REPTILE: a Tool for Replay-driven Continual Learning in IIoT

Erik Johannes Husom, Sagar Sen, Arda Goknil, Simeon Tverdal, Phu Nguyen
SINTEF Digital
Oslo, Norway
firstname.lastname@sintef.no

ABSTRACT

We present an automated Machine Learning (ML) tool designed as a continual learning pipeline to adapt to evolving data streams in the Industrial Internet of Things (IIoT). This tool creates ML experiences, starting with training a neural network model. It then iteratively refines this model using fresh data while judiciously replaying pertinent historical data segments. When applied to IIoT sensor data, our tool ensures sustained ML performance amid evolving data dynamics while preventing the undue accumulation of obsolete sensor data. We have successfully assessed our tool across three industrial datasets and affirm its efficacy in dynamic knowledge retention and adaptation.

KEYWORDS

Machine Learning, Industrial Internet of Things

ACM Reference Format:
Erik Johannes Husom, Sagar Sen, Arda Goknil, Simeon Tverdal, Phu Nguyen. 2023. REPTILE: a Tool for Replay-driven Continual Learning in IIoT. In *Proceedings of IoT Conference (IoT'23).* ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/xx.xxx/xxx_x

1 INTRODUCTION

The Industrial Internet of Things (IIoT) has ushered in a profound revolution in industrial processes, fundamentally altering the way industries operate. By facilitating the seamless collection and analysis of substantial amounts of sensor-generated data, this transformative technology has paved the way for previously unimaginable advancements. Nevertheless, within the dynamic and rapidly evolving IIoT ecosystem, the imperative to continually adapt and acquire knowledge from incoming data streams cannot be overstated. This adaptability is the linchpin for sustaining peak performance and accuracy in machine learning (ML) models. As the IIoT landscape continually reshapes itself, models must evolve alongside it, ensuring that they remain finely tuned to the intricacies of the data.

The prevailing approach in ML applications for IIoT involves learning from large, unchanging sets of multivariate sensor data to predict vital attributes like faults, quality, and energy consumption. However, the reality of IIoT is dynamic, with data evolving due to factors like tool replacement, manufacturing variations, and environmental shifts. Relying solely on static datasets is untenable

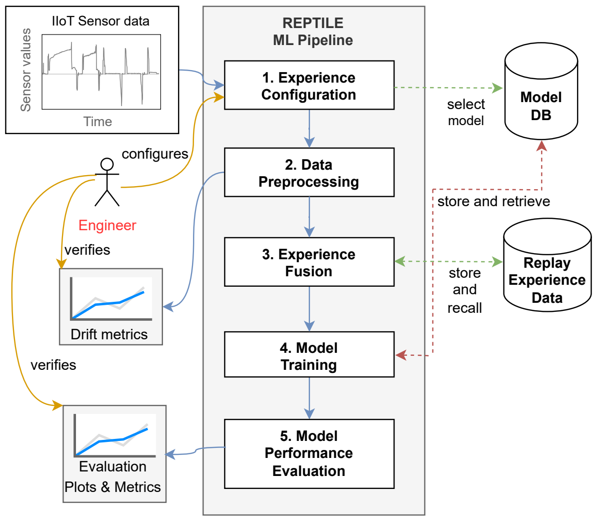

Figure 1: REPTILE Tool Overview.

for accurate predictions. Additionally, perpetual storage of high-volume, high-velocity IIoT data is unsustainable, and retraining models with an ever-expanding dataset is computationally expensive. Adaptation to the dynamic nature of IIoT data is imperative for sustainable and precise ML applications.

This paper introduces REPTILE (REPlay-driven conTInual LEarning for the Industrial Internet of Things), an innovative continual learning tool crafted to glean knowledge from dynamically evolving data streams, judiciously reutilizing pertinent historical data segments. REPTILE combats catastrophic forgetting (forgetting information from previous tasks while learning new ones) by incorporating replay-driven learning experiences, which selectively reintroduce past data alongside new information, thus preserving and updating model knowledge over time. It is conceived as a well-structured pipeline configured to orchestrate ML experiences. This configuration entails the initial training of a baseline neural network model, followed by iterative enhancements leveraging new data while selectively replaying historically significant data fragments. The culmination of these processes yields specific model versions tailored to the ever-unfolding nuances of incoming IIoT data, allowing for precise inferential and predictive capabilities.

