3

Edge Intelligence Architecture for Distributed and Federated Learning Systems

Pierluigi Dell'Acqua, Lorenzo Carnevale, and Massimo Villari

University of Messina, Italy

Abstract

In recent years, deep neural networks have achieved success in Electric Vehicles (EVs) monitoring, primarily due to their scalability with large-scale data and numerous model parameters. However, EVs rely on resource-constrained edge devices that struggle with complex models, and data privacy concerns prevent sharing data outside the owning device. Federated Learning (FL) and Knowledge Distillation (KD) have emerged as key solutions, enabling model simplification and distributed training on private data. FL allows models to be trained locally on edge devices, addressing privacy concerns while keeping data decentralized and avoiding central server dependencies. This approach requires lightweight models optimized for edge intelligence deployment. To address this challenge, we propose architectural solutions leveraging FL, KD and model compression techniques to create simplified Artificial Neural Networks (ANNs) suitable for edge devices in EVs. The proposed architecture integrates these methods into a federated environment, ensuring distributed training while maintaining computational efficiency for EV monitoring and predictive maintenance applications. By combining FL, KD, and model compression, our approach enables efficient and privacy-preserving Machine Learning (ML) models, enhancing Edge Intelligence (EI) for EV monitoring in resource-constrained settings.

Keywords: knowledge distillation, federated learning, edge computing, IoT, edge intelligence.

3.1 Introduction

The automotive industry has witnessed a substantial expansion in both the scale and intricacy of electrical and electronic system architectures. In this regard, the EVs production is becoming increasingly prevalent in the field. Therefore, the challenge of predicting diagnosing faults and improving the LI-ion Battery (LIB) lifetime in EVs becomes progressively more demanding.

Modern monitoring systems approach the battery state parameters maintenance, such as the State of Charge (SoC), State of Health (SoH), State of Power, Remaining Useful Life, within safe limits, safeguarding the battery's safety. These are based on ML and Artificial Intelligence (AI) models, which can adapt the analysis to the specific technology. Furthermore, the computational trend is moving the data elaboration to the edge, with no requiring EVs producers to share their own diagnostic data as training datasets, preserving the industrial secrets and users' privacy. Actually, edge devices are characterized by low capacity and low performance that generally do not allow complex operations. For this reason, model simplification becomes necessary.

This scenario can be intended as a specific EI scenario, where AI models and algorithms are deployed and executed on resource-constrained edge devices. EI refers to the integration of AI capabilities directly on edge devices, enabling real-time data processing, decision-making and autonomy at the edge of the network. In such contexts, the need to balance computational demands with limited hardware resources is critical. Therefore, FL strategies, that aim to train models in a distributed manner and keep data locally on users' devices to preserve the privacy, can be adopted. The basic idea of FL unfolds in several key stages: i) a model based on an ANN is centrally initialized and subsequently disseminated to various peripheral devices; ii) these devices independently train the model using their locally available data, sending back for aggregation [17] to the central server only the outcomes of this localized training, such as the model weights.

This work aims to define a comprehensive and unified EI architecture designed to the specific demands of EVs monitoring systems. The goal is to leverage the potential of edge computing and distributed AI strategies to

improve the real-time diagnosis, prediction, and overall health management, specifically of LIBs. Within this framework, the FL strategy is a cornerstone, providing an effective solution for decentralized learning. By enabling model training on local devices FL ensures that sensitive diagnostic data remains on the edge, thereby enhancing both privacy and security by preserving industrial know-how and user-specific data.

Moreover, to meet the computational efficiency requirements inherent to resource-limited edge devices, the architecture also incorporates complementary compression methods. Among these, KD not only facilitates the transfer of knowledge from a larger, more complex "teacher" model to a smaller, more efficient "student" model—making it ideal for deployment on devices with limited computational power—but also mitigates a well-known limitation of FL: KD strategies can effectively address both heterogeneous data and heterogeneous models within an FL environment [18].

In summary, this architecture proposes the integration of FL, KD, and compression techniques to create a scalable, efficient, and privacy-preserving solution for enhancing the monitoring and lifetime management of EVs within an EI scenario.

The remaining of the paper is organized as follow. Section 1.2 reports the EI background and discuss the recent advance in the literature. Section 1.3 determines the scenario where the proposed architecture may be deployed. That architecture is, therefore, discussed in Section 1.4. Finally, Section 1.5 summarizes the paper's insights and brings light on the future research activities.

3.2 Related Works

3.2.1 Edge Intelligence

In most of the cases, the computation on edge devices leverages data owned by them-self, data not shared with others for privacy reasons. In this context, EI leverages data generated at the edge of the network by applying AI directly to it. As stated in [28], there is not a formal definition of EI, though it is commonly used to describe the execution of AI models on edge devices. Therefore, they define EI as the efficient utilization of data in a cooperative edge-cloud system, where both inference and training can occur across all devices. This prospective is outlined by a six-level framework that categorizes where applications are executed, as reported in figure 3.1:

- Cloud Intelligence: a model (e.g., Deep Neural Network) is fully trained and executed in the cloud.
- Level 1 — Cloud–Edge Coinference and Cloud Training: a Deep Neural Network (DNN) model is trained in the cloud, but inference is performed through a cooperation between the cloud and the edge. In this case, part of the data is offloaded (e.g., migrated) to the cloud.
- Level 2 — In-Edge Coinference and Cloud Training: the DNN model is trained in the cloud, but inference is carried out at the network edge. In this setup, inference is handled by edge nodes or nearby devices, with data either fully or partially offloaded to the edge using device-to- device (D2D) communication.
- Level 3 — On-Device Inference and Cloud Training: the DNN model is trained in the cloud, but inference is performed entirely on the device itself, with no data being offloaded.

