Application of Federated Machine Learning in Manufacturing

1st Vinit Hegiste  
Chair of Machine Tools and Control Systems  
TU Kaiserslautern  
Kaiserslautern, Germany  
vinit.hegiste@mv.uni-kl.de

2nd Tatjana Legler  
Chair of Machine Tools and Control Systems  
TU Kaiserslautern  
Kaiserslautern, Germany  
tatjana.legler@mv.uni-kl.de

3rd Martin Ruskowski  
Innovative Factory Systems (IFS)  
German Research Center for Artificial Intelligence (DFKI)  
Kaiserslautern, Germany  
martin.ruskowski@dfki.de

Abstract—A vast amount of data is created every minute, both in the private sector and industry. Whereas it is often easy to get hold of data in the private entertainment sector, in the industrial production environment it is much more difficult due to laws, preservation of intellectual property, and other factors. However, most machine learning methods require a data source that is sufficient in terms of quantity and quality. A suitable way to bring both requirements together is federated learning where learning progress is aggregated, but everyone remains the owner of their data.

Federate learning was first proposed by Google researchers in 2016 and is used for example in the improvement of Google’s keyboard Gboard. In contrast to billions of android users, comparable machinery is only used by few companies. This paper examines which other constraints prevail in production and which federated learning approaches can be considered as a result.

Index Terms—Federated Learning, Image classification, Quality Inspection, Manufacturing

I. INTRODUCTION

Machine Learning has become part of many sectors, including our daily life. Ranging from Netflix recommendations [1] to smart healthcare [2], fraud detection [3], earthquake prediction [4], recycling and waste management [5], and the development of self-driving cars [6]. The possible applications of ML in production and manufacturing are similarly diverse [7], [8].

Besides the production of defects, the prevention and implementation of quality measures are the cost for preventing and removing defects, and costs for quality measures to be considered in total quality management [9]. ML-based approaches can be used to monitor processes or predict the product quality [10], [11]. Through the recording and analysis of the operational data of machinery, predictive maintenance helps minimise unplanned downtime and optimize tool life [10]. New insights can be gained through the use of additional sensors in machines. For example, [12] used microphones to detect chatter on a CNC lathe and thus helps to ensure optimal process parameters. Product quality control with CNN has higher adaptability to changing conditions and therefore reduces rejection rates [13], [14].

The widespread use and high accuracy of ML models can be attributed to the high availability of data, for example, from mobile or IoT devices [15]. ML as-a-service providers, such as IBM Cloud, Amazon Web Services, Google Cloud, and Microsoft Azure [16], compute the models on their high-performance servers. In this centralised architecture all data is sent to the cloud and the client sends a request to the server. Although the achievable accuracy is the highest in this case, data privacy concerns are increasing and stricter regulations such as General Data Protection Regulation (GDPR) came into force [17]. This led to a decentralised, on-device machine learning approach [18].

II. RELATED WORK

A. Federated learning

Google introduced federated learning in 2016 [19]. They developed a new concept called Federated optimisation, with the goal of creating a high-quality centralised global model based on models trained on data from unevenly distributed data across an extremely large number of nodes (edge devices, client’s machines, smart phones, etc). It is a novel technique that enables multiple users to train a global model based on local weights obtained from the local data of the users participating in the training round. By iterating this process, the global model achieves a high accuracy based on the local data from various users/devices used to train this model. The main advantage of this technique is that raw sensitive data or image never ever leaves the user’s device, and only the model weights are sent to the global server. This helps to reduce privacy concerns when training a model on sensitive data such as health care, personal chats, sensor data from manufacturing companies, and so on. Figure 1 depicts a generic diagram of the federated learning architecture, in which the machine learning model is trained on the user’s device, which could be a laptop, edge device, computer, or even a smartphone.
After training, the model weights are sent back to the server, which uses a federated learning algorithm/framework such as FedAvg [19], FedProx [20], FedMA [21], FedVision [22], etc. to generate a global federated model, which is then sent back to all users for the next update. This single iteration of sending a global model to users and receiving weights from all users before applying a federated learning algorithm to obtain the global model is referred to as a single communication round (CR).