2 RELATED WORK

IIoT data originates from a dynamic environment, demanding adaptations to maintain ML model prediction performance. Some researchers have applied ML to IIoT data at the network edge. For instance, Sun et al. [8] introduced an AI-enhanced offloading scheme for edge processing of IIoT data. Bellavista et al. [1] proposed an architecture for ML model deployment on the edge-cloud continuum, involving model updates at the edge based on local data and

IoT'23, November 7 –10, 2023, Nagoya, Japan
© 2023 Association for Computing Machinery.
ACM ISBN 978-1-4503-9517-5/23/03…$15.00
https://doi.org/xx.xxx/xxx_x

subsequent data aggregation in the cloud. However, this approach doesn't address catastrophic forgetting or knowledge retention.

Several continual learning methods have been explored in the context of IIoT, such as regularization-based approaches by Maschler et al. [5, 6] and memory-aware synapse cloning by Tercan et al. [9]. Yet, no methods discuss the deployment of continual learning in industrial settings or offer a versatile pipeline for various prediction tasks. The use of replay as a simple yet effective strategy to mitigate catastrophic forgetting in IIoT data remains unexplored.

3 TOOL OVERVIEW

REPTILE is the tool supporting our approach recently described in our research paper [7]. It is designed and implemented as an ML pipeline. Figure 1 presents the tool overview. REPTILE ingests multivariate time series data from an IIoT system, employing diverse sensors to monitor and regulate physical processes, such as CNC milling or grinding. This data includes target variables for quality measurement. Initially, the pipeline generates a baseline ML model using a static dataset extracted from raw data. The dataset has input and target variables. REPTILE can subsequently update the ML model through replay-driven learning experiences overseen by an engineer. When model performance aligns with expectations and data drift is minimal, the engineer can deploy the model as an ML service for tasks like anomaly detection, quality assessment, or equipment performance prediction in IIoT. REPTILE has five steps.

3.1 Experience Configuration (Step 1)

The engineer customizes learning and inference experiences. Three experiences can be invoked: *baseline learning*, *inference*, and *replay-driven learning*. Configurable parameters encompass experience type, feature extraction from input time series, train/test data split, the volume of data replayed with new data, the selection of ML models from the database, and model evaluation criteria.

The baseline learning experience entails training an ML model with a static dataset, involving data preprocessing, model training, experience fusion, and model evaluation (as depicted in Figure 1). When selected in the experience configuration step, REPTILE utilizes sensor data from the industrial environment. This training dataset must include the target variable for prediction. The process involves iterative refinement of control parameters, ensuring the model achieves satisfactory performance.

An inference experience involves utilizing an existing ML model to make predictions from IIoT data without engaging in model training. The pipeline operates by running the model against a new data batch. The inference service is configured to access the raw data queue, extracting sensor variables for a specified time frame and organizing them into a batch dataset. The engineer then specifies the model identifier in the Model DB. REPTILE processes the data, excluding some substeps of data preprocessing (e.g., profiling and data splitting) required for training. The configuration activates components for data cleaning, feature extraction, model retrieval from Model DB, prediction generation, and communication of results to the IIoT system (either through messages to other services or display on a user interface). Inference experiences are employed when learning experiences are not ongoing, signifying model stability under prevailing conditions.

The replay-driven learning experience is configured to employ an existing model from the Model DB alongside new data and a portion of stored historical data, as depicted in Figure 1. New and past data must adhere to the same schema for input and target variables, as defined in the configuration. The configuration further delineates the quantity of new data to be retained for future use. New data is sourced from the raw data queue and undergoes preprocessing for training purposes. Subsequently, *the Experience Fusion step* is activated, amalgamating the replay experience data (comprising part of the historical data for replay) with the new data. This combined dataset is then employed to train an existing ML model, as specified by a Model identifier in the configuration. The engineer assesses the performance of the new ML model, and if deemed satisfactory, proceeds to initiate the inference experience with the updated model.