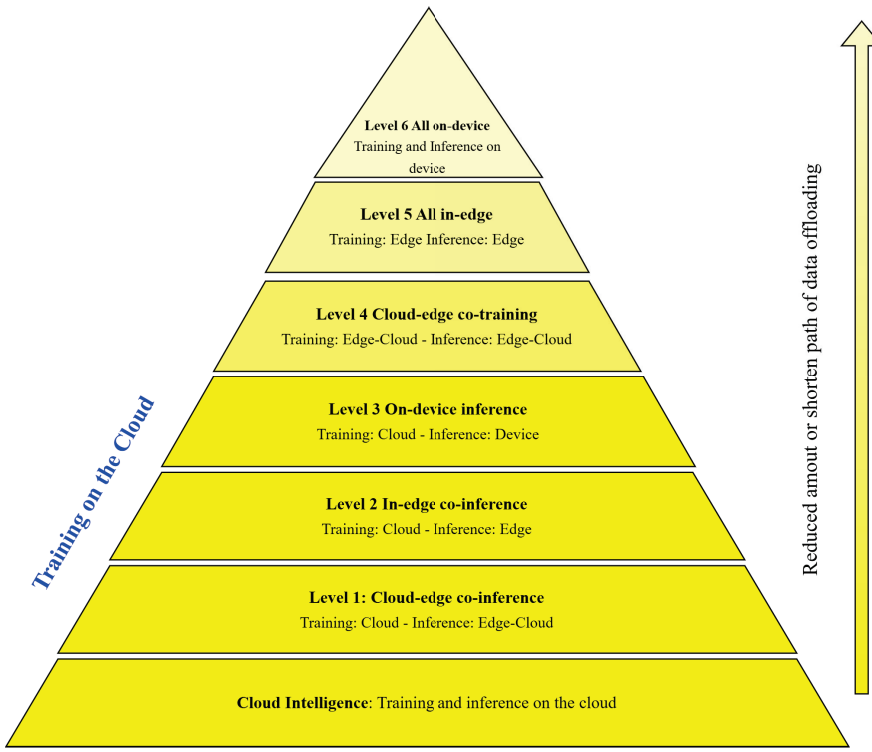

Figure 3.1 Six-level rating for EI described in [28].

- Level 4 — Cloud–Edge Cotraining and Inference: Both the training and inference of the DNN model are con- ducted in a cloud-edge cooperative manner.
- Level 5 — All In-Edge: Both training and inference of the DNN model occur at the network edge.
- Level 6 — All On-Device: Both training and inference are handled entirely on the device.

As the level of EI increases, the amount and distance of data offloading decrease. This leads to lower data transmission latency, improved privacy and reduced Wide Area Network (WAN) bandwidth costs. However, this comes at the expense of increased computational latency and higher energy consumption. Therefore, there is not universally "best" level of EI. The optimal level is application-dependent and should be determined by considering multiple factors, such as latency, energy efficiency, privacy and WAN bandwidth cost.

3.2.2 Federated Learning

Among the different technologies designed to implement training in edge devices (e.g., Aggregation Frequency Control, Gradient Compression, Gossip Training), FL is often adopted where the learning process involves a federation of devices or nodes that do not need to share their own data but only their optimized model's parameters.

FL progresses through key stages, as reported in figure 3.2. Firstly, a neural network model is initialized in the central server and distributed to peripheral clients. Therein, the model is trained with local data only, sampled by the device itself. Importantly, this training happens autonomously, without sharing raw data with a central server. Only the outcomes of the training, which are the optimized model weights, are sent back to the central server, which aggregates them using an aggregation strategy (e.g., Federated Average [17], others [20] that implements the adaptive FedAvg, Lazy and Quantizatized gradients [4], [22]). This process maintains data privacy as raw data stays on local devices. This iterative process allows the model to evolve and improve over time without necessitating the aggregation of raw data in a central repository (Figure 3.2). The DFedAvgM framework [23] operates without a central server aggregator. It's implemented on clients connected through an undirected graph, where each client performs stochastic gradient descent with momentum and communicates only with its neighbors.

