10^{-10}	0.33×10^{-10}
		$W = 15MHz$		
$E\ (J)$	1.24×10^{-4}	1.67×10^{-4}	0.53×10^{-4}	0.61×10^{-4}
$E_p\ (J)$	0.66×10^{-4}	1.67×10^{-4}	0.23×10^{-4}	0.23×10^{-4}
$E_{tr}\ (J)$	0.58×10^{-4}	0.002×10^{-4}	0.31×10^{-4}	0.38×10^{-4}
$EU\ CO_2e\ (kg)$	0.1×10^{-10}	0.12×10^{-10}	0.04×10^{-10}	0.05×10^{-10}

3 TRAINING AND RETRAINING TOWARDS ROBUST AND PERSONALIZED AI

Robust design of AI is essential for the long-term success of the semi-autonomous system design. Towards this, IntellIoT focuses on monitoring and out of distribution detection, AI model personalization, and AI model retraining.

3.1 Monitoring and Out of Distribution Detection

Federated Learning brings valuable benefits by enabling model training, refinement, and evaluation with remote data. These include enhanced privacy, support for continuous learning and hyper-personalization, and enabling in product and post-deployment learning and validation. At the same time, Federated Learning does not tackle aspects around data pre-processing, curation, and quality assessment. In the case of a centralized approach, this could be carried out by the researchers developing the model who have access and can inspect the data. In the case of Federated Learning, the modelers do not see the remote data and need to develop a range of tools enabling them to decide without inspecting the data if the training data is of sufficient quality and to detect any data, labelling or annotation issues that could affect the quality of the model.

In our solution we will develop several safeguards that enable us to preserve the integrity of the model. Firstly, the Global AI Component will use a reference of high quality, curated dataset to assess the quality of the updates received from the workers. If the updates deteriorate the performance of the model on the reference dataset, these will be ignored, and the previous model parameters will be shared for the next training round. We will as well investigate methods to remotely assess the collected data before it is used for training and without being aggregated and seen, for instance by detecting labels that do not fit the data, data elements that are out of distribution compared to the previous collected data, and data values that are inconsistent, conflicting, or out of the possible ranges. Another aspect that is important is to differentiate between rare cases and measurements that are highly valuable for training and should not be ignored, and data that is of low quality or incorrect (for instance if devices are not correctly placed).

Post-deployment, the solution will enable the monitoring of the model by collecting ground truth during model execution and leveraging (sample) data to detect performance issues, potential model drift with changes in data, and rare cases in which the model prediction is incorrect. Relevant rare cases will be collected and leveraged to refine the model and to extend the reference dataset at the coordinator.

3.2 AI Model Personalization

An important aspect in healthcare scenarios is model personalization, with challenges due to unbalanced and non-identically distributed data that may damage loss convergence and reduce model robustness. To handle such problems, a federated learning system for personalization must have "local adaptors". Globally trainable weights specialize in extraction of task-specific features from the data after the training, whereas locally trainable weights learn specific transformations of those features. Only trainable parameters are shared among the workers, while the normalization statistics remain private during the training. In experiments of such a system, the central model achieves the baseline performance. However, "local adaptors" cause discrepancy between workers' normalization statistics after the training. Therefore, the system must support the persistent model state. This setup allows a researcher or hospital employee to categorize model parameters into sharable and private and allows an evaluation procedure on local data.

To personalize recommendations for patients of cardiovascular disease in a remote patient monitoring setting, we envision creating models for adherence, blood pressure trends, arrhythmia and possibly other abnormal heart rhythms, deterioration, engagement, and mood. The recommendations will be communicated to a physician who has the role of human-in-the-loop in the healthcare use case UC2.

3.3 AI Model Retraining

One of the main challenges in the data-driven ML is that the training data may not reflect the inference scenarios. This could be either due to unknown variables that were not accounted, the presence of unseen data corresponding to rare or extreme events, or temporal and/or spatial changes yielding drifts in data distributions. Specially, when the data is generated at the network edge with on-device constraints, it is possible to have clients with immature datasets that fail to characterize the entire data distribution. Hence, it is vital to track the model performance during inference and intervene when AI fails to complete its responsible tasks.

