TUDelft

Delft University of Technology

Sustainable Machine Learning Retraining

Optimizing Energy Efficiency Without Compromising Accuracy

Poenaru-Olaru, Lorena; Sallou, June; Cruz, Luis; Rellermeyer, Jan S.; Van Deursen, Arie

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Sustainable Machine Learning Retraining: Optimizing Energy Efficiency Without Compromising Accuracy

Lorena Poenaru-Olaru*, June Sallou†, Luis Cruz*, Jan S. Rellermeyer‡, Arie van Deursen*

*Software Engineering Research Group, *Delft University of Technology*, Delft, The Netherlands
{L.Poenaru-Olaru, l.cruz, Arie.vanDeursen}@tudelft.nl
†*Information Technology Group*, *Wageningen University & Research*, Wageningen, The Netherlands
{june.sallou}@wur.nl
‡Dependable and Scalable Software Systems, *Leibniz University Hannover*, Hannover, Germany
{rellermeyer}@vss.uni-hannover.de

Abstract—**The reliability of machine learning (ML) software systems is heavily influenced by changes in data over time. For that reason, ML systems require regular maintenance, typically based on model retraining. However, retraining requires significant computational demand, which makes it energy-intensive and raises concerns about its environmental impact. To understand which retraining techniques should be considered when designing sustainable ML applications, in this work, we study the energy consumption of common retraining techniques. Since the accuracy of ML systems is also essential, we compare retraining techniques in terms of both energy efficiency and accuracy. We showcase that retraining with only the most recent data compared to all available data reduces energy consumption by up to 25%, being a sustainable alternative to the status quo. Furthermore, our findings show that retraining a model only when there is evidence that updates are necessary, rather than on a fixed schedule, can reduce energy consumption by up to 40%, provided a reliable data change detector is in place. Our findings pave the way for better recommendations for ML practitioners, guiding them toward more energy-efficient retraining techniques when designing sustainable ML software systems.**

Index Terms—**sustainable AI monitoring, Green AI, sustainable retraining, ML-enabled systems**

I. INTRODUCTION

The increasing adoption of Machine Learning (ML) and Artificial Intelligence (AI) within organizations has resulted in the development of more ML/AI software systems [1]. Although ML/AI brings plenty of business value, it is known that the accuracy of ML applications decreases over time [2]. Thus, ML developers must monitor and maintain their ML systems in production. One reason for this phenomenon is the fact that ML applications are highly dependent on the data on which they have been trained. Real-world data usually changes over time [3] – a phenomenon often referred to as concept drift [4] – which can significantly impact the normal operation of ML systems [5]. Therefore, appropriate maintenance techniques are required for the design of ML software systems. One common approach to maintaining these systems is to periodically update these applications by retrain-

ing the underlying ML models with the latest version of the data [6], [7].

On another note, the process of training machine learning models has raised substantial concerns about the carbon footprint of ML applications [8], [9]. For this reason, regularly retraining ML applications to preserve their accuracy, implies considerable energy consumption [10]. On the other hand, not retraining ML applications at all can severely impact their performance over time [3], affecting their reliability in practice. Therefore, there is a need for sustainable ML retraining methods that reduce the environmental impact of ML applications [1], [10].

Previous research [11] has explored the energy consumed during training and inference of continuous learning, which are ML applications where a machine learning (ML) model learns incrementally without requiring full retraining. Continual learning ML applications present significant challenges when incorporating domain experts' feedback into the ML model [4]. Consequently, a continual learning approach is not suitable for certain applications. A better solution is periodically retraining ML models after accumulating sufficient new samples. However, to the best of our knowledge, the effects of retraining an ML model using various retraining techniques on both accuracy and energy consumption have not been thoroughly examined. Therefore, this study aims to examine the effects of various retraining techniques on accuracy and energy consumption in real-world ML applications, to provide best practices for designing sustainable ML systems. We analyze these retraining techniques through the lens of two perspectives, namely the *data perspective* and the *frequency perspective* [10]. The *data perspective* addresses the impact of the data included in the retraining process, while the *frequency perspective* focuses on the impact of the frequency at which a model is retrained. Moreover, since some ML applications also require substantial energy for inference [12], we also explore whether the retraining strategy affects the energy consumption of the inference tasks.

In this empirical study, we use failure prediction applica-

100

tions as the case study ML application. Failure prediction applications are part of the AIOps research domain and are ML applications that aim to identify failures in large and complex software systems in order to improve their efficiency and reliability [13]. We specifically select failure prediction applications since periodic retraining is commonly applied to keep models up to date and these applications require large datasets, making them considerably energy intensive in practical settings [6], [7], [14]–[16]. We argue that an efficient model monitoring and retraining strategy can lead to significant improvements in maintaining these models. Hence, we employ three failure prediction models previously presented in the literature, which are built using three open-source real-world datasets.

Previous research has compared retraining failure prediction models using all available data versus only the most recent data, with both approaches maintaining performance over time [7]. However, retraining on all data is likely to consume more energy [10], [17]. Our study empirically evaluates the energy consumption and compares the accuracy and energy efficiency of these two retraining techniques.

Related work also suggests that periodic retraining is less sustainable compared to drift-based approaches [1], [10]. However, drift detectors consume energy [18] and can be overly sensitive [19], triggering unnecessary retraining and increasing energy consumption. Hence, in this work, we quantify the impact of drift detection-based retraining on the energy efficiency of AI-driven failure prediction systems.

The main contribution of this study is a quantitative empirical study of the impact of different retraining techniques on the trade-off between accuracy and energy consumed by ML systems during both training and inference. Our research shows that drift-based retraining approaches reduce the energy consumption of failure prediction models over time. However, the gains in energy efficiency are highly dependent on the choice of a drift detector. All experiments and analyses are openly available in a replication package[1].

II. BACKGROUND

In this section, we introduce the background knowledge on which the remainder of the paper builds, i.e., AIOps and failure prediction, concept drift, and retraining approaches.

A. AIOps

Large-scale software systems generate vast amounts of operational data, making manual analysis and inspection impractical [13]. The term AIOps, introduced by Gartner [20], refers to applying ML techniques to automate this process [6]. AIOps applications are ML applications used to monitor large systems and enhance software delivery, compliance, quality, and security in organizations [13]. A recent survey identifies four key AIOps categories: root-cause analysis, incident detection, failure prediction, and automated actions [21]. This work focuses on failure prediction, specifically analyzing two ML

applications: disk failure prediction and job failure prediction, which learn patterns from past failures to anticipate hardware failure (disk) or software failure (job). It has been previously shown [6], [7], [14], [22], [23] that these applications are severely affected by concept drift and that they employ large amounts of data to train.

B. Concept Drift

The term concept drift refers to changes in the data over time [24]. It is a ubiquitous phenomenon in real-world data, as data changes are generated by uncontrollable external factors [19]. Concept drift can severely impact the accuracy/performance of ML models over time since the ML algorithms used to build these models work under the assumption that the distribution of the data data learned during the training process should be similar to the distribution of the data on which the model is evaluated [3]. However, in the real world, this assumption often does not hold, since data are continuously changing, which can lead to noticeable drops in the model's accuracy over time.