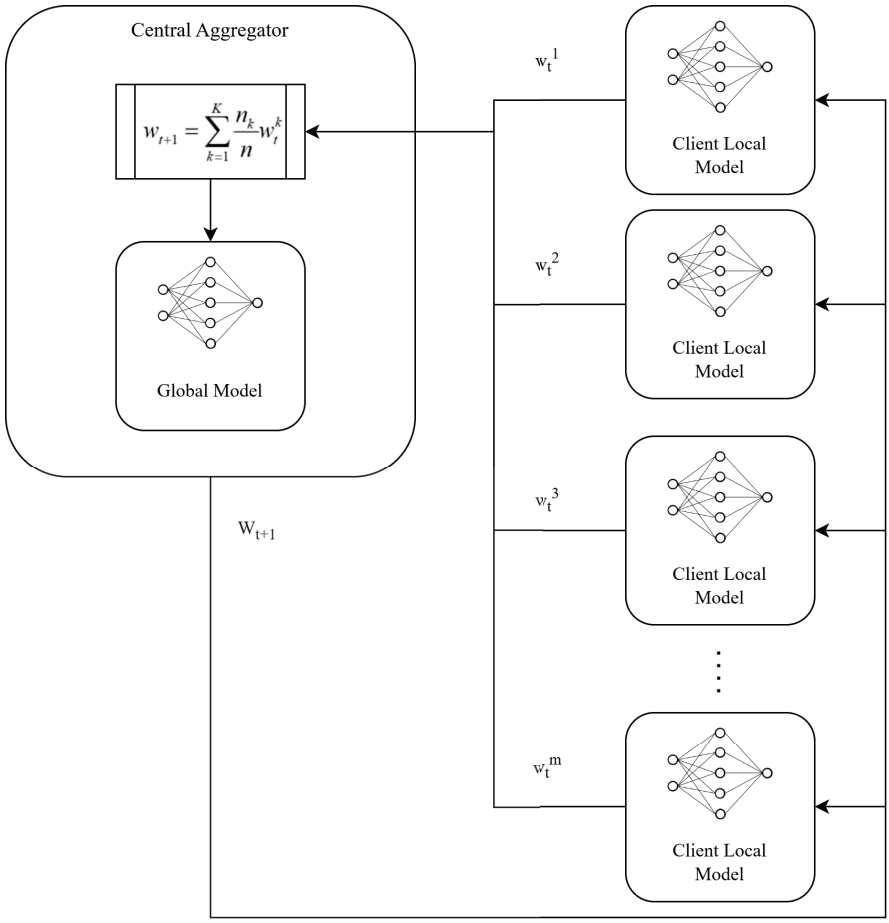

Figure 3.2 In a Federated Learning scenario, each client trains its model leveraging its own private data and sends its model parameters to a central server. The central server aggregates the parameters received from each client to enhance the performance of the central global model, which is then sent back to the clients.

This reduces communication costs and enhances privacy protection. The authors introduce DFedAvgM and its quantized algorithms offering extensive numerical verification of its performance.

3.2.3 Model Compression

In most of the cases, peripheral resource-constraint devices, with low computational capabilities, are not able to handle complex AI models because they

cannot guaranty acceptable performance in terms of energy consumption, memory footprint and latency, i.e., in inference processes.

The main idea behind quantization is to convert the weights and activation values of a neural network model from high precision to lower precision, thereby reducing memory usage and latency [15]. Typically, quantization is applied to a pre-trained neural network, a process known as post-training quantization, without any fine-tuning [3], [6].

As an alternative, the Pruning methods [8] aim to remove non-essential components from DNN models while minimizing the impact on performance. Over time, pruning techniques evolved into two categories: i) the structured pruning as channels that serve as the primary pruning units; while ii) the unstructured pruning employs heuristic techniques to eliminate insignificant parameters, such as low-weight values, gradients or Hessian statistics [15].

Authors in [28] categorize EI technologies into training and inference technologies, as shown in Table 3.1. We have enhanced this table by incorporating KD, which serves as a strategy that bridges the two categories. While KD is implemented during the training phase, its primary goal is to produce a compressed model. The basic concept of KD [10] lies in training a simpler model (e.g., the student model) to imitate the behavior of the original, larger model (e.g., the teacher model). It produces a more efficient and quicker model to be executed. The teacher-student model is reported in figure 3.3.

In [7], different orthogonal distillation techniques classifications have been proposed. For example, the KD schema differentiates offline from online. Methods falling in the first category mainly focus on improving different parts of the knowledge transfer, including the design of knowledge and the loss functions. The [25] defines the distilled knowledge as the flow in the problem resolution trajectory, [13] uses Singular Value Decomposition (SVD) and Radial Basis Functions (RBf) for accuracy enhancement and minimizing computational costs, [19] incorporates distillation loss into the training of a smaller student network whose weights are quantized. On the other hand, the online distillation strategies aim to train both teacher and student models in a unified training scheme or, in vary common scenarios, to deal with lack of teacher network. The [26] proposes a collaborative learning strategy, [12] builds a multi-branch variant of a specified network by adding auxiliary branches, [24] leverages an additional classifier facilitating collaborative learning [21] and mutual learning without the need for pre-training a high- capacity teacher model. In many real-world applications,

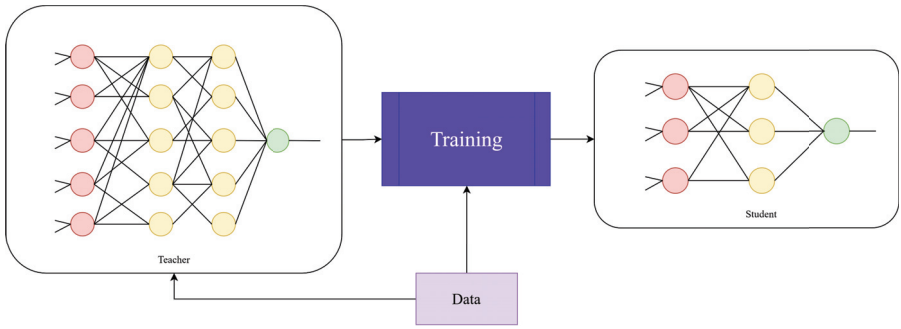

Figure 3.3 The schema illustrates the fundamental concept of KD: during the training of a simplified neural network, knowledge from a larger network is transferred to the smaller one.

Table 3.1 From Train to Inference technologies.

Technology	Highlights
Federated Learning	Training data remains on individual devices, while a shared model is trained centrally by aggregating locally computed updates, ensuring data privacy
Aggregation Frequency Control	Optimize the balance between local updates and global parameter aggregation, all within the given resource constraints
Gradient compression	Involves converting each element of the gradient vectors to a low-precision, finite-bit value. Gradient sparsification, on the other hand, reduces communication overhead by transmitting only a subset of the gradient vector values.
DNN Splitting	Select a splitting point to reduce latency as much as possible
Knowledge Distillation	Transfer the learnings of a large pre-trained model, the "teacher model," to a smaller "student model"
Model Compression	Pruning and quantization methodologies
Model Partition	Computation tasks are offloaded to edge servers or mobile devices, with a focus on optimizing both latency and energy efficiency for better performance
Model Early-Exit	Accuracy-aware
Edge Caching	Fast response towards reusing the previous results of the same task
Input Filtering	Detecting difference between inputs, avoiding abundant computation
Model Selection	Inputs-oriented optimization and accuracy-aware
Support for Multi-Tenancy	Scheduling multiple DNN-based task

FL meets limitations and several KD frameworks are designed to overcome them. For example, the [9] improves communication efficiency reducing communication among clients, [5] adopts a grouping strategy to group clients that share homogeneous resources improving communication efficiency and balancing computing resources, [1], [2], [16] focus mainly on handling model and data heterogeneity.

