Replay-Driven Continual Learning for the Industrial Internet of Things

Sagar Sen
SINTEF Digital
Sustainable Communication Technologies
Oslo, Norway
sagar.sen@sintef.no

Simon Myklebust Nielsen
University of Oslo
Department of Informatics
Oslo, Norway
simonmn@ifi.uio.no

Erik Johannes Husom
SINTEF Digital
Sustainable Communication Technologies
Oslo, Norway
erik.johannes.husom@sintef.no

Arda Goknil
SINTEF Digital
Sustainable Communication Technologies
Oslo, Norway
arda.goknil@sintef.no

Simeon Tverdal
SINTEF Digital
Sustainable Communication Technologies
Oslo, Norway
simeon.tverdal@sintef.no

Leonardo Sastoque Pinilla
University of the Basque Country
CFAA
Zamudio, Spain
edwarleonardo.sastoque@ehu.eus

***Abstract*—The Industrial Internet of Things (IIoT) leverages thousands of interconnected sensors and computing devices to monitor and control large and complex industrial processes. Machine learning (ML) applications in IIoT use data acquired from multiple sensors to perform tasks such as predictive maintenance. While remembering useful learning from the past, these applications need to adapt learning for *evolving sensor data* stemming from changes in industrial processes and environmental conditions. This paper presents a continual learning pipeline to learn from the evolving data while replaying selected parts of the old data. The pipeline is configured to produce *ML experiences* (e.g., training a baseline neural network model), improve the baseline model with the new data while replaying part of the old data, and infer/predict using a specific model version given a stream of IIoT sensor data. We have evaluated our approach from an AI Engineering perspective using three industrial case studies, i.e., predicting tool wear, remaining useful lifetime, and anomalies from sensor data acquired from CNC machining and broaching operations. Our results show that configuring experiences for replay-driven continual learning allows dynamic maintenance of ML performance on evolving data while minimizing the excessive accumulation of legacy sensor data.**

I. Introduction

The Industrial Internet of Things (IIoT) is the interconnection of sensors, computing devices (on the edge and the cloud), and industrial machines (e.g., industrial robots, machine tools, boilers) in a production network. IIoT provides streaming access to thousands of sources of time-varying data from sensors used in numerous use cases supported by Machine Learning (ML) applications [1], [2]. These applications include predictive maintenance [3], remote quality monitoring [4], and energy optimization [5].

The de-facto standard for ML applications in IIoT is to learn from a large static set of multivariate sensor data acquired over a fixed period to predict business-critical attributes such as faults, failure, quality, and energy consumption. However, *data is not static and evolves* in a dynamic IIoT environment. For instance, manufacturing data evolves due to the replacement of cutting tools, the changes in the parts manufactured, and the environmental conditions (e.g., temperature variations at the tooltip, vibrations on a shop floor, and electromagnetic interference due to coupling with other electric cables). It is impossible for ML applications to consistently make accurate predictions through learning from a static IIoT dataset. Furthermore, it is environmentally unsustainable to store high-volume and high-velocity IIoT data perpetually while computationally expensive to re-train new ML models using an infinitely growing dataset. In this paper, we answer if we can conceive and evaluate a continual learning pipeline preserving historical and topical learning performance while minimizing the IIoT data storage.

Humans can continually learn from new experiences without ignoring the past. However, artificial neural networks catastrophically forget learned knowledge if trained only with new data using standard training algorithms such as online error back-propagation. The change in neural network weights induced by training is the root cause of catastrophic forgetting of past knowledge. Replay is a mechanism in biology where the brain's hippocampus re-activates neural patterns similar to activation patterns of past waking experiences to consolidate short-term memory to long-term memory [6]. Therefore, we investigate whether we can mimic replay to train artificial neural networks in IIoT on new and evolving data without forgetting past learned knowledge.

In this paper, we propose, apply and assess an ML pipeline for replay-driven continual learning that can train a baseline model and subsequently update it based on new data minimizing catastrophic forgetting of the past. Some continual learning approaches [7]–[10] support the minimization of catastrophic forgetting in the context of IIoT. However, they focus on a specific prediction task and do not offer a generic pipeline that can be applied to various prediction tasks. They also do not discuss extensively how continual learning can be engineered and deployed for industrial settings. Our pipeline is the first one using the concept of replay as a simple yet effective approach to avoid catastrophic forgetting in multiple prediction tasks (see Section VIII) with IIoT data. We extensively discuss the engineering aspects of the pipeline and how to deploy replay-driven continual learning in industrial settings.

An engineer (we call *an ML experience engineer*) can configure our pipeline to create ML experiences (e.g., training a baseline neural network model), improve the baseline model with the new data and the replay of $X\%$ of the old data, and infer/predict using a specific version of a model given a stream of IIoT sensor data. She can specify a purge policy to delete data periodically from a message queue for incoming data after it has been consumed for learning and partially stored for replay. Furthermore, the configuration can either deploy a model for inference or pause inference (e.g., due to input drift and deteriorating performance) to start a new learning experience using new data and the replay of some old data.

We assessed our replay-driven continual learning pipeline with three different predictive inference tasks (i.e., estimating data anomalies, predicting tool wear, and anticipating tool breakage) in our IIoT case studies (see *RQ1* and *RQ2* in Section VIII). Tool wear in manufacturing is the gradual failure of cutting tools due to regular operation caused by mechanical abrasion from the workpiece, chipping, thermal cracking, and fracture. Tool wear, when undetected, causes defects in products and, consequently, abnormal production stops and the generation of industrial waste from defective parts. The main challenge with accurately predicting tool wear is to adapt to the process and environmental changes in the manufacturing process without forgetting the tool wear prediction learned from the past. We used our pipeline to create ML experiences that result in a baseline model and replay experience models with 0% replay, 20% replay, 40% replay, and 60% replay to compare their performance to the performance of a model with 100% replay. We observed that models with 0% replay result in catastrophic forgetting of learned knowledge. However, models with n% replay performed better than those with 100% replay in our case studies, indicating that the replay mechanism helps minimize catastrophic forgetting.

Our pipeline solves a model learning problem in the IIoT domain. However, numerous challenges arise in its engineering and deployment in industrial production environments. Therefore, we also analyzed and evaluated our pipeline from the AI engineering perspective [11] (see *RQ3* in Section VIII). We investigated how the replay-driven continual pipeline could be configured and deployed in a production environment. Our main contributions are as follows.

- **Novel Replay-Driven Continual Learning Pipeline.** We introduce an ML pipeline that enables the ML experience engineers to create various learning and inference experiences for ML models with IIoT data.
- **New Tool Support.** We open-source our ML pipeline for replay-driven continual learning[1].
- **Evaluation on Industrial Case Studies.** We demonstrate that replay-driven continual learning can minimize catastrophic forgetting and adapt to new data. Furthermore, we show that the pipeline requires less data to remember past experiences while performing well on new data.

The paper is structured as follows. Section II presents the background regarding continual learning, data pipelines, and AI engineering. Section III discusses the related work. In Section IV, we present an overview of our pipeline. Sections V, VII, and VI describe the core technical details. Section VIII reports on the evaluation results for three industrial case studies. In Section IX, we present some insights into the evaluation and limitations of the pipeline. We conclude the paper in Section X.

II. Background

Our pipeline employs continual learning (Section II-A) and is engineered as a data pipeline (Section II-B). We evaluated it from an AI engineering perspective (Section II-C).

A. Continual Learning

Continual learning [12] (also called incremental learning or lifelong learning) is a concept in ML (in particular deep learning) in which models continuously learn and evolve with increasing amounts of input data while retaining learned knowledge. It embraces the non-stationary world, where models need to be updated for changes in the input data (i.e., due to changes in the environment). A continual learning system should demonstrate both plasticity (acquisition of new knowledge) and stability (preservation of old knowledge) [13].