Continuous model update/retrain is a commonly known technique to mitigate the effects of concept drift on ML models over time [4], [25], [26].

From the **retraining frequency perspective** researchers [4] have distinguished between two retraining techniques, namely *periodic retraining* and *informed retraining*. Periodic retraining implies that models are retrained based on a predefined period, while informed retraining means that a model is retrained based on a data monitoring tool called a concept drift detector. In the latter situation, the concept drift detector evaluates whether the training data becomes significantly different than the data the model is evaluated on [19].

From the **retraining data perspective**, there are two techniques previously presented in the literature, namely the *sliding window* and the *full-history* retraining approach [7], [14], [27]. The sliding window approach implies that the model is retrained only on the newest data, discarding old samples, while with a full-history retraining approach, the model is retrained on all the available data until a certain point in time. Therefore, the latter retraining technique constantly enriches the training dataset with new samples once they become available.

III. RELATED WORK

A. Concept Drift in AIOps Applications

Previous work has shown that multiple ML applications within AIOps, including failure prediction model, have been affected by concept drift [6], [13], [15], [16], [27], [28]. In failure prediction applications, concept drift can be caused by different external factors, such as feature updates, user workloads, or software/hardware updates [15]. These factors can affect the behavior of the data over time, which usually impacts the accuracy of failure prediction models. This aspect makes concept drift a serious threat to the trustworthiness of failure prediction models, especially when their output is used in decision-making processes [13].

When it comes to AIOps applications, previous works recommend AIOps practitioners to periodically retrain failure prediction models [6], [7], [15]. From the retraining data perspective, both the sliding window [6], [7], [14]–[16], [27], [29] and full-history approaches [7], [14], [27] retraining approaches were employed to update AIOps models over time. From the retraining frequency perspective, the most commonly used retraining technique is the periodic retraining [6], [7], [14]–[16]. However, Lyu et al. [14] also experimented with informed retraining. The authors employed a supervised concept drift detector to monitor the error of the failure prediction model over time and retrained every time they observed a significant error increase. To compute the error, the ground truth, also known as true labels, is required. True labels refer to which sample is a "failure" and which sample is a "non-failure". However, in some AIOps applications, continuously monitoring the error over time might not be possible since obtaining true labels is expensive in time and resources. For these applications, to obtain the true labels operational engineers have to continuously perform root cause analysis to understand the main cause of the failure [13], [27], [28]. This consumes a significant amount of the time of operational engineers which makes acquiring the true labels significantly expensive. In this situation, employing an unsupervised drift detector that does not require computing the error using true labels would be preferred [19], [27]. Therefore, unlike previous work [14], in this paper, we analyze the effects of retraining based on an unsupervised drift detector. Furthermore, while previous work only took into account the accuracy when determining the effects of retraining based on drift detection, we also consider the consumed energy to ensure the sustainability of the failure prediction models over time.

B. Sustainability of ML Systems

The adoption of ML applications in the industry has seen considerable growth over the past years [8], [9]. Although numerous domains benefit from the use of machine learning, these models consume significant amounts of energy, raising concerns among researchers about the environmental impact of such applications [1], [8], [9], [12]. This led to the rise of the GreenAI research field, which encourages the development of ML applications that consume less energy while preserving their accuracy [30].

One key practice in building GreenAI systems is reporting the energy consumption of ML applications, which could raise awareness of their carbon footprint and help find more sustainable configurations [30]. Wu et al. [9] examined the carbon footprint of Facebook's ML applications and discovered that both training and inference have a significant contribution to the overall carbon footprint of the ML application. Plenty of research has been focused on understanding the environmental impact of training ML systems, such as the study of Xu et al. [31] targeting multiple computer vision applications. On the other hand, the work of Luccioni et al. [12] analyzed inference energy in tasks like image classification and language modeling. They concluded that although tasks involving images are more energy-intensive, model training remains significantly more carbon-intensive compared to inference.

While significant attention has been given to the energy consumption of training and inference in ML models, the energy used during the model's active production phase (model lifecycle) has received less focus. Trinci et al. [11] investigated the training and inference energy efficiency of continual learning algorithms for computer vision applications, but these algorithms are not directly applicable in the AIOps context due to the high costs of continuously gathering true labels [13] to perform continual learning. AIOps applications require ongoing model monitoring and updates, but these should occur in batches at predefined intervals rather than continuously. Although previous work [10] explored sustainable retraining techniques, there is a lack of empirical analysis demonstrating their effectiveness in maintaining model accuracy and reducing energy consumption over time. Omar et al. [18] have studied the energy consumption of multiple drift detectors with respect to their drift detection accuracy. However, the drift detectors analyzed operate in a supervised manner, relying on error rate computation after inference, which requires true labels. This approach is impractical for some AIOps applications, where acquiring true labels depends on human annotation [13]. Therefore, unsupervised drift detectors are more suitable, as they identify drift by analyzing differences between training and inference data without needing true labels. Our study differs from previous work [18] by employing unsupervised drift detectors and focusing on their ability to indicate the need to retrain a failure prediction model, rather than assessing the drift detection accuracy.

IV. Research Questions

The goal of our study is to assess the impact of different retraining techniques on the energy consumption of failure prediction models. With this aim, we address the following research questions (RQ):

RQ1. What is the impact of each retraining technique on the training energy consumption?
   a) What is the impact on energy consumption of employing the sliding window vs the full-history approach (retraining data perspective)?
   b) What is the impact on energy consumption of employing the periodic vs the informed retraining (retraining frequency perspective)?
   c) What is the best retraining technique overall?

RQ2. What is the impact of each retraining technique on the inference energy consumption?

V. Methodology and Experiments

In this section, we outline the methodology used to address the research questions and conduct our experiments. Specifically, we discuss the used datasets, the process for building the failure prediction models, the experimental design, and the energy consumption measurement.

A. Datasets

We employ the only three publicly available open-source AIOps datasets to build failure prediction models according to previous work [6], [14]: the Backblaze Disk Stats Dataset [32], the Google Cluster Traces Dataset [33], and the Alibaba GPU Cluster Trace Dataset [34].

The **Backblaze Disk Stats Dataset** has been previously used to build disk failure prediction models [6], [7], [14], [22], [23]. It contains information about operational hard drives available in the data center collected daily since 2013. The dataset includes both drive information (manufacturer, serial number, or capacity), as well as information related to early error detection extracted through a monitoring system implemented by the manufacturer, called SMART attributes (Self-Monitoring, Analysis, and Reporting Technology). Given that data collected before 2015 does not include SMART attributes and to align with previous work [6], we employ 12 months of data collected in 2015.

The **Alibaba GPU Cluster Trace Dataset** is relatively new, released in 2021 by the Alibaba Group, and, to the best of our knowledge, there has been only one work that employed it to build job failure prediction models [14]. It is composed of information regarding job execution extracted from a large-scale data center. The data is collected for a period of 2 months, from July to August 2020 [35], from approximately 6500 GPUs across around 1800 machines.