3.2.4 Beyond the State of the Art

Actually, the scientific literature is challenging proposing a standard architecture for EI. Many methodologies may be involved both for training and inference. FL, KD, quantization and pruning are examples of enabling key technologies available for the exploitation of AI models into the edge of the network. This work aims to propose an architecture that includes all the mentioned methodologies on a simple workflow that optimizes the deployment and execution of EI solutions for EV.

3.3 Use Case

The use case focuses on the development of health monitoring systems for LIBs lifetime and contextual risk assessment in EVs. Among the various monitoring strategies, ML-based methods provide high accuracy, although they require large datasets for effective training [14]. Due to the complexity and critical nature of managing LIBs in EVs, deploying ML systems on resource-constrained microcontroller-based platforms is crucial. This integration enables real-time monitoring and predictive maintenance, optimizing battery performance and extending operational life.

Although cloud computing offers incomparable performance that can be leveraged for centralized activities (e.g., initial training, FL central aggregation), three main reasons lead to the need to push AI computation to the edge:

- *Sensing data*: the sensing layer within EVs produces a continuous and rich flow of data that cannot be transferred to the cloud to prevent bandwidth saturation.
- *Real-time response*: in a monitoring context, it is desirable to have real-time alerts rather than waiting for a stable connection with the central processor.

- *Privacy concerns*: manufacturers are not inclined to share their own diagnostic data with cloud data centers where big data are collected and processed.

The basic flow begins with defining and training an AI model in the cloud, potentially utilizing a more complex model in a KD scenario [10]. Once trained, the model may undergo optimization techniques such as quantization or pruning to reduce its size and improve inference time. After these optimizations, the model is converted into formats compatible with microcontroller-based devices (e.g., ONNX, TinyML), enabling deployment on distributed clients within EVs.

Once each client runs its own optimized AI model, real- time monitoring takes place locally on the EV. However, as new data continuously flows from the vehicle's sensing layer, the model can be further refined. These improvements can be shared with the central cloud, as well as with other clients, when stable connection is available, in a FL environment. This decentralized learning process allows each client to contribute to the overall model improvement without the need to share raw data, thus preserving privacy while enabling continual learning and adaptation (Figure 3.4).

This use case can be classified as level 4 within the EI framework proposed by [28], where both cloud and edge devices collaborate to perform training and inference tasks. However, in scenarios where the cloud is unable to handle training (e.g., due to the lack of a training dataset), the use case

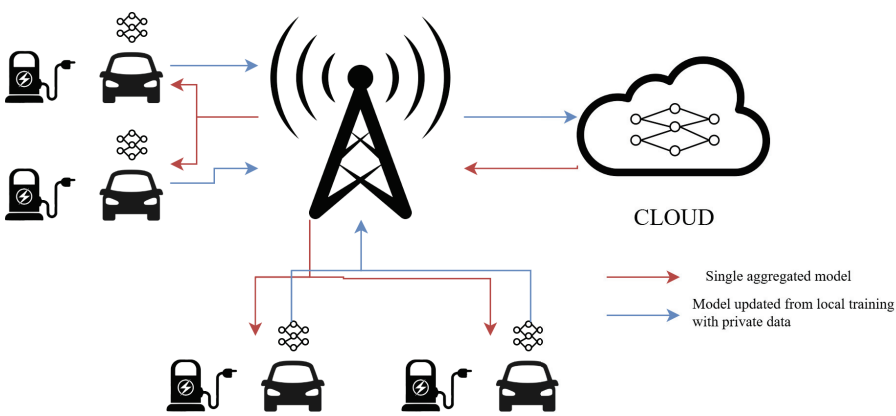

Figure 3.4 Use case scenario.

shifts to a level 5, where the edge devices will be engaged for both training and inference.

3.4 Architecture Proposal

In the context of EVs monitoring, leveraging data-driven approaches based on ML and AI algorithms is a critical point because of the limited computational and energy capacity of the machines. In the following sections, we will build the final architecture step-by-step, describing each involved component.

3.4.1 Assumption

The proposed architecture shares computation responsibility between cloud and edge computing, exploiting EI method- ologies for training and inference, such as the FL, KD, quantization and pruning. The cloud is adopted as much as possible for high-computation activities, such as serving as i) complex AI model training with existing datasets and ii) FL central node aggregator. Most of the inference and a relevant part of the training is intended to be executed on the edge.

The EVs are equipped with low-performance edge devices capable of handling models that are generally not too complex. These devices will perform monitoring and diagnostic tasks on their own sensing data, thereby preserving user privacy constraints. Moreover, since clients do not maintain a continuous connection either with the central cloud or among themselves, communication efficiency should be ensured. Clients will be responsible for performing training and sharing their model parameters to contribute to global model improvements.

3.4.2 Cluster Aggregator

The Cluster Aggregator, depicted in figure 3.5, is deployed in the cloud and primarily functions as the central aggregator for the FL system. This role is facilitated by several key modules responsible for specific tasks to ensure smooth and efficient operations within the federated framework.

