An MLOps Framework for GAN-based Fault Detection in Bonfiglioli's EVO Plant

Simon Dahdal, *Student Member, IEEE*, Lorenzo Colombi, Matteo Brina, Alessandro Gilli, Mauro Tortonesi, *Member, IEEE*, Massimiliano Vignoli, and Cesare Stefanelli, *Member, IEEE*

***Abstract*—In Industry 5.0, the scarcity of data on defective components in smart manufacturing leads to imbalanced datasets. This imbalance poses a significant challenge to the development of robust Machine Learning (ML) models, which typically require a rich variety of data for effective training. The imbalance not only restricts the models' accuracy but also their applicability in diverse industrial scenarios. To tackle this issue, our research delves into the capabilities of Deep Generative Models, with a special focus on Generative Adversarial Networks, for the generation of synthetic data. This approach is aimed at rectifying dataset imbalances, thereby enhancing the training process of ML models. We demonstrate how synthetic data can substantially bolster the performance and reliability of ML models in industrial settings. Furthermore, the paper presents an innovative MLOps pipeline and architecture, meticulously designed to incorporate Deep Generative Models (DGMs) into the entire ML development cycle. This solution is automated and goes beyond mere automation; it is self-optimizing and capable of making necessary corrections, specifically engineered to address the dual challenges of data imbalance and scarcity, thus enabling more precise and dependable ML applications in smart manufacturing.**

***Index Terms*—Deep Generative Models, MLOps, Generative Adversarial Network, Industry 5.0, Synthetic Data Generation, Imbalanced Datasets.**

I. INTRODUCTION

THE integration of Machine Learning (ML) in industrial environments leads to increased operational efficiency and drives innovation in manufacturing processes and product development. This synergy of ML and big data is at the heart of Industry 5.0 [1]–[4], marking a shift towards more intelligent, adaptable, and sustainable manufacturing [5]. Maximizing the potential of ML in industry, however, necessitates addressing the unique challenges inherent in these settings.

In fact, *imbalanced datasets* – stemming from the scarcity of data on faulty components, due to the high yield of manufacturing processes [6] – *present a significant challenge for the effective adoption of ML in Industry 5.0 applications*. Most ML models assume the availability of extensive and varied datasets but the imbalance often results in models that are both limited and biased [7]. While Chaos Engineering, more and more used in IT, is starting to be proposed in industrial environments [8], manufacturing companies cannot be expected to address this issue by purposely inducing component failures or operational disruptions for the sake of data generation – an understandably costly and risky endeavor. Instead, a more promising solution lies in the augmentation of datasets through the generation of synthetic faulty component data with Deep Generative Models (DGMs) [9].

DGMs have seen remarkable advances in recent years, and have revolutionized the field of generative AI by effectively learning to capture complex data distributions and implementing high-fidelity sampling from them. In this context, most modern approaches leveraging diffusion-based and flow-based models have almost overnight become the most relevant approaches in many use cases – starting from image and video generation. However, *Generative Adversarial Networks (GANs)* arguably remain the best approach for the generation of tabular data [10]–[12], both in terms of sample fidelity and of performance at the training and at the inference levels.

In addition, since manufacturing processes exhibit characteristics that change over time, ML models need to be continuously re-evaluated and periodically re-trained. The deployment of a newly trained GAN typically results in enhanced performance capabilities, necessitating the retraining of the associated ML model with data generated by the improved fresh GAN. This requirement underscores the need for a more effective process management approach. By ensuring that the ML model is trained with the most recent data, its accuracy and applicability to the current manufacturing context are maintained, thereby aligning with the evolving industrial demands. On top of that, the progression of any ML project from a proof of concept to production is often impeded by the absence of DevOps and MLOps expertise [13].

This paper investigates the adoption of GAN-based synthetic data generation and the realization of a robust MLOps platform in the context of a real and challenging industrial use case. We present a comprehensive solution specifically designed for the gearbox assembly and testing line of the Bonfiglioli EVO plant, but broadly applicable to any real-world industrial use case. Our solution improves Bonfiglioli's production line by introducing an ML-based pre-testing phase, that serves as an early filter to identify defects, thereby conserving resources and energy that would be otherwise used in the expensive testing phase for likely faulty gearboxes. By leveraging GAN-based synthetic generation of faulty gearbox data, our solution overcomes the low-performance issues typically exhibited by classifiers trained with imbalanced datasets. This would lead to significant cost savings other than improving time efficiency and throughput of the assembly line.

More specifically, the paper demonstrates the effectiveness of DGMs, and GANs in particular, in overcoming challenges associated with data scarcity and imbalance through synthetic data generation. We achieved significant improvements in

S. Dahdal, L. Colombi, M. Brina, A. Gilli, C. Stefanelli, and M. Tortonesi are with University of Ferrara, Italy (E-mail: {simon.dahdal, lorenzo.colombi, matteo.brina, alessandro.gilli, cesare.stefanelli, mauro.tortonesi}@unife.it)

M. Vignoli is with Bonfiglioli, Calderara di Reno, Bologna, Italy (E-mail: massimiliano.vignoli@bonfiglioli.com)

DOI: 10.36244/ICJ.2024.2.1

classifier performance using Wasserstein GANs (WGANs) and Conditional GANs (CTGANs). Compared to the baseline classifier trained on the original dataset, these approaches yielded a 6-point increase in F1-Score and a nearly 12-point improvement in F2-Score. The larger improvement in F2-Score is particularly important for our task of detecting broken gearboxes, as it reduces False Negatives. At the same time, our MLOps platform based on KubeFlow and KServe is capable of serving 200 requests per second thus satisfying – and significantly surpassing – the performance requirements of the use case for the performance of trained ML models.

