IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 5 September 2023, accepted 24 September 2023, date of publication 29 September 2023,
date of current version 25 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3320766

RESEARCH ARTICLE

A Practical Approach on Performance Assessment of Federated Learning Algorithms for Defect Detection in Industrial Applications

E. ZULUAGA[1], S. JAZIRI[1], A. TELLAECHE[2], AND I. PASTOR-LÓPEZ[2]
[1]D4K Group, Deusto Tech, University of Deusto, 48007 Bilbao, Spain
[2]Computer Science, Electronics and Communication Technologies Department, University of Deusto, 48007 Bilbao, Spain
Corresponding author: A. Tellaeche (alberto.tellaeche@deusto.es)

ABSTRACT One of the most common problems to be solved in industrial environments is the detection of defects in the manufacturing quality control of different products. The design of such automatic systems, especially if they are based on image processing, often presents difficulties related to the availability of sufficient well- labeled data for initial training. Usually, it is not easy to have enough industrial samples with defects to have a sufficiently large and balanced dataset. This research work presents a new method of hybridisation of convolutional neural networks used for defect detection in industry by means of image processing, using Federated Learning (FL) techniques. This method is able to overcome the limitations and problems presented by this type of systems stated above. In this research, it is demonstrated with examples of defect detection in textures that it is possible to reach a defect detection effectiveness above 90% on average, in problems where no dataset is available, using federated learning algorithms with classifiers based on Convolutional Neural Networks previously trained in other problems of defect detection in other types of textures. The creation of such systems for error detection on previously untrained data using federated learning and obtaining this effectiveness represents an advance on the state of the art with respect to the existing approaches, and constitutes the main contribution of this research work.

INDEX TERMS Deep learning (DL), federated learning (FL), automatic optical inspection.

I. INTRODUCTION

This section provides a detailed study of the artificial intelligence techniques used in this research. First, a review is made of the main characteristics that systems and algorithms based on Deep Learning for defect detection in industrial production systems must fulfill, followed by an exhaustive review of the main federated learning algorithms available to generalize this type of industrial classifier to obtain satisfactory results in defect detection problems in which no dataset is available.

A. DEEP LEARNING FOR DEFECT DETECTION

Defect detection (DD) is essential for preventing unanticipated equipment failures, ensuring manufacturing effectiveness, and guaranteeing operational safety [1]. In the literature, system types for defect detection and diagnosis include model-based methods, data- and signal-based methods, and knowledge-based methods [2]. Researchers have paid increasing attention to signal-based or data-driven methodologies in the modern industry, since these approaches offer excellent diagnostic accuracy. In addition, they don't necessitate empirical measurement of physical characteristics [3]. Although, data-driven techniques have two shortcomings: training and testing samples must come from the same data distribution, and they devolve on proper data preparation [4]. In recent decades, support vector machine (SVM), k-nearest neighbor (KNN), decision tree (DT), and artificial neural network (ANN) models have frequently been utilized in DD systems. Sometimes, these types of models are referred to as shallow machine learning (SML) models [5]. DD systems that depend on SML models

The associate editor coordinating the review of this manuscript and approving it for publication was Turgay Celik.

are based on statistical data. They are called traditional defect detection [6]. The enormous number of features in a model must be carefully chosen in order to operate effectively in DD systems, lower the computational complexity of the SML model, and maintain classification accuracy [7]. For this reason, researchers introduced the feature selection process in the SML model to eliminate unnecessary and duplicate features [5]. On the other hand, DL models are immensely effective with large training data sets. At a certain stage, model's performance will become stagnant. Therefore, deep learning models are often implemented to overcome these drawbacks and ensure the efficiency of data-driven approaches. The human feature selection process used in conventional defect detection can be replaced by an automated feature learning approach provided by deep learning models.

B. RELATED WORK USING DEEP LEARNING FOR DEFECT DETECTION

Deep learning (DL) is a branch of machine learning techniques that uses multiple layers of information-processing stages in hierarchical architectures to learn features or representations and classify patterns. The methods created by deep learning research has recently started to influence a variety of signal-processing and information-processing tasks in both traditional and new, enlarged fields, including machine learning and artificial intelligence. Several applications, including natural language processing (NLP) [8], image processing [9], robotics [10], and medical applications [11], have all taken advantage of deep learning. The extreme use of DL techniques is due to three factors: The recent developments in machine learning and signal/information-processing research, as well as the greatly improved chip processing capabilities (e.g., GPU units), have all contributed to the adoption of deep learning models. Convolutional neural networks (CNN), stacked auto-encoders (SAE), restricted Boltzmann machines (RBM), deep belief networks (DBN), and deep neural networks (DNN) are the most popular deep learning models that have been widely used in defect detection [12]. One example of the application of Deep Learning in defect detection can be found at [13] study, which uses deep learning to detect defects in 6G Industry 4.0 heterogeneous data environments The study presents the ADL-FDI4 framework (Advanced Deep Learning Framework for Defect Diagnosis in Industry 4.0), which receives images, videos, time series, and graphs about the same industrial event as inputs and then combines all the information in these different formats to produce the most precise defect diagnoses possible for that event. Besides, in [14], an application is proposed that uses deep learning for preventive defect detection in rotating machinery based on vibration signals. Similarly, [15] introduces an ANN-based defect classification and location method for medium voltage direct current (MVDC) shipboard power systems.

Antagonistic Generative Networks(AGN) and Covolutional Autoencoders (CA), in [16], proposed using AGN to model training data with defect-free images for texture anomaly detection. Similarly, CA is also used for detecting anomalies by considering the reconstruction error obtained by the network when trying to reconstruct an anomaly not previously trained. Spatial information using structural similarity are incorporated to improve segmentation results in CA-based anomaly detection [17].

Semisupervised Algorithm for Anomaly Detection [18] use autoencoders to create a semisupervised algorithm for anomaly detection, focusing on two different datasets; one synthetic and the other centered on railway images. However, their approach presents noisy results with only one type of defect detection, which not be applicable to cases with multiple defect types.

Triplet Network for surface Defect Detection use a specific network architechture called a triplet network to detect surface defects [19]. Nevertheless, this method is supervised and requires precise labeling of defects in the dataset, which can be challenging for industrial products with unpredictable and diverse defect forms.

Transfer Learning for Fault Detection focus on the transferability of trained parameters from unsupervised to supervised domains for fault detection, where labeled defect samples are scare [20]. However, the proposed approach in this text uses a real industrial dataset and combines CA and one-class SVM from improved defect detection results.

In all these previous examples and in the rest present in the literature, the efficient training of the network models is an essential component, taking into account three major factors for this training process: the architecture of the DL model, the size of the dataset, and the characteristics of the input data. This paper presents efficient methods for overcoming these three limiting factors, provided there are already existing deep learning models for similar defect detection applications.

FL offers a distinct advantage in tackling these three limiting factors on DD. Firtsly, in terms of the architechture of the DL model, FL facilitates collaborative learning across distributed devices, allowing the integration of diverse local models leading to more comprehensive and robust DD architectures. Secondly, regardin the size of the dataset, eliminating the need to centralize the sensitive defect data. Finally, concerning the characteristics of the input data, FL leverages data from various sources, enriching the model's ability to handle diverse defect patterns and variations encountered in real-world applications.

C. STATE OF THE ART IN FEDERATED LEARNING ALGORITHMS

Federated learning (FL) algorithms are designed to bring processing to the data instead of moving data to the processing location. This approach addresses privacy, ownership, and data localization concerns. In FL, data is distributed across decentralized devices, known as workers [21]. Each worker independently learns from its private dataset to generate a local model. The ultimate goal is to enhance a global model

by incorporating knowledge from these local models, or by aggregating knowledge from nearby devices if the focus is solely on local information.