Continual learning aims to minimize catastrophic forgetting (i.e., the failure of stability, in which new experience overwrites previous experience) in neural networks due to online back-propagation with new data. Learning requires weights in neural networks to change but changing the ones associated with past learning results in forgetting. This dilemma of remembering the past vs. updating weights for new data is known as the *stability-plasticity dilemma* [14]. *Replay-driven* or rehearsal-based approaches for continual learning involve storing part of the old data and replaying it with a mix of new data. Replay-driven approaches have been applied to image classification [15]–[17]. Experience replay has been shown to reduce catastrophic forgetting in reinforcement learning for video games [13]. Another approach to continual learning is *elastic weight consolidation* [8] which freezes the weights critical to making predictions of past knowledge or assigns them low learning rates while a new layer of neurons with higher learning rates is added to learn from new data. Similar to elastic weight consolidation, it is possible to progressively grow a neural network to achieve continual learning with new data as presented by Ashfahani and Pratama [18]. Another approach is to leverage *knowledge distillation* [19] to transfer the knowledge of one or several neural network models to a new model where a teacher model trains a student model. Distillation can augment experiences in continual learning by providing input/output pairs from past learning. It is also

[1]https://github.com/SINTEF-9012/Erdre-Continual-Learning

possible to combine replay and knowledge distillation for continual learning. Buzzega et al. [20] present dark experience replay that integrates replay, distillation, and regularization where training aims to match a network's non-normalized predictions (a.k.a. logits) sampled through the optimization trajectory to ensure adherence to past learning.

We utilize the ML research given above to devise and assess an ML pipeline of replay-driven continual learning for IIoT. We evaluate replay-driven continual learning in practice with industrial datasets and address AI engineering dimensions that enable us to put our pipeline in production for IIoT data.

B. Data Pipelines

A data pipeline is a digital infrastructure that facilitates the movement and processing of data. ML applications generally require complex data pipelines that can handle steps, including processing raw data and producing a trained (and possibly deployed) ML model [21]. The first part of such pipelines involves data cleaning, feature engineering, and data restructuring for the appropriate input format required by the given ML algorithm. It is followed by building and training an ML model evaluated and eventually deployed. Developing an ML model typically involves running several experiments to fine-tune configuration and control parameters.

Traditional version control systems like *Git* [22] easily keep track of changes in source code and control parameters but are ill-suited for tracking big data and binary files, e.g., ML models. One alternative solution is to use the Data Version Control (DVC) framework [23], [24] to support big files in data pipelines. A DVC pipeline is created by defining a set of stages. Each stage has a run command(s) concerning its dependencies to other stages. The dependencies usually involve input data, control parameters, expected output, and source code. DVC can skip the execution of certain stages and instead fetch the correct output from the cache if they have already run in the same configuration.

C. AI Engineering

Bosch et al. [11] define ”AI engineering” as an extension of software engineering with new processes and technologies for the development and evolution of AI systems. The goal of AI engineering is to address the engineering challenges of AI systems from the software engineering point of view. One needs to address numerous AI engineering concerns to implement continual learning. Below we briefly describe some AI engineering dimensions [11] we considered while devising and evaluating our pipeline for continual learning.

- **Data versioning and dependency management**. Data versioning needs to be handled since data generated even by consecutive software versions may not be compatible. In addition, data quality is crucial in training to achieve high model performance. Since data pipelines are less robust than software pipelines, they need data quality management, which can be particularly challenging when different data types and sources are used.
- **Deployment infrastructure**. Companies need an infrastructure that reliably deploys subsequent model versions, measures model performance, and raises warnings for anomalous behavior. Deployment of ML models may require a substantial change in the system architecture.
- **Quality attributes**. The key challenge of data science for ML models is to achieve high accuracy and precision. Several other quality attributes, including computation performance, the number of inferences per time unit, real-time properties, and system robustness, are also relevant from the AI engineering perspective.
- **Monitoring and Logging**. Once an ML model is deployed and operational, it is needed to monitor and log its performance. Since ML models have insufficient explainability, monitoring and logging are required to build confidence in the model's accuracy while detecting when its performance declines or is poor from the beginning.
- **Integration of models and components**. Since an AI system has not only AI components, companies need to integrate ML models with the traditional software components of the system. Software verification techniques need to be adapted to check whether AI and regular software components are integrated seamlessly. Depending on the criticality of ML models, the validation and verification activities need to be more elaborate and strict.

III. Related Work

Realizing continual learning in practice for the IIoT forms the subject of this paper. We present some background on core ML techniques for continual learning in Section II-A. This section focuses on the role of software and AI engineering in bringing continual learning into practice and its consequent impact within the context of IIoT. We present the related work on both aspects of the problem.

Continual learning leads to several AI engineering concerns to orchestrate several components in an ML pipeline. Some of these components are concerned with the acquisition of streaming IIoT data, monitoring drift in input data, managing learning and replay, monitoring ML performance, ML model storage & retrieval, (re-)deployment, and inference. Diethe et al. [25] present a *reference architecture for continual learning* that addresses the challenges of implementing continual learning in practice. They emphasize the need to handle both streaming and batch data and briefly present concepts such as self-diagnosis through monitoring, self-correction policies, and self-management of resources. They also discuss parameters that control the automation of the ML life-cycle, such as horizon (how quickly does the data point become part of the model?), cadence (how often should we retrain?), provenance (can we trace decisions back to underlying causes?), and cost (what is the retraining cost?). However, they do not propose or assess any continual learning approach with experiments. Mirzadeh et al. [26] focus on the impact of parameters such as the width and depth of neural networks in continual learning. Schwarz et al. [27] present a similar approach, called progress and compress, leveraging knowledge distillation for continual learning that avoids the need for architectural growth in neural networks. These two approaches do not explicitly address software/AI engineering challenges of deploying continual learning production. Yang et al. [28] present an Automated Machine Learning (AutoML) pipeline for anomaly detection in IoT data in dynamic environments. They enlist several ML models that can be automatically updated based on *concept drift* detection. However, their pipeline is linear and does not consider cyclical dependencies that may arise due to replay.

Industrial IoT data stems from a dynamic environment requiring adaptations to ML models to maintain prediction performance. There are works applying ML to IIoT data on edge. For instance, Sun et al. [29] present an AI-enhanced offloading scheme to process IIoT data on edge. Bellavista et al. [30] provide an architecture for ML model deployment on the edge-cloud continuum for federated learning. Their approach entails updates to initial models on edge based on local data followed by data aggregation in the cloud. It creates new models based on false negatives but does not address catastrophic forgetting or remembering past knowledge. There are some continual learning approaches [7]–[10] that focus on the minimization of catastrophic forgetting in the context of IIoT. For instance, Maschler et al. [7], [8] present a regularization-based approach based on elastic weight consolidation to realize continual learning while detecting anomalies in the metal forming process. Tercan et al. [9] use memory-aware synapse cloning to perform continual learning in a real-world predictive quality case in injection molding. None of these works discuss how to deploy continual learning in industrial settings. They focus on a specific prediction task (e.g., only predicting tool wear) and do not propose a pipeline like ours that can be applied to various prediction tasks. Furthermore, replay has not been used as a simple yet effective approach to avoid catastrophic forgetting with IIoT data.

Engineering AI systems [11] (e.g., devising continual learning techniques for industrial processes or infusing unsupervised learning into manufacturing) for industrial settings has not been discussed extensively. Angelopoulos et al. [31] present learning algorithms for fault detection in Industry 4.0 with little focus on how to design and deploy them. Some works [32]–[34] discuss unsupervised learning for predictive maintenance and anomaly detection without mentioning AI engineering aspects. Only recently, Husom et al. [35] explicitly discuss and evaluate their AI-based approach (i.e., an unsupervised learning pipeline for sensor data validation) for industrial settings from the AI engineering perspective. Our work presents the field knowledge of how a continual learning pipeline is engineered and its learning and inference experiences are employed, from which researchers can benefit.