II. Background and Related Work

In the last decade, data has been seen as the key driver of progress across various industries, emphasizing its role in sparking new ideas, insights, products, and better decision-making. This aligns with the vision of *Clive Humby's*, who stated in 2006 that "data is the new fuel," highlighting the need for vast amounts of high-quality information for progress and efficiency. Additionally, thanks to the insights gained from recent deep learning approaches, the extensive data collected from industrial plant sensors has emerged as the predominant driving force. However, as contemporary data-driven and ML-powered applications begin to emerge in various industries the reliance on substantial, high-quality data is essential. Challenges such as data incompleteness, poor quality, and inadequate quantity can pose significant obstacles [14]. To address this issue various solutions can be found in literature, including data-generation techniques [15].

A. Generative Adversarial Networks (GANs)

Generative AI has been demonstrated to be a revolutionary field, especially with the introduction of Deep Generative Models (DGM). These state-of-the-art models, which originate from the fusion of generative algorithms and deep learning, have the great capabilities to generate new, realistic data samples that replicate the features and patterns of the training data, called synthetic data. There are several architectural designs of DGMs, such as GANs and Variational Autoencoders, Energy-based models, Autoregressive models, and Diffusion models. They stand out with their ability to generate data that closely mirrors real-world data distributions. In essence, the objective of training DGM is to grasp an unknown probability distribution from a typically limited number of independent and identically distributed samples. Upon successful training, DGM can be employed to assess the likelihood of a given sample and generate new samples resembling those from the unknown distribution. [12]

In recent years, the popularity of Diffusion models has surged due to their ability to generate samples of excellent quality, especially in image generation. However, GANs still remain the reference solution for generating tabular data [10]–[12]. Initially introduced by Goodfellow et al. [16] leveraging game theory concepts, GANs are based on 2 networks the Generator and the Discriminator. The generator learns the data distribution through unsupervised learning to produce authentic adversarial samples. Simultaneously, the Discriminator distinguishes between real and synthetic (generated) samples. The learning process involves iterative updates to both the generator and the discriminator. The generator function produces samples from noise input, while the discriminator tries to distinguish real samples from the synthetic samples.

The peculiar architecture of GANs allows them to efficiently generate data that closely mimic real examples [17]. The use of a neural network to model the loss function allows GAN-based models to have a lower number of parameters compared to other DGMs, with significant advantages in terms of higher data sampling speed and low training time [9]. The GAN capability to generate data concurrently, rather than sequentially, also enhances their speed, making them more applicable and effective in real-world scenarios. The lower computational demand and high efficiency of GANs make them a powerful and efficient tool for creating high-quality data, that is fundamental in Industry 5.0.

Many types of GANs have been introduced in literature such as Deep Convolutional GAN, Conditional GAN, Pix2Pix GAN, Cycle GAN, and others [17]. One of the most interesting variants is the *Wasserstein GAN (WGAN)*, introduced by Arjovsky et al. in 2017 [18]. WGAN proposes a new cost function using the Wasserstein distance, also known as the Earth mover's distance, which is used to measure the distance between two probability distributions. This makes the training of WGANs generally much more stable than that of traditional GANs and significantly reduces the occurrence of the mode collapse phenomenon that usually makes GANs overfit on a specific class of data and therefore, preventing it from generating samples that belong to the targeted class. The Wasserstein loss also provides a more meaningful measure of convergence and thus more useful insights into generator performance. Structurally, the network remains largely the same, except that the activation function of the discriminator's output layer is replaced with a linear function instead of a Sigmoid function.

In addition to WGAN, another GAN-based generative model that has shown great potential is the *Conditional Tabular GAN* (CTGAN) [19], which is specifically designed to generate synthetic tabular data. CTGANs are also designed to reduce the mode collapse phenomena [20]. For this reason, CTGAN is particularly suited for use cases affected by data scarcity. CTGAN innovatively introduces a 'mode-specific normalization' technique for processing continuous features. This approach involves treating each feature independently. Firstly, a variational Gaussian mixture model is fitted to the feature. Subsequently, normalization of each value within the feature is performed using the mean and variance of the corresponding Gaussian component from the mixture. Furthermore, CTGAN modifies the GAN's loss function to enhance the generation of categorical features. This modification includes an additional term in the loss function: the cross-entropy between the one-hot encoding of the input and that of the generated data. This adjustment aids in conditioning the output of the categorical features, ensuring higher fidelity in the synthesized data. Another novelty introduced by the CTGAN is the Training-by-Sampling process, focusing on replicating the original dataset's feature distribution through strategic data sampling and conditional vector construction [20].

However, the presence of misclassified examples in the training set could impact the CTGAN generation performance. Since CTGANs are designed to condition the generator's output, their process can be adversely affected by biases and inaccuracies stemming from these misclassified data points in the original dataset. As a result, CTGANs may learn to generate an increased number of false positives and negatives. This highlights the importance of clean and accurately classified training data for the effective functioning of CTGANs.

B. Machine Learning Operations (MLOps)

MLOps is a relatively new discipline that emerged when machine learning models began to be deployed in production environments. It can be seen as a specialization of DevOps, which focuses on software development principles and practices. However, MLOps is specifically oriented towards the automation of ML workflows. [21]. A cardinal aspect is automation where possible because automating an end-to-end ML pipeline helps avoid manual tasks and reduces delays [22]. However, there also could be some manual steps and tasks to be done. For example, when the model performance goes down, below a certain threshold, a retraining process is triggered and as a result, a specialized ML engineer could decide whether to approve the update for the deployment. Another crucial MLOps aspect is the guarantee of reproducibility and repeatability of the experiments other than versioning of the models, code, and data [21].