Google uses this federated learning technique in their application called Gboard (Google keyboard), where the data of users who use this application is stored locally on their devices [23]. Google sends a global model to these devices, which train it with local data and send the updated weights and gradients back to the Google server. On the server side, an algorithm called FedAvg [19] is used, which computes the average of these weights (from all clients) and updates the model weights with these averaged weights. This revised model is returned to the clients for training on newly generated data. The question of methodology arises, because in this case, the local devices require computing power and memory, as well as frequent communication between the devices and the server, which will require high bandwidth in order to exchange machine learning model parameters. Google finds a solution to this problem by training the model on mobile devices while they are charging, connected to Wi-Fi, and not in use. Only under the aforementioned conditions are model parameters sent and a new update model is downloaded for training on local data [23].

**B. Basic architecture and Algorithms**

McMahan et al. mentioned two algorithms in [19], namely FedSGD and FedAvg. In FedSGD, the clients train the entire local data set over one epoch and then send the updated local weight gradients to the central server. This server then averages all of the weight gradients from all of the clients, applies the weight update, and sends the updated weights to the clients. The important point to note here is that in this entire process, all clients locally take one step of gradient descent, as shown in figure 2.

FedSGD algorithm provides as a base to FedAvg algorithm where more computation is performed on the local side such that each client iterates multiple times before the averaging step i.e. increasing the epochs and introducing mini batches as shown in figure 3. This paper concluded that with FedAvg algorithm the number of communication rounds between a server and all clients can be reduced by properly hyper-tuning the number of local epochs and the batch size for all the clients. The main issue with federated learning is the communication rounds, as these model weights must be transferred from all clients to the server and back again, and in real life, problems such as internet outage, power outage, and so on can impede the execution of a single communication round. As a result, having hundreds of such communication rounds increases the algorithm’s failure in an uncontrolled environment. Thus, increasing the epochs on the local client...
side while decreasing the number of communication rounds is an effective solution. FedAvg serves as the foundation for frameworks/algorithms such as FedProx [20], FedMA [21], FedVision [22], etc. Additionally, this federated learning can be subdivided into a number of categories [24], out which we use Horizontal federated learning [25]. Horizontal federated learning (also called as sample-based federated learning) is used in scenario when data sets share the same features but are different in samples [25]. Our use case has the similar scenario (refer section III-A) where the clients’ data set share similar features but are different in samples.

### III. Federated Learning in Manufacturing for Quality Inspection

The introduction of federated learning is advantageous for businesses, research institutions, and even individuals attempting to collaborate on applications such as quality inspection, anomaly detection, object detection, etc. This technique can be useful for inter-company applications in which two or more companies train a model collaboratively, as well as intra-company applications in which a company with multiple branches or production locations wishes to train a deep learning model for any of the aforementioned applications. Using horizontal federated learning [25], we constructed a USB quality inspection model for our use case. This use case aims to construct a global USB classification model capable of classifying all mechanical forms of USB port errors/faults present in the data set. This use case includes three clients with their own local data, as depicted in the figure 4, which are trained using a VGG-19 model [26]. We have experimented with this entire data set trained using VGG-19 model and a global federated model trained using 3 clients (the total data used is same but each USB sticks’ data set is represented as an individual client), where the starting weights used are pretrained ImageNet weights [27]. In regards to federated machine learning, we also conducted experiments to determine the difference in the total number of Communication Rounds required by the federated learning algorithm when using the fine-tuned VGG model and a custom fully-connected layered VGG model (Transfer learning). Our application’s workflow adheres to FedAvg [19] (figure 3) because it is the foundation for federation learning and easy to implement.