3.2 Data Preprocessing (Step 2)

IIoT's multivariate sensor data enters a raw data queue using message queue software like MQTT or Apache Kafka. Each message includes a timestamp, variable, and value. To create a dataset for an experience, the configuration designates a timestamp range and a set of sensor variables. In addition, the experience configuration can specify which data to delete as it's consumed for learning or inference. This dataset undergoes standard ML pipeline substeps: *data profiling*, *cleaning*, *feature engineering*, *test/train data splitting*, *scaling*, and *sequentialization* (breaking data into sub-sequences).

3.3 Experience Fusion (Step 3)

This step involves amalgamating newly acquired data with historical data from previous experiences. It enables storing and retrieving a portion of the dataset employed within the pipeline for replay purposes. When a model is generated or updated, part of the dataset can be designated to be preserved as replay experience data and further utilized in subsequent replay-driven learning experiences. Specific subsets of the datasets processed through the pipeline are replayed to prevent the detrimental effects of catastrophic forgetting (i.e., the failure of stability, in which new experience overwrites the previous experience). When model updates occur, the recalled replay experience data is seamlessly integrated into the dataset and subsequently incorporated into the model training step.

3.4 Model Training (Step 4)

The input and output sub-sequences play a pivotal role in configuring and training the ML model within the REPTILE tool. REPTILE provides the capability to specify learning parameters and choose from various ML model architectures, including Dense Neural Networks (DNNs)/Fully Connected Neural Networks (FCNNs), Convolutional Neural Networks (CNNs) [4], and Long Short-Term Memory (LSTM) [2]. A portion of the training dataset, typically around 20%, is reserved for use as a validation dataset before the actual training process commences. REPTILE incorporates an automated mechanism that halts training if the validation set's prediction error ceases to improve, thereby preventing model overfitting. The resulting ML model is stored in the Model Database (Model DB) for subsequent evaluation. Engineers have the flexibility to configure

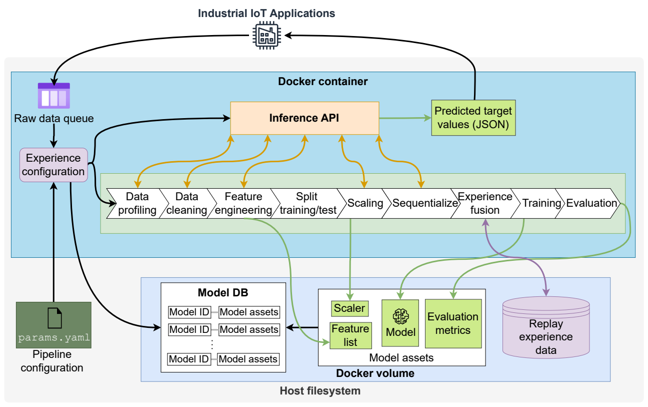

Figure 2: Deployment of REPTILE as a Docker Container.

REPTILE to either initiate model creation from scratch or select any prior model from the Model DB as a starting point.

3.5 Model Performance Evaluation (Step 5)

The test dataset (an independent and previously unseen dataset) serves as a critical tool for evaluating model performance while mitigating potential bias stemming from hyperparameter adjustments during model training. This evaluation process involves a comparison between the model's predictions and the actual ground truth values, offering insights into the model's accuracy in predicting the target variable. To facilitate this assessment, REPTILE generates prediction plots on the test dataset. For regression models, metrics such as Mean Squared Error (MSE), coefficient of determination (R^2 score), and Mean Absolute Percentage Error (MAPE) are employed to gauge performance, while classification models are assessed using metrics like accuracy and F1-score. These metrics provide quantitative measures of how well the model aligns with the real-world data it aims to predict.

4 IMPLEMENTATION & AVAILABILITY

REPTILE is enclosed within a Docker container, providing portability and effortless deployment compatibility across diverse IIoT ecosystems. This container harmoniously combines the training and inference components, as depicted in Figure 2. Docker containers furnish a uniform and self-contained environment, facilitating the installation and configuration of REPTILE on an array of hardware and software platforms. This installation ensures consistent performance and adaptability, underscoring the framework's versatility in addressing the requirements of varied IIoT environments.