In literature it is possible to find various works on the ML life-cycles [23], [24] but some activities are consistently mentioned. These common activities include data engineering, model engineering, operations, and support tools, as detailed by Faubel et al. [25]. Firstly, data engineering comprises activities that are exclusively related to data like data collection, analysis, and preparation. Secondly, model engineering refers to all the steps that serve to create a model. This step includes activities like model building, training, evaluation, selection, and packaging. All the activities mentioned before can be either performed manually or in an automated way. The operations part aims to maintain the quality of the model in a real-world scenario. Operations include continuous integration and continuous deployment (CI/CD), model testing, deployment, and monitoring. Lastly, support tools provide services like versioning, infrastructure management, and automation. Infrastructure management is a key component of an MLOps system and must address various challenges. These challenges include the heterogeneity of both hardware (CPUs, GPUs, etc.) and software (operating systems, ML modules, libraries, etc.), the lack of standards due to costly legacy machines and their inability to communicate with newer systems, resource management, which involves the efficient allocation and utilization of computational resources to optimize performance and cost, and scalability, as the infrastructure must expand to accommodate the increasing size and complexity of ML models [25].

In greater detail, an MLOps pipeline necessitates various technical components, as detailed in work by Kreuzberger et al. [21]. This comprehensive list includes certain ML-related components such as a feature store, a model registry, and a metadata store. Moreover, to adhere to CI/CD principles, a source code repository and a CI/CD component are required. Furthermore, to enable automated workflow (pipeline) an orchestration component is essential. Lastly, it is usually needed to include a model serving component and an infrastructure orchestrator. In particular, the infrastructure orchestrator plays a crucial role in providing the required computation resources. Depending on the level of automation of an MLOps pipeline application, it can be classified using MLOps maturity models. The most widely used are those developed by Google and Microsoft. Google's model comprises three levels. Starting from 0 denoting no automation, progressing to level 1 with ML pipeline automation, and culminating at level 2, with CI/CD Pipeline automation. [26]. In contrast, the Microsoft model consists of five levels and its structure includes both MLOps and DevOps aspects.

Today, MLOps in Industry 4.0 still is in the initial adoption phase, as highlighted by Faubel et al. in [25] but the development and use of AI in the various branches are increasing exponentially [27]. This is particularly evident when considering small and medium-sized enterprises (SMEs) that have recently embarked on their digital transition journey or find themselves in the middle of a transition phase [4]. Notably, SMEs often struggle with a limited IT workforce and insufficient expertise in the ML sector, as well as limited financial resources to effectively build ML-powered applications. Therefore, it is crucial to choose a suitable strategy. Adopting Google's MLOps level one may introduce unnecessary complexity and a steep learning curve for these businesses. Moreover, the physical nature of the production environment introduces additional challenges and constraints, such as safety concerns as illustrated in [13]. As a pragmatic alternative, SMEs may benefit more from a lower level of automation that prioritizes simplicity, aligning better with their operational capacity and limited skill sets. Considering that SMEs represent 99% of all businesses in the EU as refereed by the "Annual Report on European SMEs 2022/2023" published by the European Union, developing an approachable set of standards and best practices could have a significant societal impact in a relatively short time.

III. BONFIGLIOLI INDUSTRIAL USE CASE

Bonfiglioli (https://bonfiglioli.com) is a leading manufacturing company that designs and manufactures a wide range of gear motors, drive systems, planetary gear motors, reducers, and inverters with over 130 years of experience. Bonfiglioli is increasingly adhering to Industry 5.0 best practices, and implementing efficient and environmentally sustainable processes.

Within the wide range of manufacturing lines that Bonfiglioli is running, and constantly improving, the gearbox assembly and testing line located in the EVO Plant represents a particularly interesting one. This line utilizes sophisticated machinery for automated precision assembly and thorough testing processes. The assembly line is composed of three workstations (WS) as illustrated in Fig. 1. **WS1 - Differential Assembly** is responsible for the assembly of the differential

part of the gearbox. It collects data on insertion forces and tightening torque. **WS2 - Gearbox Assembly** similarly engages in the assembly process, here focusing on the entire gearbox. It gathers data on insertion forces and tightening torque, akin to the WS1 station. **WS3 - End-of-Line Testing** is the last station of the plant, which receives the assembled gearbox and implements the testing phase.

In turn, WS3 consists of two distinct machines. The first one is dedicated to generating cycle data through specific stress tests on the component under examination. This involves conducting two phases of static analysis at 800 revolutions per minute (RPM) and two phases of ramp analysis at 11,000 RPM. These tests are critical for assessing the component's performance under varying operational conditions. Concurrently, a second machine operates to monitor vibrational data. This machine is specifically tasked with tracking vibration levels as various RPM thresholds are surpassed. The data collected from this machine is crucial for understanding the vibrational characteristics of the component under different operational speeds, which is vital for ensuring its reliability and structural integrity.

To further improve the efficiency of the Bonfiglioli EVO gearbox assembly and testing line, the manufacturing process has been extended by integrating a pretesting phase, positioned between WS2 and WS3, that offers substantial benefits in enhanced sustainability and reduced costs. The pretesting phase serves as an early filter to identify defects, thereby conserving resources and energy that would be otherwise used in the more expensive final testing phase. It aids in minimizing waste by detecting quality issues at an earlier stage, allowing for rectification or recycling before extensive testing. Not only do the added early quality checks ensure higher overall product quality, aligning with sustainable manufacturing practices by reducing environmental impact, an aspect increasingly vital in today's market, but they also improve the overall good output of the assembly line. The realization of an effective pretesting phase requires addressing the issue of training an accurate classifier using an imbalanced dataset collected from the WS1 and WS2 machines. At the same time, there is the need to design and develop an MLOps platform capable of deploying, running, monitoring, and managing and updating ML models.

The dataset collected from WS1 and WS2 is centered around two processes: tightening torques and press-fit forces. The roughly 700 metrics contained by the dataset are critical as they directly impact the assembled gearbox's functionalities. An accurate and multi-step evaluation process, conducted in collaboration with domain experts, has led to the identification and prioritization of 66 key features deemed most influential on the gearbox's final performance and readiness. Based on these input features and the output of the tested gearboxes from the WS3 testing machine, we managed to construct the final dataset used to to train the ML pretest classifier.