The aim of FL is to ensure that the learning achieved from the global model closely matches the results that would have been obtained from a completely centralized model trained on the entire dataset. The FL process involves several steps, which can be summarized as follows:

a. Generate an initial global model.
b. Select workers for participation.
c. Distribute the global model to the selected workers
d. Retrain the network or wait for the next round if not selected.
e. Receive updates from each worker
f. Aggregate the modifications provided by the device to improve the overall model
g. Iterate through multiple rounds until the global model converges, based on predefined termination criteria.

This learning process enables the resolution of traditional application problems that involve data from sources.By applying learning to smaller datasets, FL reduces the computational power required. Additionally, it facilities collaboration for those whose ability to share data is limited by its volume or privacy restrictions.

Furthermore, FL does not assume that the data must be independent and identically distributed (IID). It supports non-IID data, which means it can handle data with the following characteristics:

a. Biases in feature or label distributions.
b. Unbalance in available data.
c. Different characteristics for the same label, or vice versa.

Moreover, while FL architecture itself provides improvements in data privacy, it is not enough. Additional layers must be applied to mitigate common attacks to which this network is still vulnerable.

Finally, it is important to note that FL should not be confused with distributed models, as the latter primarily aim to enhance computing capacity by creating robust networks with minimal possibility of failure. In FL, data security and the independence of participating devices in the learning process are of greater importance. Therefore, less robust means of communication, such as WiFi, can be utilized.

1) ARCHITECTURES

To obtain a global model, two approaches can be adopted. The first approach is based on a centralized architecture where a central server acts as the network coordinator. The second approach involves operating solely with local devices, where each device can assume the role of coordinator.

In the centralized architecture, multiple local devices, referred to as workers, are coordinated by a central server (Fig. 1). Each worker is responsible for training the network and transmitting parameter updates to the server. The central server then aggregates the received parameters from all selected workers based on the chosen learning method. [22].

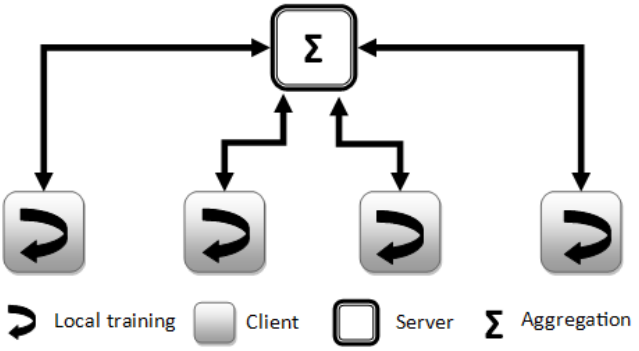


FIGURE 1. Centralized architecture.

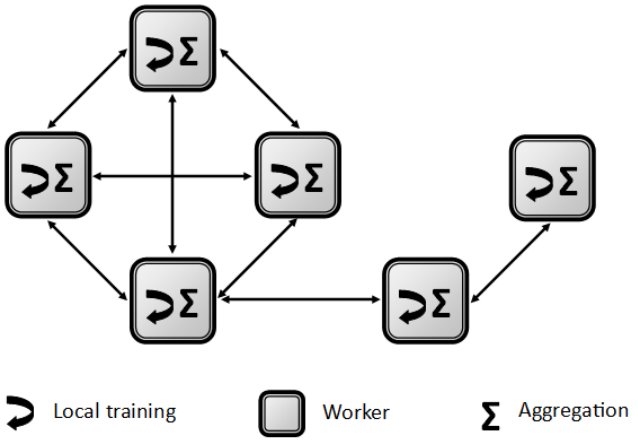


FIGURE 2. Decentralized architecture.

The decentralized architecture arises from the need to address two main challenges: the bottleneck that occurs in the central server when capturing parameters from workers, and the limitations in computing power when dealing with a large number of parameters sent by workers

In this architecture (Fig 2), all workers do not need to be directly connected to each other. Instead, each worker shares its updated parameters with a nearby worker, which takes on the role of an aggregator. This approach results in reduced communication and training time. Each worker improves the model's performance by incorporating its own parameters as well as the parameters received from nearby devices [23].

2) CATEGORIZATION OF FEDERATED LEARNING

FL can be categorized into three main groups in terms of the feature spaces (X), labels (Y) and samples (I). These categories depend on the kind of feature that each worker shares, resulting in different scenarios [24].

The first category is the horizontal case, where each worker utilizes the same feature space and labels, but the samples they possess are different (Fig. 3).

$$Xi \equiv Xj, \quad Yi \equiv Yj, \quad Ii \neq Ij \quad \forall D_i, D_j, \quad i \neq j \qquad (1)$$

In the vertical category, workers utilize supplementary data, meaning that each worker has data on specific features

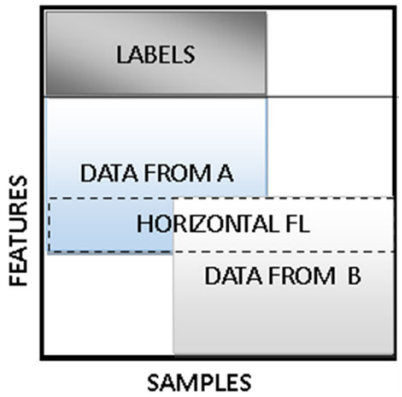


FIGURE 3. Horizontal federated learning.

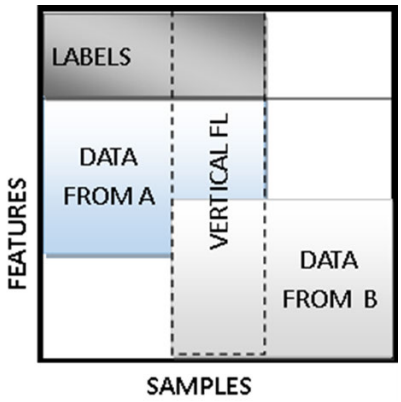


FIGURE 4. Vertical federated learning.

that, on their own, do not make up a complete training model (Fig. 4). Consequently, samples are shared, and it is necessary to have the features of worker A and worker B to be able to infer from the data.

$$Xi \neq Xj, \quad Yi \neq Yj, \quad Ii \equiv Ij \quad \forall D_i, D_j, \quad i \neq j \qquad (2)$$

Lastly, federated transfer learning can be applied when the data of each worker differs in both the samples and the feature space (Fig. 5). This is an important enhancement as it addresses the challenge of extending the scope of existing federated learning algorithms.

$$Xi \neq Xj, \quad Yi \neq Yj, \quad Ii \neq Ij \quad \forall D_i, D_j, \quad i \neq j \qquad (3)$$

3) LEARNING ALGORITHMS

Learning algorithms play a crucial role in federated learning by aggregating models from multiple nodes into a single global model. These algorithms aim to accelerate convergence, reduce training time, or improve the overall performance of the model.

While there are emerging algorithms designed specifically for decentralized architectures, their focus lies more on data security and worker validation rather than enhancing model's performance.

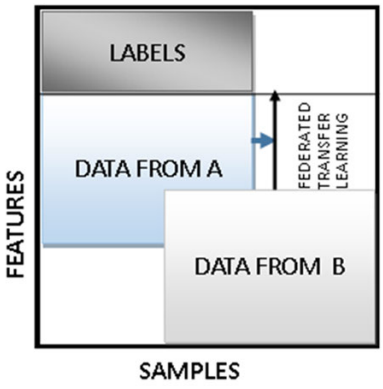


FIGURE 5. Federated transfer learning.