1) *Model Aggregation Module*: This module is going to execute the FL core function. It generates the global model by aggregating (e.g., Federated Average [17]) trained models from the peripheral devices. This process ensures that the central model continuously improves by integrating

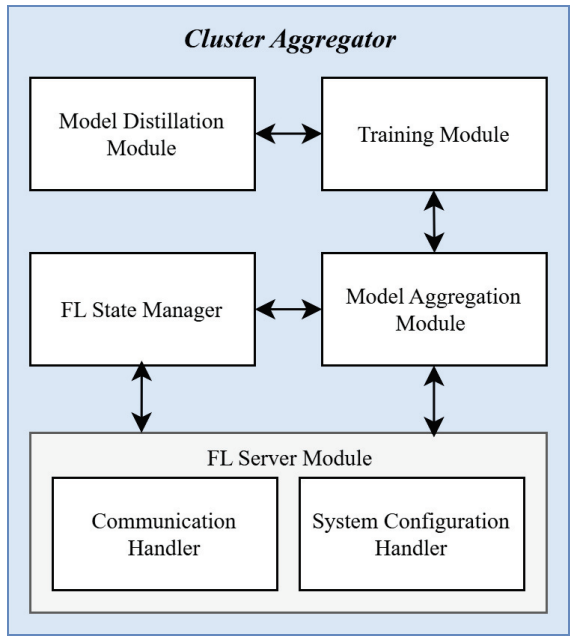

Figure 3.5 Cluster Aggregator schema designed to handle FL central aggregator tasks and to implement a distillation framework adaptable during the training process.

knowledge from distributed nodes without directly accessing their raw data, preserving privacy. This module is designed to handle different forms of model updates and can incorporate advanced techniques, such as weighted averaging, depending on the specific characteristics of the distributed models.

2) *FL State Manager*: It plays a critical role in maintaining synchronization between the cloud-based aggregator and peripheral devices. It tracks the status of each client, ensuring that the aggregation process considers only those clients that have successfully completed their local model training. The state manager keeps a record of which devices are actively participating in each round of FL, their connectivity status, and whether their contributions are valid for aggregation. This component also manages potential failures or delays in communication, ensuring the system can handle interruptions and continue functioning smoothly. Its role becomes even more significant in EVs scenario, where each EV changes its position very frequently and a stable connection cannot be guaranteed.

3) *System Configuration Handler*: The handler is tasked with managing the configuration of the entire system. This module ensures that the software environment is correctly set up, with all dependencies and configurations aligned for optimal performance. Additionally, it handles dynamic updates to system settings, such as changing communication protocols or modifying the aggregation frequency. Moreover, it guarantees that all components are correctly initialized and maintained throughout the lifecycle of the FL process.
4) *Communication Handler*: It manages the communication between the cloud-based aggregator and peripheral de- vices. It sets up and oversees the data transfer channels, ensuring that the communication is both efficient and secure. Given the distributed nature of FL, reliable communication is crucial for transmitting model updates, hyperparameters, and any other necessary metadata between clients and the cloud. The Communication Handler also implements protocols to minimize latency, reduce communication overhead, and ensure data integrity during transfer.
5) *Training and Model Distillation Modules*: The Training Module on the cloud side is activated when the global model is initialized and trained using an existing dataset. Once this initial training phase is completed, the global model is shared with the peripheral clients to begin the federated learning process. The clients use the global model as a starting point, performing local training on their own data and subsequently sharing their model updates with the central aggregator.

In addition to the standard training workflow, the Model Distillation Module plays a crucial role by implementing one or more KD strategies [7], [10], [19] during the training process. These strategies can be employed for several reasons:

- When the global model does not achieve an acceptable level of performance, KD can be used to refine it further by leveraging smaller, more efficient models that capture the key patterns of the original data
- Mainly in regression problem, different KD strategies make up for the lack of the dataset used for training
- [11], [27], allowing the transfer of knowledge from the pre-trained model to the global model without needing access to the original data

- When the FL process is subject to constraints, such as heterogeneous models and/or non-IID data across clients, KD can help align the learning processes [18].

By integrating KD into the training pipeline, the system can enhance the robustness and flexibility of the FL process, improving model performance in scenarios where traditional FL might face limitations.

All Together, the modules described above form a robust infrastructure that supports efficient and secure FL and their combination enables distributed training and aggregation while preserving user privacy, ensuring system reliability and maintaining overall system integrity.

3.4.3 Cloud Components

Although the Cluster Aggregator is the main component deployed on the Cloud, other elements need to be integrated to simplify model training and ANN-based model simplification, making them deployable on resource-constrained devices embedded in EVs (Figure 3.6).

1) *Compression Server:* It is introduced to reduce model complexity and size, addressing the challenges posed by the low computational performance and limited storage capacity of edge devices. It utilizes various handlers (i.e., software components capable of managing specific functionalities) to apply common compression techniques such as quantization, pruning and sparsification. These techniques are crucial for optimizing models to efficiently run on edge devices with constrained resources.
   Quantization reduces the precision of the numerical values used to represent the model's parameters, thereby decreasing both the model's size and its computational requirements. This allows edge devices to process models more efficiently without compromising significant accuracy.
   Pruning involves removing redundant or non-contributing weights from the model, which not only reduces its complexity but can also enhance performance by simplifying the model's structure. This results in a leaner, faster model that is more suitable for deployment in resource-limited environments.
   Sparsification, on the other hand, introduces sparsity into the model by setting insignificant weights to zero. This spar- sity can be leveraged by specialized hardware to accelerate computations, further enhancing the model's performance on edge devices.

