Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 217 (2023) 436–445

Procedia Computer Science

www.elsevier.com/locate/procedia

4th International Conference on Industry 4.0 and Smart Manufacturing

Compression scenarios for Federated Learning in Smart Manufacturing

Seif Allah EL Mesloul Nasri[a*], Ihsan Ullah[b*], and Michael G Madden[b]

[a]*Departement of Electronic, Badji Mokhtar University, Annaba 23000, Algeria*
[b]*School of Computer Science, University of Galway, Galway H91 TK33, Ireland*

Abstract

Recent advances in Industrial Internet of Things (IIoT) and communication technologies have provided new concepts of smart manufacturing and paved the way for the new era of Industry 4.0. The huge amount of generated data from machinery, connected systems, and edge devices can be exploited to extract actionable insights that can enhance the decision-making process in smart manufacturing, targeting to improve the return of investment. Conventionally, smart decisions are obtained from centralised data using predefined machine/deep learning models. To achieve better accuracy, these models require training based on diverse, unbiased, and comprehensive datasets, which may not be feasible in distributed and remote facilities due to the high scalability of smart manufacturing and data privacy concerns. Federated learning stands as a primary solution to these issues by bringing collaborative smart manufacturing without requiring centralised data training rather than model sharing. However, these models with many parameters result in big size, which put pressure on the communication channels due to frequent transfers from server to clients and vice versa and may face communication delays and model corruption. In addition, bigger models create issues for edge devices with limited memory and processing capacity which needs consideration. Therefore, effective compression methods are mandatory to ensure Communication-Efficient Federated Learning. In this paper, first, we discuss some possible application opportunities of FL in smart manufacturing, especially for the automotive industry but that can be extended for others. Secondly, we highlight FL communication challenge in smart manufacturing for which we present the compression concept and objectives in FL. It is followed by discussing the state-of-the-art compression techniques in FL. Thirdly, a special focus is given to NNR and DeepCABAC for their demonstrated high compression gains with a wide variety of deep models. Finally, we gave some directions for compressing the FL models in the future.

Keywords: Inudstry 4.0; Smart manufacurring; Federated Learning; Compression; NNR; DeepCABAC.

* The first two authors have equal contribution. Email address: ihsan.ullah@nuigalway.ie, seifallah.nasri@gmail.com

10.1016/j.procs.2022.12.239

1. Introduction

Industry 4.0 aims to boost manufacturing with more interlinked, comprehensive, and holistic control and decisions over an integrated ecosystem [1]. The industrial internet of things (IIoT) is one of the paradigms that is associated with the emergence of Industry 4.0. The goal of IIoT is to create continuous connectivity between machines, sensors, edges, central devices, and users, operating in industrial contexts [2]. This expanded connectivity results in a huge amount of data growing exponentially [3]. Machine Learning (ML) is a paradigm in which algorithms are used to construct models from data, and the models are then used for predictions and decision-making. Deep Learning (DL) is a sub-field of ML in which the models are based on large neural networks (NN), which are particularly well suited to working with very large amounts of data. DL models train and learn from the data that is generated by sensors connected to the distributed edge devices. As a result, these models try to solve data-driven problems and take smart action automatically [4]. The performance and accuracy of the DL model outputs are affected by the amount and diversity of data used in training them. Traditionally, there are two options for training data in an industrial context [5]. The first one is to use cloud-centric architectures, where data is sent from its site to the cloud to build a centralized learning model. However, centralized data storage and processes increase leakage risks of sensitive manufacturing data. Without rigorous data privacy practices, companies are reluctant to share their raw data with industrial competitors. In the same context, General Data Protection Regulation (GDPR), California Consumer Privacy Act (CCPA), and other international data policies are presenting more challenges to cloud-centric data storage and processing [6]. Personal data about employees, suppliers, customers, and trading partners are now protected by these regulations. Smart factories are not an exception, they need to comply with the rules. The second option termed distributed learning [7] or on-site learning, is applied to provide definite data privacy and protection by preserving the generated data on its local devices. Instead of sending data to a central server each time, a model is distributed by the server to edge devices. This distributed model then is trained and deployed by the device using limited local data. This training is in the form of finetuning step with personalised data local to the client. However, this personalized deployment after fine tuning does not benefit from peer experience and data to optimize the device performance and increase accuracy rates. To address these issues, Federated Learning (FL) was introduced in 2016 by Google [8]. FL enables the training of ML and DL models without the need to send local data to a central server. FL supports a collaborative training approach, where many individual edge devices train the same model using their data while benefiting from their peers' data having access to it. The local devices (clients) in FL download a model from a trusted federated server. Then, training upon that downloaded model proceeds on each device. Once the model has been finetuned, the resulting updated model is transmitted to the federated server which will ensure the aggregation of all the fine-tuned models that are received from all the client edge devices. The received models are aggregated, and the resulting model is broadcasted to all clients (edge devices) to start the next round of fine-tuning. This process continues for some time until certain criteria are met. It provides an obvious advantage in terms of protecting and preserving industrial data and gives small companies with limited data sets an opportunity to provide more accurate decisions through a larger pool of data. Hence, each company doesn't need to collect and label a large amount of data rather they can use a pre-trained model and finetune it with their limited data. While FL has advantages, there are some challenges [9] that might slow down FL generalization over a large scale of smart manufacturing applications. Most IIoT devices are characterized by their limited computation and storage capabilities. This makes deploying DL models on edges, which are known for the voracious appetite for computational power, a strict challenge. Communication in smart manufacturing uses mostly wireless networks which offer limited bandwidth to IIoT devices compared to wired networks. This fact has created a major challenge to deploy FL on edges, called the communications bottleneck. Even though FL eliminates the process of sharing extensive amounts of data from edge devices to a central server, the overall communication cost of sharing model updates from thousands of devices can be expensive and create a bottleneck during bandwidth allocations while allocating channels to participating clients in a FL [10]. To reduce the communication overhead, various solutions have been proposed, that fall into two main approaches. The first approach is based on reducing the iteration numbers between edge devices and servers using periodic communication schemes [11]. The second approach aims to reduce the size of the shared model in each iteration by utilizing compression methods [12, 13]. The organization of the paper is as follows. Firstly, in Section 2, we briefly discuss challenges and opportunities of deploying FL in Smart manufacturing of automotive industry, with a focus on the communication challenge. Section 3 presents some relevant research that addresses the communication overhead using compression