The **Google Cluster Traces Dataset** has been previously used to build job failure prediction models [6], [7], [14], [36]. The dataset includes information about jobs executed on a large-scale cluster at Google collected for 29 days in May 2011.

B. Machine Learning Models

When building the failure prediction models, i.e., disk failure prediction for the Backblaze dataset and job failure prediction for the Alibaba and Google datasets, we replicate the pipelines presented in previous work [6], [7], [14], [22], [23], [36]. Thus, we use the same features and follow the same model design strategy presented in previous studies. More details about features are specified in our replication package.

1) Labels and Features: The Backblaze dataset contains an attribute indicating whether a drive failed the day after data collection, which serves as the label for our failure prediction model. It consists of approximately 7 million samples, with 0.05% labeled as failures and 99.95% as non-failures. For the prediction model, we selected 19 temporal SMART features from previous research [6], [7], [14], [22], [23]. Of these, 11 are non-cumulative (raw) values from the last day, while 8 are cumulative (raw diff) changes over one week compared to the previous day's values.

To create the final Alibaba dataset, we remove unfinished jobs and those ending in less than five minutes, except those labeled with the status "fail." The final dataset consists of 701,000 samples, with 34.5% labeled as failures and 65.5% as non-failures. To train the job failure prediction model, we use 12 features, comprising 6 configuration (conf) features and 6

temporal (temp) features calculated over a 5-minute window since job submission.

The Google dataset lacks a direct attribute for failed or non-failed jobs. Each job (sample) can have multiple events (fail, finish, kill, submit, update, evict, schedule) and states (pending, dead, running, unsubmitted). Following previous research [6], [7], [14], a job is labeled as "fail" only if its final state is "fail." We remove jobs with incomplete records and those that finished within five minutes of submission, as they do not provide sufficient metrics for predicting failure. The final dataset includes approximately 625,000 samples, with 1.5% labeled as failures and 98.5% as non-failures. For the job failure prediction model, we use 15 features: 9 configuration (conf) features and 6 temporal (temp) features calculated over a 5-minute period since job submission.

2) Model Building Pipeline: We built three model failure prediction models, one disk failure prediction, and two job failure prediction models, corresponding to the three datasets employed in this study, Backblaze, Google, and Alibaba, respectively. To closely resemble the failure prediction models presented in previous work [6], [7], [14], [22], [23], [36] we build a monthly failure prediction model for the Backblaze dataset, a daily failure prediction model for the Google dataset and a weekly failure prediction model for the Alibaba dataset. We split each dataset in half. The first part is used to train each model, while the second part is further divided into smaller subsets, each corresponding to a specific period (day, week, or month, depending on the dataset). These subsets mimic the real-world scenario in which new data is generated periodically and are used for inference. Fig. 1 shows a detailed overview of our model-building pipeline.

The model-building pipeline is similar for all our three failure prediction models unless specified otherwise. The first step of the pipeline is preprocessing the data through scaling. Similar to previous work [6], [7], [14], we employ a standard scaler which we fit on the training data and apply on the inference data such that all values of the features range in a predefined interval.

The Backblaze and Google datasets suffer from high-class imbalance, meaning that the number of failure samples is tremendously lower compared to the number of non-failure samples. For the Alibaba dataset, the class imbalance is not as severe. Therefore, similar to previous work [6], [7], [14], we perform downsampling for the Backblaze and Google datasets obtaining a class balance ratio of 1:10. This indicates that for each failure sample, there are 10 non-failure-samples.

The last steps of the model-building pipeline are model training and inference. For this step, we employ a Random Forest classifier as it is one of the most popular classifiers used in failure prediction achieving the highest performance in distinguishing between failure and non-failure samples [6], [7], [14], [22], [23]. While training the model, hyperparameter tuning is performed through Randomized Search, similar to previous work [14]. The trained model is further used for inference.
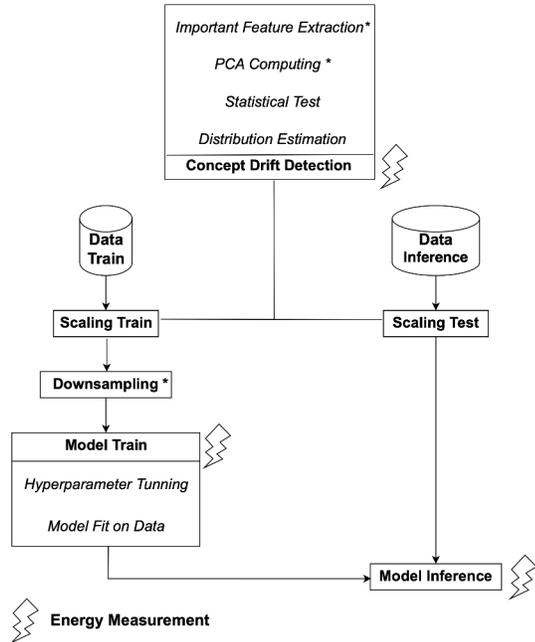
Fig. 1: Energy measurements in model pipeline. The symbol '*' indicates that the specific step was done only when applicable.

3) Failure Prediction Evaluation Metric: Similar to previous work [6], [7], [14], we employ ROC AUC to evaluate the accuracy of the failure prediction model. In this paper, we will use the terms ROC AUC and accuracy interchangeably. ROC AUC is a metric that measures the model's ability to assign higher predicted probabilities to the positive class compared to the negative class. The highest ROC AUC score is 1.0 and shows that all samples are correctly classified. We choose this metric due to the high-class imbalance in all datasets.

C. Retraining Approaches

In our experiments, we investigate the energy consumption of different retraining techniques. From the retraining data perspective, we analyze the effects of two retraining techniques, namely the *full-history approach* and the *sliding window approach* [7]. In full-history retraining, the model is retrained using all data accumulated up to the given retraining time. In sliding window retraining, at each retraining step, new data is incorporated while the oldest data, corresponding to one period, is discarded. Unlike full-history retraining, this method maintains a relatively constant training dataset size.

From the retraining frequency perspective, we analyze the effects of two retraining techniques, namely the *informed* vs *periodic* retraining [4]. In the periodic retraining approach, the model is retrained at predefined intervals, regardless of whether the data distribution has changed. To remain consistent with prior work [6], [7], [14], we perform retraining at different intervals depending on the dataset: monthly for Back-

blaze, weekly for Alibaba, and daily for Google. In contrast, the informed retraining approach triggers model updates only when a significant shift in the data distribution is detected. To identify such shifts, we employ the Kolmogorov-Smirnov (KS) statistical test [3], [19], [24], [37], a widely used unsupervised drift detection technique.