IIoT raw data enters the container via a message queue and can be employed for model training and inference, contingent on the presence of valid target values. Configuration details for the pipeline, encompassing control parameters and experience fusion settings, are conveyed to the Docker container via a YAML file. When creating models from scratch, iterative experimentation and parameter refinement are typically required, necessitating multiple pipeline runs with YAML file adjustments. Intermediate data is cached using DVC [3], enabling automatic stage skipping if prior runs employ the same parameters, thereby optimizing computational efficiency and resource utilization.

Within the Docker container, an Inference API is deployed, serving real-time applications by furnishing an interface that enables IIoT systems to query the pre-trained model. The API orchestrates the requisite pipeline steps for input data preprocessing and leverages the trained model to furnish predicted target values. This setup facilitates seamless interaction between IIoT systems and the model, streamlining real-time decision-making processes.

REPTILE manages storage and data through a combination of file-based storage and DVC. This integrated approach establishes dedicated directories within the application to store raw data, models, and essential metadata, creating a structured hierarchy for efficient data access and retrieval. DVC further enhances this setup by tracking changes to these assets and maintaining a version-controlled history of experiments conducted with the tool. This integration preserves a clear separation between REPTILE's core functionalities and underlying data storage processes, ensuring organized and streamlined data management.

Docker technology is employed to encapsulate the API along with its requisite dependencies, creating a self-contained and portable unit for uniform deployment across varied platforms. When the container is operational, Docker volumes facilitate the enduring storage of essential assets and data files. This configuration ensures the continued availability and editability of data on the host device, even after the container has concluded its execution.

5 EVALUATION

This section presents key findings from the evaluation conducted to address the following research questions [7].

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

We assessed REPTILE using three IIoT datasets (obtained from the Bosch CNC machining of aluminum workpieces, the broaching of aeroengine discs, and the manufacturing of piston rods) for three different predictive tasks (predicting tool wear, remaining useful lifetime, and anomalies from sensor data).

To address *RQ1*, we computed 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 aeroengine discs and time until the tool change during the manufacturing of piston rods. Table 1 lists the R^2 scores for the baseline model evaluated across five sub-datasets representing sensor data from broaching turbine disc slots.

Table 1: 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

As broaching progresses sequentially, the model's performance worsens on more recent data. The baseline model, trained on the earliest slots (Slot Set 1), performs poorly when tested on the latest

Table 2: Performance scores of the replay models trained on Bosch CNC dataset.

Data	0% replay		20% replay		60% replay		100% replay	
F1-sc.:	past	new	past	new	past	new	past	new
M 19	0.681	0.593	0.620	0.454	0.657	0.443	0.690	0.450
M 20	0.005	NC	0.659	NC	0.619	NC	0.718	NC
M 21	0.002	0.000	0.571	0.778	0.628	0.747	0.672	0.762
M 21	0.485	0.689	0.644	0.830	0.630	0.784	0.751	0.904

slots. Tool wear gradually increases in each slot, shifting beyond the model's training distribution. The R^2 score dips below zero, indicating that the model struggles even to outperform a basic estimator predicting the average target value. The initial baseline model performance is unsatisfactory, and it quickly becomes obsolete with each subsequent dataset, rendering it unusable.

The baseline models examined in the case studies exhibited a noticeable decline in predictive performance on more recent data. This phenomenon signifies a crucial observation: if left static, ML models tend to become less effective at capturing the evolving patterns in the IIoT data they aim to predict. The implications of these findings underscore the necessity of implementing continual learning techniques to ensure that the models remain adaptable and up-to-date. In other words, continual learning becomes imperative to counteract the inherent model performance degradation over time. By periodically updating the models with new data and preserving past knowledge, we can mitigate the risk of their predictive capabilities deteriorating as the data landscape evolves.

To answer *RQ2*, we conducted a performance comparison of replay models trained on our datasets using varying replay percentages. Replay involves merging a specified percentage of historical data from a replay experience reservoir with fresh IIoT data to retrain an existing ML model from the Model DB. This process is only initiated when we detect drift and subpar ML model performance during inference experiences. Table 2 summarizes the performance of replay-driven continual learning for the Bosch dataset. It provides the identifier for a stream of incoming new datasets, the historical performance (F1-score) of recent datasets, and the performance on the segment of the new dataset designated for testing.