techniques. We also discuss the state-of-the-art neural network compression standard NNR which has demonstrated high performance in DL models compression as well as its feasibility for FL in Section 4. Section 5 provides our perspective on the future and how we aim to address the compression challenges in FL in a holistic approach that takes into consideration the applied learning models, the type of data and the industrial context. Finally, in Section 5, we will conclude our work.

2. Federated Learning in Smart Manufacturing (Automotive Industry)

Although FL is still in its infancy, reports on usage of FL [14] in various applications suggests that there are huge spaces for it to accelerate the development of many fields including governance, education, telecommunication, healthcare, finance, smart manufacturing, etc. Taking the example of smart manufacturing, FL could be applied to improve the complete journey of a product from initial design, development, service, and disposal. In the smart automotive industry, hundreds (or sometimes thousands) of workstations, machines, and sensors are connected and require the deployment of ML models to effectively perform the assigned tasks such as robotic arms, fault detection, or object recognition [15]. FL can bring a lot of benefits to the entire product life cycle in the context of the automotive industry for large companies as well as small enterprises by giving them access to full intelligence. FL can be effectively incorporated into the following processes of the smart automotive industry:

2.1. Product Design

In automotive industry, generative design is used to iteratively generate several designs within the constraints provided [16]. Recently, Reinforcement Learning [17] and DL [18] are also used for various purposes to generate these designs. Based on that a FL approach can be applied to those generative design techniques that uses DL in their system to improve the generative designs of automotive components. FL enables the reduction of the design cycle across companies by making full use of the distributed data protected by privacy rules. The FL-enabled generative design will sustain engineers to produce lightweight, environment-friendly with less carbon emission, stronger, and more sustainable vehicles. Hence, introducing FL in generative design could accelerate automotive parts production while protecting data and maintaining safety standards.

2.2. Defect Detection

Defect detection in the automotive industry is used for early error detection and anomaly diagnosis. It helps in enhancing production line efficiency and improving automotive parts quality. Automotive part defect detection has evolved from manual methods to traditional classification methods to AI-enabled methods using computer vision and DL models [19]. Lack of data is a major challenge to deploy DL models in this context. FL-enabled defect detection enables automotive companies with similar production activities to be part of collaborative platforms and exchange intelligence through global models without concern about their data privacy (see Figure 1). As an example, many big machinery providers e.g. robotic arms for welding providers do collect data about usage of the machines from smart manufacturers to improve the performance of their robotic arms and related IoT nodes monitoring the usage and environment. Rather than collecting data from smart manufacturers, they can send a model to train it on the consumer premises, resulting in an enhanced system trained over data from various localities. Hence making it capable of detecting defects that were never raised on premises of consumer one but were reported and trained over data on premises of consumer two. For example, changes in humidity and temperature in a manufacturing plant (consumer one) might affect the manufacturing process resulting in defects in the products that it never faced before but as the temperature and humidity raise the FL-based defect detection system will raise an alarm and stop the process because the model recognises the situation from its experience that it learned in data from consumer two. Eventually, defect/fault detection in a smart manufacturer will play a key role that can avoid the high cost and reputational damages.

2.3. Enhancing Vision Systems

Autonomous vehicles use computer vision systems based on deep learning models for various applications e.g. object detection and avoidance [20], and lane detection [21]. These models are also updated regularly for enhancing their performance for which data from the real world (vehicles on the road) is collected. However, with FL an unsupervised or self-supervised model can be trained on the edge device (in this case an autonomous vehicle with

processing set up) rather than collecting data or more specifically labelled data from the owner. A small model can be easy to transfer and train on the vehicle at a specific time. In addition, the privacy-preserving nature of FL will avoid any unwanted situation e.g. information leakage of route and speed, yet give enough information to predict any critical issues by a predictive maintenance system.

2.4. Future Mobility Testbeds

Virtual and augmented reality are positively impacting many industries including the automotive industry. As an example, VR/AR can simulate environments and scenarios where autonomous vehicles will be tested for detecting objects in the vehicle's vicinity [22]. Smart manufacturers, as well as future mobility testbed providers (testing area where the vehicles of the future are tested for all scenarios before allowing them to be used in public e.g. FMC Ireland who are working in stimulating research and innovation activities in the area of drone delivery, Autonomous Connected Electric Shared Vehicles (ACES), including Connected and Autonomous Vehicles (CAV) in Ireland) can make use of FL to train global shared models for improving performance in normal as well as challenging scenarios before certifying them suitable for certain roads.

