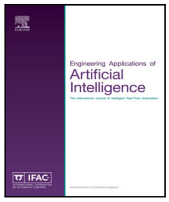
Engineering Applications of Artificial Intelligence 124 (2023) 106610

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



A multi-label Continual Learning framework to scale deep learning approaches for packaging equipment monitoring ☆

Davide Dalle Pezze [a,*], Denis Deronjic [a], Chiara Masiero [a], Diego Tosato [b], Alessandro Beghi [a], Gian Antonio Susto [a]

[a] Department of Engineering, University of Padova, Gradenigo, 6/b, Padova, 35131, Italy
[b] Galdi S.R.L., Enrico Fermi, 43, Postioma (TV), 31038, Italy

ARTICLE INFO

Dataset link: https://data.mendeley.com/datasets/4nhx2x67cd, https://github.com/dallepezze/bat-ocdm

Keywords:
Multi-label classification
Continual Learning
Alarm forecasting
Industry 4.0

ABSTRACT

Continual Learning aims to learn from a stream of tasks, being able to remember at the same time both new and old tasks. We propose a scenario that holds immense appeal for various real-world applications, where a model adapts to handle a stream of machines with distribution shifts Tests on real packaging data proved the feasibility of Continual Learning for addressing such problems. Our study uncovers the limitations of previous algorithms in the Domain Incremental Learning. Our research presents a novel approach for tackling multi-label tasks in Continual Learning, achieving superior performance compared to existing approaches found in the literature. Our method not only achieves optimal performance but also has logarithmic complexity, significantly reducing computation times.

1. Introduction and main contributions

Manufacturing machinery may now gather and communicate pertinent data regarding its condition thanks to IoT and Industry 4.0 (Essien and Giannetti, 2020; Langarica et al., 2019; Li et al., 2020; Maggipinto et al., 2022; Susto et al., 2018; Yuan et al., 2020). Alarm records, in particular, are frequently accessible for legacy equipment and have a far smaller memory and transmission footprint than sensor readings. Alarm analysis is a low-cost alternative or useful supplement to monitoring solutions based on raw sensor data. For instance, anticipating future alarms enables operators to implement the best corrective measures quickly. Therefore, alarm Forecasting (AF) plays an essential role in safely managing process operations. As introduced in Dalle Pezze et al. (2021), it is frequently reasonable to rephrase AF as a multi-label classification task.

Generally, Machine learning has proven valuable in optimizing processes, improving efficiency, and enhancing overall performance. It has found success in various industrial applications, yielding positive outcomes (Puruncajas et al., 2020; Dalle Pezze et al., 2021; Li et al., 2014; Bajic et al., 2018).

But in the industrial setting, one of the major difficulties in using Machine Learning (ML) approaches in production is that it is often difficult to adapt to process variability (Ma et al., 2022). For example, new machines could act differently than ones already deployed. Instead of learning a new model for each piece of equipment or starting from scratch every time, it would be preferable for scalability to update the current model when fresh data are available. Indeed, repeated training from scratch would require increasing amounts of memory and computing power as new units were added. However, in the classic paradigm, model retraining using only new data leads to a sharp decrease in performance on previously learned tasks, a phenomenon known as Catastrophic Forgetting (CF) (Goodfellow et al., 2013). A branch of research called Continual Learning (CL) tries to reduce CF and make it possible to train models using an incoming stream of training data (Parisi et al., 2019; Delange et al., 2021). In CL, training samples come in subsequent tasks, and the trained model can access only a single task at a time.

Proposed CL approaches for multi-label classification were proposed for the Class Incremental Learning (CIL) scenario (Van de Ven and Tolias, 2019), where new labels appear overtime. To achieve the scalability required by a production-grade solution, we propose a CL approach for AF, framed as a Domain Incremental Learning (DIL) Multi-Label classification task. In DIL (Van de Ven and Tolias, 2019), the

☆ This study was partially carried out within the MICS (Made in Italy – Circular and Sustainable) Extended Partnership and received funding from Next-GenerationEU (Italian PNRR – M4 C2, Invest 1.3 – D.D. 1551.11-10-2022, PE00000004). Moreover, this study was also partially carried out within the PNRR research activities of the consortium iNEST (Interconnected North-Est Innovation Ecosystem) funded by the European Union Next-GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS00000043).