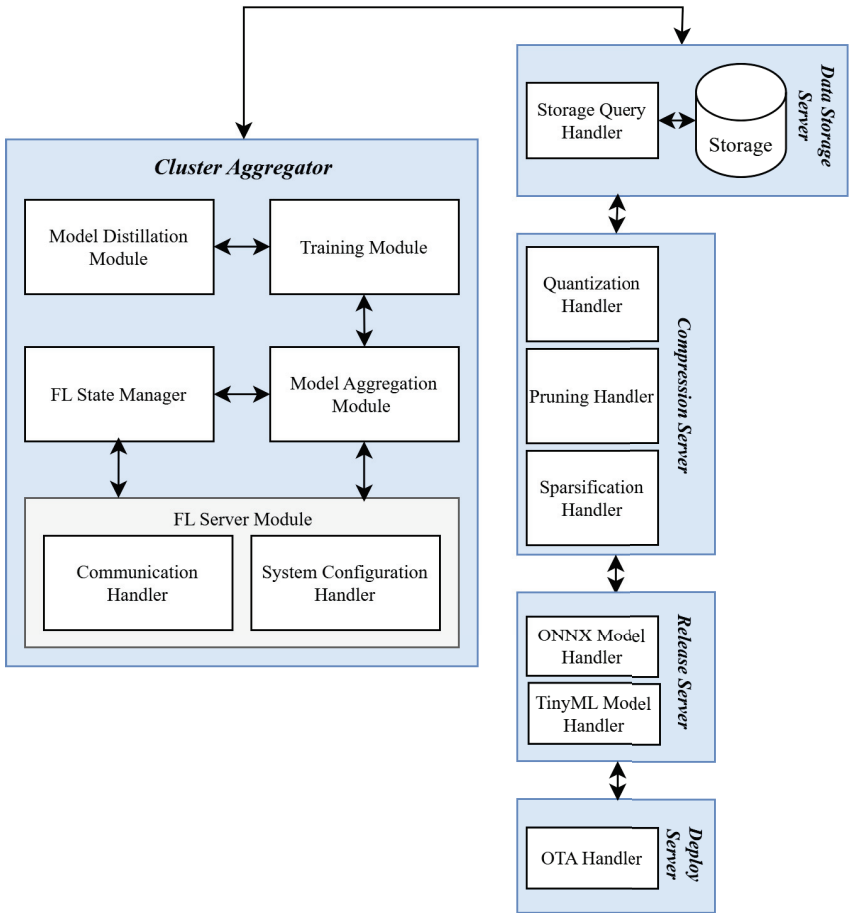

Figure 3.6 Software components deployed in the Cloud.

Together, these compression techniques enable the deploy- ment of complex models on edge devices, ensuring efficient operation while maintaining the balance between performance and resource utilization.

2) *Release Server*: The server is responsible for preparing ANN-based models in specific formats, such as ONNX, to enable seamless integration and deployment across a wide range of platforms and devices. The ONNX format, in particular, is highly valued for its interoperability between different machine learning frameworks, allowing models to be trained in one framework and deployed in another with minimal

conversion effort. This flexibility is crucial in environments where multiple frameworks are in use, ensuring that models can be efficiently transferred and utilized without compatibility issues. Additionally, the Release Server plays a critical role in the context of TinyML, where models must be optimized for execution on ultra-low-power devices such as microcontrollers and sensors. In this scenario, the server supports the conversion of models into highly compact formats suitable for deployment on resource-constrained edge devices. By integrating model compression techniques and optimizing for reduced memory and power consumption, the Release Server ensures that even complex ANN-based models can run efficiently in embedded systems.

3) *Deploy Server*: AI models will be deployed leveraging the Over-the-air (OTA) protocol that allows remote updates on microcontrollers without requiring physical access or direct connections. This approach is particularly beneficial for IoT and embedded systems where devices are often distributed in locations that are difficult to reach.
   The OTA deployment process begins with a central server preparing the new firmware or software update. The micro- controller periodically checks for updates via a secure wireless communication channel, such as Wi-Fi or cellular networks. When an update is available, the microcontroller downloads the update package and performs integrity checks. If the update passes the verification process, it is stored in a dedicated memory partition on the device. Finally, the microcontroller reboots and switches to the new firmware version, ensuring minimal downtime and continuous operation.
   Security plays a critical role in the OTA update process. To prevent unauthorized or malicious updates, encryption methods, authentication protocols, and secure boot mechanisms are often employed to ensure the integrity and authenticity of the update process.
4) *Data Storage Server*: Storage space provides crucial functionality. As its name implies, this server is responsible for storing large datasets required for the training process. It ensures that data is readily accessible and managed efficiently, supporting the extensive data requirements of modern machine learning algorithms. The Data Storage Server also handles data preprocessing and augmentation tasks, preparing the data in a suitable format for training.

These integrated components work in tandem to create a robust and efficient pipeline for deploying ANN-based models on resource-constrained

devices embedded in EVs. By addressing the challenges of model size, complexity, and data management, the system ensures that high-performance models can operate effectively even in environments with limited computational resources.

3.4.4 Distributed Agent

The FL framework is intrinsically a distributed framework. In the proposed architecture (figure 3.7), a Distributed Agent is hosted by each peripheral client, which resides on an edge device within an EV.

1) *FL Client Module*: Within the agent, this module is responsible for establishing communication with the central cluster, receiving the global model's weights, and sending back the updated weights after performing local training on its local dataset. This module plays a crucial role in the federated learning process, ensuring that the client's contributions are incorporated into the global model. It is important to emphasize the complexity of the task managed by the Communication Handler. This component must work in coordination with the FL Participation Handler to address the asynchronicity of communication. Due to the intermittent nature of connectivity between each EV and the central aggregator, as well as among the EVs themselves, there is no guarantee of continuous communication. This sporadic connectivity necessitates robust mechanisms to ensure that updates are transmitted accurately and efficiently whenever a connection becomes available, thereby maintaining the integrity and effectiveness of the federated learning process.
2) *Inference Module*: The local ANN-based model will be utilized in inference tasks to implement the monitoring process. This module takes the trained model and applies it to real-time data gathered from the EV, enabling functions such as predictive maintenance, performance optimization, and anomaly detection. By leveraging the local model, the EV can make intelligent decisions without relying on constant cloud connectivity, thus enhancing the system's reliability and responsiveness. Additionally, the architecture ensures data privacy and security, as the FL approach allows data to remain on the edge device. Only the model updates, which are less sensitive than raw data, are shared with the central cluster. This decentralized approach not only enhances privacy but also reduces the bandwidth required for data transmission, which is critical in mobile and resource-constrained environments like EVs.