2.5. Supply Chain Management

Supply chain management is a complex operation for the automotive industry. Even social and political factors influence the total business outcomes. By collecting and analysing data, AI-enabled supply chain optimisation helps the automotive industry to minimize risks and become more agile by with identified issues and providing a more accurate schedule to clients [23]. Encouraging automotive suppliers to participate in a shared FL platform to create optimized global models, will help all participants to achieve improved supply chain management.

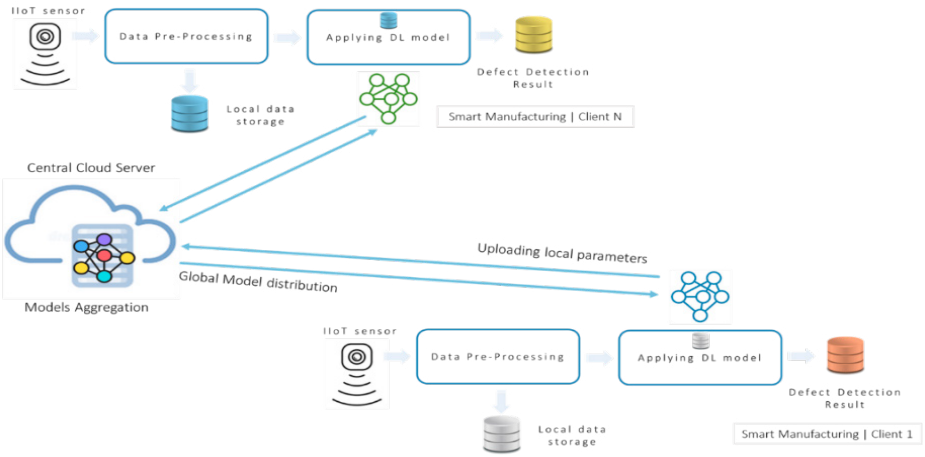

Figure 1 A simplified FL concept for defect detection in smart manufacturing

To deploy FL in all the cited applications, a similar protocol to [24] is followed which is described below:

1. Initialization: The central server trains and generates an initial global model based on the limited available data.
2. Clients' selection: The clients are selected by the central cloud server based on a common targeted task. Clients are edge-connected nodes that can be sensors, operating machines, or automation systems (e.g. industrial control systems). These selected nodes can be from the same company or/and from different companies engaged in one FL network. Here an operating machine can be a computer or robot designed to make an industrial task. Whereas, a sensor or an automation system, both can have calculation units and internal memories to do DL model training and deployment. The capacity of each node type makes the difference.
3. Distribution: The cloud central server distributes the initial model to be trained on the selected edge clients using the local raw data.
4. Uploading local training parameters: After training on local data, all the selected clients upload the resulted training parameters to the cloud server for aggregation tasks.
5. Aggregation: a new version of the global model will be developed in the central server based on some criteria

e.g. averaging the uploaded models. This aggregated model provides more accurate results compared to the initial model trained on a single data source [8].

6. Testing and tuning: The aggregated global model is distributed to clients who test them on their data. The model is fine-tuned if requested based on the test results. The global model is then updated and distributed to all clients for another iteration.
7. Iteration: steps from 3 to 6 are repeated until the global model achieves the targeted performance and accuracy.

The above process shows that there is a huge downlink and uplink during the communication rounds between the central cloud server and the multiple distributed clients. Literature [25, 26] also shows that the wireless network to the IoT nodes is not strong nor do the IoT nodes have too much storage and processing capacity. Specific communication mechanisms including compression techniques must be involved to handle this huge amount of information traffic that characterises FL deployment. This challenge will be discussed in the following section.

3. Compression Techniques in Federated Learning

The total number of IIoT edge devices of multiple smart factories taking part in the FL process could result in hundreds or thousands of nodes in a large-scale industrial context. Most IIoT devices have low computation capacity and use wireless networks for communication which are relatively slow compared to wired networks. Furthermore, in FL training there is massive communication traffic between the distributed client and the central server issued from the communication rounds that happened before achieving the desired performance of the global model. Furthermore, current deep neural network models can exceed hundreds of gigabytes in certain complex architectures with billions of parameters e.g. [27] which is about 100 GB. The model size not only requires big space for storage as well as loading in runtime memory, but it also requires hundreds of GPUs and about a month to train a single model. Furthermore, it also requires a huge amount of data to train on. Transferring such models from varying communication and processing capabilities increases the chances of model corruption and delays in aggregating and updating models. All these facts could create a communication bottleneck when deploying FL. To reduce the communication delays and cost in FL, several methods have been suggested based on two mainstream directions. The first concept consists of reducing the communication cost by lowering the frequency of model updating iterations using periodic communication strategies. The second concept is directed toward the compression of the updated models between the clients and the server for each communication round. The proposed compression techniques must reduce the model size while maintaining the accuracy level of the targeted task. Compression in FL can be classified in three categories based on the three objectives [28], which are:

(1) Gradient compression: this consists of reducing the model update size transmitted from clients to the central server for FL model aggregation based on averaging.
(2) Global model compression: the objective of this compression type is to reduce the broadcasted global model size, from server to client.
(3) Reducing local computation cost: employing techniques for the overall training process in a way that reduces the computation cost for the local training. This is not a compression approach; however, it is an important requirement to consider when we apply a compression technique. One point to note is that using compression algorithms (e.g. DEFLATE) to reduce bandwidth, one needs to bear in mind that there are increased computation costs for compression and decompression. Therefore, in a device that has limitations on both bandwidth and CPU, it is not guaranteed that compression will help. Therefore, regarding the low computation capacity of IIoT devices, applied compression techniques should not be too heavy in terms of calculation. Hence, reducing the overall training computation within the edge devices will facilitate FL deployment.