IV. Overview of the Approach

Humans learn from experiences over their lifetime. These experiences include learning continuously from new information by sensing the real world, replaying to consolidate memory during sleep, and using the learned knowledge to infer decisions under new circumstances. This paper answers if we can design a machine learning pipeline to mimic learning, replay, and inference as experiences generated from IIoT data. To this end, we present a replay-driven continual learning pipeline for IIoT data (see Figure 1).

The pipeline receives multi-variate time series data as *input* from an IIoT system that employs several sensors to monitor a physical process and take action to control it. For instance, position, velocity, acceleration, torque, temperature, force, and vibration sensors are used to control an IIoT system for manufacturing (e.g., CNC milling, broaching, and grinding). The input also contains labels or target variables provided by human experts or other automated processes that measure product/process quality. The pipeline's initial *output* is an ML model generated by a *baseline learning experience* using a static dataset extracted from a raw data queue. The meta-information of the dataset should include input and target variables, and the dataset should contain values for each. The pipeline can output updated ML models by undergoing *replay-driven learning experiences* (training with new data and replaying past data). These experiences are configured and orchestrated by an *ML experience engineer* observing model performance and drift. If the engineer deems that the model performance is satisfactory and the data is not drifting, she can configure the pipeline to deploy an ML model through an inference service. This ML model as a service can have *inference experiences* where it predicts anomalies, process/product quality, or equipment performance (e.g., tool wear prediction) in IIoT. Baseline models are produced, updated, and deployed, through continual learning and inference, based on dynamic changes due to input *data drift* and *concept drift* in ML performance. We present the pipeline in six stages:

Experience configuration. The ML experience engineer configures the learning and inference experiences of our pipeline through the experience configuration stage. She can invoke three experiences: a *baseline learning experience*, an *inference experience*, and a *replay-driven learning experience*. We explain their details in Sections V, VI, and VII. Parameters include experience type, which features to extract in the input times series, train/test split, how much data to replay along with new data, which ML model to train from the model database, and model evaluation parameters.

Data pre-processing. Multivariate sensor data from IIoT arrives in a message queue we call *raw data queue* (implemented using messages queue software such as MQTT and Apache Kafka), where each message contains a timestamp, variable, and value. Data is extracted from this queue by specifying (as part of the configuration) a timestamp range and set of sensor variables to create a dataset (CSV file) for an experience. The experience configuration can also specify which data to purge/delete over time when it is consumed for learning/inference. The dataset for any experience goes through several sub-stages common in ML pipelines: *data profiling*, *data cleaning*, *feature engineering*, *splitting test/training data*, *data scaling*, and *data sequentializing (splitting data into sub-sequences)*. Data profiling includes computing the non-linear *maximum information coefficient* [36] and the linear *Pearson's correla-*

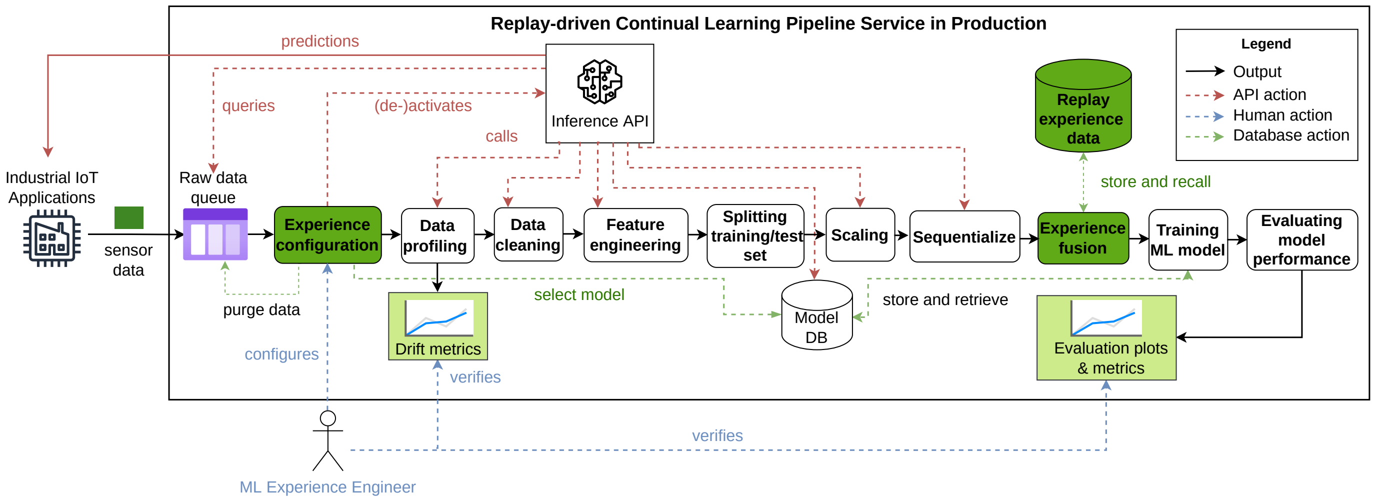

Fig. 1. Replay-Driven Continual Learning.

tion coefficient [37] to find correlations between data columns of different sensors. It generates statistical quantities for each column and alerts if any column contains several zeros or missing values. Data cleaning uses this output to automatically remove unwanted data (e.g., columns with several constant or null values). Affected observations are removed (a missing value is replaced by zero if it is consistent with the behavior of the affected variable).

Feature engineering extracts statistical properties, called *features*, from the raw input data that exhibit invariance to noise. Furthermore, the feature-based representations of time-series data [38] perform well in classifying tasks at a fraction of the computational cost of processing raw time-series data. The input and output data can be split into *training* and *test* datasets. The pipeline uses the training set to train the ML model and tune hyper-parameters (see *Training ML Model*). The test dataset is locked away during training and used as an unbiased dataset to evaluate model performance.

The datasets contain measurements from different sensors with varying value ranges and are, thus, scaled [39] for a comparable influence during training. The training (including validation dataset) and test datasets are *restructured* into input and output sub-sequences of the specified *window size* since predictions rely on a window of time-varying observations from input sensors and the desired window of output values.

Experience fusion. Experience fusion entails combining new data and data from past experiences. This stage stores or recalls the replay part of the dataset used in the pipeline. When a model is created or updated, part of the dataset can be configured to be stored as replay experience data and used later when creating replay-driven learning experiences. Sampled parts of the datasets going through the pipeline are replayed to avoid catastrophic forgetting. When a model is updated, replay experience data recalled in the experience fusion stage and integrated into the dataset is passed to the training stage.

Training ML model. The input and output sub-sequences are used to configure and train the ML model. Our pipeline enables specifying learning parameters and selecting ML model types (architectures). We consider Dense Neural Networks (DNNs)/Fully Connected Neural Networks (FCNNs), Convolutional Neural Networks (CNNs) [40], and Long Short-Term Memory (LSTM) [41] to train ML models. A small part of the training dataset (e.g., 20%) is set apart as a *validation dataset* before training. Our pipeline automatically stops training if the prediction error of the validation set stops improving, preventing the over-fitting of the model to the training data. It saves the ML model for evaluation in the model database (Model DB).An ML experience engineer can configure the pipeline to create a model from scratch or select any previous model in the Model DB as a starting point.

Evaluating model performance. The test dataset is an unforeseen dataset used to evaluate the model performance to minimize bias due to hyper-parameter tuning in model training. We compare the model output and the ground truth to assess how accurately the model predicts the target variable. To this end, the pipeline generates the plots of predictions on test data. Mean Squared Error (MSE), coefficient of determination (R^2 score), and Mean Absolute Percentage Error (MAPE) are metrics used to evaluate the performance of regression models, and accuracy and F1-score are used for classification models. We use the R^2 score as a metric for the performance of regression models in this paper (see Sections VIII-B and VIII-B). For the classification models, we assess the model performance using F1-score (see Sections VIII-B and VIII-B).