In our case, the proposed algorithms are intended to be applied in a centralized architecture to achieve more accurate results from the global model. However, they can be adapted for decentralized aggregation if needed.

For the centralized algorithms, each worker performs E epochs for the stochastic gradient descent (SGD) updates with a mini-batch size of B. The K workers are indexed by k, and each worker has n_k samples of local data. Consequently, the number of local SGD iterations, denoted by τ_k, can vary significantly across workers and is calculated as $\tau_k = [En_k/B]$.

The selection of workers S_t for each round is determined by randomly choosing a number of workers, denoted as C, which will perform the calculations. The value m_t represents the number of samples in this selection.

In the algorithms, w_t^k represents the indexed weight for each worker, while w_t denotes the global average of the weights for the selected workers. In addition, the learning rate η determines the influence of the loss function values during local training on the weights of the convolution layers.

The simplest ways to aggregate information in a model are given by the two typologies presented below:

- FedSGD: This method is based on the stochastic gradient descent. Each worker shares their local gradient with a central server. Then, the server averages them by weighting the number of training samples for each gradient step. However, this approach is computationally inefficient because the worker has to communicate with the central server twice after each gradient descent step [25].
- FedAvg: Building upon the previous algorithm, FedAvg replaces the gradient averaging step with weight averaging (Algo. 1). This turns out to be a much more efficient algorithm in terms of communication, as it allows several gradient descents among each transmission of information [26].

In the aforementioned algorithms, it is assumed that the number of local updates is the same for all workers $\tau_k = (\tau$ for each worker $k)$. However,in practice, this uniformity of updates is often not achieved due to the heterogeneity

Algorithm 1 FedAvg

Server execute:
Initialize W_0
for each round t=1,2,... **do**
  $m \leftarrow max(CK, 1)$
  $S_t \leftarrow (random\ set\ of\ m\ workers)$
  for $each\ worker k \in S_t\ in\ parallel$ **do**
    $w_{t+1}^k \leftarrow WorkerUpdate(k, w_t)$
  end for
  $m_t \leftarrow \sum_{k \in S_t} n_k$
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
end for
Client update(k,w):
$B \leftarrow$ (split P_k into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in B$ **do**
    $w \leftarrow w\eta \nabla l(w; b)$
  end for
end for
return w to server

Algorithm 2 FedProx

Server execute:
Initialize W_0
for each round t=1,2,... **do**
  $m \leftarrow max(CK, 1)$
  $S_t \leftarrow (random\ set\ of\ m\ workers)$
  for each worker $k \in S_t\ in\ parallel$ **do**
    $w_{t+1}^k \leftarrow WorkerUpdate(k, w_t)$
  end for
  $m_t \leftarrow \sum_{k \in S_t} n_k$
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
end for
Client update(k,w):
$B \leftarrow$ (split P_k into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in B$ **do**
    $w \leftarrow w\eta \nabla l(w; b) + \frac{\mu}{2} \|w - w^t\|^2 where \mu > 0$
  end for
end for
return w to server

among workers in terms of their local dataset size or computational speed. As a result, convergence to a minimum can significantly deviate from the global minimum [27].

For this reason, other kinds of algorithms are developed to deal with heterogeneous data, like the following typologies:

- FedProx: It can be understood as an adaptation of FedAvg. This algorithm adds a proximal term to the local loss function to avoid abrupt deviations from the global minimum. This means that there is only a change in the step 14 of Algorithm 1 (Algo. 2). So, the server continues to average the weights of the selected workers [28].
  This proximal term is given by the square of the Euclidean distance between local and global weights. Besides, an adjust parameter (μ) is introduced to control the degree of fitting to the global model.FedProx demonstrates significantly more stable and accurate convergence compared to FedAvg when dealing with heterogeneous data.
  When $\mu = 0$ the FedAvg algorithm is acquired.
- FedSkip: As with the previous algorithm, FedSkip can also be understood as an adaptation of FedAvg (Algo. 3). Wherein the server algorithm is modified instead of making changes to the workers. It introduces Iskip stages, during which the server shuffles the weights instead of averaging them (Fig. 6). By doing so, each worker gains access to additional data while preserving privacy. Since the server implements these modifications, the workers are unaware the weights they receive originate from another worker or are an average of the weights [29].
- FedDC: It is an algorithm designed to address non-linear optimized functions, distinguishing itself from simple

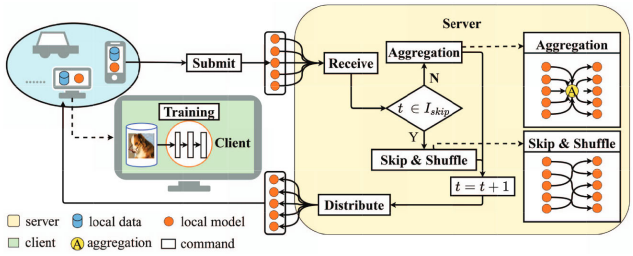


FIGURE 6. FedSkip aggregation process.

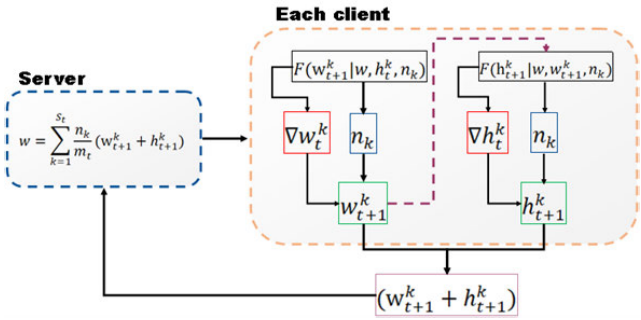


FIGURE 7. FedDC aggregation process.

weight averaging (Fig. 7). It acknowledges the presence of drift between local and global models [30], which it utilizes to enhance the robustness and speed of model convergence through learning the model drift (Algo. 4). Each worker calculates this drift and sends it to the server. This drift is given by:

$$h_{t+1}^k = h_t^k + (w_t^k - w_t) \tag{4}$$

The previous drift h_t^k, worker weights w_t^k, and global weights w_t are involved in the computation of this drift.

Algorithm 3 FedSkip

Server execute:
Initialize W_0
for each round t=1,2,... **do**
  $m \leftarrow max(CK, 1)$
  $S_t \leftarrow (random\ set\ of\ m\ workers)$
  if $t \in I_{skip}$ **then**
    for each worker $k \in S_t$ *in parallel* **do**
      $w_{t+1}^k \leftarrow w_t^{random(k)}$
    end for
  else
    for each worker $k \in S_t$ *in parallel* **do**
      $w_{t+1}^k \leftarrow WorkerUpdate(k, w_t)$
    end for
  end if
  $m_t \leftarrow \sum_{k \in S_t} n_k$
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$
end for
Client update(k,w):
$B \leftarrow$ (split P_k into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in B$ **do**
    $w \leftarrow w\eta\nabla l(w; b)$
  end for
end for
return w to server