These objectives go side by side, and they are complementary to each other. In practice, the uplink communication from clients to a server in FL is generally slower than the downlink communication due to the resource-limited capacity edge devices i.e., limited capacity of IIoT [29]. Therefore, Objective 1 could have the most significant effect on the overall communication cost as there is more to be gained when applying more aggressive compression structures to the averaging process across the distributed clients. A significant amount of existing research has been applied to address Objective 1 [30, 31, 32, 33, 34], while Objective 2 has been less widely studied [35]. Objectives 1, 2, and 3 could be addressed jointly to improve the overall compression gain.

Figure 2 shows a simplified structure of how compression could be applied in the FL context. The server broadcasts an initial model to the clients participating in the FL network. Once received by the client, a training cycle starts on a private dataset. The resulting updated gradients are then processed and prepared for transmission. Compression is a crucial step to reduce the cost of uplink communication. Conventionally, there are two types of compression, lossy and lossless techniques. Both methods could be combined and applied to the updated weights to achieve enhanced compression rates. Since weights can be considered as continuous-valued variables, quantization is applied as a lossy compression and entropy encoding is applied then as a lossless compression. Applied compression techniques depend on different factors such as the client's energy and computation resources, In addition to the available bandwidth for transmission. In practice, devices involved FL network may have different hardware and software compositions ranging from IIoT capacity-limited devices to powerful machines. FL heterogeneity nature is an actual challenge that involves considering multiple scenarios when selecting the compression techniques to be deployed on edge clients. Before aggregating the received updates and forming the global model, decoding techniques should be applied, to retrieve the updated gradients, aligned with the deployed encoding methods at the client's level. The learned global model may be subject to compression before starting a second FL round to reduce its size without decreasing its accuracy rate. A threshold can be applied each time based on the resulted global model size, downlink capacity, and the overall clients' capabilities to decide if the global model requires a compression before transmission or not. It is

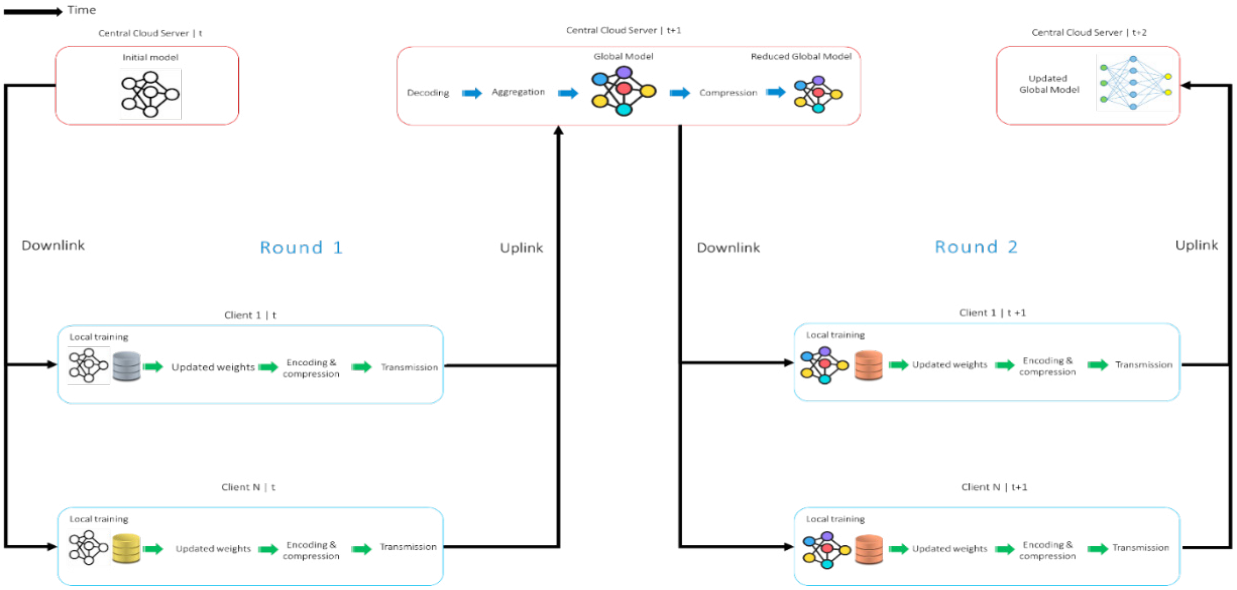

Figure 2 Compression directions in FL communication

also suggested to consider the compression targeted rate for more communication efficiency. Until recent times, there is not a ready-to-use compression method to apply directly in all FL networks.

Multiple and dynamic compression scenarios can be used in one FL network to deal with the communication bottleneck. In the following subsections, we provide a short review of the relevant compression types proposed by the FL research community based on Sparsification and Quantization.