Sections V, VII, and VI explain the details of the replay-driven continual-learning in Figure 1.

V. Baseline Learning Experience

The first step in our pipeline involves creating a *baseline learning experience* by training an ML model with a static dataset (running through the pipeline stages of data profiling, data cleaning, feature engineering, train/test set split, scaling, sequentializing, training, experience fusion and evaluation in Figure 1). When the ML experience engineer chooses the baseline learning experience in the experience configuration stage, sensor data collected from the industrial environment is the input to the pipeline stages. This training dataset needs to contain the target variable or label that the model should learn to predict. Data profiling is important for creating the baseline model since the correlations between the input and target variables are leveraged when deciding what input features to use for the model. The baseline learning experience involves reiterating the stages and fine-tuning control parameters during each iteration to produce a model with acceptable performance. Feature selection, scaling method, and window size for sequentializing are parameters affecting the model performance. The training stage typically relies on a large set of control parameters, defining the size and configuration of the neural network and the training duration. In the experience fusion stage, the engineer specifies the amount of data for future replay where the model needs to be updated. After training and evaluation, the model is stored in the Model DB.

VI. Inference Experience

An inference experience refers to utilizing an existing ML model to produce a prediction from IIoT data. The pipeline does not concern itself with training but runs the model against a batch of new data. The engineer invokes the inference service via REST API, the command line (of a virtualized Docker container), or using a Graphical User Interface (all three options call the pipeline components). The inference service is configured to query/read the raw data queue to obtain sensor variables for a given time range and put them into a *batch dataset*. Next, the engineer specifies the identifier of the model in Model DB. The pipeline pre-processes the data with some exceptions (there is no need to prepare for training by profiling and splitting the data). The configuration invokes components to clean the input data, extract features, load a trained model from Model DB, obtain a prediction, and send it back to the IIoT system either as a message to another service or by displaying it on a user interface. The inference experience is invoked when learning experiences are not occurring (the model is stable under the current circumstances).

VII. Replay-driven Learning Experience

Learning experiences use new incoming data to improve past ML models, including the baseline model. The engineer monitors the inference process to recognize (a) when input IIoT data drifts away from aggregate statistical properties of data used to create past models in Model DB (data drift) and (b) when the performance of the current model is poor for the incoming data. Once she observes consistent deviation in performance, she invokes the replay-driven learning experience in the experience configuration stage.

The replay-driven learning experience is configured to use an existing model in Model DB with new data and part of the past data stored in the *Replay Experience Data reservoir* (see Figure 1). New and past data should have the same schema for input and target variables/labels in the configuration. The configuration also specifies the amount of new data maintained in the *Replay Experience Data reservoir* for future use. New data are extracted from the raw data queue and preprocessed for training. The configuration invokes the *Experience Fusion stage* to combine the replay experience data (part of past data for replay) with the new data. The combined dataset is used to train an existing ML model from Model DB (specified by a Model identifier in the configuration). The engineer evaluates the performance of the new ML model. If the performance is satisfactory, she invokes the inference experience with the new model (see Section VI).

As part of the configuration, the engineer may keep inference active or inactive on incoming data during the replay-driven learning experience. To automatically start the replay-driven learning experience, she may specify thresholds and constraints for the statistical properties of input data based on past observations and ML performance metrics. These constraints can be domain-specific, and it is hard to trace and verify their consequences as many new models can be created automatically in the Model DB without supervision.

VIII. Evaluation

We evaluate replay-driven continual learning on three industrial case studies to address three Research Questions (RQ)s:

- ***RQ1.*** *How does a baseline model based on static legacy data perform on new IIoT data for predictive inference?*
- ***RQ2.*** *How does replay-driven continual learning perform on new and old IIoT data for predictive inference?*
- ***RQ3.*** *How can our approach be run in industrial production environments?*

A. Subjects of the Evaluation

We applied our approach to categorize signals (normal and anomalous) and predict tool wear (the gradual failure of cutting tools due to regular operation) and remaining useful lifetime of a tool in three case studies.

1) Bosch CNC Machining Dataset: The dataset is a collection of real-world industrial vibration data obtained from three milling CNC machines (each executing fifteen processes) in a real-world production plant using a smart data collection system over a two-year period. The machines were processing aluminum workpieces in an operation sequence on aluminum parts (see Figure 2).

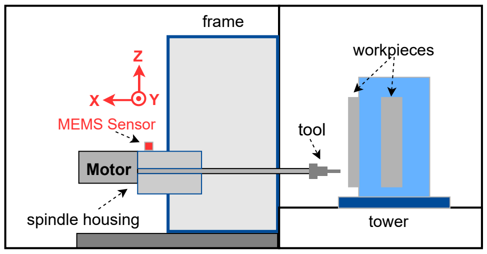

Fig. 2. Schema of the 4-axis machining center with mounted sensor [42].

The acceleration was measured using a tri-axial accelerometer (Bosch CISS sensor [43]) mounted to the rear end of the spindle housing. Each sensor was at a constant distance to the tool center point, while the three axes of the accelerometer were in alignment with the linear motion axis of the machine (see the sensor coordinate system in Figure 2). The accelerometer X- Y- and Z-axes were recorded using a sampling rate of 2 kHz for four different time frames, each lasting five months from February 2019 until August 2021, and labeled accordingly (i.e., normal and anomalous data).

The dataset contains labels on whether the accelerometer data are anomalous. We create a predictive model (a classification model) to categorize the signals as normal or anomalous.

2) Broaching of Airplane Turbine Discs: Broaching is a manufacturing process for forming internal or external round, flat, or contoured surfaces. A broaching machine pushes a multi-toothed cutting tool, a broach, into a workpiece to remove material (see Figure 3). Slots of various dimensions are cut at high production rates. Our dataset was collected from three broaching tools in a broach tool holder broaching fifty slots for around three hours. The broaching operation was to broach *fir tree slots* on jet engine turbine discs. We had three data sources: (i) two accelerometers with a sample rate of 12.8 kHz, (ii) a data logger with a rate of 250 Hz, and (iii) tool wear measured with an optical microscope. We use the accelerometer data as input to predict the tool wear measured in millimeters.

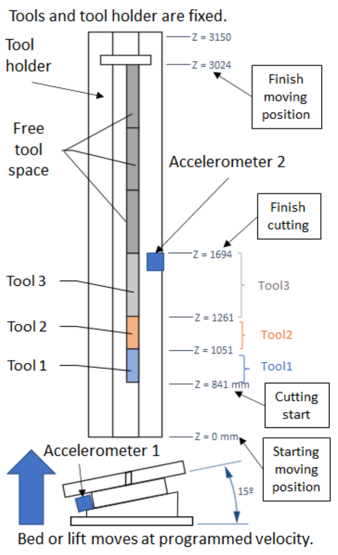

Fig. 3. Broaching machine.

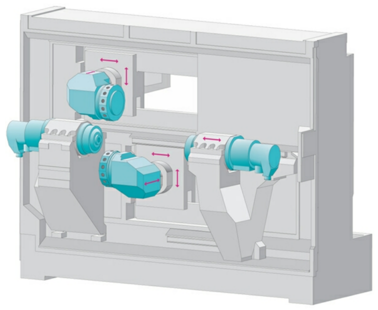

Fig. 4. Index C65 lathe generating the piston rod dataset [44].

3) Piston Rod Manufacturing Dataset: This dataset was recorded during the machining process of piston rods (pneumatic cylinder components) at the Center for Industrial Productivity (CiP) learning factory in Darmstadt, Germany. The piston rods made of stainless steel were manufactured in batches of four in a turning machine (i.e., Index C65 lathe in Figure 4). The machine was equipped with a Siemens 840d CNC control streaming relevant internal signals to a host PC storing the data. The dataset contains the part IDs, machine control data, and timestamp information on when tools were changed in the lathe. The process data (machine control data) includes torque, current, spindle speed, and feed rate for every record with a sampling rate of 5 Hz. Using the process data, we predict the remaining useful lifetime of the tool.