Algorithm 4 FedDC

Server execute:
Initialize W_0
for each round t=1,2,... **do**
  $m \leftarrow max(CK, 1)$
  $S_t \leftarrow (random\ set\ of\ m\ workers)$
  for each worker $k \in S_t$ *in parallel* **do**
    $w_{t+1}^k \leftarrow WorkerUpdate(k, w_t)$
  end for
  $m_t \leftarrow \sum_{k \in S_t} n_k$
  $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t}(w_{t+1}^k + h_{t+1}^k)$
end for
Client update(k,w):
B $\leftarrow$ (split P_k into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in B$ **do**
    $w \leftarrow w\eta\nabla l(w; b) + \frac{\alpha}{2} R_k(w_k; h_k, w) + G_k(w_k; g_k, g)$
  end for
end for
return w to server

Additionally, the algorithm incorporates a penalized term into the local loss function, akin to the proximal term in FedProx. However, instead of solely calculating the difference between local and global weights, it incorporates the previous drift as well.

$$R_k(w_k; h_k, w) = \|h_k + w_k - w\|^2, \forall k \in K \quad (5)$$

Furthermore, an essential aspect of this algorithm is the inclusion of a gradient correction term to regulate gradient stochastic optimization. This term is determined by computing the difference between the local gradient (g_k) and the average of local gradients (g), divided by the product of the learning rate (η) and the number of training iterations per round (n).

$$G_k(w_k; g_k, g) = 1/\eta n < g_k - g > \quad (6)$$

D. ARTICLE ORGANIZATION

This article is organized in different sections as follows: The introduction section has already presented an in-depth study of the deep learning techniques used for defect detection and an updated state of the art of the most common federated learning algorithms available. Following this first main section in the article, Section II will present precisely the main research contributions, to join with Section III, proposing the description of the research problem thought to assess the validity of the method proposed. Section IV describes data, tests and metrics used, to end presenting final valuable results, and finally Section V summarizes the conclusions obtained and presents the path for different future works that could continue the work started with this research

II. MAIN RESEARCH AIM AND CONTRIBUTIONS

This paper presents a novel research work, applying federated learning algorithms to effectively train defect detection systems on complex textures where no initial dataset is available. Consequently, the main objective of this research work is the definition of a method for hybridizing the configuration of defect detection systems in similar environments, resulting in a system capable of generalising the defect detection in the original proposed problem.

More specifically, these are the main contributions that should be highlighted:

- A specific design method for automatic texture defect detection systems is presented based on Deep Learning without the availability of an associated dataset.
- The validation of using different Federated Learning algorithms in the design of industrial systems is confirmed and demonstrated.
- The presented approach obtains results comparable, if not superior, to those obtained by classical dataset-based training techniques for these types of systems.

III. PROBLEM DESCRIPTION AND METHODS

In industrial manufacturing, processes are often controlled locally, or, in other words, in a distributed manner. Each machine has an embedded quality control system that supervises the local operation. That being said, it is also true that similarities are common among these processes. For example, defect detection and monitoring of textures are

common in many different machines that process laminated materials (wood, paper, metal, etc.). Dimensional control is also common to all the processes involved in manufacturing industrial metal parts in the automotive industry. In addition to all this, industrial monitoring applications are increasingly integrating the advantages provided by AI.

For these previously explained situations, the AI models in charge of quality monitoring in each machine can improve their performance by broadening and integrating different models from many different machines into a single global model, learning characteristics from these models in other machines that perform similar monitoring procedures. This kind of generalization is executed through the use of FL.

Therefore, this research presents a series of tests using Federated Learning aggregation algorithms in systems using convolutional neural networks for quality control, demonstrating that it is possible to create high-performance supervisory systems for environments where no pre-labeled dataset is available for training. A schematic explanation of the proposed approach is displayed in Fig. 8.

For the design of the local defect detection algorithms for each of the training textures, two different types of convolutional neural networks (CNN) have been chosen. The first of these, the Squeezenet model [31], is a network designed for mobile devices or devices with limited processing capacity. It has been tested in the state of the art providing very good accuracy in relation to its small footprint. The second selected network is the Darknet19 model [32], a medium-sized network, widely used in object detection models, such as YOLOv2 project [32].

The training of the different network architectures and the execution of the federated learning algorithms will be carried out in a server machine locally, with the following hardware configuration:

- CPU: Intel Core i7-10700F
- RAM: 16 GB DDR4
- GPU: NVidia GeForce RTX 2060 (computation capability of 7.5)
- SSD: Samsumg 870 1TB

This machine is running Ubuntu 22.04 LTS as operating system. The software used for network training has been MATLAB 2023A along with the Deep Learning Toolbox. This software selection provides in-depth control of the training procedure, including input image adaption to the input layer of the network during the training process. It also provides the access to the network trained parameters in a very straightforward way. For the implementation of the Federated Learning approaches, Tensorflow and Tensorflow-federated libraries have been used.

IV. COMPARATIVE ANALYSIS AND PERFORMANCE EVALUATION

This section presents the technical development of the proposed solution. The first subsection details the different datasets used, both the one used to train the two CNNs proposed as classifiers, and the one used in the test stage, a different dataset not used for training. The second section details in depth the training process, detailing hyperparameters of the networks and metrics used to evaluate the results obtained with the training textures. Finally, the last two sections detail the federated learning techniques to hybridise both CNNs, ending with the analysis of the results per generic classifier obtained by detecting faults in the test dataset.

TABLE 1. Mean defect size for each texture.

Texture	Area(pix)
Wood	13.256
Tile	3173
Grid	3655
Carpet	61.587
Leather	9066

A. DATASETS FOR DEFECT DETECTION IN TEXTURES

Two datasets have been utilized in this research. The first dataset, the Anomaly Detection Dataset from MVTec GmbH has been employed to train the local models of the CNN networks selected, with one model dedicated to each of the five basic textures present in this dataset.

The second dataset is the Kolektor dataset. This dataset will be used to assess the performance of the general model created from the local models (Fig. 8), employing different FL algorithms.

The Anomaly Detection Dataset of the German business MvTec GmbH's [33] was chosen as the ideal test dataset for comparison purposes. This data is specifically designed to address the challenges faced by machine learning models in classifying anomalies. Its task involves finding the optimal segmentation of defective images that closely resembles the correct images.

The dataset is composed by ten different objects of any kind presenting defects (cables, bottles…) and five fundamental textures, all of them common in industrial manufacturing processes. These fundamental textures are: carpet, grid, leather, tile and wood. Each of these basic textures have five differently classified defects each. For this research work, only this part of the dataset will be used as the solution presented is focused in error detection in general textures.

These textures were selected due to the model's requirement of recognizing subtle variations in an environment that closely resembles the correct one. This poses a significant challenge for machine learning algorithms (Table 5), while humans find this task relatively easy when provided with a reference pattern. To address this issue, the mean defect size for each texture is obtained.

Additionally, these datasets are representative of real-world inspections in the industry, making them the most suitable and challenging scenario for evaluation. The primary challenge arises when new types of anomalies emerge during production, requiring the generated model to identify them.