#### A. Data set

Our data set consists of three different types of USB sticks. Figure 4 depicts three different USB stick with four classes each. Each USB stick’s data functions as local data for each client, hence in our federated learning scenario we have three clients. Huawei Openlab Munich’s [28] USB stick acted as client1’s data, SmartFactory-KL’s [29] USB stick acted as client2’s data and Chair of Machine Tools and Control Systems’s (at Technische Universität Kaiserslautern) [30] USB stick acted as client3’s data. Client1 possesses a hexagonal USB drive with a rotating body that protrudes the USB plug from the body. Client2 has a rectangular, blue USB stick that resembles a LEGO brick, while Client3 has a similar USB stick that is red in color. The distinguishing characteristic of this data set is that the error on each client is unique. Client1 has little stickers on its USB port that represent a scratch error, Client2 has a damaged USB port as its fault, and Client3 has a rusty USB port as its error. The “HIDDEN” class represents a picture in which the USB port is not clearly visible to the camera, making it impossible to determine whether the USB port of the USB stick is damaged/inoperative. And the “EMPTY” class is comprised of blank images and also images of non-USB objects such as wires, screws, etc.

---

**Fig. 4. Different USB sticks and classes belonging to individual clients**

**Fig. 5. External data set of USB with 3.0 port for testing purpose**

All three Clients share the “OKAY” class, where the camera can clearly observe an error-free USB port. The data set consists of images taken from various angles and different
lighting conditions. The total number of images per client is kept unequal (figure 6) to demonstrate real life scenarios in which custom data sets are not always independent and identically distributed i.e. non-IID data set. For testing purpose we also used a USB 3.0 (figure 5) which had no errors to ensure if the final model created (whether federated or normal) is not over-fitted. This USB3.0 data was manually annotated and had a total of 3071 out of which 255 are “HIDDEN” and rest are “OKAY” images.

B. Architecture

VGG-11, VGG-16, VGG-19 [26] and a custom 5 layer CNN (similar to one mentioned in [19]) were used to train the entire USB data set (figure 4) and an external USB 3.0 data set was used to test these trained models. VGG-19 was chosen as the deep learning architecture for this federated application, since it performed the best among all other architectures. Pretrained ImageNet weights were used for the VGG network for training these models. VGG has a total of approximately 139 million parameters out of which 124 millions contribute to its fully connected layers (FC layer). VGG can be used to train a custom data set by fine-tuning the last layer’s output equal to the number of labels in the custom data set. For fine-tuned VGG as well as for transfer learning we use the pretrained VGG weights as the initial weights. The pretrained weights also act like a common weight initializer for all the clients taking part in the federated learning, as stated and proved by [19] that models with common initial weights are likely to reach a lower loss gradient and with less CRs.

For the application of disaster classification, [31] applies transfer learning to VGG architecture with a custom classifier layer and trains a model by sharing and federating only the classifier layers’ weights. Hence in this research, we tested with both VGG-19 convolution layers as feature extractor, as well as retraining the fine-tuned VGG-19 network for federated learning; the results of these experiments may be seen in the part in section IV. While training the entire VGG-19 model, the total number of parameters to train are around 139 millions, but as the last classifier layer is changed from 1000 to 4 as there are four classes as outputs, the total trainable parameter in our case is approximately 134 millions as shown in figure 7. Whereas the total number of trainable parameters in case of custom fully connected layer for transfer learning are 25,954,564 (approx. 26 millions) which can be seen in details in figure 7. It is common knowledge that the greater the number of parameters, the denser the model and the longer it takes to train a model with a high number of parameters. The global federated model acquired by training the entire fine-tuned VGG network (from now on referred to as Process1) is around 550 MB in size, where as the model produced by transfer learning and the weights of only FC layers (further referred as Process2) has model weights size around 140 MB.