B. Results

This section discusses the results of our case studies, addressing, in turn, each of the RQs.

RQ1: How does a baseline model based on static legacy data perform on new IIoT data for predictive inference?: We address RQ1 by computing the performance of baseline models using (a) F1-score on the binary classification of anomalous behavior in the Bosch CNC machining of aluminum workpieces and (b) R^2 score for regression models predicting tool wear during broaching of aero-engine discs and time until the tool change during the manufacturing of piston rods.

Table I presents the baseline model's performance scores (F1-score) across five different datasets for CNC machining across time. The number in the name of the dataset in the left column indicates the year and month the data was collected. Each dataset was collected several months apart, enabling us to study how the model performance changes over time.

TABLE I
BASELINE MODEL PERFORMANCE ON BOSCH CNC DATASET.

Evaluated Data	F1-score
M01 2019 02	0.891
M01 2019 08	0.217
M01 2020 02	NC
M01 2021 02	0.025
M01 2021 08	0.489

The baseline model has an F1-score of 0.891 when tested on the first dataset (i.e., *M01 2019 02* refers to the first data set) with test data from the same process in which we trained the model. The F1-score is significantly lower on the other datasets and ranges from 0.489 to 0.025, which may indicate a change in the underlying pattern. The dataset *M01 2020 02* has only non-anomalous vibrations, and the F1-score is, therefore, non-calculable (NC). The F1 score and the imbalance in the datasets have a significant correlation (see Tables I and II). The baseline model was trained on a dataset having a relatively high proportion of anomalous vibrations. Therefore, we have poor F1-scores on *M01 2019 08* and *M01 2021 02*, which have few anomalous vibrations, and a better score on *M01 2021 08*, which has a similar distribution of anomalous vibrations to that of the first dataset. Even considering the imbalance, we can see that the performance has deteriorated from the first to the last dataset, which shows the need for updating the model through continual learning.

TABLE II
DISTRIBUTION OF ANOMALOUS VIBRATIONS ON BOSCH CNC DATASET

Dataset	Anomalous vibrations
M01 2019 02	21.2%
M01 2019 08	3.1%
M01 2020 02	0.0%
M01 2021 02	0.8%
M01 2021 08	21.2%

Table III presents the performance scores (R^2 scores) of the baseline model of the broaching use case. The broaching dataset is divided into five sub-datasets; each contains sensor data acquired from broaching six to ten fir-tree slots on the turbine disc. The slot broaching occurs sequentially over time; the evaluation of the five datasets shows how the baseline model performs worse when evaluated on more recent data. The baseline model was trained on Slot Set 1, i.e., the sensor data collected from broaching the first six slots. When the baseline model is evaluated on test data from the same dataset, it performs poorly (see the first row in Table III). However, the training was mainly on the earliest slots of the dataset; the testing was on the last ones. The tool wear increases gradually in each slot, implying that the target variable moves out of the training distribution. The R^2 score is lower than zero when the model performs worse than predicting the mean target value for every prediction, i.e., the model cannot beat a naive estimator outputting the average target value of the test set for any input values. The initial performance of the baseline model is unsatisfactory since the tool wear moves into a range that the model has not seen before. Furthermore, the performance degrades quickly in each subsequent dataset, and the baseline model becomes outdated and even more unusable.

TABLE III
BASELINE MODEL PERFORMANCE ON BROACHING DATASET.

Evaluated Data	R^2 Score
Slot Set 1	-6.086
Slot Set 2	-155.462
Slot Set 3	-5666.836
Slot Set 4	-8137.291
Slot Set 5	-7439.583

Table IV presents the performance scores (R^2) of the baseline model of the piston rod use case. The piston rod dataset is divided into five sub-datasets having roughly equal sizes (approximately 150 production cycles in each sub-dataset). The model has an R^2 score of 0.539 when tested on Piston Rod Set 1. However, the performance quickly deteriorates when the model is evaluated on subsequent sets. The R^2 score is negative already on the second dataset and down to -1.110 on the last dataset.

TABLE IV
BASELINE MODEL PERFORMANCE ON PISTON ROD DATASET.

Evaluated Data	R^2 Score
Piston Rod Set 1	0.539
Piston Rod Set 2	-0.052
Piston Rod Set 3	-0.592
Piston Rod Set 4	-0.347
Piston Rod Set 5	-1.110

Answer to RQ1: The baseline models in the case studies showed deteriorating performance over time when evaluated on newer data. The results indicate the need for continual learning to update the models to avoid model performance decline.

RQ2: How does replay-driven continual learning perform on new and old IIoT data for predictive inference?: Replay entails the fusion of a certain percentage of past data from a replay experience reservoir with new IIoT data to train an existing ML model in the Model DB. This experience is only invoked when we observe the drift and poor ML model performance during inference experiences. New vibration data from the IIoT stream is curated periodically to detect anomalies during Bosch CNC machining; Table V summarizes the performance of replay-driven continual learning for the Bosch dataset. Similarly, new vibration data acquired from broaching fir-tree slots on airplane turbine discs are grouped as slot sets to predict tool wear in a broaching tool. Table VI presents the performance of our approach in the broaching dataset. Table VII shows the equivalent results for the piston rod dataset. Tables V, VI, and VII give the identifier for a batch of new datasets arriving over time, performance (F1-score or R^2 score) for the datasets in the recent *past*, and performance on the part of the *new* dataset reserved for testing. We evaluated replay-driven continual learning for 0%, 20%, 60%, and 100% replay. 100% replay is equivalent to traditional ML, where training and testing use the dataset without new data.

The Bosch CNC machining dataset has very few rare events or anomalies. All degrees of replay work well in predicting anomalies in new data from CNC machining (see Table V). However, the F1-score could not be computed (NC) in one incoming set (M01 2020 02) since there were no anomalous vibrations. When using 0% replay, training on datasets with very few anomalies caused the F1-score to decrease significantly, making the model seem unusable. Training on the most recent dataset with a considerable number of anomalies caused the model to regain its performance on past data. All other degrees of replay (20%-100%) prevented catastrophic forgetting and helped the model detect anomalous vibrations by replaying a portion of older data.

The time scale for predicting tool wear from CNC broaching is much smaller than in the Bosch case. The number of times an expensive broaching tool (sometimes costing several thousand euros) is used does not exceed two orders of magnitude. Therefore, there is a need to rapidly detect tool wear after broaching a few slots. Detecting tool wear is a regression problem as it is measured in millimeters. We observe in Table VI that 0% replay results in steady deterioration in the performance of predicting tool wear from Slot Set 2 to Slot Set 5. Increasing the amount of replay to 20% improves the performance on past data for 20% replay, 60% replay, and 100% replay. However, there is a varying poor performance in predicting tool wear for the most recent new unforeseen data across all percentages of replay. An explanation for poor and variable performance on new unforeseen data but good performance on the most recent past data could be that new data is considerably different from anything seen before due to increasing tool wear. The physics knowledge of the tool wear process could significantly enhance our understanding of how vibrations affect tool wear and why it is consistently out-of-distribution. Both traditional ML (due to 100% replay) and replay-driven continual learning perform poorly on new data calling for further study of the phenomena.

We do not have direct measurements of tool wear in the piston rod use case, but we have timestamps of when the

TABLE V
PERFORMANCE SCORES OF THE REPLAY MODELS TRAINED ON BOSCH CNC DATASET.