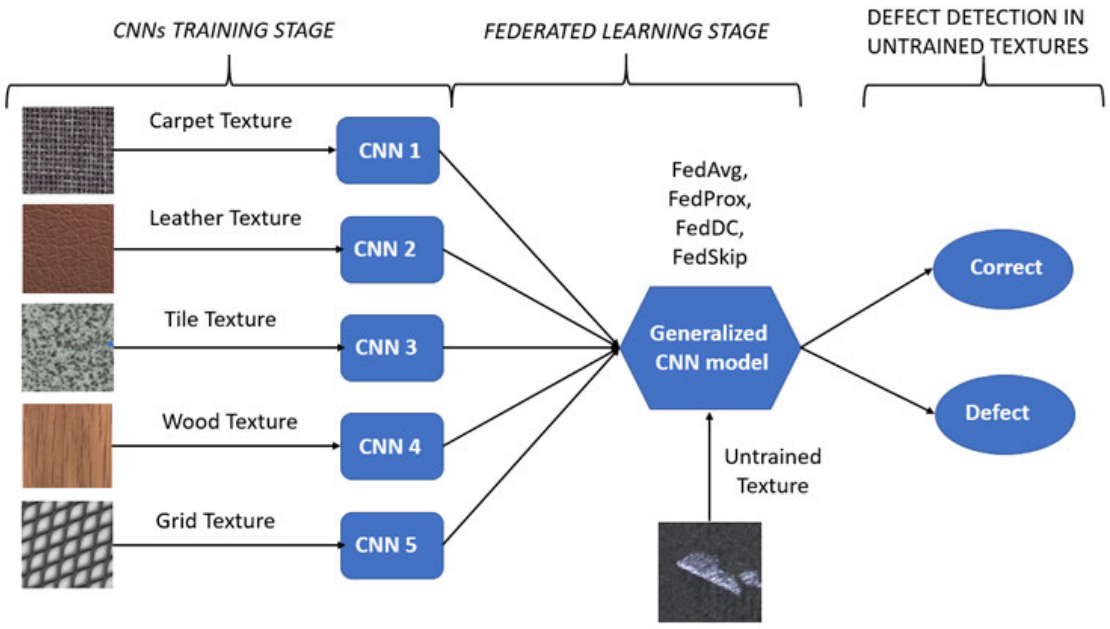


FIGURE 8. Overall schema of the proposed idea.

TABLE 2. Number of images for each class and texture.

	Correct	Defect
Wood	247	60
Tile	230	84
Grid	264	57
Carpet	280	89
Leather	245	92

TABLE 3. Sample of the textures on MVTec dataset.

	Wood	Tile	Grid	Carpet	Leather
Correct					
Defect 1					
Defect 2					
Defect 3					
Defect 4					
Defect 5					

Furthermore, the data reflects a common imbalance situation found in the industry, where the majority of data is defect-free, and access to anomaly data for training purposes is often limited (Table 2).

Table 3 presents examples of fundamental textures employed in this study, are shown in . Except for the tile, which has a size of 840 * 840 pixels, all the images in the dataset for these fundamental textures have a 1024 * 1024 resolution. Furthermore, it is worth nothing that the grid dataset is in grayscale, unlike the other datasets that are provided in RGB format.

In order to simplify the data set, all types of anomalies are reduced to a single class called 'defect' and the correct image group is preserved in the 'correct' class. Thus, the algorithm will have the same outputs, the same labels, but different samples and input features.

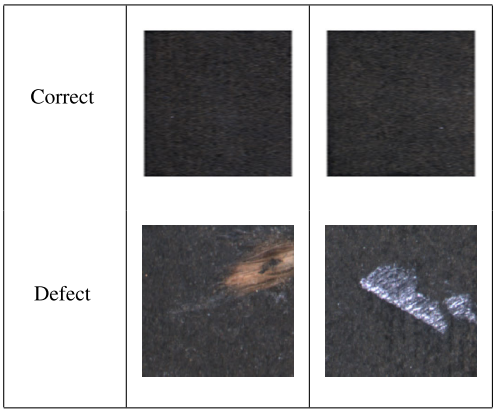
The second dataset used in this research work as a testing dataset is the Kolektor Surface Defects dataset [34].

This publicly available dataset is created with images of defective production items that Kolektor Group submitted and tagged. The image sizes are 230 * 630 pixels. The images were taken in a well regulated industrial setting.

The dataset consists of 356 images with visible defects and, 2979 images without any defects. On the surface of the item, there are several types of defects, such as scratches, minor spots or surface imperfections.

The dataset is divided into training images and model testing images. The training set is composed of 2085 correct and 246 defect images. The testing set of the images has a total number of 894 correct images and 110 defect images. The defective images contain the different errors previously mentioned

TABLE 4. Samples of the kolektor dataset.



This dataset also provides ground truth (GT) images to distinguish between defective and non-defective surfaces, but these labelling information has not been used in this research work

In order to adapt this dataset for the testing phase of this research, all types of anomalies are reduced to a single one called 'defect' and the correct image group is preserved in the 'correct' class (Table 4), as has been done previously in the AD dataset.

B. TRAINING OF CNNS FOR EACH INITIAL TEXTURE

For this study, two state-of-the-art CNN architectures were selected to train the local defect detectors in each of the initial training textures: Squeezenet and Darknet19.

Due to the characteristics of the Anomaly Detection Dataset, and the existing imbalance between the number of correct samples and samples with defects in each of the five textures to be used for the training of the CNNs, data augmentation techniques have been applied to the original dataset samples to obtain a more balanced dataset. More precisely, the transformations applied to the images were as follows:

- Horizontal and vertical translation within a range of Â±30%
- Scaling by increasing or decreasing the original image by 10%
- Image rotation within an angle of ± 90%
- Image reflection in any of its orientations

Finally, the size of the input images in the two datasets has different resolutions, even within the same dataset. In order to standardize the input data, the images of the datasets have been scaled to a final resolution of 224 * 224, the same as preconfigured on the Squeezenet and Darknet19 input layers.

As explained in previous sections, the two previously selected network models will be trained with each of the five textures present in the anomaly detection dataset, obtaining a total of 10 trained CNNs, 5 with Squeezenet architecture and 5 with Darknet19 architecture.

TABLE 5. Hyperparameters of neural networks.

Hyperparameters	Value
Optimizer	sgdm
Initial Learning Rate	3e-4
Mini Batch Size	10
Max number of Epochs	30
Number of Layers (Darknet19)	66
Number of Layers (Squeezenet)	70

TABLE 6. Results of the first training of Squeezenet.

	Carpet	Grid	Leather	Tile	Wood
Initial accuracy	60.0%	50.0%	70.0%	10.0%	0.0%
Final accuracy	97.3%	82.8%	100.0%	73.0%	100.0%
Initial loss	0.8292	0.6784	0.8288	2.0793	2.9490
Final loss	0.0177	0.0182	0.0098	0.0098	0.0014

TABLE 7. Results of the first training of Darknet19.

	Carpet	Grid	Leather	Tile	Wood
Initial accuracy	60.0%	60.0%	60.0%	60.0%	50.0%
Final accuracy	98.7%	98.4%	100.0%	96.8%	100.0%
Initial loss	0.5703	0.8061	0.6886	0.8220	1.1799
Final loss	0.0214	0.0045	0.0002	0.0010	0.0004

For both networks, the stochastic gradient descent (SGD) with momentum optimizer has been selected due to its proven good results when training CNNs for classification [25]. In both cases and according to the literature, the momentum γ has been established to 0.9.

The training parameters of the networks were obtained iteratively, performing different training tests with the networks until the best possible results were obtained for the 5 textures. The table below resumes the main hyper-parameters used to train the neural networks.