C. Implementation
Using TCP-IP protocol within the same network, an application was developed in which three independent computers acted as clients and had access to the server IP where weights were sent and global weights were received. But for faster results and experimentation, we demonstrated that all the three clients and a server were deployed on the same system but had no direct connection and without access to one another’s data. This application and experimentation was performed on Ubuntu 20.4 with a 32GB memory Nvidia GeForce RTX 3090 graphics card. To simulate a federated learning scenario, each client’s local data was saved independently and was only accessible to that specific client. In our instance, three unique scripts were executed for three distinct clients, and the local weights were stored in a server-accessible location. Figure 8 elaborates more on our custom federated learning workflow on the client side. In accordance with the schematic 8, during the first communication round when there is no global model available in the mentioned path, the client utilises the pretrained ImageNet weights for VGG-19 as the initial weights to train the local epochs. After training, the weights are stored to a designated directory, and then the model’s accuracy is evaluated using the test data set. All the clients use the same seed point for common weight initialisation (for fully connected layer initialisation in Process2) and to acquire the same train-test data split on every run. On the server side, once all the clients have finished training the local data, the server creates VGG models equal to the number of clients in the CR to load the client weights and one extra model to store the weights derived from the federated learning algorithm. The federated learning algorithm used for this application is "FedAvg" [19] (as explained in subsection II-B, figure 3) where the resulting global model is achieved by averaging the weights of all the clients and creating a global model which acts as the initial weights for the next communication round for all the clients. After the first communication round when the global weights are available, (as described in figure 8) the client architecture will load these weights and first check the accuracy of this global model on the client’s local data and then use them as the initial weights for retraining the local data.

In case of Process1 all VGG weights are stored and federated, but in case of Process2 only the fully connected layers’ weights are stored and federated. Successful completion of the resulting global model occurs when the global model achieves the desired result accuracy on all of the client’s local test data. By using a custom classifier layer with fewer parameters, as depicted in figure 7, the average time required for a single epoch for all clients in Process1 is approximately 34 seconds, but for Process2 it is just 19.5 seconds.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

The results achieved from global federated model from both Process1 and Process2 are mentioned in table I. Results show that Process1 achieved a far better accurate global model with as less as five communication rounds (CRs). As the communication round increased we noticed that the accuracy of global model on client’s local data decreasing and hence we achieved an optimal global model at the 5th CR (the local epochs of each clients was set to 5 epochs). Similarly in case of Process2, as there are less trainable parameters as compared to Process1 (subsection III-B), it required more local epochs as well as more CRs. We experimented with local epochs set to 5, 10, 15 and 20 where as CRs for each of them were taken to around 100 CRs. The best model was always produced around 13th to 16th CR for the respective local epoch number. In this case local epoch set to 10 and 15th CR gave us a good model as compared to other hyper parameters.

| Model Type | Local Test Data Accuracy | External USB 3.0 Data Set Accuracy |
|------------|-------------------------|----------------------------------|
| Normal deep learning with VGG | 99.9% | 99.9% |
| Federated fine-tuned VGG model (Process1) | 99.9% | 97.4% |
| Federated transfer learning VGG model (Process2) | 99.9% | 85% |

Fig. 9. Confusion matrix for classification of external dataset with normal deep learning method using VGG-19 model