However, naively training a classifier using the imbalanced dataset led to suboptimal performance. More specifically, using a logistic regression model we obtained a relatively high number of False Negatives, as discussed in Section IV-B. To address this issue, we designed a GAN-based synthetic generator for faulty component data.

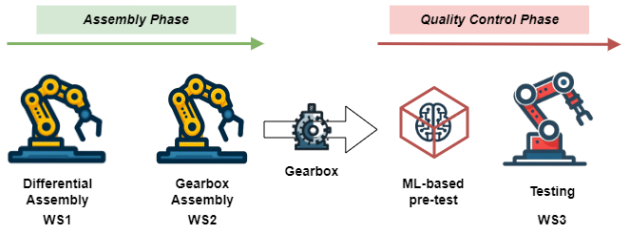


Fig. 1: Bonfiglioli gearbox assembly line at the EVO plant.

A. Design of GAN-based Synthetic Data Generation

Like with all artificial neural network-based solutions, the model architecture represents a critically important factor for the effectiveness of Generative Adversarial Networks (GANs). Model architecture design involves many decisions, starting from the number of layers and the number of neurons in each layer for both the Generator and Discriminator networks, as well as the activation function, the possible adoption of regularization constraints, the tuning of hyperparameters, etc. – all factors that can significantly affect a model's performance. Hence, employing an automated tool for the training process of a GAN can markedly improve its effectiveness, making such a tool an essential aid for practitioners.

In this specific case, Optuna (https://optuna.readthedocs.io/) was employed to systematically search for the optimal configuration of architecture and hyperparameters. For example, aspects such as the number of layers in each network, the number of neurons in each layer, the size of the input noise vector, the number of training epochs, and the size of the training batch are defined using the capabilities of the optimizer. Specifically, Optuna requires upper and lower bounds for the hyperparameter search space and, based on an objective function that needs to be minimized (or maximized), the search for the optimal combination of hyperparameters is done by selecting one of various heuristics given at the disposition of the users of Optuna. The parametric model of the GAN at hand has been specified and passed to Optuna as follows: The generator is composed of an input layer of the same size as the noise, which $\in [10, 500]$, N hidden layers with $N \in [1, 5]$, and many neurons in each layer that $\in [32, 512]$. The activation function chosen for each hidden layer is *LeakyReLU*, to avoid the dying ReLU phenomenon, where neurons become inactive and stop learning, effectively rendering them useless by causing the gradient to become zero. The output layer, on the other hand, has a size equal to the number of features in the dataset, as it is required to produce synthetic output data. The activation function for the output layer is *Tanh*, suitable as the dataset is initially normalized between -1 and 1. As for the optimizer, *RMSProp* has been selected, as often proposed in literature due its adaptive Learning Rate, Convergence Speed and stability even on very nonstationary problems [18]. Regarding the discriminator, the only changes are in the input layer, which has a size equal to the number of features in the data, as its function is to receive input data and decide whether it is synthetic or not. The output layer has a size of 1 and utilizes a linear activation function. Regarding the training,

a batch training methodology has been chosen, with the batch size determined by Optuna.

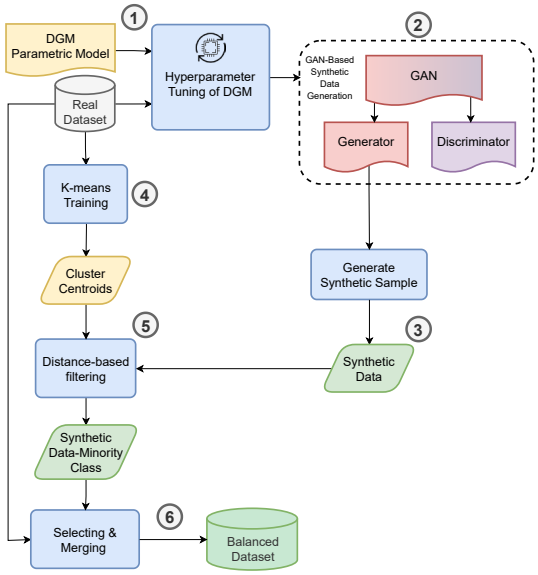


Fig. 2: WGAN Dataset Balancing algorithm.

Fig. 2 illustrates the data augmentation workflow using WGAN. The goal was to generate examples belonging to the minority class. For this reason, the workflow includes WGAN training, data generation and filtering steps. The latter is needed due to the presence of mislabeled examples in the training set and the WGAN's poor performance in generating categorical data. In detail, the steps are the following:

1) From the parametrically defined model, hyperparameter optimization is performed, minimizing the objective function.
2) The WGAN that achieves the minimum value of the objective function is stored in the Model Registry.
3) The best performing WGAN is used to generate a synthetic dataset much larger than the original one
4) A K-means model is trained on the original dataset to identify the centroids of two clusters, which presumably correspond to the two classes of gearboxes: 'good' and 'broken'.
5) Once the centroid of the minority class is identified, the synthetic data are filtered measuring the Euclidean distance between each example of the minority class centroid. Empirically, it has been found that even a simple approach such as the Euclidean distance has given good results. This process retains at least N example less distant than d_{max} from the centroid. N represents the difference between the number of examples in the majority and minority classes in the original dataset, and d_{max} is the empirically chosen maximum distance a point can have from the centroid to be classified as belonging to the minority class.
6) The synthetic filtered data is used to balance the initial dataset.

Moreover, for comparison purposes, we tested a second straightforward GAN-based model, called. CTGAN. It was configured specifically to generate examples belonging to the minority class. Specifically, were used the Synthetic Data Vault (SDV) [28] CTGAN implementation. SDV is a comprehensive Python library tailored for generating tabular synthetic data and offers various synthesizers including the CTGAN model.