To further analyze the impact of different drift detection strategies on energy consumption, we explore multiple variations of drift detection. This process consists of two primary steps: first, estimating the distributions of both the training and inference data, and second, applying the KS test to compare these distributions. We examine three distinct approaches to drift detection. The first approach, KS-ALL, estimates distributions using all available model features. The second, KS-PCA, reduces the feature dimensionality such that 95% of the variance in the data is preserved using Principal Component Analysis (PCA) before estimating distributions. The third approach, KS-FI, selects only the most relevant features, filtering out those with a feature importance ranking below the mean, as determined by Gini importance. Both KS-PCA and KS-FI introduce a feature selection step before distribution estimation. While KS-PCA applies PCA for dimensionality reduction, KS-FI retains only the most informative features based on their importance ranking.

D. Experimental Design & Setting

In our experimentation, we study 8 retraining configurations based on two retraining perspectives: data and frequency.

We repeat our experiments to mitigate bias in our results caused by the randomness of the ML model. We allocate a one-week budget for each dataset, running each configuration multiple times. For the Backblaze and Alibaba datasets, we repeat the experiments 30 times using 30 different random seeds. The Google dataset is one of the most computationally intensive, therefore only 5 random seeds are computed. However, no significant variation in the results was observed by experimenting with different random seeds. Furthermore, when collecting energy consumption data, we shuffle the configurations to reduce the risk of background activities or factors such as temperature impacting only a category of experiments, following recommendations for energy studies [38]. All our conclusions are verified using the Wilcoxon statistical test with a confidence level of 95%.

The experiments are run on a machine with an AMD Ryzen 9 7900X processor (12 physical cores, 24 threads), 64 GB RAM (2x32GB DDR5 @ 5.600MT/s), and an MSI Geforce RTX 4090 (24GB GDDR6X memory) graphic card. The operating system is Ubuntu 22.04.3, with Linux kernel version 6.2.0.

E. Energy Consumption

We measure energy at the level of the *model pipeline* and the *model lifecycle*. The model pipeline level refers to the components of the failure prediction model that are measured. The model lifecycle refers to how the total energy consumed by a certain process is consumed throughout the lifecycle of

the machine learning model. In this subsection, we further describe which tool we employ to measure energy consumption.

1) Energy Measurement Tool: Energy measurement tools help quantify the sustainability of different experimental settings. To measure the consumed energy, we employ the CodeCarbon [39] Python package. The energy consumed by the CPU and RAM is measured through RAPL and the energy consumed by the GPU is measured through NVIDIA Management Library [31]. CodeCarbon measures the duration of each experiment (in seconds) and the consumed energy (in kWh). Furthermore, this energy measurement tool has been previously used to measure the energy consumed by drift detectors and the energy consumed during training and inference [11], [18].

2) Measuring Energy Consumption of the Model: In our experiments, we measure the energy consumption of multiple steps in the failure prediction model lifecycle as shown in Figure 1. In this work, we are solely interested in understanding the impact of different model retraining techniques on the energy consumed during the model lifecycle. Thus, we are not measuring the energy of all the steps required in building the models, such as data scaling or downsampling to balance the two classes. In our experiments, we measure three types of energy, namely *training energy*, *drift detection energy* and *inference energy*. Furthermore, in our results, we depict the cumulative values of the energy of these models. For example, the training energy is composed of the initial training of the model and energy consumed to perform retraining either periodically (if the configuration is Periodic) or when the drift detection indicates (if the configuration is drift detection based). For the Static configuration, we perform training only once and never retrain further.

The **training energy** is the total energy consumed by training/retraining. This energy measurement is composed of the energy measured during the hyperparameter tuning phase, in which the best hyperparameters are chosen to fit the training data, and the energy measured during the phase in which the best model is fitted on the training data as described in Fig. 1.

Drift detection energy refers to the total energy consumed in detecting data drift. Depending on the detector—KS-ALL, KS-PCA, or KS-FI—this energy is measured in two or three components. KS-ALL includes energy used for estimating the data distribution and applying a statistical test to determine significant changes. KS-PCA adds a third measurement for the energy spent on dimensionality reduction using PCA. Similarly, KS-FI requires three measurements, but instead of PCA, it measures the energy used to extract and filter important features.

The **inference energy** represents the energy consumed by applying the trained model to perform inference and extract the predictions.

VI. Results

In this section, we present our results and answer the research questions.

A. Impact on Training Energy

To answer RQ1, we depict our results in Fig. 2, presenting at the top the results of ROC AUC and at the bottom the total training energy combined with the total drift detection energy for each retraining technique for each dataset, when applicable. In the case of the Static model, the total training energy contains only the initial model training. When it comes to the Periodic model, there is no total drift detection energy, thus we solely depict the energy consumed during the periodical model retraining process. The drift detection energy is solely considered for the KS-ALL, KS-FI, and KS-PCA models.

1) RQ1a: Full-History vs Sliding Window Retraining: In this experiment, we evaluate the sliding window and full-history approaches in terms of model accuracy (ROC AUC) and energy consumption (joules). To enhance the generalizability of our conclusions, we assess these retraining strategies in two scenarios: periodic retraining (Periodic) and drift-based retraining using the KS-ALL, KS-FI, and KS-PCA methods. Additionally, we include a baseline scenario with a static model (Static), trained only once, to assess the overall benefits of retraining.

In Fig. 2 we show that in almost all situations the full-history approach is more energy-intensive when compared to the sliding window approach. The only exception can be observed in the situation of KS-ALL for the Alibaba dataset, where there is no significant difference (p-value of 0.67) in terms of the amount of energy consumed using a sliding window or a full-history retraining approach. This exception can be explained by the fact that the ROC AUC for KS-ALL_FH is similar to the ROC AUC for KS-ALL_SW and to the ROC AUC of the Static model. This shows that in this particular situation for both full-history and sliding window approaches the drift detector did not identify any drift, and, therefore, no retraining has been performed besides the initial model training. In all other situations, the energy consumed by retraining using a sliding window approach was statistically lower than the energy consumed by retraining using a full-history approach (p-values below 0.05). For instance, when it comes to the Periodic model, we can notice an almost 25% decrease in energy consumed while retraining using the sliding window approach vs the full-history approach.

> **RQ1a Answer**
>
> A sliding window approach consumes significantly less energy than a full-history approach.

2) RQ1b: Periodic vs Informed Retraining: From Fig. 2 we can notice that in most cases the train energy consumption is reduced when using an informed retraining (retraining based on drift detection) technique for all datasets.