Fig. 10. Confusion matrix for classification of external dataset with global federated fine-tuned model (Process1)
Figure 9, 10 and 11 illustrates the confusion matrix for classification of the external test data set images (figure 5) by all the three models (3rd model being the normal deep learning model). Similarly, table I demonstrates the best accuracy of all three models on both local and external dataset. For Process2 to achieve the accuracy shown in table I, it required minimum 15 CR. Confusion matrix shown in figure 11 demonstrates that the model cannot perform well on an external test data set. A live classification of USB sticks was done, where the federated transfer learning model (Process2 model) performed the worst. Training a fine-tuned federated VGG model (Process1) outperforms a Transfer learning custom federated model (where both models use pre-trained ImageNet weights) because ImageNet data set [27] does not contain very similar images/objects used for our custom use case. As a result, freezing weights, which acts as a feature extractor in transfer federated learning, did not perform well for quality inspection or manufacturing, as the images are too specific and private to the use case. In contrast, the Process1 model weight size is approximately 500 MB, which can make the transmission of model weights between server and clients challenging. A reference model is trained with the entire data set using normal deep learning methods with the same architecture and with 30 epochs to have a baseline to compare the global federated model. Table I and confusion matrix from figures 9 and 10 illustrate that the model trained with entire data set performs slightly better on the local test set as well as the external data set. But the global federated model (Process1) which is trained by just sharing the model weights between clients, does not lack much behind when compared to the normal deep learning model. To test the robustness of these models, we created a live classification scenario that was slightly different in terms of environment, lighting conditions, and angles than the environment in which the train data set was created.

Figure 12 depicts the classification of a client1 USB stick using fine-tuned federated global model (left side) and normal deep learning (right side). It is interesting to note that client2’s USB stick has client1’s fault and is appropriately categorised as “NOT_OKAY” by both classification models (lower half). Figure 12 demonstrates that the global federated model is able to classify the image with greater confidence than the model trained using conventional deep learning (with entire data set). This proves that a federated model can perform as well as a standard deep learning model, even without access to raw image data and by just exchanging model weights. In our case the global federated model generalises more effectively than the standard deep learning model as it can be seen by the confidence score from both image classifications as illustrated in figure 12. Due to GDPR law, the primary application of federated learning in Europe is training a global model for a given specific application by several parties/clients, without revealing private or sensitive data.

Figure 13 shows the live classification of this federated use case, in which the global model is compared to each of the three clients trained for 30 epochs on their own local data set, and all of them have the identical architecture (VGG-19 architecture with last layer output as 4). This is to demonstrate that in normal circumstances, each client only has access to it’s own data and is limited on training a deep learning model based on its local data. Client3’s error USB stick in figure 13 is accurately predicted by client3 and the global federated model. Client1 identifies the USB as “NOT_A_USB” (which is just another term for “EMPTY class”) since it has never encountered a USB of that shape. For client2 the error dataset consists of damaged USB sticks and hence it classifies the USB as “OKAY”. Figure 14 shows the live classification of client2’s USB stick containing client1’s error, which was successfully classified by client1’s model because the error belongs to its data set, but with a lower confidence score because this USB stick was never in client1’s data set. The image is classified as ”OKAY” by both client2 and client3 model, since the error does not belong to either clients’ data.
V. Conclusions

With the implementation of the new GDPR regulations [17], federated learning can be utilised to build a robust deep learning model on private or sensitive data. In this paper we prove that application like quality inspection in manufacturing, where the error/faulty images in production line are few in number and also fewer in variety; using federation learning, multiple parties can collaborate to train a robust quality inspection model for similar products or use case. A method wherein the convolution layers of VGG act as a feature extractor and the weights of custom classifier layers are federated to create a global model (model size 140MB) required at least 15CR which has a movable cap on top that exposes the USB port. This third-party USB stick is likewise accurately classified by the global federated model, which strengthens our use case’s reliance on federated learning approach. Federated learning is therefore a useful tool in smart factories if numerous partners wish to construct a quality inspection model for a particular product or use case.