B. Kubeflow

We implemented the MLOps platform on top of Kubeflow (https://kubeflow.org/), an MLOps framework developed and maintained by Google. Kubeflow aims to make it easier for organizations to develop, deploy, and manage ML workloads. Kubeflow includes Katib for hyper-parameters tuning, KServe for model serving, Jupyter Hub, and Kubeflow pipeline. It is possible to interact with these features through a web-based GUI. Due to Kubeflow's architecture, which is micro-services oriented and based on Kubernetes, it can be seamlessly integrated with other software components operating on top of Kubernetes. In addition, Kubeflow can be installed in every cloud or a local single-node Kubernetes cluster.

Using Kubeflow Pipelines (KFP) [29] it becomes possible to build and execute portable and scalable ML workflows. A pipeline is represented as a Directed Acyclical Graph, where each node is a component. At runtime, each component execution corresponds to a single container. Components can be exported for later use, in YAML format. This way, components are highly portable and facilitate code reuse. Pipelines are programmable using KFP SDK that lets easily convert a Python function into a container component using a simple Domain Specific Language. Another way to create a component is by using a custom Docker image, permitting the inclusion of different programming languages in the pipeline. From the pipeline SDK users can define, save, and execute pipelines. This allows triggering execution programmatically, enabling continuous training strategy. Kubeflow, also, provides a simple method to pass data between each step using its artifact store. An artifact is every object created during the execution. For example, an artifact could be a dataset or a set of metrics. For each artifact, metadata information is saved in a MySQL database, while the object is saved in a MinIO object-store. Therefore, the artifact store uses both MySQL and MinIO as backends.

C. MLOps Pipeline Implementation

1) Runtime Environment: For all of these reasons and the features stated above, Kubeflow has been chosen as the MLOps framework for testing deployment. The decision was made to deploy Kubeflow using raw manifests on a Kubernetes single-node cluster. Specifically, the Kubernetes environment chosen was MicroK8s, favored for its straightforward installation process and valuable add-ons. The Kubernetes node utilized was configured with 16 virtual CPUs, 32GB of RAM, and 50GB of storage memory.

2) Kubeflow Pipeline: The ML pipeline is implemented via KFP version 2, leveraging the Python KFP SDK. The pipeline is made of various components, as illustrated in Fig. 3. The

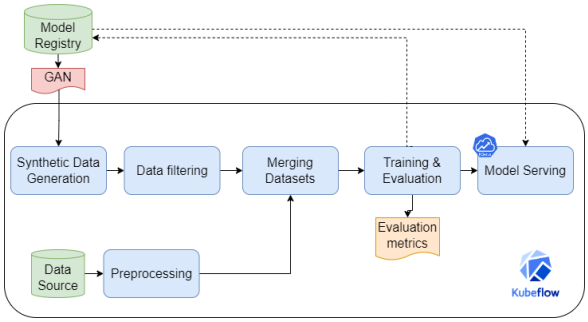
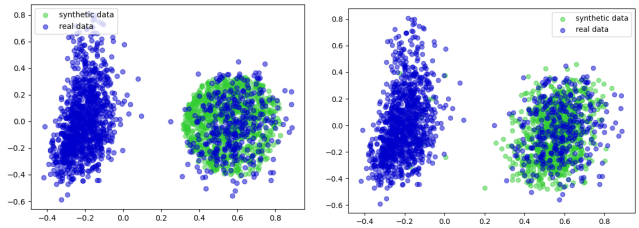


Fig. 3: Kubeflow MLOps pipeline.



(a) WGAN (b) CTGAN

Fig. 4: Original data and synthetic data plotted after the reduction to 2 dimensions with PCA.

initial component retrieves the latest version of the dataset stored in a CSV file from a Kubernetes volume and transfers it to Kubeflow's artifact store for subsequent utilization. This step is also responsible for acquiring the pre-trained GAN. The serialized GAN can be stored in the same volume as the dataset, or a specialized model registry, such as Mlflow. The adoption of a model registry offers many advantages, including decoupling the GAN development processes from the classifier training phase. Subsequently, the dataset undergoes preprocessing where the meaningless features are deleted and inputs are separated from outputs. Concurrently, the newest GAN available in the model registry is retrieved and used to generate a synthetic dataset. The synthetic dataset is filtered to keep only the examples belonging to the minority class and is used to balance the real dataset.

In the subsequent steps, the classifier, specifically a Logistic Regression model (LogReg), is trained on the enriched and balanced dataset. LogReg was specifically chosen for its inherently interpretable nature which is of great importance for manufacturing applications that require thoroughly evaluating and possibly certifying decision-making elements. However, other ML classifiers, including Decision Trees, Support Vector Machines, Gradient Boosting Machines, etc., could be used. After the training the model undergoes evaluation, and metrics are exported for comparative analysis through the Kubeflow dashboard. As a final step, the model is deployed into production using Kubeflow's built-in serving framework: *KServe* (https://kserve.github.io). KServe offers a Kubernetes Custom Resource Definition to enable out-of-the-box deployment of trained models onto various widely used serving runtimes, such as TFServing, TorchServe, Triton, and many others.

A scikit-learn inference service has been deployed using the KServe Python SDK. The predictor is configured with a minimum and a maximum number of replicas set at, respectively, 1 and 10. Additionally, constraints for the predictor pod resources have been defined, limiting it to use 0.5 CPU and 0.5 GB of memory. Finally, as shown in Fig. 3, the model can be saved in the model registry, to prepare it for subsequent utilization. This practice serves several purposes, including tracking the model and facilitating further evaluation. Additionally, the stored model can be deployed into production at a later stage.