When it comes to the Alibaba dataset, the energy consumed by training based on the KS-ALL drift detector is significantly similar to the energy consumed in the Static configuration. This shows that the KS-ALL drift detector does not identify any change in the data, and thus, any need for retraining,
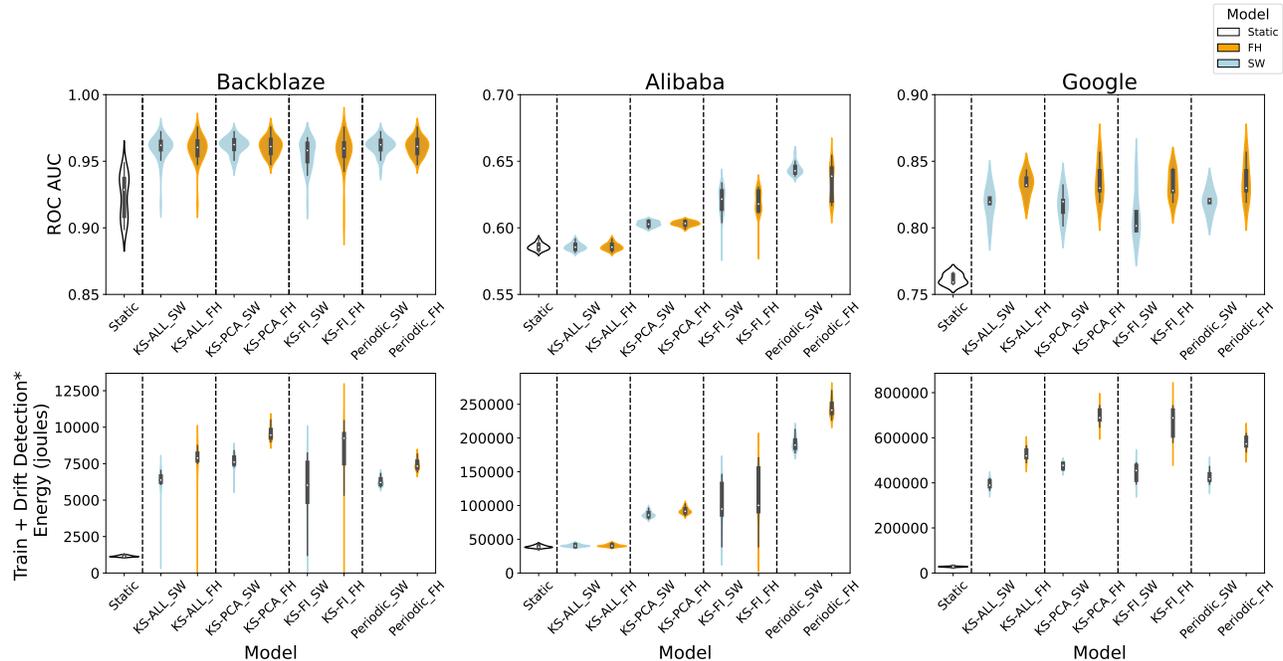
Fig. 2: ROC AUC and energy consumption of each retraining technique. Note: for the informed retraining (KS-ALL, KS-PCA, and KS-FI all configurations) the energy consumed is calculated during both the retraining and the drift detection.

consuming almost the same amount of energy as the Static model. However, there is a significant difference in the ROC AUC between KS-ALL and the other configurations (at least 4%), showing that retraining is required. Thus, this drift detector is not able to properly identify the drift in this dataset, and should not be used as an indicator of when to retrain. We can also notice that some informed retraining techniques consume more energy than periodic retraining techniques. For instance, retraining a model for the Google dataset using a KS-FI or a KS-PCA drift detector is more energy-intensive than periodically retraining the model. The reason for this is the fact that these drift detectors are too sensitive to changes in data and signal the need to retrain often. Since these detectors also consume energy and the number of times the model required retraining is not significantly reduced, these configurations consume more energy compared to periodic retraining. On the other hand, for the Google dataset, we can observe that some drift detectors, KS-FI and KS-PCA, are considerably sensitive and constantly indicate the need for retraining. For this reason, they consume more energy compared to periodic retraining and this finding shows that informed retraining is not always more sustainable than periodic retraining.

In Table I, we present the percentage of the energy consumed solely by the drift detector out of the energy consumed by both training and drift detection. This table shows that the energy consumed by the drift detector is relatively low (less than 4%). This shows that incorporating an unsupervised drift detector does not bring too much overhead to the total energy consumption. We can further see from Table I that the KS-

TABLE I: Percentage of energy consumed by each drift detector from the energy consumed for training + drift detection.

Drift Detector	Retraining	Backblaze	Alibaba	Google
KS-ALL	SW	3.17	3.90	0.41
	FH	2.65	3.91	0.37
KS-PCA	SW	1.77	1.54	0.26
	FH	1.50	1.53	0.21
KS-FI	SW	1.63	0.47	0.13
	FH	1.16	0.47	0.11

ALL drift detector consumes more energy than KS-PCA and KS-FI for all datasets. This can be explained by the fact that this drift detection technique derives the data distribution from the entire feature space, while the KS-PCA and KS-FI perform feature reduction before deriving the data distribution. Thus, unsupervised drift detectors incorporating feature reduction are generally sustainable.

> **RQ1b Answer**
>
> Informed retraining usually consumes less energy than periodic retraining when the drift detector is properly chosen.

3) Best Overall Retraining Approach: When choosing the best overall retraining approach, we have to consider which retraining technique achieves the highest ROC AUC while lowering energy consumption compared to periodic retraining. Furthermore, we consider the Static model as a lower bound in terms of ROC AUC.

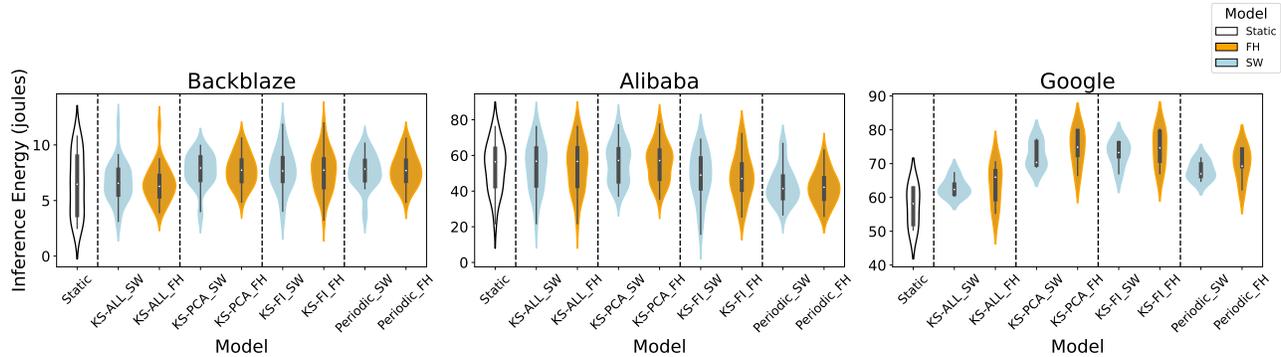To answer this research question (RQ1c), we examine the

Fig. 3: Energy consumed during inference for each retraining technique.

energy consumed for each retraining technique with respect to its corresponding ROC AUC improvement depicted in Figure 2. When it comes to the Backblaze datasets, there is no statistical difference in ROC AUC among configurations that include a drift detector (KS-ALL, KS-PCA, and KS-FI) and the periodic configuration. However, the KS-FI using a sliding window approach consumes the least energy. The KS-FI_SW is the best overall for the Alibaba dataset, achieving the highest ROC AUC with the lowest energy use. Regarding the Google dataset, the retraining technique that consumes the least amount of energy while achieving the highest performance is the KS-ALL drift detector. These experiments demonstrate that no single retraining technique is universally optimal. Therefore, further experimentation is necessary to identify the most sustainable approach for each case.