The results obtained for this network detecting defects in each of the training textures have been the following. Accuracy metric is defined as:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{7}$$

where:

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

The training parameters for the Darknet19 were established with the same base values. That means, the mini-batch size set to 10 and the learning rate set to 0.0003, obtaining the following results in defect detection and classification:

Figures 9 and 10 show graphically the results summarised respectively in tables 6 and 7. These graphs represent both the accuracy obtained and the evolution of the loss function in the

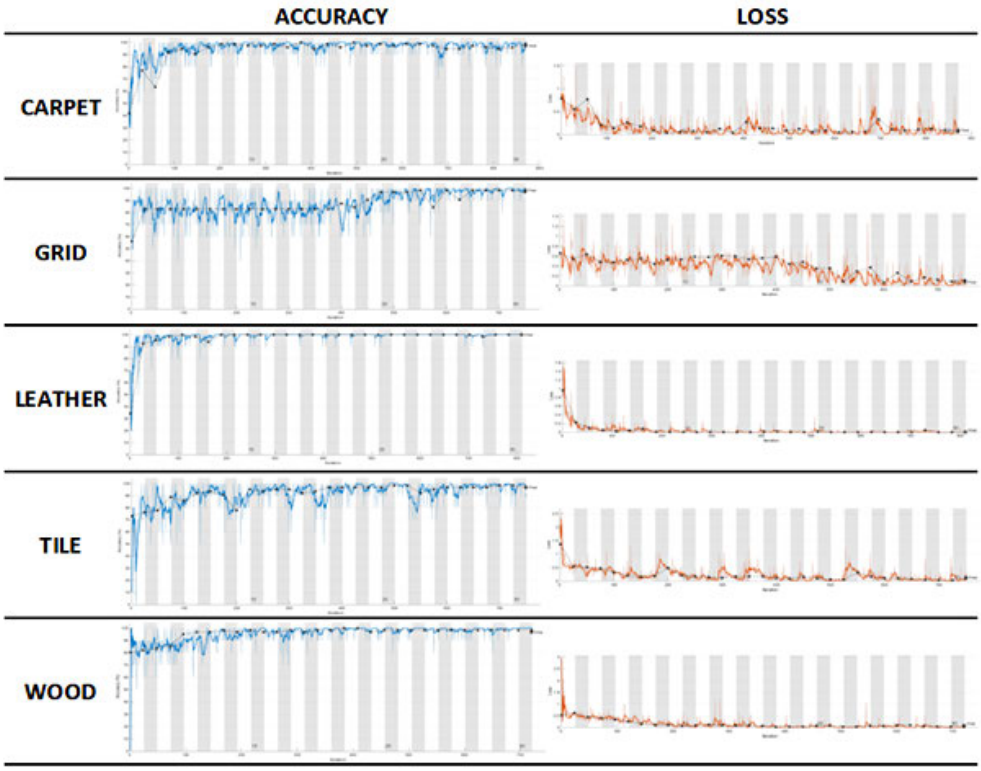


FIGURE 9. Accuracy and loss graphs for the first training of squeezenet.

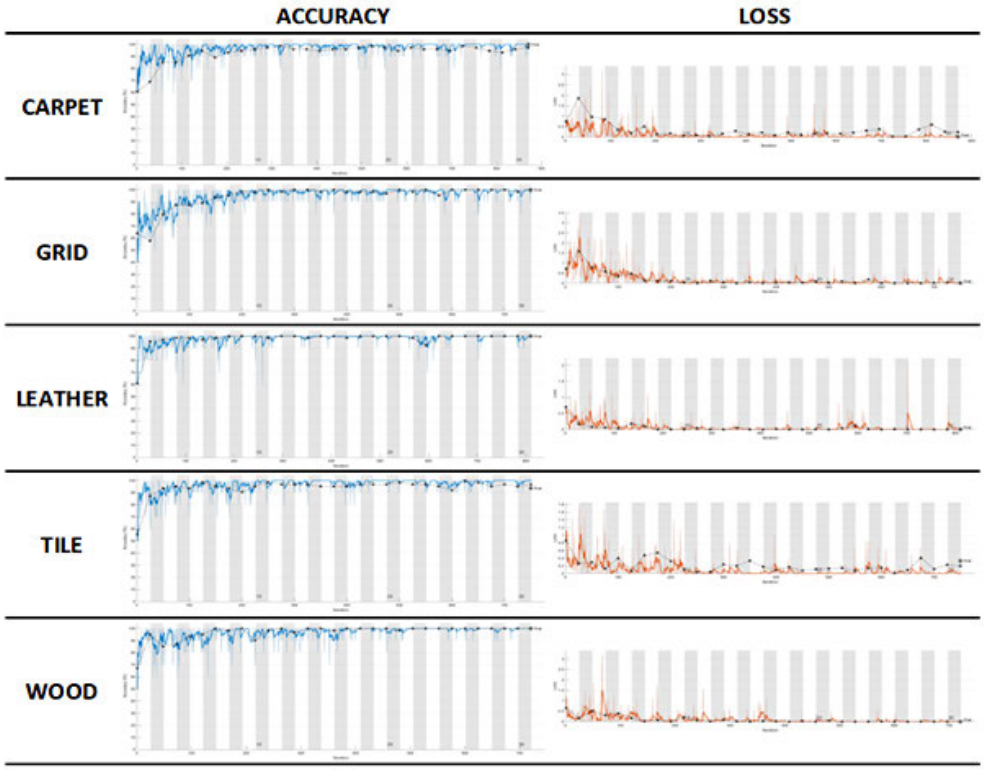


FIGURE 10. Accuracy and loss graphs for the first training of Darknet19.

learning of the different base textures trained independently. As can be seen, the training accuracy for each of the two networks (Squeezenet and Darknet19) is generally very good. However, and more specifically, it can be observed that in the

TABLE 8. Results for Kolektor dataset.

Algorithm	Network	Accuracy	F1 Metric
FedAvg	Squeezenet	84.95%	91.76%
	Darknet19	95.70%	97.62%
FedProx	Squeezenet	88.24%	93.35%
	Darknet19	89.11%	93.83%
FedSkip	Squeezenet	89.25%	94.19%
	Darknet19	95.70%	97.67%
FedDC	Squeezenet	80.84%	88.62%
	Darknet19	82.67%	89.64%

case of the Squeezenet network, the learning of the Grid and Wood textures has been more difficult, requiring more epochs to reach a level of accuracy comparable to that obtained for the rest of the textures. In these cases, the loss function minimises its value much more slowly, which gives an idea at first sight of the difficulty of training these textures. In the case of the CNN Darknet19, as this network is structurally larger, the accuracy and the loss function obtained in each of the textures are somewhat better than those obtained with the Squeezenet network. In the same way as with the previous network, it can be seen that the Grid structure is the one that presents the greatest difficulties. Finally, once the two trained network architectures for the different textures have been obtained, the Federated Learning algorithms will be applied on them, in order to obtain two models that generalise the detection of defects on different textures and surfaces.

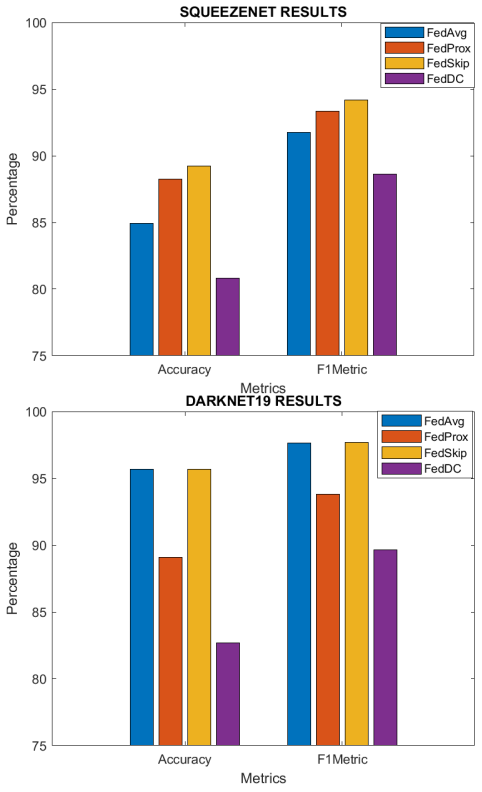


FIGURE 11. Results for kolektor dataset.