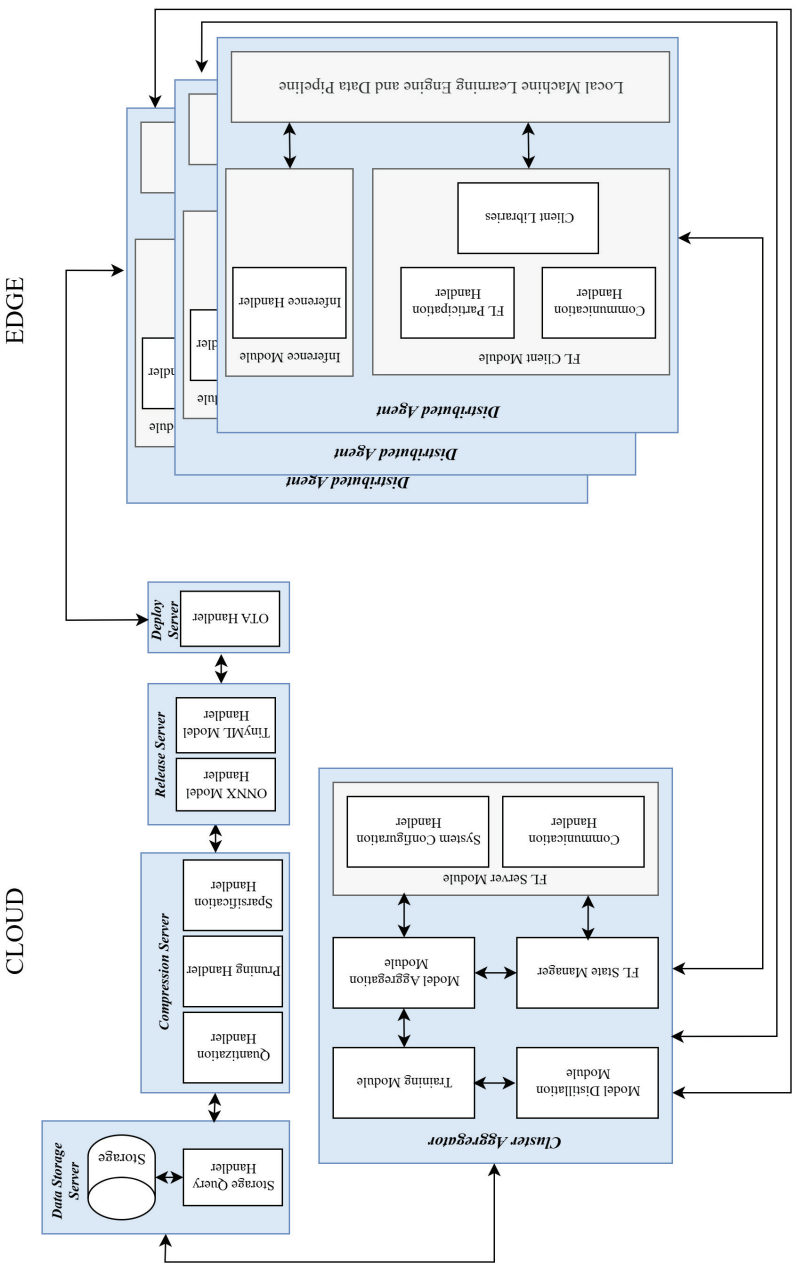

Figure 3.7 The final architecture includes a Cluster Aggregator, deployed in the Cloud, and Distributed Agents, deployed on resource-constrained edge devices.

Overall, the proposed architecture provides a scalable and efficient solution for deploying FL in EVs. By addressing communication challenges, ensuring data privacy, and enabling local inference, the system enhances the capabilities of EVs to perform complex tasks autonomously and effectively.

3.5 Challenges

The implementation of distributed artificial intelligence architecture presents several key challenges:

- *Communication among devices with limited Internet connectivity and energy capacity*: The intermittent availability of network connections, coupled with restricted energy resources and with the mobility of the clients, poses significant difficulties for effective data transmission and distributed computing and FL state maintenance.
- *Generalization of neural network architectures, AI frameworks*: A critical challenge lies in ensuring that AI solutions can generalize across diverse neural network models, software frameworks, and hardware platforms, enabling broad applicability and scalability.
- *Deployment on several architectures*: Deploying AI models on microcontrollers, each with different architectures and processing capabilities, adds a layer of complex- ity. Ensuring compatibility and performance optimization across these heterogeneous systems requires careful consideration of both software adaptation and hardware constraints.
- *Trade-offs in training and compression methodologies*: Striking the right balance between various training approaches and data compression techniques is essential to optimize performance while managing the limitations of resource-constrained devices

3.6 Conclusion

In this work, we designed and proposed an architecture aimed at implementing an Edge Intelligence scenario. Thanks to its modularity, each EI rating layer, from 1 to 5, can be realized.

Among different technologies, Federated Learning has been identified as a distributed training strategy to prevent data sharing and preserve privacy. Since edge devices typically have low-performance hardware, we also included modules for AI model compression and simplification.

Although the architecture has been designed for a generic EI scenario, we identified the monitoring system of Electric Vehicles as a potential use case

where the proposed architecture could be applied after development. Furthermore, a list of interesting challenges and open points has been identified and is presented in this paper.

Acknowledgements

This work is partially supported by the Horizon Europe "Open-source deep learning platform dedicated to Embedded hardware and Europe" project (Grant Agreement, project 101112268 - NEUROKIT2E).

References

[1] S. Ahmad and A. Aral, "*FedCD: Personalized Federated Learning via Collaborative Distillation*," 2022 IEEE/ACM 15th International Conference on Utility and Cloud Computing (UCC), Vancouver, WA, USA, 2022, pp. 189-194, https://doi.org/10.1109/UCC56403.2022.00036.

[2] Ilai Bistritz, Ariana J. Mann, and Nicholas Bambos. 2020. *Distributed distillation for on-device learning*. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20). Curran Associates Inc., Red Hook, NY, USA, Article 1894, 22593–22604.