TABLE II: Energy estimation for using a drift detection-based retraining approach vs a periodic approach over the period of one . In the Drift Detection column, we depict the most sustainable drift detection-based retraining technique for each dataset, namely KS-FI for Backblaze and Alibaba and KS-ALL for Google. Note: Only the sliding window retraining technique is presented since it is the most sustainable. In this table, K implies that the value needs to be multiplied by 1.000 and M implies that the value needs to be multiplied by 1.000.000

Drift Detector	Retraining	Periodic	Drift Detection
Backblaze	6 months	6.3K	5.9K
	1 year estimate	12.6K	11.8K
Alibaba	1 month	191.3K	106.9K
	1 year estimate	2.3M	1.3M
Google	2 weeks	427.0K	393.7K
	1 year estimate	11.1M	10.2M

Each of our datasets was evaluated during periods of different lengths, namely 6 months for Backblaze, 1 month for Alibaba, and 2 weeks for Google. These lengths are strongly dependent on the size of each dataset and on how failure prediction models were designed in previous work [6], [7], [14]. Therefore the retraining frequency for each failure prediction model is different, monthly retraining for Backblaze, weekly retraining for Alibaba and daily retraining for Google. To better understand the impact of retraining periodically vs

retraining based on drift detection over a longer period, we estimated the energy consumed on training + drift detection over one year for each dataset. In this experiment, we assume no change in the energy consumed during the given period and the rest of the year. For instance, if the energy consumed by periodic retraining during the available 6 months for Backblaze is 6.3K, we assume that in the following 6 months, the model will consume the same amount of energy.

In Table II we show the energy consumed by the most sustainable retraining technique for each dataset (KS-FI for Backblaze and Alibaba and KS-ALL for Google) vs periodic retraining and an estimate of energy consumed over one year. Our results show that retraining using a suitable drift detector can significantly reduce energy consumption over one year. More specifically it can reduce the energy consumed by failure prediction models for Google and Alibaba by around 1000 kilojoules (around 40% less energy for Alibaba and 10% for Google) and the energy consumed by failure prediction models for Backblaze by almost 1 kilojoule (around 7%). Therefore, employing a drift detector that is not too sensitive and can identify when a model requires retraining is substantially beneficial in the long term compared to periodic retraining.

> **RQ1c Answer**
>
> There is no one-size-fits-all solution for model retraining and different options should be considered before opting for a particular technique. Although there is evidence of the benefits of using an informed retraining technique, the main challenge lies in choosing an appropriate drift detector and in the frequency of concept drift occurrences. Furthermore, the optimal informed retraining technique should be paired with a sliding window retraining approach to minimize the consumed energy.

B. Impact on Inference Energy

Our study aims to understand whether different model retraining techniques can impact the energy consumed during inference. Therefore, in Fig. 3 we depict the distribution of the energy consumed during inference for each model across all

random seeds. From Fig. 3 we can notice that the variation in inference energy consumption for each retraining technique is relatively small. Therefore, although each retraining technique is different in terms of the frequency of retraining a model and the data, the energy on inference is not impacted.

We verified whether there is a significant difference between the energy consumed during the inference for all analyzed scenarios (p-values higher than 0.05). For the Google dataset, there was no significant difference in energy consumed during inference between all the analyzed retraining techniques. For the Alibaba dataset, both periodic retraining techniques (Periodic_SW and Periodic_FH) are significantly different from the rest in terms of energy consumed during inference. However, the difference in consumed energy is only around 10 joules. When it comes to Backblaze, there was no significant difference in the energy consumed during inference between the Static and KS-ALL or between KS-PCA, KS-FI, and Periodic models. However, although for instance the Static and Periodic models consume significantly different amounts of energy according to the Wilcoxon statistical test (p-value 0.04), this difference was only around 1.5 joules. Thereby, the difference in energy consumed during inference between the retraining techniques is either not significant or extremely low.

> ### RQ2 Answer
>
> In general, the employed retraining technique does not affect the energy consumption of inference tasks.

VII. Discussion and Implications

In this section, we will highlight the general findings derived from this study and we will discuss our results. Our findings aim to help ML practitioners build more sustainable ML software systems and understand how to experimentally assess different retraining techniques in terms of sustainability. The remainder of this section will present each of our findings followed by a discussion and its implications.

> ### Finding 1
>
> Retraining using only the newest data is typically more sustainable than retraining on all available data and has a negligible influence on the model's performance.

In our work, we validate the claim of the authors [10] that employing a sliding window approach when retraining ML models is more sustainable with experimental evidence. When comparing the sliding window and the full-history approach we noticed that for all datasets the difference in model performance, ROC AUC, is minimal, while the difference in energy consumption is considerable. For the Backblaze dataset, the ROC AUC obtained by both retraining approaches is the same, while for the Alibaba dataset, the sliding window approach achieved an ROC AUC of 1% higher than the full-history approach, showing that the model accuracy benefits from deleting older samples. The only dataset where the full-history

approach has a higher ROC AUC than the sliding window approach is the Google dataset. However, from all three datasets, the Google dataset is the shortest in terms of sample collection (29 days instead of 2 months for Alibaba and one year for Backblaze). Therefore, the reason why a full-history approach can be more beneficial from the accuracy perspective is that the model might require more training samples before deploying the model in production. Furthermore, the difference in energy consumed by retraining the Google dataset with a full-history approach vs a sliding window approach is significantly high, ranging from 130 kilojoules to 222 kilojoules depending on the configuration, while the ROC AUC gain is only a maximum of 1.5%. Due to the minimal accuracy improvement and the high energy consumption associated with using a full-history approach, we recommend that ML practitioners adopt a sliding window retraining technique when developing Green ML applications.

> ### Finding 2
>
> Retraining a model based on a drift detector can benefit both the model's performance and the energy consumed **only** if the drift detector is properly chosen.

While answering RQ1b and RQ1c, we noticed that integrating a drift detector as an indicator of when to retrain can be beneficial only if we know that the detector is not too sensitive or the detector can identify drifts. We had an example of employing a drift detector that was not able to identify any drift for the Alibaba dataset, namely the KS-ALL. In this situation, using this drift detector leads to obtaining the same ROC AUC as not retraining the model at all (Static), while slightly increasing the overall energy consumption of the model in production, since the energy consumed by the drift detector needs to be considered. We also had an example of what happens if the drift detector is too sensitive and constantly signals drifts in the experiment with the Google dataset (drift detectors KS-PCA and KS-FI). Here we observed that if a drift detector is too sensitive, the amount of energy consumed by a configuration that retrains based on drift detection becomes higher than the amount of energy consumed during periodic retraining, while sometimes (in the case of KS-FI) lowering the ROC AUC by 1%. However, for all three datasets, there was at least one configuration involving drift detection that both reduced the consumed training energy and preserved the ROC AUC. For this reason, we argue that a drift detector can benefit both the consumed energy and the model's performance only if it is properly chosen.