Within the scope of IntelIIoT, the HITL plays a major role in terms of intervening the semi-autonomous systems to ensure the success of task completion. In agriculture use case, a human driver may take over the control of the tractor to avoid obstacles, a physician may provide expert opinions in the healthcare use case, and a human driver may take over the grabbing and placing of workpieces to improve the precision of the task under the manufacturing use case. With each of such interventions, the edge devices expose to new training data due to experts' actions, which can be in return used to improve and refine the AI models. In this view, the growth of training data over time (and over failures) can be utilized to continuously improve the AI via retraining. Here, retraining is envisioned from two directions that are illustrated in Figure 13.

Retraining as initiating FL over the network: When clients are exposed to numerous human interventions, it indicates that the existing AI models require retraining. Since the training data has been expanded over new interventions, the new knowledge that can be obtained by such interventions needs to be shared over the network to improve all local AI models. In this regard, retraining is carried out similar fashion of training under FL, except for that the local datasets are now grown with previously unseen new data samples.

Retraining as updated model discovery: The clients of the IoT networks are expected to be online or offline depending on their roles and due to some external factors (e.g., some tractors could be under repair, patient devices are set to offline, a manufacturing line is under renovations). Hence, such client devices are likely to be relying on AI models that are not up to date with recent retraining iterations. In this view, it is inefficient to initiate retraining over the entire network simply due to the absence of updated AI models. Hence, it is essential to seek for the updated AI models in the network with the aid of IAKM system. Situations where the updated AI models can be discovered with the aid of IAKM system, clients simply download the updated models. Otherwise, the retraining falls back to the retraining as initializing FL.

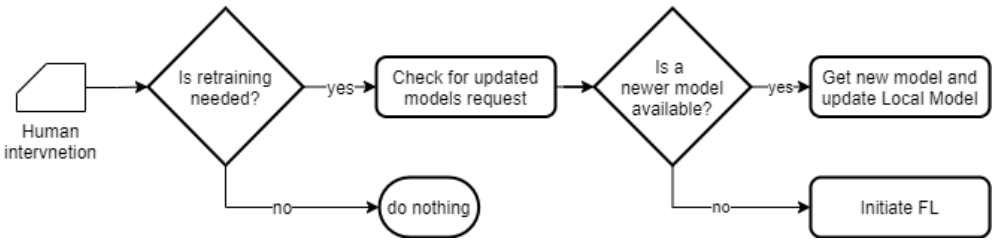

Figure 13. Simplified decision diagram for retraining

3.4 Summary

The goal of the above three tasks is to reduce the human interventions over time. The maturity of these tasks will be evaluated after the deployment and testing of the AI models in the three use cases over Cycle 2.

4 CONCLUSION & FUTURE DIRECTION

The Task 3.2 of IntellIoT focuses on several key (i) algorithmic components towards enabling distributed AI over IoT networks considering the resource constrains due to communication and computation and (ii) software components that enable FL over agriculture, healthcare, and manufacturing use cases while enabling semi-autonomous behaviour via use case-specific AI components. During the Cycle 1, the research work has been carried out to lay the foundation for developing software components and then, the preliminary use case-specific models are developed. This includes data collection, data cleaning and refining, and planning and designing ML architectures complying with the requirements set by the use case-specific tasks.

Cycle 2 will focus on completion of research on ML model reduction (KPI 3.2) as well as further refining ML models addressing the aspects of robustness and personalization. It is worth highlighting that some choices that are currently implemented in terms of ML architectures, definitions of inputs, monitoring, and detection of out of distributions are subject to change during the Cycle 2 to enhance the overall performance of each use case.