3.1. Sparsification

Sparsification compression approaches are used to compress models in FL during transmission to the central server. Gradient sparsification is an approach to compress the model weight vector (gradient) to improve communication performance. Aji et. al. [36] proposed a gradient sparsification method that only transmits the large gradients after truncating the smallest ones. In this method, gradient updates are positively skewed, after that 99% smallest updates are mapped to zero then sparse matrices are exchanged. The gradient communication speed was increased by up to 22% on the used neural machine translation (NMT) over 4 GPUs (Titan Xs), without damaging the final accuracy. The sparse ternary compression (STC) method is proposed in [13] to provide compression for FL in both communication directions (uploading/downloading). Tests were conducted over different FL models with various datasets. An analysis provided in the paper revealed that the convergence rate can be reduced by factors like training

on Non-IID small data portions and when a subgroup of clients participate in the communication iterations. The STC provided faster convergence compared to the reported algorithms such as FedAVG for both mentioned factors. The proposed STC method showed effectiveness with bandwidth with constrained conditions. The same research group introduced another compression method called Sparse Binary Compression (SBC) to significantly reduce the communication cost in distributed learning environments [37]. SBC combines methods of communication delay and gradient sparsification with a new binarization technique. SBC employs communication delay methods [9] after each local iteration where gradients are not transmitted. It is also found in [32] that communication delay and gradient sparsity are considered two separate types of sparsity techniques in Stochastic Gradient Descent (SGD). Although they noticed a slight decrease in accuracy from these methods, higher compression gains are provided compared to regular Synchronous Distributed Stochastic Gradient Descent (DSGD) and Federated Averaging.

3.2. *Quantization*

Quantization is another technique that addresses compression in the FL context. It consists of reducing the entropy of the model parameters to a reduced set of values without negatively affecting the model accuracy. A dense quantization method was proposed by [38] to quantize the weights of the DL model while minimising the degradation of accuracy. This method combines quantization, pruning, and Huffman coding, and it has achieved a significant compression rate of 49×, without accuracy loss. TernGrad was introduced in [39] to stochastically quantize gradients to ternary. A compression rate of 16× is achieved for a complex DL model in a decentralised learning setting with a slight accuracy decrease of 2%. FedPAQ is proposed in [40] as a framework to lower the overall communication cost. Quantization techniques presented in [41] are applied in this research to quantize each local model before getting uploaded. Consequently, the communication overhead is reduced. Liu et al. [42] proposed a Hierarchical Quantized FL method that leverages client edge cloud hierarchy and quantizes model weights. Their proposed approach enhances the communication efficiency in a FL environment by performing model parameters quantization and partial edge aggregation.

4. Compression Standards for Federated Learning

FL requires iterated communication/transfer of NN parameters (modal) between multiple devices which might create a communication overhead and limits the FL benefits if applied on devices with limited resources. As outlined before, several research methods proposed specific compression algorithms to reduce the NN representation size. Based on subsection 3.1 and 3.2, sparsification and quantization compression approaches showed optimized results in FL, but only for certain targeted applications with specialized conditions. This fact raised the demand for NN compression methods that can be applied to various neural network models in different scenarios and contexts. The Moving Picture Expert Group MPEG (ISO) has been working since 2017 on a general, easy-to-use standard for neural network compression. The successive calls and working activities of MPEG led to the first-ever standard for Neural Network Compression and Representation (NNR) as part 17 of ISO/IEC 15938 [43]. The first version of the NNR standard includes deep context-adaptive binary arithmetic coding (DeepCABAC) as the core encoding and decoding engine, in addition to quantization and pre-processing techniques like sparsification, pruning, low-rank decomposition, unification, local scaling, and batch norm folding (see Figure 3). These compacted methods in the NNR standard provided a compression efficiency of up to 97% for transparent coding cases without decreasing the neural network model accuracy [44]. The work on Neural Network Coding (NNC*)* was extended in 2021, to consider coding of incremental neural network updates in FL applications, where parameter-wise updates of neural networks are continuously transmitted. Accordingly, the second edition of NNC is developed (ISO/IEC 15938-17 ed2). Adding tools such as specific temporal adaption in DeepCABAC and structured sparsification improved further the compression efficiency of NNC ed2 to compression ratios for incremental data of <1% per epoch as reported in [45]. To evaluate NNR compression

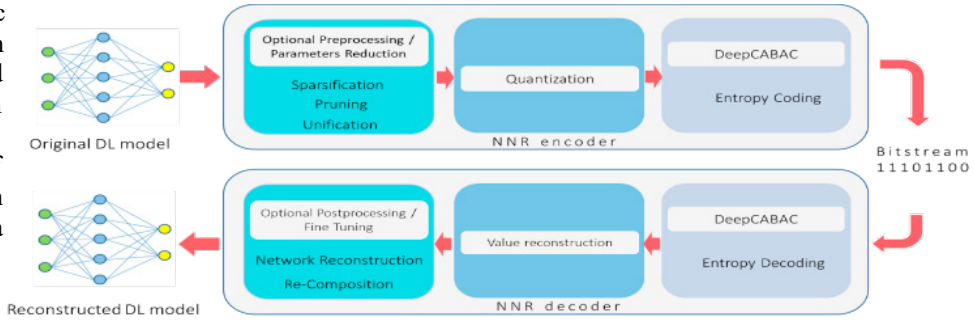

Figure 3 NNR Coding and Decoding Structure [40]

performance in [44], different neural network models were used according to the evaluation framework document [46]. The utilised model set is composed of three image classification models (VGG16, ResNet50, and MobileNetV2), a model for audio classification (DCase), and an image autoencoder (UC12B). The standard reference software Neural Network Compression Test Model (NCTM), version 6.0 was used to carry out the experiments. Overall, obtained results show the high compression efficiency of NNR. For example, in comparison to the original models' sizes, compression ratios of 2.96%, 3%, and 4.12% were achieved without accuracy degradation for VGG16, UC12B, and DCase models, respectively. NNR outperformed other methods considerably, i.e., a compression ratio of 2.98% forVGG16 using NCTM compared to a compression ratio of 7.74% when using BZIP.