As a part of the experimentation (using Process1), we even tried training the clients with common random initial weights (training from scratch) but it took at least 20 CR to even notice the progress of accuracy of the global model on the clients’ local data set and even the global model achieved from 100\textsuperscript{th} did not perform well while classifying the external data set (figure 5) or even with the live classification. With random initial weights (different initial weights for all clients) the results where worst (it took 35 CR to even notice a change in local accuracy) than the ones with common initial weights as proved by [19]. Hence using a pretrained ImageNet model for fine-tuned VGG-19 proved to be the best architecture in our scenario.
to achieve the desired accuracy on an external dataset, but performed poorly in a live classification scenario. Whereas a finely-tuned VGG model (model size 558MB) required only 5 CR to develop a global model that generalised exceptionally well on the USB images. In the live classification scenario, the fine-tuned global federated model detected all errors with a slightly higher confidence score than the VGG model trained on centralised data. Using pre-train ImageNet weights as initial weights for a fine-tune VGG network trained on custom data in a federated context created a classification model with fewer communication rounds. We are amongst the few that have successfully experimented with federated learning using a custom data set in manufacturing, where the global model was able to generalise well despite being constructed by simply exchanging model weights, therefore keeping the clients’ local data secure and private.

VI. ACKNOWLEDGEMENT

This project was done in collaboration with Huawei Technologies Duesseldorf GmbH, at the European Research Center in Munich. Thanks for the support of Prof. Juergen Grotepass and Kirill Fridman in this application that also has been demonstrated on Hannover Fair, May 31st 2022.1

REFERENCES

[1] X. Amatriain, “Big & personal,” in Proceedings of the 2nd International Workshop on Big Data, Streams and Heterogeneous Source Mining Algorithms, Systems, Programming Models and Applications, ser. ACM Conferences, W. Fan, Ed. New York, NY: ACM, 2013, pp. 1–6.

[2] S. U. Amin, M. S. Hossain, G. Muhammad, M. Alhussein, and M. A. Rahman, “Cognitive smart healthcare for pathology detection and monitoring,” IEEE Access, vol. 7, pp. 10 745–10 753, 2019.

[3] A. Thennakoon, C. Bhagyan, S. Premadasa, S. Mihiranga, and N. Kuruwitaarachchi, “Real-time credit card fraud detection using machine learning,” in Proceedings of the 9th International Conference on Cloud Computing, Data Science and Engineering - Confluence 2019. Piscataway, NJ: IEEE, 2019, pp. 488–493.

[4] Q. Wang, Y. Guo, L. Yu, and P. Li, “Earthquake prediction based on spatio-temporal data mining: An lstm network approach,” IEEE Transactions on Emerging Topics in Computing, vol. 8, no. 1, pp. 148–158, 2020.

[5] M. Erkinay Ozdemir, Z. Ali, B. Subeshan, and E. Asmatulu, “Applying machine learning approach in recycling,” Journal of Material Cycles and Waste Management, vol. 23, no. 3, pp. 855–871, 2021.

[6] L. Li, K. Ota, and M. Dong, “Humanlike driving: Empirical decision-making system for autonomous vehicles,” IEEE Transactions on Vehicular Technology, vol. 67, no. 8, pp. 6814–6823, 2018.

[7] D. T. Pham and A. A. Alify, “Machine-learning techniques and their applications in manufacturing,” Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture, vol. 219, no. 5, pp. 395–412, 2005.

[8] R. Rai, M. K. Tiwari, D. Ivanov, and A. Dolgui, “Machine learning in manufacturing and industry 4.0 applications,” International Journal of Production Research, vol. 59, no. 16, pp. 4773–4778, 2021.

[9] L. J. Porter and P. Rayner, “Quality costing for total quality management,” International Journal of Production Economics, vol. 27, no. 1, pp. 69–81, 1992.

[10] A. K. Choudhary, J. A. Harding, and M. K. Tiwari, “Data mining in manufacturing: a review based on the kind of knowledge,” Journal of Intelligent Manufacturing, vol. 20, no. 3, pp. 501–521, 2009.

[11] J. Lenz, T. Wuest, and E. Westkämper, “Holistic approach to machine tool data analytics,” Journal of Manufacturing Systems, vol. 48, pp. 180–191, 2018.

[12] R.-Y. Yang and R. Rai, “Machine auscultation: enabling machine diagnostics using convolutional neural networks and large-scale machine audio data,” Advances in Manufacturing, vol. 7, no. 2, pp. 174–187, 2019.