New Data	0% replay		20% replay		60% replay		100% replay	
F1-score:	past	new	past	new	past	new	past	new
M01 2019 08	0.681	0.593	0.620	0.454	0.657	0.443	0.690	0.450
M01 2020 02	0.005	NC	0.659	NC	0.619	NC	0.718	NC
M01 2021 02	0.002	0.000	0.571	0.778	0.628	0.747	0.672	0.762
M01 2021 08	0.485	0.689	0.644	0.830	0.630	0.784	0.751	0.904

TABLE VI
PERFORMANCE SCORES OF THE REPLAY MODELS TRAINED ON BROACHING DATA SET.

New Data	0% replay		20% replay		60% replay		100% replay	
R^2 Score:	past	new	past	new	past	new	past	new
Slot Set 2	0.089	-8.207	0.086	-7.831	-0.032	-8.100	0.035	-7.882
Slot Set 3	-0.028	-57.522	0.716	-65.488	0.702	-57.952	-0.075	-1297.407
Slot Set 4	-0.723	-24.926	0.699	-301.906	0.714	-294.650	0.748	-261.217
Slot Set 5	-0.841	-54.646	0.805	-82.818	0.809	-52.086	0.891	-56.382

TABLE VII
PERFORMANCE SCORES OF THE REPLAY MODELS TRAINED ON PISTON ROD MANUFACTURING DATASET.

New Data	0% replay		20% replay		60% replay		100% replay	
R^2 Score:	past	new	past	new	past	new	past	new
Piston Rod Set 2	0.088	0.349	0.341	0.494	0.321	0.420	0.421	0.484
Piston Rod Set 3	-0.524	0.785	0.437	0.692	0.506	0.556	0.515	0.616
Piston Rod Set 4	-0.045	0.569	0.362	0.530	0.425	0.496	0.434	0.365
Piston Rod Set 5	-0.180	0.409	0.335	0.309	0.394	-0.032	0.379	-0.237

tool was replaced (which happened ten times during the data collection) obtained from the visual tool inspection. We predict the useful remaining tool lifetime by using the tool replacement info. With 0% replay, the performance is poor for past data but much better for new data (see Table VII). We expect this poor performance to happen when data changes over time without any updates in the model. Using 20% replay or more significantly improves the model performance on past data, which indicates the need for using methods for retaining past knowledge. We have high model performance on the new data when using 100% replay for each case (except for Piston Rod Set 5). The tool was replaced at irregular intervals during the data collection. These irregular intervals may indicate that the process behavior was significantly different across the five sub-datasets and explain why we have varying performances with no clear trend across the datasets.

Early machining data often contains only good vibrations (or a few bad vibrations reflecting tool wear) since tool wear increases in time while bringing new unforeseen patterns resulting in imbalanced data. Hence, the prediction performance of ML models is relatively good initially and gradually deteriorates, e.g., in the Broaching and Piston Rod cases (Tables VI and VII). With 0% replay, we observe performance deteriorates for baseline and new data in the Broaching and the Piston Rod datasets. Using 20% replay significantly mitigates the performance deterioration on the baseline; we still have poor performance on new data due to the imbalance caused by very few new bad patterns. There is, however, a slight improvement in predicting wear for new data for the Broaching case at Slot 5 (Table VI). The performance for 60% and 100% replay in the Piston Rod case is worse than the performance for 20% replay at Piston Rod Set 5, indicating that excessive use of baseline data may make performance poor on new data. Therefore, selecting data for 20% replay is important to maintain the prediction performance.

Answer to RQ2: We conclude that 20% replay is adequate to maintain good performance on continual learning, minimizing catastrophic forgetting. However, both continual learning and traditional ML with 100% replay perform poorly on new unforeseen data that is likely out-of-distribution from what has been seen before.

RQ3: How can our approach be run in industrial production environments?: To address RQ3, we report the lessons learned while engineering replay-driven continual learning in two different industrial settings. We summarize our experience based on the AI engineering dimensions in Section II-C.

Data versioning and dependency management: DVC [24] is a popular framework keeping track of large data and binary files like ML models (see Section II). Using DVC to cache replay data and different model versions can be time-saving in a setting where one wants to experiment with various factors such as replay percentage, training epochs, and train/test split. However, the challenges in using DVC due to the cyclical nature of continual learning, with training, evaluating, inference, and back-to training interfere with the versioning system of a DVC pipeline. Therefore, we developed our pipeline with custom data and model management, where the location of data and models are parameters specified by the engineer. This practice contrasts with using DVC automatically managing data and model versions and locations.

Deployment infrastructure: The pipeline ingests batch data arriving over periods or as a stream and is deployable as a virtualized Docker container. Raw batch data can be ingested from sensors by (i) connecting to a database on an IIoT system, (ii) pulling data from an IIoT system using protocols such as REST, OPC-OA, and MQTT, or (iii) reading time-stamped CSV files dumped by an IIoT system in a live folder. The engineer configures an experience to extract data for learning from sources such as a folder, database, or message broker for inference or learning. She monitors drift in input data and concept drift as reported by the pipeline and determines which experience to run for replay-driven continual learning. Invocation of an experience results in the orchestration of specific pipeline components (see Figure 1). Deployment in continual learning requires that training and inference share the model repository and computing infrastructure as models are generated continually over time when the engineer deems a model update necessary. The engineer needs to specify rules that put the deployed system into either baseline training, replay-driven training, or inferencing when performance is stable. This practice is different from traditional model deployment (where we deploy a static model after training).

Quality Attributes: Data quality attributes completeness, currency, and accuracy are prerequisites for creating and updating a reliable AI model but are also necessary for meaningful inference. We profile the data and present statistical properties to the ML experience engineer. The engineer may also specify domain-specific assertions on the data using tools such as Great Expectations [45]. These expectations are continually verified during profiling datasets extracted from the raw data queue. The engineer can evaluate the trustworthiness of predictions based on the quality of the data.

Monitoring and Logging: Continual learning presents challenges in monitoring data inflow and detecting when and how to update ML models. The first challenge is incoming data purge for inference and learning experiences in resource-constrained systems (e.g., the edge) having limited data storage. A purge policy should determine when to remove what data from the raw data queue for what experience. To this end, we should log data consumed in the data queue. The second challenge is manually monitoring input distribution and model performance for input and concept drifts (respectively) to invoke a new experience. To address this challenge, we can train Bayesian neural networks that estimate uncertainty in the model as confidence intervals. These intervals can automate the continuation of inference experiences (for short ones) or trigger a new learning experience (for long intervals and high uncertainty). However, expert-guided or automated experiences may lead to numerous datasets and models in the pipeline and its storage system. ML models should be aggregated over time, while others should be purged to maintain a sustainable system. Auditability of the continual learning may require different levels of logging to search for transformations of data and models created in a cyclical loop. Systematically addressing these challenges still remains an open problem.

Integration of models and components: The inference experience requires that ML models (typically stored in a Model DB with unique identifiers) be integrated into a callable inference software service (see Figure 1). The engineer specifies which ML model to use in experience configuration for an inference experience. She gives the Model ID in a parameter file to run the pipeline and infer all incoming data in the raw data queue. We consider the continual learning pipeline as standalone software that we can use for learning and inferencing based on its configuration. We believe that both inferencing and learning should be part of the same system, unlike traditional ML (models are deployed in a device without access to a continually evolving database of new ML models).

Answer to RQ3: We advocate engineering a standalone continual learning pipeline where learning and inference are coupled closely. The pipeline requires monitoring of data accumulation and model creation to enable the purging of old data already consumed for learning. Furthermore, models should be aggregated and discarded to keep the pipeline sustainable. Comprehensive management of data and model life-cycle in continual learning is still an open problem.

C. *Threats to Validity*

Internal validity. We show that replay of past data can minimize catastrophic forgetting. However, ML performance (the F1-score and R^2 score) may be affected by data redundancy over time. It is common in manufacturing that the same process is repeated often. It might be difficult to distinguish the effect of replay vs. redundancy in new data. We observe vibration data varies considerably over time in our case studies, making our experiments legitimate.