5 ANNEX

List of abbreviations

- AGV: Autonomous Guided Vehicles
- AI: Artificial Intelligence
- AR: Augmented Reality
- BP: Blood Pressure
- CML: Communication for Machine Learning
- CNN: Convolutional Neural Network
- CSI: Channel State Information
- DRAM: Dynamic Random-Access Memory
- E2E: End-to-End
- Fitrec: Fitness Recommendation
- FL: Federated Learning
- GPR: Gaussian Process Regression
- GPU: Graphics Processing unit
- HBM: High-Bandwidth Memory
- HDD: Hard Disk Drive
- HITL: Human-in-the-Loop
- HR: Heart Rate
- IID: Independent and Identically Distributed
- IoT: Internet of Things
- LSTM: Long Short-Term Memory
- MEC: Mobile Edge Computing
- ML: Machine Learning
- MLC: Machine Learning for Communication
- MPI: Message Passing Interface
- MSE: Mean Squared Error
- MSI: Model State Information
- NBDT: Neural-Backed Decision Trees
- NG: Next Generation
- NN: Neural Network
- PS: Parameter Server
- RB: Resource Block
- SDD: Solid State Drive
- SE: Spectral Efficiency
- SRAM: Static Random-Access Memory
- TPU: Tensor Processing Unit
- URLLC: Ultra-Reliable Low Latency Communication
- VCU: Vehicle Control Unit
- VR: Virtual Reality

6 REFERENCES

[1] J. Park, S. Samarakoon, M. Bennis and M. Debbah, "Wireless network intelligence at the edge," *Proceedings of the IEEE*, vol. 107, p. 2204–2239, 2019.

[2] N. a. G. A. Farsad, "Neural network detection of data sequences in communication systems," *IEEE Transactions on Signal Processing,* vol. 66, no. 21, pp. 5663-5678, 2018.

[3] M. a. C. U. a. S. W. a. Y. C. a. D. M. Chen, "Artificial neural networks-based machine learning for wireless networks: A tutorial," *IEEE Communications Surveys & Tutorials,* vol. 21, no. 4, pp. 3039-3071, 2019.

[4] T. a. H. T. Ben-Nun, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," *ACM Computing Surveys (CSUR),* vol. 52, no. 4, pp. 1-43, 2019.

[5] T. G. Dietterich, "Steps toward robust artificial intelligence," *AI Magazine,* vol. 38, no. 3, pp. 3-24, 2017.

[6] J. Konecny, H. B. McMahan, D. Ramage and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527,* 2016.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, 2017.

[8] J. Konecny, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492,* 2016.

[9] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, 2019.

[10] B. Zhang, A. Davoodi and Y. H. Hu, "Exploring energy and accuracy tradeoff in structure simplification of trained deep neural networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems,* vol. 8, p. 836–848, 2018.

[11] Y. Wang, M. Sheng, X. Wang, L. Wang and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications,* vol. 64, p. 4268–4282, 2016.

[12] M. S. Elbamby, C. Perfecto, C.-F. Liu, J. Park, S. Samarakoon, X. Chen and M. Bennis, "Wireless edge computing with latency and reliability guarantees," *Proceedings of the IEEE,* vol. 107, p. 1717–1737, 2019.

[13] C. G. a. D. Boneh, "A Fully Homomorphic Encryption Scheme," *Stanford University [PhD Thesis],* 2009.

[14] Nvidia, "NVIDIA A100 Tensor Core GPU," [Online]. Available: https://www.nvidia.com/en-us/data-center/a100/.

[15] "InfiniBan Architecture Specification," InfiniBand Trade Association, [Online]. Available: https://www.infinibandta.org/ibta-specification/.

[16] D. K. Panda, "RDMA-Based Networking Technologies and Middleware for Next-Generation Clusters and Data Centers," [Online]. Available: http://conferences.sigcomm.org/sigcomm/2018/files/slides/kbnet/keynote_1.pdf.

[17] Google, "Cloud TPU," [Online]. Available: https://cloud.google.com/tpu.

[18] L. a. Y. J. a. Z. Y. a. W. W. a. L. A. a. S. S. L. a. X. Z. a. K. T. Wang, "Superneurons: Dynamic GPU memory management for training deep neural networks," in *Proceedings of the 23rd ACM SIGPLAN symposium on principles and practice of parallel programming*, 2018.

[19] 3GPP, "Study on New Radio Access Technology: Radio Access Architecture and Interfaces," 3GPP TR 38.801, 2017.