> ### Finding 3
>
> Although inference is not affected, the energy consumption of training a model is severely impacted by the retraining strategy.

Throughout our experiments, we noticed that the differ-

ence in consumed energy during inference among different retraining strategies is either not significant or extremely small. This shows that inference energy consumption is in our experimental setup not influenced by the employed retraining technique. However, we solely experimented with the Random Forest classifiers since this is the state of the art of failure prediction, which is the case study of this paper. Therefore, our conclusions apply to this classification algorithm, but future work should investigate whether the same conclusions hold when employing other classification algorithms or when investigating other study cases, such as deep learning applications, which are more energy-intensive when it comes to inference.

> **Finding 4**
>
> The retraining frequency has a significant impact on the energy consumption.

While answering RQ1c, we performed a one-year estimation to compare the long-term benefits of employing drift-based retraining with periodic retraining. Throughout this experiment, we also noticed that the most energy-intensive model is the failure prediction model for the Google dataset (approx. 10 megajoules consumed during one year), followed by Alibaba (approx. 1 megajoules) and Backblaze (approx. 6 kilojoules). The main difference between these models in the energy consumed over one year is the retraining frequency, since Backblaze and Google models are retrained every month and the Alibaba dataset is retrained every week. Therefore, the retraining frequency has a significant impact on the model's energy consumption. However, the retraining frequency is context dependent because it must consider the business context and how the ML application is used in real world. Hence, given the tremendous impact of the retraining frequency on the energy consumed, we recommend ML practitioners to design ML applications that require retraining less often if the business context allows.

VIII. Threats to Validity

a) External validity.: Our study focuses solely on Failure Prediction models, where concept drift is a known issue [6], [7], [14]. To support replication, we provide a replication package with all the code necessary to execute experiments. All employed datasets are publicly available (Backblaze, Google, Alibaba). We use Random Forests, the state-of-the-art for this task, but results may vary with other algorithms. Due to time constraints, we limited our scope, though future work should explore other domains and models.

b) Internal validity.: Our experiments replicate state-of-the-art literature on failure prediction models [6], [7], [14] and we acknowledge that different configurations can challenge the results we collect. Nevertheless, our replication package paves the work for future research to expand the scope of our subjects.

c) Construct validity: We measured energy only for training, inference, and drift detection, as other pipeline parts should not affected by retraining methods. Due to a one-week experiment budget, we ran fewer repetitions for the Google dataset, but results remain valid since 30 seeds (on Alibaba and Backblaze) showed little variation compared to 5 (used for Google).

IX. Conclusion

In this study, we investigated the effects on training and inference energy consumption of multiple retraining techniques (sliding window vs. a full-history and periodic vs. informed). We conduct our experiments on a real-world ML application, namely failure prediction.

We provide empirical evidence that retraining a model only on the newest data (sliding window) is more sustainable than retraining a model on all available data (full-history) [1], [10]. Furthermore, employing a full-history approach does not always come with a benefit in terms of the models' performance compared to sliding windows. We further showed that retraining based on unsupervised drift detectors is better than retraining periodically w.r.t. energy consumption only when the drift detection technique is not too sensitive and is capable of identifying drift. To understand the long-term benefits of employing a drift detection-based retraining technique, we showed that using an appropriate drift detector is estimated to decrease energy consumption by up to 40% during one year compared to period retraining. Furthermore, we demonstrated that in most situations the retraining approach does not influence the energy consumed during inference.

The general conclusions derived from this study should serve as a guideline for ML practitioners to build Green ML applications. If the business context allows, practitioners should design ML applications that require less frequent retraining. For example, an ML application retrained monthly is more sustainable than retrained weekly. Practitioners should discard old data when retraining the model and should adopt a drift detection-based retraining approach to optimize energy efficiency. Unsupervised drift detectors that perform feature reduction (KS-FI and KS-PCA) are usually more sustainable than the ones that do not (KS-ALL). We recommend practitioners carefully choose the drift detector that is the most suitable for their datasets and analyze their consequences on the ML model's consumed energy and accuracy in the long term.

Based on our findings, drift detection-based retraining can solely be beneficial in terms of both energy consumption and the model's accuracy if the drift detector is properly chosen. Therefore in the future, we plan to develop a framework that enables ML practitioners to check whether a drift detector is too sensitive to data changes or incapable of detecting drifts. In this paper, we solely focused on the energy consumption of AIOps applications since these applications are widely used in industry [40], they use plenty of data, and they are sensitive to concept drift [6]. We plan to expand this study to other energy-intensive applications where model update is also critical, such as large language models or other deep learning applications.

REFERENCES

[1] Heli Järvenpää, Patricia Lago, Justus Bogner, Grace Lewis, Henry Muccini, and Ipek Ozkaya. A synthesis of green architectural tactics for ml-enabled systems. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Society*, ICSE-SEIS'24, page 130–141, New York, NY, USA, 2024. Association for Computing Machinery.

[2] Eric Breck, Shanqing Cai, Eric Nielsen, Michael Salib, and D. Sculley. The ml test score: A rubric for ml production readiness and technical debt reduction. *2017 IEEE International Conference on Big Data (Big Data)*, pages 1123–1132, 2017.

[3] Firas Bayram, Bestoun S. Ahmed, and Andreas Kassler. From concept drift to model degradation: An overview on performance-aware drift detectors. *Knowledge-Based Systems*, 245:108632, 2022.

[4] João Gama, Indrundefined Žliobaitundefined, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 2014.

[5] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. Data management challenges in production machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, page 1723–1726, 2017.

[6] Yingzhe Lyu, Heng Li, Mohammed Sayagh, Zhen Ming (Jack) Jiang, and Ahmed E. Hassan. An empirical study of the impact of data splitting decisions on the performance of aiops solutions. *ACM Trans. Softw. Eng. Methodol.*, 30(4), jul 2021.

[7] Yingzhe Lyu, Gopi Krishnan Rajbahadur, Dayi Lin, Boyuan Chen, and Zhen Ming (Jack) Jiang. Towards a consistent interpretation of aiops models. *ACM Trans. Softw. Eng. Methodol.*, 31(1), 2021.

[8] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluís-Miquel Munguía, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training, 04 2021.

[9] Carole-Jean Wu et al. Sustainable ai: Environmental implications, challenges and opportunities. In *Proceedings of Machine Learning and Systems*, volume 4, pages 795–813, 2022.

[10] L. Poenaru-Olaru, J. Sallou, L. Cruz, J. S. Rellermeyer, and A. van Deursen. Retrain ai systems responsibly! use sustainable concept drift adaptation techniques. In *2023 IEEE/ACM 7th International Workshop on Green And Sustainable Software (GREENS)*, pages 17–18, 2023.