IV. Experimental Evaluation

Let us demonstrate how WGAN and CTGAN adoption can improve the performance of the pre-testing phase in the Bonfiglioli assembly line.

A. GANs Performance

Starting with a visual comparison of the synthetic data generated by the two trained GANs. Fig. 4 illustrates this relationship. To create this visualization comparison, Dimensionality Reduction was employed. In detail, the Principal Component Analysis (PCA) algorithm was used to reduce the dimensionality of the dataset from 66 features which were heavily cleaned into 2 dimensions. The PCA has managed to create 2 distinct clusters, the left one is denser presenting the good gearboxes while the right cluster represents the broken gearboxes. For Both WGAN and CTGAN, We can observe the positioning of the **synthetic data** and the **original data**. We can observe that the CTGAN has successfully created a distribution that closely matches the original, dividing the data into two clusters that resemble the original clusters.

Secondly, for a more precise and detailed comparison of the two GANs performances, we employed a range of distance and similarity metrics that best describe data distributions. These metrics were selected due to their superior capability to elucidate the differences in the data distributions created by each GAN, offering a deep and thorough analysis of their respective performances. A brief description of each used metric: Starting with the *Wasserstein distance* (WD) indicating a close similarity between the original and synthetic datasets. A second metric is *Kolmogorov-Smirnov (KS) D statistic*, which represents the maximum difference between two empirical cumulative distribution functions. The larger the D value, the more significant the difference between the two distributions, the actual gearboxes and the synthetic gearboxes. The KS D statistic is based on the KS test which was used to compare the differences in probability distribution features data between the actual gearboxes and the generated synthetic gearboxes. The *KS-complement* (KS-C) was a supplementary index of the KS test, and its value equaled $1 - (D\ statistic)$. The *Correlation Similarity* (CS) has been computed. This metric measures the correlation between a pair of numerical columns and computes the similarity between the real and synthetic data. A score closer to 1.0 shows a perfect pairwise correlation. Finally, we

measured the *Jensen Shannon distance* (JS), an alternative to measure the distance between two probability distributions. All the previously presented distance metrics were applied feature-wise between original and synthetic datasets to then calculate the mean of the scores.

A summary table of compression metric results between the two GANs can be seen in Table I. The results demonstrate a marginally superior performance of the CTGAN compared to the WGAN, confirming the observations made in earlier visual comparison. An important evaluation of the inference time is crucial. The WGAN model has achieved 0.14 seconds to generate 1,000 samples and 0.32 seconds for 10,000 samples. Conversely, the CTGAN model was slower, requiring 0.24 seconds for 1,000 samples and 0.53 seconds for 10,000 samples, possibly due to its distinct architecture. However, both models exhibited swift inference performance.

TABLE I
GANs PERFORMANCE.

	KS-C	CS	WD	JS
WGAN	0.759	0.922	0.024	0.309
CTGAN	0.799	0.931	0.022	0.261

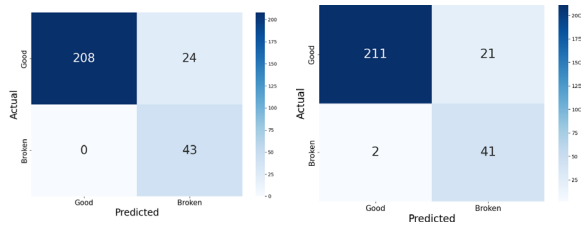
B. Classifier Prediction Performance

The last crucial test was to test how much the LogReg model improved by augmenting the original dataset with the synthetic data. The test was done as follows: Firstly, the cleaned original dataset was split into training and testing sets. Subsequently, the training dataset was augmented with the WGAN using the methodology described earlier (see section III-A). Three LogRegs were trained, one on just the original training set, a second with a WGAN-augmented training dataset, and a third one with the conditioned CTGAN-augmented training set. The confusion matrix of the results is shown in Fig. 5. The number of False Negatives has been reduced to zero in the case of WGAN and near zero in the case of CTGAN, this is due to the presence of mislabeled in the generated synthetic data as explained in the section II-A. More comprehensive and detailed metrics are reported in Table II.

Although CTGAN showed better performance in similarity metrics, WGAN, enhanced with unsupervised filtering, resulted in more effective classifier training. These results are attributed to the process of managing mislabeled data points in the dataset. A data augmentation by CTGAN failed to resolve and, in some cases, might worsen the situation by generating mislabeled data points introducing more biases and errors into the process. The final results showcase the achievement of the goal of reducing the number of incorrectly classified broken gearboxes (False Negative) and improving the performance of the classifier model in terms of key metrics such as Recall, F2 score, and the G-mean. This improvement underscores the enhanced ability of the model to accurately identify and classify gearbox conditions, marking a noteworthy advancement in predictive capabilities. In the light of zero waste tolerance, we can conclude that WGAN fits better in this specific use case, despite CTGAN performing similarly.

TABLE II
LOGREG MODEL PERFORMANCE METRICS [30].

	Original dataset	WGAN augmented	CTGAN augmented
Accuracy	0.90	0.91	**0.92**
Precision	0.65	0.64	**0.66**
Recall	0.81	**1.00**	0.95
F1 Score	0.72	**0.78**	**0.78**
F2 Score	0.77	**0.90**	0.88
G-Mean	0.86	**0.95**	0.93



(a) trained with WGAN balanced augmented dataset

(b) trained with CTGAN balanced augmented dataset

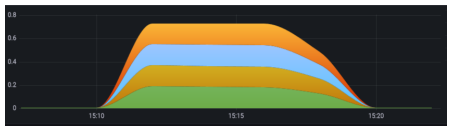
(c) trained with original balanced augmented dataset

Fig. 5: Comparison between confusion matrices of LogReg classifiers trained with different datasets.