In the Bosch CNC dataset, rare events or anomalies are infrequent. All levels of replay effectively predict anomalies in new CNC machining data. However, F1-score calculation was not feasible (NC) for one dataset (M 20) as no anomalous vibrations occurred. Using 0% replay led to a substantial F1-score decline due to limited anomaly data, seemingly rendering the model unusable. Yet, training on the most recent dataset with more anomalies restored model performance on prior data. All other replay levels (20%-100%) prevented catastrophic forgetting and aided in detecting anomalous vibrations by reintroducing older data segments.

Our experiments indicate that employing a 20% replay mechanism is sufficient to uphold commendable performance in continual learning while mitigating the risk of catastrophic forgetting. Nevertheless, both continual learning and traditional ML with 100% replay exhibit subpar performance when confronted with new, unforeseen data that likely deviates from previously encountered distributions.

To address *RQ3*, we analyzed REPTILE's deployment in industrial environments. DVC [3] simplifies replay data and model version management for experiments but faces challenges in cyclic continual learning. Our custom pipeline empowers engineers to specify data and model locations, diverging from DVC's automated approach. REPTILE, deployable in a Docker container, manages batch data from periodic or streaming sources, including IIoT systems. Engineers configure experiences for learning data extraction, monitor drift, and decide between baseline training, replay-driven training, or inference for stable performance. This approach differs from traditional static model deployment. Continual learning has challenges in data storage, drift detection, and model management. Managing data storage on resource-constrained systems and detecting drift require careful policies. Bayesian neural networks can automate decisions, but data and model accumulation require periodic maintenance. The auditability of continual learning remains an open challenge. ML models from a Model DB are integrated into a callable service for inference. Engineers specify the model using a Model ID in the configuration, enabling the pipeline to process incoming data. This approach also differs from traditional ML, where models lack access to evolving model databases.

6 CONCLUSION

We introduced REPTILE, a tool for replay-driven continual learning to adapt to evolving data streams in IIoT. The tool's main features include: (1) replaying selected portions of historical data alongside new data, enabling ML models to adapt and improve as new information becomes available, (2) the dynamic updating of ML models based on changing data patterns, ensuring that models remain accurate and effective in evolving IIoT environments, and (3) a high degree of configurability, allowing engineers to define and customize various aspects of the learning and inference experiences.

ACKNOWLEDGMENTS

The work has been conducted as part of the InterQ project (958357) and the DAT4.ZERO project (958363) funded by the European Commission within the H2020 Programme.

REFERENCES

[1] Paolo Bellavista, Roberto Della Penna, Luca Foschini, and Domenico Scotece. 2020. Machine learning for predictive diagnostics at the edge: An IIoT practical example. In *ICC'20*. 1–7.
[2] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[3] iterative.ai. Visited in 2022. Open-source Version Control System for Machine Learning Projects. https://dvc.org/.
[4] Yann LeCun et al. 1989. Generalization and network design strategies. *Connectionism in perspective* 19 (1989), 143–155.
[5] Benjamin Maschler, Thi Thu Huong Pham, and Michael Weyrich. 2021. Regularization-based Continual Learning for Anomaly Detection in Discrete Manufacturing. *Procedia CIRP* 104 (2021), 452–457.
[6] Benjamin Maschler, Hannes Vietz, Nasser Jazdi, and Michael Weyrich. 2020. Continual learning of fault prediction for turbofan engines using deep learning with elastic weight consolidation. In *ETFA'20*, Vol. 1. 959–966.
[7] Sagar Sen, Simon Myklebust Nielsen, Erik Johannes Husom, Arda Goknil, Simeon Tverdal, and Leonardo Sastoque Pinilla. 2023. Replay-driven continual learning for the industrial internet of things. In *CAIN'23*. IEEE, 43–55.
[8] Wen Sun, Jiajia Liu, and Yanlin Yue. 2019. AI-enhanced offloading in edge computing: When machine learning meets industrial IoT. *IEEE Network* 33, 5 (2019), 68–74.
[9] Hasan Tercan, Philipp Deibert, and Tobias Meisen. 2022. Continual learning of neural networks for quality prediction in production using memory aware synapses and weight transfer. *Journal of Intelligent Manufacturing* 33, 1 (2022), 283–292.