External validity. The external threat to validity concerns the generalizability of continual learning to all types of IIoT data. We mitigate this threat by using three representative production lines manufacturing different components for different companies and generating data in different sizes and natures.

Construct validity. Construct threats to validity concern the degree to which our measure accurately assesses what it's supposed to. Datasets may contain sensor data not correlated, which could introduce noise while measuring the performance of ML models. To mitigate this threat, we asked domain experts to identify the sensor data for predictions.

IX. Discussion

Generalizability: We have experimented with using replay-driven continual learning in the IIoT domain, where the repetition of a task such as manufacturing is a hallmark. Our results are based on the assumption that the time series data from the same manufacturing process reflect a repetitive process with small variations that should be continually learned. We cannot generalize our conclusions to other domains unless we experiment with data of the same nature.

Safety Concerns: We should consider implementing continual learning from sensor data with caution in safety-critical scenarios where a malfunction or failure may cause significant damage to humans or the environment. One approach to address safety concerns is to have a learning process human experts monitor and control (a feedback loop where the system output is constantly monitored and any abnormal behavior is detected and flagged for inspection). Additionally, safety-critical systems can be designed with redundancy and fail-safes to prevent catastrophic failures. Another approach is to use a hybrid learning system that combines machine learning algorithms with human expert knowledge. The algorithms are trained on sensor data to detect patterns and anomalies, and human experts provide feedback and input to ensure that the system functions safely and correctly. In both cases, it is important to test and validate the system before its deployment in a safety-critical environment. The testing involves simulating various scenarios to ensure the system responds appropriately to unexpected events. It is also crucial to ensure that the system is continuously monitored and updated to adapt to changes in the environment or system behavior.

Actionable Guidelines for Engineers: Engineers should configure the pipeline to create a baseline model based on incoming sensor data for the ideal operation of the IIoT system. The baseline model should be used for inference and then for updates when the engineer detects a drop in performance. The engineer should monitor performance metrics on new and unforeseen data to assess training on new data. She should also evaluate the pros and cons of forgetting past knowledge.

Selection of Replay Data: We assume a realistic scenario where new data constantly arrives, and the engineer has little room to select data carefully. Hence, 20% of old training input/output pairs are chosen randomly from each old dataset. However, we can employ other data selection techniques, such as those that aim to maintain a balance between classes and a range of values.

X. Conclusion

In this paper, we presented a replay-driven continual learning pipeline that trains a baseline model and updates it subsequently based on new data minimizing catastrophic forgetting of the past. We assessed our pipeline with three IIoT case studies. Our results show that the performance of models using static data quickly degrades due to data drift. However, replaying 20% of the past data whenever we encounter new data maintains performance. Furthermore, we discussed how we handled some AI engineering dimensions in our pipeline.

A. Future work

New continual learning metrics: Continual learning introduces new aspects of the learning process that go beyond what is measured by the performance metrics applied in conventional ML settings. A new set of metrics (e.g., performance maintenance, forward transfer, and backward transfer) has been designed for lifelong learning scenarios [46]. These metrics enable a more formal expression of how well continual learning strategies work, e.g., whether models can adapt to new data and to what degree they lose or gain knowledge on old tasks. Using such metrics will improve the evaluation of lifelong learning scenarios and give a better picture of model reliability and the continual learning process.

Continual balancing: Various techniques (e.g., random oversampling, randomly duplicating data points from the minority class, and removing data points from the majority class) have been proposed to balance datasets. Some of these techniques (e.g., the Synthetic Minority Oversampling Technique) are computationally expensive [47]. It is hard to know in advance which balancing methods paired with which learning methods will perform best on a given dataset. We plan to implement the selection of balancing techniques in the pipeline.

Uncertainty estimation: We can quantify uncertainty to analyze and verify the input and outputs of ML-enabled software systems and, hence, improve the trustworthiness of such systems. One way forward is to use uncertainty estimation [48] to autonomously guide the generation and deployment of new learning experiences using recent data. Bayesian Neural Networks (BNNs) can determine the uncertainty of a model's performance in a confidence interval for the prediction [49]. When the interval is above a threshold, we should store new data and train a new model based on new and some old data to minimize *catastrophic forgetting*.

Heterogeneous architectures: The pipeline should execute learning and inference experiences on heterogeneous edge devices. We plan to investigate the generation of *intermediate representations* (IR) in Multi-level Intermediate Representation (MLIR) [50] for our pipeline. MLIR can be translated to lower-level representations such as LLVM [51] and compiled for x86 (32-bit and 64-bit), ARM, PowerPC, MIPS, and RISC-V across various devices on the edge-cloud continuum.

Green AI and Continual Learning: Research in Green AI addresses reducing power consumption for inference and training of models on the edge-cloud continuum. We aim to investigate continual learning techniques for sustainable IoT including the batteryless edge [52], [53]. Our plan also includes tackling the challenge of continual learning in the context of the edge-cloud continuum, where diverse data storage and computational capabilities demand swift and efficient coordination of training, inference, and knowledge aggregation to attain optimal performance while meeting Green AI criteria.

Acknowledgment

The work has been conducted as part of the InterQ project (958357) funded by the European Commission within the Horizon 2020 research and innovation programme.

References

[1] S. Sen, E. J. Husom, A. Goknil, S. Tverdal, P. Nguyen, and I. Mancisidor, "Taming data quality in ai-enabled industrial internet of things," *IEEE Software*, 2022.

[2] M. Isaja, P. Nguyen, A. Goknil, S. Sen, E. J. Husom, S. Tverdal, A. Anand, Y. Jiang, K. J. Pedersen, P. Myrseth *et al.*, "A blockchain-based framework for trusted quality data sharing towards zero-defect manufacturing," *Computers in Industry*, vol. 146, p. 103853, 2023.

[3] J. Zenisek, F. Holzinger, and M. Affenzeller, "Machine learning based concept drift detection for predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106031, 2019.

[4] B. Zhou, T. Pychynski, M. Reischl, E. Kharlamov, and R. Mikut, "Machine learning with domain knowledge for predictive quality monitoring in resistance spot welding," *Journal of Intelligent Manufacturing*, vol. 33, no. 4, pp. 1139–1163, 2022.

[5] T. Ahmad, H. Zhu, D. Zhang, R. Tariq, A. Bassam, F. Ullah, A. S. AlGhamdi, and S. S. Alshamrani, "Energetics systems and artificial intelligence: Applications of industry 4.0," *Energy Reports*, vol. 8, pp. 334–361, 2022.

[6] T. L. Hayes, G. P. Krishnan, M. Bazhenov, H. T. Siegelmann, T. J. Sejnowski, and C. Kanan, "Replay in deep learning: Current approaches and missing biological elements," *Neural Computation*, vol. 33, no. 11, pp. 2908–2950, 2021.

[7] B. Maschler, T. T. H. Pham, and M. Weyrich, "Regularization-based continual learning for anomaly detection in discrete manufacturing," *Procedia CIRP*, vol. 104, pp. 452–457, 2021.

[8] B. Maschler, H. Vietz, N. Jazdi, and M. Weyrich, "Continual learning of fault prediction for turbofan engines using deep learning with elastic weight consolidation," in *ETFA'20*, vol. 1, 2020, pp. 959–966.

[9] H. Tercan, P. Deibert, and T. Meisen, "Continual learning of neural networks for quality prediction in production using memory aware synapses and weight transfer," *Journal of Intelligent Manufacturing*, vol. 33, no. 1, pp. 283–292, 2022.

[10] J. Hua, Y. Li, W. Mou, and C. Liu, "An accurate cutting tool wear prediction method under different cutting conditions based on continual learning," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 236, no. 1-2, pp. 123–131, 2022.

[11] J. Bosch, H. H. Olsson, and I. Crnkovic, "Engineering AI systems: A research agenda," in *Artificial Intelligence Paradigms for Smart Cyber-Physical Systems*, 2021, pp. 1–19.