* Corresponding author.
E-mail addresses: davide.dallepezze@unipd.it (D. Dalle Pezze), denis.deronjic@studenti.unipd.it (D. Deronjic), chiara.masiero@statwolf.com (C. Masiero), diego.tosato@galdi.it (D. Tosato), alessandro.beghi@unipd.it (A. Beghi), gianantonio.susto@unipd.it (G.A. Susto).

https://doi.org/10.1016/j.engappai.2023.106610
Received 21 April 2023; Received in revised form 3 June 2023; Accepted 6 June 2023
Available online 28 June 2023

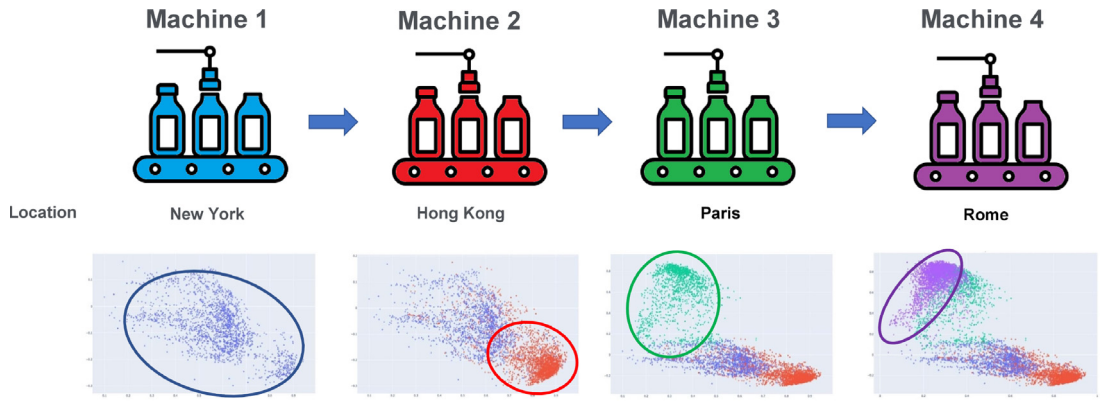


Fig. 1. High-level representation of the continual learning scheme. New machines are gradually placed in various locations across the world. The data distribution varies from earlier machines due to a number of factors, including different sensors, recipes, etc. Therefore, it is necessary a model that can manage data from older equipment while simultaneously adjusting to new machine data..

Table 1
Some abbreviations used in the work.

Abbreviation	Description
AF	Alarm Forecasting
CF	Catastrophic Forgetting
CL	Continual Learning
CIL	Class Incremental Learning
DIL	Domain Incremental Learning

labels remain the same for each task. In our setup, new machines act as a stream of new tasks. Our main contributions are the following:

(i) We propose a scenario that holds immense appeal for various real-world applications, where machines are gradually installed over time, necessitating the model's ability to adapt to distribution shifts and retain the knowledge of previous data.
(ii) We thoroughly investigate the intriguing and unexplored Domain Incremental Learning. Our study highlights the limitations of certain previous algorithms when applied in this context.
(iii) Our research presents a novel approach for tackling multi-label tasks in Continual Learning, achieving superior performance compared to existing approaches found in the literature.
(iv) Our method also has logarithmic complexity, significantly reducing computation times.
(v) To demonstrate the feasibility of addressing this significant problem through Continual Learning, we conducted tests on real data sourced from the packaging industry, utilizing the formulation of Alarm Forecasting (Tosato et al., 2020).

The rest of the paper is organized as follows. Section 2 provides an overview of the literature about AF and CL applied to multi-label classification. In Section 3, we introduce a novel replay-based approach for the multi-label DIL scenario after describing a multi-label formulation for AF. In Section 4 we validate the proposed method on a real-world AF problem from the Packaging Industry. Finally, Section 5 provides some concluding remarks and describes envisioned future works (see Table 1).

2. Related work

2.1. Alarm forecasting for predictive maintenance

The goal of alarm data analysis is to translate alarm logs into actionable insights for operators. Many approaches aim to detect abnormal behaviors using Anomaly Detection algorithms (Domingues et al., 2018; Yen et al., 2013; Du et al., 2017) or Fault Detection and Classification approaches (Fan et al., 2020). Other approaches aim to provide useful feedback to users before abnormal behaviors occur. AF plays a fundamental role in this scenario. For example, in Xu et al. (2019) historical alarm sequences are exploited using Bayesian estimators. In Zhu et al. (2016), a probabilistic model based on an N-gram model is proposed to predict the probability of alarm occurrence, given the previous alarms. Due to the limitations of N-gram models, more advanced approaches based on neural network architectures were proposed. In Cai et al. (2019), alarm log information is embedded using Word2Vec and a Long short-term memory (LSTM)-based deep learning model is designed to predict the next alarm. In Villalobos et al. (2020), the authors propose an approach that combines LSTM neural networks to forecast the future measurements of various sensors with Residual Neural Networks to predict the future occurrence of alarms based on estimated future sensor measurements. In Dalle Pezze et al. (2021), the authors reframe AF as a multi-label classification problem and provide a more general approach that is viable in situations where raw sensor readings are not available.