C. USE OF FEDERATED LEARNING ALGORITHMS TO GENERALISE CNN MODELS FOR NEW TEXTURES

For the implementation of FedAvg it is not necessary to adjust any special parameters. The only feature that can change is the percentage given by the number of local samples out of the total number of samples when there is a difference in the number of workers selected. This percentage is then used to get the weighted average for the global model.

FedProx uses a proximal term, μ, for improving global convergence. During the training, various values of μ were tested. After observing the responses, it was determined empirically that μ value that better fits Squeezenet is 0.75 and for Darknet is 0.3. In this research, every worker in the FL algorithm has the same adaption, so it is used the same value for all of them.

In the case of FedDC, it has only one parameter that adjusts the penalized term that is added to the loss local function of each worker. To determine this value of α in [35], it is stated that the typical value is a very low value between 0.005 and 0.1. Empirically, it was determined that the value that generates the best results is 0.1.

FedSkip [29] employs the same process as FedAvg. During the local training step, clients receive a model from the server and update it on their own data. Then, they submit the revised model to the server. However, FedSkip has a different treatment in a special step $t = I_{skip}$, when the server skips the aggregation and distributes these client models to different clients via shuffle. As a result, at the t^{th} step, the k^{th} client model that takes part in the training will receive a different initialization based on the stage.

Training the data with the FedSkip algorithm does not require specific parameters. It is enough to identify the skip step before starting the training. Therefore, the process of training starts by browsing the list of data textures, training the model with each texture, and adding the new weights to the FedAvg function. When the index of the loop reaches the skip step $t = I_{skip}$, the training is skipped and the list of textures is shuffled.

D. RESULTS

In this section, the results of testing the Kolektor Surface-Defect dataset [34] are presented. The model is trained by calculating the federated learning function of weights generated after training with the following textures: leather, wood, tile, grid, and carpet. The federated learning algorithms is applied by calculating the FedAvg, FedSkip, FedProx and FedDC functions. These functions have been applied to darkNet19 and SqueezeNet network architectures previously trained independently for each texture, as show in the previous section.

The efficiency of the algorithm is tested by measuring the accuracy, previously defined in 7 and the F1 metrics.
F1 metric can be defined as follows, taking into account that this research is a bi-class classification problem (defect,

correct):

$$F_1 Metric = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{8}$$

where, again:

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

The table below summarizes the results of both metrics, after generalizing both Squeezenet and Darknet models with four federated learning algorithms (FedAvg, FedProx, FedSkip, and Fed DC).

V. CONCLUSION AND FUTURE WORK

This research work demonstrates the advantages offered by Federated Learning algorithms when creating classifiers or artificial intelligence systems in the absence of a quality dataset associated with the problem to be solved. As has been demonstrated in the different tests that have been carried out, it is possible to obtain defect classifiers in industrial environments based on images, without having a specific dataset for the problem in question, obtaining error detection results that are equivalent, if not superior, to current systems that need to have a dataset with enough quality. This is the main contribution of this work.

In this work, this analysis has been carried out on quality control systems based on CNNs, but it is estimated that the results obtained in this type of classifiers could be extrapolated to other AI systems, provided that the previously proposed conditions of similarity in the problem to be solved are met.

In this research, special emphasis has been put on demonstrating the validation of the selected federated learning algorithms (FedAvg, FedProx, FedSkip and FedDC) for the generalisation of error detection models based on convolutional neural networks. However, in both fields, there are many combinations of algorithms available beyond those presented in this paper. In industrial detection tasks, there are many algorithms already established in the state of the art, either based on neural networks or on other approaches, such as support vector machines, statistical classifiers (Bayes), etc. On the other hand, federated learning techniques are in continuous evolution, with new algorithms being created based on different distributed or centralised architectures, and taking into account data privacy as something fundamental in their implementation. Therefore, depending on the problem to be solved, numerous combinations of federated learning techniques can be used to generalise machine learning algorithms. The optimal combination of these techniques will depend mainly on the typology of the problem to be solved and the sensitivity of the data to be handled, therefore this path can open different interesting research paths, which can be considered in future work related to the case study presented here.

REFERENCES

[1] S. R. Saufi, Z. A. B. Ahmad, M. S. Leong, and M. H. Lim, "Challenges and opportunities of deep learning models for machinery fault detection and diagnosis: A review," *IEEE Access*, vol. 7, pp. 122644–122662, 2019, doi: 10.1109/ACCESS.2019.2938227.

[2] H. Wang, S. Li, L. Song, and L. Cui, "A novel convolutional neural network based fault recognition method via image fusion of multi-vibration-signals," *Comput. Ind.*, vol. 105, pp. 182–190, Feb. 2019, doi: 10.1016/j.compind.2018.12.013.

[3] L. Eren, T. Ince, and S. Kiranyaz, "A generic intelligent bearing fault diagnosis system using compact adaptive 1D CNN classifier," *J. Signal Process. Syst.*, vol. 91, no. 2, pp. 179–189, Feb. 2019, doi: 10.1007/s11265-018-1378-3.

[4] L. Wen, L. Gao, and X. Li, "A new deep transfer learning based on sparse auto-encoder for fault diagnosis," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 1, pp. 136–144, Jan. 2019, doi: 10.1109/TSMC.2017.2754287.

[5] L. Wang, Z. Zhang, H. Long, J. Xu, and R. Liu, "Wind turbine gearbox failure identification with deep neural networks," *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1360–1368, Jun. 2017, doi: 10.1109/TII.2016.2607179.

[6] P. K. Wong, J. Zhong, Z. Yang, and C. M. Vong, "Sparse Bayesian extreme learning committee machine for engine simultaneous fault diagnosis," *Neurocomputing*, vol. 174, pp. 331–343, Jan. 2016, doi: 10.1016/j.neucom.2015.02.097.

[7] X. Yu, F. Dong, E. Ding, S. Wu, and C. Fan, "Rolling bearing fault diagnosis using modified LFDA and EMD with sensitive feature selection," *IEEE Access*, vol. 6, pp. 3715–3730, 2018, doi: 10.1109/ACCESS.2017.2773460.

[8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12 pp. 2493–2537, Aug. 2011.

[9] N. Wahab, A. Khan, and Y. S. Lee, "Two-phase deep convolutional neural network for reducing class skewness in histopathological images based breast cancer detection," *Comput. Biol. Med.*, vol. 85, pp. 86–97, Jun. 2017, doi: 10.1016/j.compbiomed.2017.04.012.

[10] H. A. Pierson and M. S. Gashler, "Deep learning in robotics: A review of recent research," *Adv. Robot.*, vol. 31, no. 16, pp. 821–835, Aug. 2017, doi: 10.1080/01691864.2017.1365009.

[11] A. M. Karim, M. S. Güzel, M. R. Tolun, H. Kaya, and F. V. Çelebi, "A new generalized deep learning framework combining sparse autoencoder and Taguchi method for novel data classification and processing," *Math. Problems Eng.*, vol. 2018, pp. 1–13, Jun. 2018, doi: 10.1155/2018/3145947.

[12] G. Qiu, Y. Gu, and Q. Cai, "A deep convolutional neural networks model for intelligent fault diagnosis of a gearbox under different operational conditions," *Measurement*, vol. 145, pp. 94–107, Oct. 2019, doi: 10.1016/j.measurement.2019.05.057.