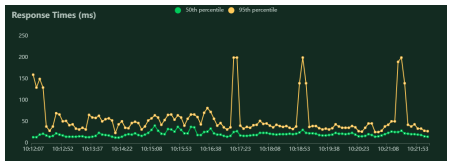
C. Classifier Serving Performance

To assess the performance of KServe, an observability stack has been deployed, specifically, the one integrated into MicroK8s has been used. This stack incorporates Prometheus for metrics scarping and Grafana for visualization. Additionally, an open-source load testing tool, Locust (https://locust.io/), was employed in the evaluation process. Key metrics, such as response time, number of pods, and pod resource consumption, were selected to provide meaningful insight into the performance characteristics of the KServe deployment.

Initially, a Locust test, with a peak of 20 users sending 10 requests per second (RPS), was started. This way, the inference service had to handle, approximately, 200 RPS. While in an idle state, only one pod is instantiated, as the RPS increased, the KServe autoscaler dynamically deployed new pods, ultimately reaching a total of 4 pods. Moreover, analyzing the resource usage revealed an equitable distribution of requests between each pod. Nonetheless, the Locust dashboard indicated that the 95th percentile of requests were served in less than 56ms. This test indicates how Kubernetes-backed KServe can easily scale in response to increasing load, maintaining efficient resource usage, as well as good performance under heavy load conditions.



(a) CPU usage of each Pod.



(b) Response time Locust chart.

Fig. 6: CPU usage and response time charts.

V. CONCLUSIONS & FUTURE WORK

Machine Learning (ML) presents a plethora of compelling and high-impact applications in Industry 5.0, although the end-to-end integration of ML-powered applications in the industrial scenario still presents many challenges. To address those issues, we realized a robust framework that automatically rebalances datasets by incorporating synthetic data generated by GANs, thus allowing to improve the performance of ML model training. Specifically, by employing a WGAN, we succeeded in training a binary classifier that can distinguish between good and broken gearboxes. This framework contributes meaningfully to Industry 5.0, particularly in terms of sustainability and Zero Defect Manufacturing. Furthermore, we realized a robust architectural pipeline for orchestrating the ML model deployment, serving, and update – thus ensuring levels of error resilience that are suited for the industrial environment. While our framework was designed for, and validated on, a real-life manufacturing line – namely the Bonfiglioli EVO gearbox assembly and testing line – it is of wide applicability. Looking ahead, we aim to measure the results of using synthetic data in Industry 5.0, such as estimating cost savings and throughput improvements in the assembly line and the acceleration that can be given to creating next-generation ML-based applications. Additionally, we plan to enhance the lifecycle management of the generated synthetic data, thereby extending the reproducibility of the ML models beyond on-the-fly generation. Finally, we also intend to explore more advanced unsupervised and semi-supervised anomaly detection techniques to identify broken gearboxes.

REFERENCES

[1] R. Venanzi *et al.*, "Enabling adaptive analytics at the edge with the Bi-Rex Big Data platform," *Computers in Industry*, vol. 147, p. 103 876, 2023. DOI: 10.1016/j.compind.2023.103876

[2] A. Frankó *et al.*, "Applied Machine Learning for IIoT and Smart Production Methods to Improve Production Quality, Safety and Sustainability," *Sensors*, vol. 22, no. 23, 2022. DOI: 10.3390/s22239148

[3] E. Jantunen *et al.*, "Maintenance 4.0 World of Integrated Information," in *Enterprise Interoperability VIII*. Springer International Publishing, 2019. DOI: 10.1007/978-3-030-13693-2_6

[4] A. E. Frankó *et al.*, "A Survey on Machine Learning based Smart Maintenance and Quality Control Solutions," *Infocommunications Journal*, vol. 13, no. 4, pp. 28–35, 2021. DOI: 10.36244/ICJ.2021.4.4

[5] P. Varga *et al.*, "Data-Driven Workflow Execution in Service Oriented IoT Architectures," in *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 203–210, 2018. DOI: 10.1109/ETFA.2018.8502665

[6] L. Alzubaidi *et al.*, "A survey on deep learning tools dealing with data scarcity: Definitions, challenges, solutions, tips, and applications," *Journal of Big Data*, vol. 10, no. 1, 2023. DOI: 10.1186/s40537-023-00727-2

[7] T. B. Nyíri *et al.*, "What can we learn from Small Data," *Infocommunications Journal*, vol. 15, no. SI, pp. 27–34, 2023. DOI: 10.36244/ICJ.2023.5.5

[8] M. Fogli *et al.*, "Chaos Engineering for Resilience Assessment of Digital Twins," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 2, pp. 1134–1143, 2024. DOI: 10.1109/TII.2023.3264101

[9] S. Bond-Taylor *et al.*, "Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7327–7347, Nov. 2022. DOI: 10.1109/tpami.2021.3116668

[10] S. De *et al.*, "Deep Generative Models in the Industrial Internet of Things: A Survey," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 9, pp. 5728–5737, 2022. DOI: 10.1109/TII.2022.3155656

[11] D. Ezeh *et al.*, "An SDN Controller-Based Framework for Anomaly Detection Using a GAN Ensemble Algorithm," *Infocommunications Journal,* vol. XV, no. 2, pp. 29–36, June 2023. DOI: 10.36244/ICJ.2023.2.5

[12] L. Ruthotto *et al.*, "An introduction to deep generative modeling," 05 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/gamm.202100008, DOI: 10.1002/gamm.202100008

[13] C. Hegedűs *et al.*, "Tailoring MLOps Techniques for Industry 5.0 Needs," in *2023 19th International Conference on Network and Service Management (CNSM)*, 2023. pp. 1–7. DOI: 10.23919/CNSM59352.2023.10327814

[14] Y.-T. Chen *et al.*, "On the Private Data Synthesis Through Deep Generative Models for Data Scarcity of Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 1, pp. 551–560, 2023. DOI: 10.1109/TII.2021.3133625