This work aims to adopt a CL framework to achieve a production-ready solution for AF, where to be scalable, it must learn with new pieces of equipment that arrive overtime.

2.2. Continual learning

CL tries to learn from a stream of tasks with non-stationary distribution. New experience is constantly being acquired over time, while old experience is still relevant and needs to be preserved. In the classic paradigm, model retraining using the new data leads to a sharp decrease in performance on previously learned tasks, a phenomenon known as Catastrophic Forgetting(CF) (Goodfellow et al., 2013). Continual Learning is commonly linked to Deep Learning models, overlooking classic machine learning techniques like clustering (Borlea et al., 2022). Deep learning models are frequently utilized in Continual Learning scenarios due to their ability to learn intricate patterns and representations from extensive datasets. Neural networks, such as Multi-Layer Perceptron (MLP), recurrent neural networks (RNNs) (Salehinejad et al., 2017), or convolutional neural networks (CNNs) (Verma et al., 2022), excel at capturing and retaining information across various time steps or data samples, thus making them well-suited for continual learning tasks.

Usually CL considers three scenarios: Task Incremental Learning (TIL), Class Incremental Learning (CIL) and Domain Incremental Learning (DIL) (Van de Ven and Tolias, 2019). CIL refers to a CL scenario where the model is required to learn and adapt to new classes of data over time. This indicates that during inference, it is expected that the model will be able to accurately categorize both existing and new classes. With domain incremental learning, only the distribution of the input data changes; the set of labels in the output remains constant.

Such a scenario fits the use-case of our research, in which we are interested in consistently predicting the same classes (i.e. alarms) on various machines that have been deployed throughout time and have a distinct data distribution (see Fig. 1). Models in the CIL and DIL scenario must be able to complete each task seen so far and determine which task is being provided to them. One key distinction of Task Incremental Learning (TIL) compared to Domain Incremental Learning (DIL) or Class Incremental Learning (CIL) is the inclusion of the Task ID during the testing phase. This simplifies the problem by reducing its complexity, as the model only needs to focus on solving the specific task at hand. This indicates that in order for TIL approaches to work, the practical application below must be able to provide the task id during inference, which is frequently not feasible in real-world problems.

The literature often summarize the CL approaches in 3 groups, regularization-based (Kirkpatrick et al., 2017; Zhang et al., 2022), dynamic architecture methods (Rusu et al., 2016; Yoon et al., 2017) and replay-based (Chaudhry et al., 2019; Rolnick et al., 2019). Some well-known methods of replay-based family are Experience Replay (ER) (Rolnick et al., 2019), Gradient Episodic Memory (GEM) (Lopez-Paz and Ranzato, 2017), and iCaRL (Rebuffi et al., 2017). They all work by explicitly retrain on a limited subset of stored samples while training on new tasks. In particular, Experience Replay (ER), despite being straightforward, has shown extraordinary effectiveness and produced excellent results in many works as discussed below. During training on new tasks, the model replays samples from the buffer to retain knowledge from previous tasks, mitigating catastrophic forgetting. By replaying past experiences, Experience Replay helps to maintain a balance between learning new tasks and preserving knowledge from previous tasks.

Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), Synaptic Intelligence (SI) (Zenke et al., 2017), and Memory Aware Synapses (MAS) (Aljundi et al., 2018) are popular regularization-based techniques (De Lange et al., 2021). EWC functions by forbidding abrupt changes to crucial parameters. The similar concept is also used by SI and MAS. However, it is demonstrated how the replay-based methods outperform the regularization-based methods in the multi-label setting, as tested in Kim et al. (2020).

Belonging to the architecture-based methods there are PackNet, PathNet and Piggyback. Generally, such methods work by keeping task-specific weights. This implies that during inference they need of the task id and therefore can work only in the TIL scenario, which make them unfeasible for many practical applications, including our use-case that belong to the Domain Incremental scenario.

The related literature suggests that the Rehearsal (also known as replay-based) approach appears to be a strong solution to CF (Pellegrini et al., 2020; Buzzega et al., 2021; Masarczyk et al., 2022; Kim et al., 2020).