[12] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu, "Embracing change: Continual learning in deep neural networks," *Trends in cognitive sciences*, vol. 24, no. 12, pp. 1028–1040, 2020.

[13] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[14] W. C. Abraham and A. Robins, "Memory retention–the synaptic stability versus plasticity dilemma," *Trends in neurosciences*, vol. 28, no. 2, pp. 73–78, 2005.

[15] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *CVPR'17*, 2017, pp. 2001–2010.

[16] P. Mazumder, P. Singh, and P. Rai, "Few-shot lifelong learning," in *AAAI-21*, vol. 35, no. 3, 2021, pp. 2337–2345.

[17] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari, "End-to-end incremental learning," in *ECCV'18*, 2018, pp. 233–248.

[18] A. Ashfahani and M. Pratama, "Autonomous deep learning: Continual learning approach for dynamic environments," in *SDM'19*, 2019, pp. 666–674.

[19] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[20] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: a strong, simple baseline," *Advances in neural information processing systems*, vol. 33, pp. 15 920–15 930, 2020.

[21] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in *ICSE-SEIP'19*, 2019, pp. 291–300.

[22] J. Loeliger and M. McCullough, *Version Control with Git: Powerful tools and techniques for collaborative software development.* " O'Reilly Media, Inc.", 2012.

[23] R. Kuprieiev, D. Petrov, P. Redzyński, S. Pachhai, C. da Costa-Luis, A. Schepanovski, P. Rowlands, I. Shcheklein, J. Orpinel, F. Santos, A. Sharma, Zhanibek, Gao, B. Taskaya, D. Hodovic, A. Grigorev, Earl, N. Dash, nik123, G. Vyshnya, maykulkarni, M. Hora, Vera, S. Mangal, W. Baranowski, C. Wolff, A. Maslakov, A. Khamutov, K. Benoy, and O. Yoktan, "Dvc: Data version control - git for data & models," https://doi.org/10.5281/zenodo.4544110, Feb. 2021.

[24] iterative.ai, "Open-source version control system for machine learning projects," https://dvc.org/, Visited in 2022.

[25] T. Diethe, T. Borchert, E. Thereska, B. Balle, and N. Lawrence, "Continual learning in practice," *arXiv preprint arXiv:1903.05202*, 2019.

[26] S. I. Mirzadeh, A. Chaudhry, D. Yin, T. Nguyen, R. Pascanu, D. Gorur, and M. Farajtabar, "Architecture matters in continual learning," *arXiv preprint arXiv:2202.00275*, 2022.

[27] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & compress: A scalable framework for continual learning," in *ICML'18*, 2018, pp. 4528–4537.

[28] L. Yang and A. Shami, "Iot data analytics in dynamic environments: From an automated machine learning perspective," *Engineering Applications of Artificial Intelligence*, vol. 116, p. 105366, 2022.

[29] W. Sun, J. Liu, and Y. Yue, "Ai-enhanced offloading in edge computing: When machine learning meets industrial iot," *IEEE Network*, vol. 33, no. 5, pp. 68–74, 2019.

[30] P. Bellavista, R. Della Penna, L. Foschini, and D. Scotece, "Machine learning for predictive diagnostics at the edge: An IIoT practical example," in *ICC'20*, 2020, pp. 1–7.

[31] A. Angelopoulos, E. T. Michailidis, N. Nomikos, P. Trakadas, A. Hatziefremidis, S. Voliotis, and T. Zahariadis, "Tackling faults in the industry 4.0 era—a survey of machine-learning solutions and key aspects," *Sensors*, vol. 20, no. 1, p. 109, 2020.

[32] M. Wocker, N. K. Betz, C. Feuersänger, A. Lindworsky, and J. Deuse, "Unsupervised learning for opportunistic maintenance optimization in flexible manufacturing systems," *Procedia CIRP*, vol. 93, pp. 1025–1030, 2020.

[33] C.-Y. Chen, S.-C. Chang, and D.-Y. Liao, "Equipment anomaly detection for semiconductor manufacturing by exploiting unsupervised learning from sensory data," *Sensors*, vol. 20, no. 19, p. 5650, 2020.

[34] E. Wescoat, M. Krugh, A. Henderson, J. Goodnough, and L. Mears, "Vibration analysis utilizing unsupervised learning," *Procedia Manufacturing*, vol. 34, pp. 876–884, 2019.

[35] E. J. Husom, S. Tverdal, A. Goknil, and S. Sen, "Udava: an unsupervised learning pipeline for sensor data validation in manufacturing," in *Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI*, 2022, pp. 159–169.

[36] D. N. Reshef, Y. A. Reshef, H. K. Finucane, S. R. Grossman, G. McVean, P. J. Turnbaugh, E. S. Lander, M. Mitzenmacher, and P. C. Sabeti, "Detecting novel associations in large data sets," *science*, vol. 334, no. 6062, pp. 1518–1524, 2011.

[37] P. Sedgwick, "Pearson's correlation coefficient," *Bmj*, vol. 345, 2012.

[38] C. H. Lubba, S. S. Sethi, P. Knaute, S. R. Schultz, B. D. Fulcher, and N. S. Jones, "catch22: Canonical time-series characteristics," *Data Mining and Knowledge Discovery*, vol. 33, no. 6, pp. 1821–1852, 2019.

[39] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*, 2012, pp. 9–48.

[40] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, vol. 19, pp. 143–155, 1989.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] M.-A. Tnani, M. Feil, and K. Diepold, "Smart data collection system for brownfield cnc milling machines: A new benchmark dataset for data-driven machine monitoring," *Procedia CIRP*, vol. 107, pp. 131–136, 2022.

[43] Bosch Connected Devices and Solutions GmbH, https://www.bosch-connectivity.com/products/industry-4-0/connected-industrial-sensor-solution/downloads/, Visited in 2022.

[44] *BETRIEBSANLEITUNG*, INDEX-Werke GmbH & Co. KG, 2006.

[45] Great Expectations, https://greatexpectations.io/, Visited in 2022.

[46] A. New, M. Baker, E. Nguyen, and G. Vallabha, "Lifelong learning metrics," *arXiv preprint arXiv:2201.08278*, 2022.

[47] V. Kumar, G. S. Lalotra, P. Sasikala, D. S. Rajput, R. Kaluri, K. Lakshmanna, M. Shorfuzzaman, A. Alsufyani, and M. Uddin, "Addressing binary classification over class imbalanced clinical datasets using computationally intelligent techniques," *Healthcare*, 2022. [Online]. Available: https://www.mdpi.com/2227-9032/10/7/1293

[48] N. Jourdan, S. Sen, E. J. Husom, E. Garcia-Ceja, T. Biegel, and J. Metternich, “On the reliability of machine learning applications in manufacturing environments,” in *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

[49] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher *et al.*, “A survey of uncertainty in deep neural networks,” *arXiv preprint arXiv:2107.03342*, 2021.

[50] C. Lattner, M. Amini, U. Bondhugula, A. Cohen, A. Davis, J. Pienaar, R. Riddle, T. Shpeisman, N. Vasilache, and O. Zinenko, “Mlir: Scaling compiler infrastructure for domain specific computation,” in *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. IEEE, 2021, pp. 2–14.

[51] C. Lattner, “Llvm and clang: Next generation compiler technology,” in *The BSD conference*, vol. 5, 2008, pp. 1–20.

[52] A. Goknil and K. S. Yildirim, “Toward sustainable iot applications: Unique challenges for programming the batteryless edge,” *IEEE Software*, vol. 39, no. 5, pp. 92–100, 2022.

[53] F. Erata, E. Yildiz, A. Goknil, K. S. Yildirim, J. Szefer, R. Piskac, and G. Sezgin, “Etap: Energy-aware timing analysis of intermittent programs,” *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 2, pp. 1–31, 2023.