4.1. DeepCABAC

DeepCABAC is the core engine of the NNR standard. It is considered a universal compression method dedicated to compressing deep neural network parameters. DeepCABAC is based on a Context-based Adaptive Binary Arithmetic Coder which was firstly designed for H.264 video coding standard [47], then applied to its successor H.265 video coding. CABAC is a lossless entropy coding that has different probability modes for various settings. it has three main coding steps: 1. It starts by converting all non-binary data to binary symbols. 2. Each bit is assigned to a probability model. 3. The Arithmetic coder is employed to optimize the probability estimate. To adapt a coder for NN compression, some requirements given in have to be considered [48], such as:

a) The codec should be universal and appropriate to encode any type of neural network models.
b) Providing high coding efficiency while minimizing the encoding/decoding complexity with a negligible accuracy degradation.
c) Some deep neural networks are composed of millions of parameters that can have redundant representations. An effective codec should reduce this redundancy to gain more storage space and deal with the constrained communication.

DeepCABAC fulfils these requirements and efficiently encodes/decodes the neural network quantized weight parameters. More information about DeepCABAC coding tools is detailed in [48 - 50]. In [49] DeepCABAC was evaluated against a reported baseline compression method (bZip) in a FL context of 10 clients using the CIFAR-10 dataset. DeepCABAC versatility for FL has been demonstrated by using different NN architectures (LeNet, VGG-11, VGG-16). Obtained results, confirmed DeepCABAC outperformance in all tested NNs architectures against bZip by reducing the total communication cost to less than 4% of the original size on all reported models without accuracy degradation of the deployed resulting model.

5. Future Directions

In the previous sections, we discussed compression types and standards. However, an unexplored approach is using a deep pyramidal model that is specifically designed to result in fewer parameters with good performance e.g., pyramidal CNN [51], 3DPyraNet [52]. In [51], a strict pyramidal structure is imposed on existing CNN models that end up in fewer parameters without losing performance. In [52], a 3D pyramidal model was proposed in which a new partially shared locally connected weighting scheme was introduced that resulted in fewer parameters by design compared to other deep models e.g., 3DCNN. Such models can be explored in FL that will result in a smaller model to be transferred from server to the clients and the server without significant delays. On the other hand, existing models can be utilised by both reducing their sizes using knowledge or in transfer learning. The knowledge distillation approach is more suitable to be utilized for testing scenarios i.e., learn the model once (a bigger model) and then extract/train a smaller network (student network) that can be used at test time. Slow training is a problem due to various issues. However, the inhibitory neuron concept can be used to rapidly refine/train a pretrained model [53]. However, these two are not as ideal as the prior models for the FL approach. Therefore, we aim to implement prior unexplored models for FL along with an industrial partner. The industrial partner will provide us with a real dataset and to contribute creating a FL platform across different companies that share the same production activities. The communication performance of the FL platform will be evaluated using the NNC standard with a set of deep learning models of the same classification purpose. We will also focus on deploying the NNR standard and another competitive baseline compression to identify weaknesses and propose optimized solutions. More tests must be conducted on real use cases to facilitate NNR adoption for FL industrial applications. Effective in-parallel compression techniques must consider within resources-limited IIoT devices for dealing with the continuously collected data and the communicated models to ensure an efficient FL concept within the targeted task.

6. Conclusions

In this paper, we first highlighted the concept of industry 4.0 and smart manufacturing and how using IIoT devices is generating a great amount of data that can be exploited to provide a smart decision. We have also presented the need of applying federated learning to boost further smart manufacturing, four examples from the automotive industry were presented for this matter. This paper also highlighted the communication overhead challenge in federated learning and its current existing solutions. A brief review of the possible compression techniques in FL was presented with more focus on the NNR standard and its core coding tool DeepCABAC. This paper suggests various usages of model compression in federated learning for industry 4.0 and smart manufacturing. In future work, we will work on its simulation and testing as well as on highlighting its limitations, pros, and cons. In addition, we aim to work on the research perspectives presented in the previous section for model compression which will reduce communication overhead in a federated learning-based system.

References

[1] Wang, Shiyong, Jiafu Wan, Di Li, and Chunhua Zhang. "Implementing smart factory of industrie 4.0: an outlook." International Journal of Distributed Sensor Networks 12, no. 1 (2016): 3159805.

[2] Sisinni, Emiliano, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. "Industrial internet of things: Challenges, opportunities, and directions." IEEE Transactions on Industrial Informatics 14, no. 11 (2018): 4724-4734.

[3] Yin, Shen, and Okyay Kaynak. "Big data for modern industry: challenges and trends [point of view]." Proceedings of the IEEE 103, no. 2 (2015): 143-146.