[11] Tomaso Trinci, Simone Magistri, Roberto Verdecchia, and Andrew D. Bagdanov. How green is continual learning, really? analyzing the energy consumption in continual training of vision foundation models, 2024.

[12] Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power hungry processing: Watts driving the cost of ai deployment? In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, page 85–99, New York, NY, USA, 2024. Association for Computing Machinery.

[13] Yingnong Dang, Qingwei Lin, and Peng Huang. Aiops: Real-world challenges and research innovations. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 4–5, 2019.

[14] Yingzhe Lyu, Heng Li, Zhen Ming Jiang, and Ahmed E. Hassan. Assessing the maturity of model maintenance techniques for aiops solutions. *arXiv preprint arXiv:2311.03213*, 2023.

[15] Yangguang Li, Zhen Ming (Jack) Jiang, Heng Li, Ahmed E. Hassan, Cheng He, Ruirui Huang, Zhengda Zeng, Mian Wang, and Pinan Chen. Predicting node failures in an ultra-large-scale cloud computing platform: An aiops solution. *ACM Trans. Softw. Eng. Methodol.*, 2020.

[16] Qingwei Lin, Ken Hsieh, Yingnong Dang, Hongyu Zhang, Kaixin Sui, Yong Xu, Jian-Guang Lou, Chenggang Li, Youjiang Wu, Randolph Yao, Murali Chintalapati, and Dongmei Zhang. Predicting node failure in cloud service systems. In *ESEC/FSE 2018: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018.

[17] Roberto Verdecchia, Luís Cruz, June Sallou, Michelle Lin, James Wickenden, and Estelle Hotellier. Data-Centric Green AI: An Exploratory Empirical Study. In *ICT4S 2022 - 8th International Conference on ICT for Sustainability*, 2022.

[18] Rafiullah Omar, Justus Bogner, Joran Leest, Vincenzo Stoico, Patricia Lago, and Henry Muccini. How to sustainably monitor ml-enabled systems? accuracy and energy efficiency tradeoffs in concept drift detection. In *2024 International Conference on ICT for Sustainability (ICT4S)*, 2024.

[19] L. Poenaru-Olaru, L. Cruz, A. van Deursen, and J. S. Rellermeyer. Are concept drift detectors reliable alarming systems? - a comparative study. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 3364–3373, 2022.

[20] Adnan Masood and Adnan Hashmi. *AIOps: Predictive Analytics & Machine Learning in Operations*. 2019.

[21] Qian Cheng, Doyen Sahoo, Amrita Saha, Wenjing Yang, Chenghao Liu, Gerald Woo, Manpreet Singh, Silvio Saverese, and Steven C. H. Hoi. Ai for it operations (aiops) on cloud platforms: Reviews, opportunities and challenges. *ArXiv*, 2023.

[22] Farzaneh Mahdisoltani, Ioan Stefanovici, and Bianca Schroeder. Proactive error prediction to improve storage system reliability. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pages 391–402, 2017.

[23] Mirela Madalina Botezatu, Ioana Giurgiu, Jasmina Bogojeska, and Dorothea Wiesmann. Predicting disk replacement towards reliable data centers. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 39–48, 2016.

[24] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31:2346–2363, 2019.

[25] João Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In Ana L. C. Bazzan and Sofiane Labidi, editors, *Advances in Artificial Intelligence − SBIA 2004*, 2004.

[26] Maayan Harel, Koby Crammer, Ran El-Yaniv, and Shie Mannor. Concept drift detection through resampling. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, 2014.

[27] Lorena Poenaru-Olaru, Luís Cruz, Jan S. Rellermeyer, and Arie van Deursen. Is your anomaly detector ready for change? adapting aiops solutions to the real world. In *CAIN24, 2nd International Conference on AI Engineering - Software Engineering for AI*, 2024.

[28] Minghua Ma, Shenglin Zhang, Dan Pei, Xin Huang, and Hongwei Dai. Robust and rapid adaption for concept drift in software system anomaly detection. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, pages 13–24, 10 2018.

[29] Yong Xu, Kaixin Sui, Randolph Yao, Hongyu Zhang, Qingwei Lin, Yingnong Dang, Peng Li, Keceng Jiang, Wenchi Zhang, Jian-Guang Lou, Murali Chintalapati, and Dongmei Zhang. Improving service availability of cloud systems by predicting disk error. In *Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference*, page 481–493, 2018.

[30] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. Green ai. In *Communications of the ACM*, volume 63, pages 54–63, 2020.

[31] Yinlena Xu, Silverio Martínez-Fernández, Matias Martinez, and Xavier Franch. Energy efficiency of training neural network architectures: An empirical study. 01 2023.

[32] Backblaze Inc. Backblaze hard drive stats. backblaze b2 cloud storage. https://www.backblaze.com/cloud-storage/resources/hard-drive-test-data.

[33] Charles Reiss, John Wilkes, and Joseph L. Hellerstein. Google cluster-usage traces: format + schema. Technical report, Google Inc., November 2011. Revised 2012.03.20. Posted at http://code.google.com/p/googleclusterdata/wiki/TraceVersion2.

[34] Alibaba Group. 2021. alibaba cluster trace program. https://github.com/alibaba/clusterdata.

[35] Weng Qizhen, Xiao Wencong, Yu Yinghao, Wang Wei, Wang Cheng, He Jian, Li Yong, Zhang Liping, Lin Wei, and Ding Yu. Mlaas in the wild: Workload analysis and scheduling in large-scale heterogeneous gpu clusters. In *9th USENIX Symposium on Networked Systems Design and Implementation*, 2022.

[36] Nosayba El-Sayed, Hongyu Zhu, and Bianca Schroeder. Learning from failure across multiple clusters: A trace-driven approach to understanding, predicting, and mitigating job terminations. In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pages 1333–1344, 2017.

[37] Grace A. Lewis, Sebastián Echeverría, Lena Pons, and Jeffrey Chrabaszcz. Augur: a step towards realistic drift detection in production ml systems. In *Proceedings of the 1st Workshop on Software Engineering for Responsible AI*, 2023.

[38] Luís Cruz. Green software engineering done right: a scientific guide to set up energy efficiency experiments. *Blog post*, 2021.

[39] Benoit Courty, Victor Schmidt, Sasha Luccioni, Goyal-Kamal, Marion Coutarel, Boris Feld, Jérémy Lecourt, Liam Connell, Amine Saboni, Inimaz, supatomic, Mathilde Léval, Luis Blanche, Alexis Cruveiller, ouminasara, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Hugues de Lavoreille, Niko Laskaris, Edoardo Abati, Douglas Blank, Ziyao Wang, Armin Catovic, Marc Alencon, Michał Stęchły, Christian Bauer, Lucas Otávio N. de Araújo, JPW, and Minerva Books. mlco2/codecarbon: v2.4.1, May 2024.

[40] Anas Dakkak, Jan Bosch, and Helena Olsson. Towards aiops enabled services in continuously evolving software-intensive embedded systems. *Journal of Software: Evolution and Process*, page e2592, 2023.