[13] T. Mezair, Y. Djenouri, A. Belhadi, G. Srivastava, and J. C.-W. Lin, "A sustainable deep learning framework for fault detection in 6G industry 4.0 heterogeneous data environments," *Comput. Commun.*, vol. 187, pp. 164–171, Apr. 2022, doi: 10.1016/j.comcom.2022.02.010.

[14] B. A. Tama, M. Vania, S. Lee, and S. Lim, "Recent advances in the application of deep learning for fault diagnosis of rotating machinery using vibration signals," *Artif. Intell. Rev.*, vol. 56, no. 5, pp. 4667–4709, Oct. 2022, doi: 10.1007/s10462-022-10293-3.

[15] N. K. Chanda and Y. Fu, "ANN-based fault classification and location in MVDC shipboard power systems," in *Proc. North Amer. Power Symp.*, Boston, MA, USA, Aug. 2011, pp. 1–7, doi: 10.1109/NAPS.2011.6025172.

[16] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *Proc. Int. Conf. Inf. Process. Med. Imag.*, Boone, NC, USA, Jun. 2017, pp. 146–157, doi: 10.1007/978-3-319-59050-9_12.

[17] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger, "Improving unsupervised defect segmentation by applying structural similarity to autoencoders," in *Proc. 14th Int. Joint Conf. Comput. Vis., Imag. Comput. Graph. Theory Appl.*, Prague, Czech Republic, Feb. 2019, pp. 372–380, doi: 10.5220/0007364503720380.

[18] M. S. Minhas and J. Zelek, "Semi-supervised anomaly detection using autoencoders," Nov. 2020, *arXiv:2001.03674*.

[19] B. Staar, M. Lütjen, and M. Freitag, "Anomaly detection with convolutional neural networks for industrial surface inspection," *Proc. CIRP*, vol. 79, pp. 484–489, Jan. 2019, doi: 10.1016/j.procir.2019.02.123.

[20] S. Maldonado and J. López, "Dealing with high-dimensional class-imbalanced datasets: Embedded feature selection for SVM classification," *Appl. Soft Comput.*, vol. 67, pp. 94–105, Jun. 2018, doi: 10.1016/j.asoc.2018.02.051.

[21] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, 2021, doi: 10.1561/2200000083.

[22] G. Liu, W. Shen, L. Gao, and A. Kusiak, "Active federated transfer algorithm based on broad learning for fault diagnosis," *Measurement*, vol. 208, Feb. 2023, Art. no. 112452, doi: 10.1016/j.measurement.2023.112452.

[23] W. Y. B. Lim, J. S. Ng, Z. Xiong, J. Jin, Y. Zhang, D. Niyato, C. Leung, and C. Miao, "Decentralized edge intelligence: A dynamic resource allocation framework for hierarchical federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 3, pp. 536–550, Mar. 2022, doi: 10.1109/TPDS.2021.3096076.

[24] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, Jan. 2019, doi: 10.1145/3298981.

[25] S. Ruder, "An overview of gradient descent optimization algorithms," Insight Centre Data Anal., NUI Galway, Connacht, Ireland, Tech. Rep., Jun. 2017.

[26] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, Apr. 2017, pp. 1273–1282.

[27] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Proc. 34th Int. Conf. NeurIPS*, Red Hook, NY, USA, Dec. 2020, pp. 7611–7623, doi: 10.5555/3495724.3496362.

[28] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proc. MLSys*, vol. 2, Mar. 2020, pp. 429–450. [Online]. Available: https://proceedings.mlsys.org/paper_files/paper/2020

[29] Z. Fan, Y. Wang, J. Yao, L. Lyu, Y. Zhang, and Q. Tian, "FedSkip: Combatting statistical heterogeneity with federated skip aggregation," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Orlando, FL, USA, Nov. 2022, pp. 131–140, doi: 10.1109/ICDM54844.2022.00023.

[30] L. Gao, H. Fu, L. Li, Y. Chen, M. Xu, and C.-Z. Xu, "FedDC: Federated learning with non-IID data via local drift decoupling and correction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, New Orleans, LA, USA, Jun. 2022, pp. 10102–10111, doi: 10.1109/CVPR52688.2022.00987.

[31] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and <0.5 MB model size," in *Proc. Int. Conf. Learn. Represent.*, Toulon, France, Feb. 2016, pp. 1–13.

[32] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 6517–6525. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2017/html/Redmon_YOLO9000_Better_Faster_CVPR_2017_paper.html

[33] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "MVTec AD—A comprehensive real-world dataset for unsupervised anomaly detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 9584–9592, doi: 10.1109/CVPR.2019.00982.

[34] J. Božič, D. Tabernik, and D. Skočaj, "Mixed supervision for surface-defect detection: From weakly to fully supervised learning," *Comput. Ind.*, vol. 129, Aug. 2021, Art. no. 103459, doi: 10.1016/j.compind.2021.103459.

[35] L. Gao, H. Fu, L. Li, Y. Chen, M. Xu, and C.-Z. Xu, "FedDC: Federated learning with non-iid data via local drift decoupling and correction," Tech. Rep., 2022, vol. 28.

E. ZULUAGA received the B.S. degree in process and products innovation engineering from the University of the Basque Country, in 2022. He is currently pursuing the M.Sc. degree in electronic, automation, and industrial control with the University of Deusto, while actively collaborating on research in the field of federated learning algorithms as part of his internship with the D4K Research Group, DeustoTech.

S. JAZIRI received the M.Sc. degree in intelligent complex systems from the Polytechnic School of Tunis, in 2022. The master's thesis was based on learning assistance strategies for exoskeleton robots using multi-task reinforcement learning. She joins the Research Group D4K as a Research Assistant for the Elkartek EGIA Project, intending to further her academic pursuit by pursuing a doctoral thesis. Her primary responsibilities involve conducting research and developing solutions in the field of artificial intelligence applied to industry, with a particular emphasis on reinforcement learning and federated learning.

A. TELLAECHE received the M.Sc. degree in automation and electronics engineering from the University of Deusto, in 2001, and the Ph.D. degree (Hons.) in systems engineering and automation from the Computer Science Faculty, UNED, in 2008, with a focus on machine vision and pattern recognition techniques.

He has more than 17 years of experience in industrial companies, the last 12 years with Tekniker Foundation as a Researcher with the Smart and Autonomous Systems Unit, working in machine vision, robotics, industrial computing, and pattern recognition systems. He has also been a part-time Lecturer with the Department of Computer Science, Electronics and Communication Technologies, Teaching Machine Vision and Robotics, University of Deusto, since 2010, where he has been a full-time Lecturer, since 2019. He has authored several articles in peer-reviewed international journals and contributions to several congresses related to his field of expertise and an H-index of 15 with more than 1300 citations according to Google Scholar. He has also participated in many research projects in calls, such as European FP7 and H2020, and in national calls.

I. PASTOR-LÓPEZ received the degree in computer engineering, in 2007, the master's degree in information security, in 2010, and the Ph.D. degree (cum laude) in computer science, in 2013. He participated in the Program in Big Data and Business Intelligence, in 2016. He is currently with Deusto University, where he is focusing on big data analytics, opinion mining, and computer vision. He is the author of several scientific papers reviewed by peers in conferences and indexed journals. He has participated in the gestation, scientific development, and technical development of numerous competitive projects and contracts with companies, the latter with several successful cases of knowledge transfer actions. In addition, he is a member of the Scientific Committee of several congresses, such as CISIS, SOCO, and ICEUTE. He is a Reviewer of many journals, including the *Journal Citation Reports* (JCR) as the Magazine of Engineering and Industry–DYNA.