[4] Wang, Jinjiang, Yulin Ma, Laibin Zhang, Robert X. Gao, and Dazhong Wu. "Deep learning for smart manufacturing: Methods and applications." Journal of Manufacturing Systems 48 (2018): 144-156.

[5] AbdulRahman, Sawsan, Hanine Tout, Hakima Ould-Slimane, Azzam Mourad, Chamseddine Talhi, and Mohsen Guizani. "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond." IEEE Internet of Things Journal 8, no. 7 (2020): 5476-5497.

[6] Pardau, Stuart L. "The California consumer privacy act: Towards a European-style privacy regime in the United States." J. Tech. L. & Pol'y 23 (2018): 68.

[7] Verbraeken, Joost, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. "A survey on distributed machine learning." ACM Computing Surveys (CSUR) 53, no. 2 (2020): 1-33.

[8] Konečný, Jakub, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. "Federated learning: Strategies for improving communication efficiency." arXiv preprint arXiv:1610.05492 (2016).

[9] Li, Tian, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. "Federated learning: Challenges, methods, and future directions." IEEE Signal Processing Magazine 37, no. 3 (2020): 50-60.

[10] Shahid, Osama, Seyedamin Pouriyeh, Reza M. Parizi, Quan Z. Sheng, Gautam Srivastava, and Liang Zhao. "Communication efficiency in federated learning: Achievements and challenges." arXiv preprint arXiv:2107.10996 (2021).

[11] McMahan, Brendan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data." In Artificial Intelligence and Statistics, pp. 1273-1282. PMLR, 2017.

[12] Alistarh, Dan, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. "QSGD: Communication-efficient SGD via gradient quantization and encoding." Advances in Neural Information Processing Systems 30 (2017).

[13] Sattler, Felix, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. "Robust and communication-efficient federated learning from non-iid data." IEEE Transactions on Neural Networks and Learning Systems 31, no. 9 (2019): 3400-3413.

[14] Developed by the Learning Technology Standards Committee, "IEEE Guide for Architectural Framework and Application of Federated Machine Learning", in IEEE Std 3652.1-2020 (2021): 1-69.

[15] Boopalan, Parimala, Swarna Priya Ramu, Quoc-Viet Pham, Kapal Dev, Praveen Kumar Reddy Maddikunta, Thippa Reddy Gadekallu, and Thien Huynh-The. "Fusion of Federated Learning and Industrial Internet of Things: A survey." Computer Networks (2022): 109048.

[16] Gavacová, Jana, Miroslav Veres, and Matús Grznár. "Computer aided generative design of automotive shaped components." Acta Technica Corviniensis-Bulletin of Engineering 7, no. 2 (2014): 19.

[17] Jang, Seowoo, Soyoung Yoo, and Namwoo Kang. "Generative design by reinforcement learning: enhancing the diversity of topology optimization designs." Computer-Aided Design 146 (2022): 103225.

[18] Yoo, Soyoung, Sunghee Lee, Seongsin Kim, Kwang H Hwang, Jong H Park, & N Kang. "Integrating deep learning into CAD/CAE system: generative design and evaluation of 3D conceptual wheel." Structural and Multidisciplinary Optimization 64, no. 4 (2021): 2725-2747.

[19] Ren, Jing, Rui Ren, Mark Green, and Xishi Huang. "Defect detection from X-ray images using a three-stage deep learning algorithm." In 2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE), pp. 1-4. IEEE, 2019.

[20] Gupta, Abhishek, Alagan Anpalagan, Ling Guan, and Ahmed Shaharyar Khwaja. "Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues." Array 10 (2021): 100057.

[21] Boukerche, Azzedine, and Zhijun Hou. "Object detection using deep learning methods in traffic scenarios." ACM Computing Surveys (CSUR) 54.2 (2021): 1-35.

[22] Sportillo, Daniele, Alexis Paljic, and Luciano Ojeda. "On-road evaluation of autonomous driving training." In 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pp. 182-190. IEEE, 2019.

[23] Wenzel, Hannah, Daniel Smit, and Saskia Sardesai. "A literature review on machine learning in supply chain management." In Artificial Intelligence and Digital Transformation in Supply Chain Management: Innovative Approaches for Supply Chains. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 27, pp. 413-441. Berlin: epubli GmbH, 2019.