[15] Z. Pödör *et al.*, "A Practical Framework to Generate and Manage Synthetic Sensor Data," *Infocommunications Journal*, vol. XIV, no. 2, pp. 64–72, June 2022. DOI: 10.36244/ICJ.2022.2.7

[16] I. J. Goodfellow *et al.*, "*Generative Adversarial Networks*," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[17] Y. Li *et al.*, "The theoretical research of Generative Adversarial Networks: an overview," *Neurocomputing*, vol. 435, pp. 26–41, 2021. DOI: 10.1016/j.neucom.2020.12.114

[18] M. Arjovsky *et al.*, "Wasserstein GAN," 2017. [Online]. Available: https://arxiv.org/abs/1701.07875

[19] L. Xu *et al.*, "Modeling Tabular data using Conditional GAN," 2019. [Online]. Available: https://arxiv.org/abs/1907.00503

[20] O. Habibi *et al.*, "Imbalanced tabular data modelization using CTGAN and machine learning to improve IoT Botnet attacks detection," *Engineering Applications of Artificial Intelligence*, vol. 118, p. 105 669, 2023. DOI: 10.1016/j.engappai.2022.105669

[21] D. Kreuzberger *et al.*, "Machine Learning Operations (MLOps): Overview, Definition, and Architecture," *IEEE Access*, vol. 11, pp. 31 866–31 879, 2023. DOI: 10.1109/ACCESS.2023.3262138

[22] L. Colombi *et al.*, "A machine learning operations platform for streamlined model serving in industry 5.0," in *NOMS 2024-2024 IEEE Network Operations and Management Symposium*, 2024. pp. 1–6. DOI: 10.1109/NOMS59830.2024.10575103

[23] "Practitioners guide to mlops: A framework for continuous delivery and automation of machine learning." [Online]. Available: https://services.google.com/fh/files/misc/practitioners_guide_to_mlops_whitepaper.pdf

[24] R. Subramanya *et al.*, "From DevOps to MLOps: Overview and Application to Electricity Market Forecasting," *Applied Sciences*, vol. 12, no. 19, 2022. DOI: 10.3390/app12199851

[25] L. Faubel *et al.*, "Mlops challenges in industry 4.0," *SN Computer Science*, vol. 4, no. 6, p. 828, 2023. DOI: 10.1007/s42979-023-02282-2

[26] G. Symeonidis *et al*., "MLOps - Definitions, Tools and Challenges," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, 2022. pp. 0453–0460. DOI 10.1109/CCWC54503.2022.9720902

[27] D. Ficzere *et al*., "AI Toolbox Concept for the Arrowhead Framework," in *2023 19th International Conference on Network and Service Management (CNSM)*, 2023. pp. 1–7. DOI: 10.23919/CNSM59352.2023.10327821

[28] N. Patki *et al*., "The synthetic data vault," in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, Oct 2016. pp. 399–410. DOI: 10.1109/DSAA.2016.49

[29] E. Bisong, *Kubeflow and Kubeflow Pipelines*. Berkeley, CA: Apress, 2019, pp. 671–685. ISBN 978-1-4842-4470-8. [Online]. Available: DOI: 10.1007/978-1-4842-4470-8_46

[30] A. Géron, Hands-on Machine Learning with Scikit-Learn, *Keras, and TensorFlow*. O'Reilly, 8 2019.

Simon Dahdal (Student Member, IEEE) is a PhD Student at the Engineering Department of the University of Ferrara, in the Distributed Systems Research Group. He currently works on projects that focus on the application of Big Data and Machine Learning in High-Stakes Environments, in particular Industry 4.0/5.0 and Human Assistance & Disaster Recovery scenarios.

Lorenzo Colombi He is a research assistant at the University of Ferrara, in the Distributed Systems Research Group. He currently works on projects that focus on the application of Big Data, Machine Learning, and Machine Learning Operations in High-Stakes Environments, such as Industry 4.0/5.0 and Human Assistance & Disaster Recovery scenarios.

Matteo Brina He is a research assistant at the University of Ferrara, in the Distributed Systems Research Group. He currently works on projects that focus on the application of Big Data and Machine Learning in Industrial Environments.

Alessandro Gilli He is a research assistant at the University of Ferrara, in the Distributed Systems Research Group. He currently works on projects that focus on the application of Big Data and Machine Learning Operations in both Industry 4.0/5.0 and Human Assistance & Disaster Recovery scenarios.

Mauro Tortonesi (Member, IEEE) is an Associate Professor and the head of the Big Data and Compute Continuum research laboratory at the University of Ferrara, Italy. He received the Ph.D. degree in computer engineering from the University of Ferrara, in 2006. He was a Visiting Scientist with the Florida Institute for Human & Machine Cognition (IHMC), Pensacola, FL, USA, from 2004 to 2005 and with the United States Army Research Laboratory, Adelphi, MD, USA, in 2015. He participates / has participated with several roles in a wide number of research projects in the distributed systems area, with particular reference to Compute Continuum, Big Data, and IoT solutions in industrial and military environments. He has co-authored over 100 publications and has 4 international patents.

Massimiliano Vignoli He is the Chief Data Scientist at Bonfiglioli S.P.A., currently leading several predictive maintenance projects. He has extensive experience in managing the full life cycle of machine learning applications, from development to production.

Cesare Stefanelli (Member, IEEE) received the Ph.D. degree in computer science engineering from the University of Bologna, Italy, in 1996. He is currently a Full Professor of distributed systems with the Engineering Department, University of Ferrara, Italy. At the University of Ferrara he coordinates a Technopole Laboratory dealing with industrial research and technology transfer. He holds several patents, and coordinates industrial research projects carried on in collaboration with several companies. His research interests include distributed and mobile computing in wireless and ad hoc networks, network and systems management, and network security.