[3] Y. Cai et al., "*ZeroQ: A Novel Zero Shot Quantization Framework*," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 13166-13175, doi: 10.1109/CVPR42600.2020.01318.

[4] Tianyi Chen, Georgios B. Giannakis, Tao Sun, and Wotao Yin. 2018. *LAG: lazily aggregated gradient for communication-efficient distributed learning*. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18). Curran Associates Inc., Red Hook, NY, USA, 5055–5065.

[5] Z. Chen, P. Tian, W. Liao, X. Chen, G. Xu and W. Yu, "*Resource-Aware Knowledge Distillation for Federated Learning*," in IEEE Transactions on Emerging Topics in Computing, vol. 11, no. 3, pp. 706-719, 1 July-Sept. 2023, doi: 10.1109/TETC.2023.3252600.

[6] Jun Fang, Ali Shafiee, Hamzah Abdel-Aziz, David Thorsley, Georgios Georgiadis, and Joseph H. Hassoun. *Post-training piecewise linear quantization for deep neural networks*. page 69–86, Berlin, Heidelberg, 2020. Springer-Verlag.

[7] Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. \. Int. J. Comput. Vision 129, 6 (Jun 2021), 1789–1819. https://doi.org/10.1007/s11263-021-01453-z.

[8] Stephen Joset' Hanson and Lorien Y. Pratt. *Comparing biases for minimal network construction with back-propagation*. NIPS'88, page 177–185, Cambridge, MA, USA, 1988. MIT Press.

[9] Chaoyang He, Murali Annavaram, and Salman Avestimehr. 2020. *Group knowledge transfer: federated learning of large CNNs at the edge*. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20). Curran Associates Inc., Red Hook, NY, USA, Article 1180, 14068–14080.

[10] Hinton, G.E., Vinyals, O., & Dean, J. (2015). *Distilling the Knowledge in a Neural Network. ArXiv, abs/1503.02531*.

[11] Myeonginn Kang and Seokho Kang. *Data-free knowledge distillation in neural networks for regression*. Expert Systems with Applications, 175:114813, 2021.

[12] Xu Lan, Xiatian Zhu, and Shaogang Gong. *Knowledge distillation by on-the-fly native ensemble*. NIPS'18, page 7528–7538, Red Hook, NY, USA, 6 2018. Curran Associates Inc.

[13] Seung Hyun Lee, Dae Ha Kim, and Byung Cheol Song. 2018. *Self-supervised Knowledge Distillation Using Singular Value Decomposition*. In Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part VI. Springer-Verlag, Berlin, Heidelberg, 339–354. https://doi.org/10.1007/978-3-030-01231-1_21.

[14] Yi Li, Kailong Liu, Aoife M. Foley, Alana Zuĺlke, Maitane Berecibar, Elise Nanini-Maury, Joeri Van Mierlo, and Harry E. Hoster. *Data-driven health estimation and lifetime prediction of lithium-ion batteries: A review*. Renewable and Sustainable Energy Reviews, 113:109254, 2019.

[15] Li, Z.; Li, H.; Meng, L. *Model Compression for Deep Neural Networks: A Survey*. Computers 2023, 12, 60. https://doi.org/10.3390/computers12030060.

[16] Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. 2020. *Ensemble distillation for robust model fusion in federated learning*. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS '20). Curran Associates Inc., Red Hook, NY, USA, Article 198, 2351–2363.

[17] McMahan, H.B., Moore, E., Ramage, D., Hampson, S., & Arcas, B.A. (2016). *Communication-Efficient Learning of Deep Networks from Decentralized Data*. International Conference on Artificial Intelligence and Statistics.

[18] Alessio Mora, Irene Tenison, Paolo Bellavista, and Irina Rish. *Knowledge distillation in federated learning: A practical guide*. In Kate Larson, editor, Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24, pages 8188–8196. International Joint Conferences on Artificial Intelligence Organization, 8 2024. Survey Track.

[19] Polino, A., Pascanu, R., & Alistarh, D. (2018). Model compression via distillation and quantization. *ArXiv, abs/1802.05668*.

[20] Sashank Reddi et al. *Adaptive Federated Optimization*. arXiv e-prints, page arXiv:2003.00295, February 2020.

[21] Guocong Song and Wei Chai. 2018. *Collaborative learning for deep neural networks*. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18). Curran Associates Inc., Red Hook, NY, USA, 1837–1846.

[22] Jun Sun, Tianyi Chen, Georgios B. Giannakis, and Zaiyue Yang. 2019. Communication-efficient distributed learning via lazily aggregated quantized gradients. Proceedings of the 33rd International Conference on Neural Information Processing Systems. Curran Associates Inc., Red Hook, NY, USA, Article 303, 3370–3380.

[23] T. Sun, D. Li and B. Wang, "*Decentralized Federated Averaging,*" in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 4, pp. 4289-4301, 1 April 2023, https://doi.org/10.1109/TPAMI.2022.3196503.

[24] Wu, Guile & Gong, Shaogang. (2020). *Peer Collaborative Learning for Online Knowledge Distillation*. 10.48550/arXiv.2006.04147.

[25] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. *A gift from knowledge distillation: Fast optimization, network minimization and transfer learning*. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7130–7138, 2017.

[26] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu. *Deep mutual learning*. In 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4320–4328, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[27] Tianxun Zhou, Keng-Hwee Chiam. *Synthetic data generation method for data-free knowledge distillation in regression neural networks*.

Expert Systems with Applications, Volume 227, 2023, 120327, ISSN 0957-4174, https://doi.org/10.1016/j.eswa.2023.120327.

[28] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo and J. Zhang, "*Edge Intelligence: Paving the Last Mile of Artificial Intelligence with Edge Computing*," in Proceedings of the IEEE, vol. 107, no. 8, pp. 1738-1762, Aug. 2019, https://doi.org/10.1109/JPROC.2019.2918951.