[13] M. Bahaghighat, F. Abedini, M. S’hoyan, and A.-J. Molnar, “Vision inspection of bottle caps in drink factories using convolutional neural networks,” in 2019 IEEE 15th International Conference on Intelligent Computer Communication and Processing (ICCP), S. Nedevschi, R. Potelea, and R. R. Slavescu, Eds. Piscataway, NJ: IEEE, 2019, pp. 381–385.

[14] T. Wang, Y. Chen, M. Qiao, and H. Snoussi, “A fast and robust convolutional neural network-based defect detection model in product quality control,” The International Journal of Advanced Manufacturing Technology, vol. 94, no. 9, pp. 3465–3471, 2018.

[15] R. Kontar, N. Shi, X. Yue, S. Chung, E. Byon, M. Chowdury, J. Jin, W. Kontar, N. Masoud, M. Nouiehed, C. E. Okwudire, G. Raskutti, R. Saigal, K. Singh, and Z. Ye, “The internet of federated things (iott): A vision for the future and in-depth survey of data-driven approaches for federated learning.” [Online]. Available: http://arxiv.org/pdf/2111.05326v1

[16] P. Dube, T. Suk, and C. Wang, “Ai gauge: Runtime estimation for deep learning in the cloud,” in 2019 31st International Symposium on Computer Architecture and High Performance Computing. Piscataway, NJ: IEEE, 2019, pp. 160–167.

[17] P. Voigt and A. von dem Bussche, “The eu general data protection regulation (gdpr),” A Practical Guide, 1st Ed., Cham: Springer International Publishing, vol. 10, p. 3152676, 2017.

[18] B. K. Mathew, J. C. Ng, and J. L. Zerbe, “Using proxies to enable on-device machine learning,” 2018.

[19] Brendan McManih, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas, “Communication-efficient learning of deep networks from decentralized data,” Artificial Intelligence and Statistics, pp. 1273–1282, 2017.

[20] T. Li, A. K. Sabu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, “Federated optimization in heterogeneous networks,” 2018. [Online]. Available: https://arxiv.org/abs/1812.06127

[21] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, “Federated learning with mixed averaging,” 2020. [Online]. Available: https://arxiv.org/abs/2002.06440v1

[22] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, T. Chen, H. Yu, and Q. Yang, “Fedvision: An online visual object detection platform powered by federated learning,” in AAAI 2020 - 34th AAAI Conference on Artificial Intelligence. AAAI press, I 2020, pp. 13 172–13 179.

[23] “Google ai blog: Federated learning.” [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html (as accessed on 11.07.2022)

[24] “Understanding federated learning terminology.” [Online]. Available: https://blog.openmined.org/federated-learning-types/ (as accessed on 11.07.2022)

[25] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated machine learning,” ACM Transactions on Intelligent Systems and Technology, vol. 10, no. 2, pp. 1–19, 2019.

[26] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. International Conference on Learning Representations, ICLR, 2015.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248–255.

[28] “Huawei openlab munich.” [Online]. Available: https://www.huawei.com/en/news/2016/3/Huawei-Openlabs-Munich (as accessed on 11.07.2022)

[29] “Smart factory kaiserslautern,” as accessed on 11.07.2022. [Online]. Available: https://smartfactory.de/ (as accessed on 11.07.2022)

[30] “Chair of machine tools and control systems,” as accessed on 11.07.2022. [Online]. Available: https://www.mv.uni-kl.de/en/wskl (as accessed on 11.07.2022)

[31] Z. Zhang, N. He, D. Li, H. Gao, T. Gao, and C. Zhou, “Federated transfer learning for disaster classification in social computing networks,” Journal of Safety Science and Resilience, vol. 3, pp. 15–23, 3 2022.

1https://www.youtube.com/watch?v=-ZzbiHO846k