[24] Li, Tian, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. "Federated learning: Challenges, methods, and future directions." IEEE Signal Processing Magazine 37, no. 3 (2020): 50-60.
[25] Sikimić, Miljan, Momčilo Amović, Vladimir Vujović, Bojan Suknović, and Dragan Manjak. "An overview of wireless technologies for IoT network." In 2020 19th International Symposium INFOTEH-JAHORINA (INFOTEH), pp. 1-6. IEEE, 2020.
[26] Sisinni, Emiliano, Abusayeed Saifullah, Song Han, Ulf Jennehag, and Mikael Gidlund. "Industrial internet of things: Challenges, opportunities, and directions." IEEE Transactions on Industrial Informatics 14, no. 11 (2018): 4724-4734.
[27] Ramesh, Aditya, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. "Zero-shot text-to-image generation." In International Conference on Machine Learning, pp. 8821-8831. PMLR, 2021
[28] Kairouz, Peter, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz et al. "Advances and open problems in federated learning." Foundations and Trends® in Machine Learning 14, no. 1–2 (2021): 1-210.
[29] Zheng, Sihui, Cong Shen, and Xiang Chen. "Design and analysis of uplink and downlink communications for federated learning." IEEE Journal on Selected Areas in Communications 39, no. 7 (2020): 2150-2167.
[30] Konečný, Jakub, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. "Federated learning: Strategies for improving communication efficiency." arXiv preprint arXiv:1610.05492 (2016).
[31] Suresh, Ananda Theertha, X. Yu Felix, Sanjiv Kumar, and H. Brendan McMahan. "Distributed mean estimation with limited communication." In International Conference on Machine Learning, pp. 3329-3337. PMLR, 2017.
[32] Alistarh, Dan, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. "QSGD: Communication-efficient SGD via gradient quantization and encoding." Advances in Neural Information Processing Systems 30 (2017).
[33] Basu, Debraj, Deepesh Data, Can Karakus, and Suhas Diggavi. "Qsparse-local-SGD: Distributed SGD with quantization, sparsification and local computations." Advances in Neural Information Processing Systems 32 (2019).
[34] Horváth, Samuel, Chen-Yu Ho, Ludovit Horvath, Atal Narayan Sahu, Marco Canini, and Peter Richtárik. "Natural compression for distributed deep learning." arXiv preprint arXiv:1905.10988 (2019).
[35] Chraibi, Sélim, Ahmed Khaled, Dmitry Kovalev, Peter Richtárik, Adil Salim, and Martin Takáč. "Distributed fixed point methods with compressed iterates." arXiv preprint arXiv:1912.09925 (2019).
[36] Aji, Alham Fikri, and Kenneth Heafield. "Sparse communication for distributed gradient descent." arXiv preprint arXiv:1704.05021 (2017).
[37] Sattler, Felix, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. "Sparse binary compression: Towards distributed deep learning with minimal communication." In 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1-8. IEEE, 2019.
[38] Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." arXiv preprint arXiv:1510.00149 (2015).
[39] Wen, Wei, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. "Terngrad: Ternary gradients to reduce communication in distributed deep learning." Advances in neural information processing systems 30 (2017).
[40] Reisizadeh, Amirhossein, A Mokhtari, H Hassani, A Jadbabaie, & R Pedarsani. "Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization." In Int Conference on Artificial Intelligence and Statistics, pp. 2021-2031. PMLR, 2020.
[41] Jiang, Peng, and Gagan Agrawal. "A linear speedup analysis of distributed deep learning with sparse and quantized communication." Advances in Neural Information Processing Systems 31 (2018).
[42] Liu, Lumin, Jun Zhang, Shenghui Song, and Khaled B. Letaief. "Hierarchical quantized federated learning: Convergence analysis and system design." arXiv preprint arXiv:2103.14272 (2021).
[43] Chang, Shih-Fu, Thomas Sikora, and Atul Purl. "Overview of the MPEG-7 standard." IEEE Transactions on Circuits and Systems for Video Technology 11, no. 6 (2001): 688-695.
[44] Kirchhoffer, Heiner, Paul Haase, Wojciech Samek, Karsten Müller, Hamed Rezazadegan-Tavakoli, Francesco Cricri, Emre B. Aksu et al. "Overview of the neural network compression and representation (NNR) standard." IEEE Transactions on Circuits and Systems for Video Technology 32, no. 5 (2021): 3203-3216.
[45] Neural Network Coding, www.hhi.fraunhofer.de, accessed: June 2022
[46] Evaluation Framework for Compressed Representation of Neural Net- works, MPEG document N17929, ISO/IEC JTC 1/SC 29/WG 11, Oct. 2018
[47] Marpe, Detlev, Heiko Schwarz, and Thomas Wiegand. "Context-based adaptive binary arithmetic coding in the H. 264/AVC video compression standard." IEEE Transactions On Circuits and Systems for Video Technology 13, no. 7 (2003): 620-636.
[48] Wiedemann, Simon, Heiner Kirchhoffer, Stefan Matlage et al. "Deepcabac: A universal compression algorithm for deep neural networks." IEEE Journal of Selected Topics in Signal Processing 14, no. 4 (2020): 700-714.
[49] Neumann, David, Felix Sattler, Heiner Kirchhoffer, Simon Wiedemann, Karsten Müller, Heiko Schwarz, Thomas Wiegand, Detlev Marpe, and Wojciech Samek. "Deepcabac: Plug & play compression of neural network weights and weight updates." In 2020 IEEE International Conference on Image Processing (ICIP), pp. 21-25. IEEE, 2020.
[50] Wiedemann, Simon, Heiner Kirchhoffer, Stefan Matlage, Paul Haase, Arturo Marban, Talmaj Marinc, David Neumann et al. "Deepcabac: Context-adaptive binary arithmetic coding for deep neural network compression." arXiv preprint arXiv:1905.08318 (2019).
[51] Ullah, Ihsan, and Alfredo Petrosino. "A strict pyramidal deep neural network for action recognition." In International Conference on Image Analysis and Processing, pp. 236-245. Springer, Cham, 2015.
[52] Ullah, Ihsan, and Alfredo Petrosino. "About pyramid structure in convolutional neural networks." In 2016 International Joint Conference on Neural Networks (IJCNN), pp. 1318-1324. IEEE, 2016.
[53] Ullah, Ihsan, Sean Reilly, and Michael G. Madden. "Enhancing Semantic Segmentation of Aerial Images with Inhibitory Neurons." In 2020 25th International Conference on Pattern Recognition (ICPR), pp. 5451-5458. IEEE, 2021.