[20] Qualcomm, "Snapdragon 845 Mobile Platform," [Online]. Available: https://www.qualcomm.com/products/.

[21] GSMARENA, "Apple iPhone XS Max," [Online]. Available: https://www.gsmarena.com/apple_iphone_xs_max-9319.php.

[22] GSMARENA, "Google Pixel 3 XL," [Online]. Available: https://www.gsmarena.com/google_pixel_3_xl-9257.php.

[23] E. Briman, "Another look at AlphaGo vs. Lee Sedol: The Power Angle," CEVA, [Online]. Available: https://www.ceva-dsp.com/ourblog/artificial-intelligence-leaps-forward-mastering-the-ancient-game-of-go/.

[24] N. Farsad, M. Rao and A. Goldsmith, "Deep learning for joint source-channel coding of text," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[25] S. Dorner, S. Cammerer, J. Hoydis and S. Ten Brink, "Deep learning based communication over the air," *IEEE Journal of Selected Topics in Signal Processing,* vol. 12, p. 132–143, 2017.

[26] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," *IEEE Transactions on Signal Processing,* vol. 66, p. 5438–5453, 2018.

[27] O. Naparstek and K. Cohen, "Deep multi-user reinforcement learning for dynamic spectrum access in multichannel wireless networks," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, 2017.

[28] U. Challita, L. Dong and W. Saad, "Proactive resource management for LTE in unlicensed spectrum: A deep learning perspective," *IEEE transactions on wireless communications,* vol. 17, p. 4674–4689, 2018.

[29] P. Popovski, O. Simeone, F. Boccardi, D. Gunduz and O. Sahin, "Semantic-effectiveness filtering and control for post-5G wireless connectivity," *Journal of the Indian Institute of Science,* vol. 100, p. 435–443, 2020.

[30] J. Wang and G. Joshi, "Cooperative SGD: A unified framework for the design and analysis of local-update SGD algorithms," *Journal of Machine Learning Research,* vol. 22, p. 1–50, 2021.

[31] J. Liu and Q. Zhang, "Offloading schemes in mobile edge computing for ultra-reliable low latency communications," *Ieee Access,* vol. 6, p. 12825–12837, 2018.

[32] "Agriculture Safety," The National Institute for Occupational Safety and Health (NIOSH), [Online]. Available: https://www.cdc.gov/niosh/topics/aginjury/default.html.

[33] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konecny, S. Mazzocchi, H. B. McMahan and others, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046,* 2019.

[34] D. Ho, *NBDT: Neural-Backed Decision Trees,* EECS Department, University of California, Berkeley [Msc Thesis], 2020.

[35] S. a. S. J. Hochreiter, "Long short-term memory," *Neural computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[36] "Classes of Heart Failure," American heart association, [Online]. Available: https://www.heart.org/en/health-topics/heart-failure/what-is-heart-failure/classes-of-heart-failure.

[37] J. Ni, L. Muhlstein and J. McAuley, "Modeling heart rate and activity data for personalized fitness recommendation," in *The World Wide Web Conference*, 2019.

[38] D. Liu and J. Yu, "Otsu method and K-means," in *2009 Ninth International Conference on Hybrid Intelligent Systems*, 2009.

[39] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[40] M. M. Wadu, S. Samarakoon and M. Bennis, "Federated learning under channel uncertainty: Joint client scheduling and resource allocation," in *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, 2020.

[41] M. M. Wadu, S. Samarakoon and M. Bennis, "Joint Client Scheduling and Resource Allocation under Channel Uncertainty in Federated Learning," *IEEE Transactions on Communications*, 2021.

[42] M. Krouka, A. Elgabli, C. B. Issaid and M. Bennis, "Energy-Efficient Model Compression and Splitting for Collaborative Inference Over Time-Varying Channels," *arXiv preprint arXiv:2106.00995*, 2021.

[43] "Greenhouse gas emission intensity of electricity generation in Europe," European Environment Agency, [Online]. Available: https://www.eea.europa.eu/data-and-maps/indicators/overview-of-the-electricity-production-4/assessment.

[44] H. Pishro-Nik, "Introduction to probability, statistics, and random processes," 2016.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.