Multi-label classification plays a crucial role in various real-world applications where each instance can be associated with multiple labels simultaneously. However, in the context of CL, the research focus on multi-label classification has been relatively limited. Despite the significance of multi-label tasks, there have been only two main replay-based approaches specifically designed for multi-label classification proposed in the literature:

(i) The first approach is Partitioning Reservoir Sampling (PRS) (Kim et al., 2020). The idea is that it is sufficient to allocate a portion of the memory to the minority classes to retain a balanced knowledge of present and past experiences.
(ii) The second approach is called Optimizing Class Distribution in Memory (OCDM) (Liang and Li, 2022). It aims at speeding up the processing time required to select the samples, compared to PRS, resulting in a significant improvement. OCDM is a greedy approach that selects a subset of samples such that the final distribution of the labels in memory is as close as possible to a uniform target distribution.

Both approaches are derived from the Online CL (OCL) scenario, where, in contrast to classic CL, a single batch of the task is shown each time. Thus, in OCL, multiple epochs on the same batch are not allowed. The PRS and OCDM were proposed for the CIL scenario (Van de Ven and Tolias, 2019), where new labels may appear for each new task.

In our case, we are considering a Domain Incremental Leaning (DIL) scenario (Van de Ven and Tolias, 2019) instead, where the set of output labels remains the same, but the input distribution changes based on the task. In fact, the set of alarm codes is the same for each piece of equipment, while the alarm frequency is different for the different deployments. In this work, we start from the OCDM to design a task-aware replay-based approach, as detailed in the next section.

3. Proposed approach

3.1. Alarm forecasting as a multi-label classification task

To validate the proposed CL multi-label approach in the DIL scenario, we evaluate it on a real-world dataset from the dairy products packaging industry (Tosato et al., 2020). In the same vein as in Dalle Pezze et al. (2021), the goal is to produce a list of distinct alarms that are likely to occur in a future time window.

Given the set of all alarm codes $\mathcal{A}$ and the subset of target alarm codes to be predicted $\mathcal{A}_{out} = \{a_1, \dots, a_L\} \subseteq \mathcal{A}$, the target of the multi-label classification problem is the n-hot encoding $y = (y_1, \dots, y_L) \in \mathcal{Y}$, such that $y_i = 1$ if alarm a_i will occur in the future time window of length D_{out}, otherwise $y_i = 0$. We represent the input data x as the normalized count of distinct alarms that occurred in a previous time window of fixed duration D_{in}. Thus, $x = (c_1, \dots, c_N) \in \mathcal{X}$, $N = |\mathcal{A}|$ and, $c_j = C_j / \sum_k C_k$ where C_j is the number of alarms of type j occurred in the input window.

To recap, we reframe AF as the task of estimating the following map from the input set to the label space:

$$\begin{aligned} h &: \mathcal{X} \to \mathcal{Y} \\ (c_1, \dots, c_N) &\mapsto (y_1, \dots, y_L). \end{aligned} \tag{1}$$

In dealing with multi-label classification, we need to keep in mind two key challenging aspects: (i) the possibly overwhelming size of the output space, since the number of possible combinations of labels grows exponentially as the number of class labels increases (Zhang and Zhou, 2013), and (ii) unequal label distribution in most multi-labeled datasets (Charte et al., 2019; Van Horn and Perona, 2017; Kim et al., 2020). Moreover, many business-related critical alarms are rare (Cai et al., 2019). Therefore, we use Weighted Focal Loss (Lin et al., 2017), an idea from Object Detection and Segmentation, to train the classification model. Results from Dalle Pezze et al. (2021) indicate that this choice translates into better classification performance for low-frequency alarms.

We consider a Multi-layer Perceptron (MLP) as a model for multi-label classification. Indeed, according to Dalle Pezze et al. (2021), approaches based on neural networks seem to achieve better performance on the AF task at hand than those based on classic ML paired with problem transformation to convert multi-label problems to single-label problems, either single-class or multi-class (Madjarov et al., 2012; Read et al., 2009). The structure of the MLP model is detailed in Fig. 2.l Since the sigmoid activations in the output layer share the same hidden representation, this model takes into account the relations among different labels.

3.2. Continual learning classifier design

The industrial scenario is a dynamic environment in which new machines are installed over time. CL provides tools that enable ML solutions to scale efficiently. In this work, we consider new equipment pieces as new tasks, setting our problem in the DIL scenario (Van de Ven and Tolias, 2019) as previously stated. This means that the set of

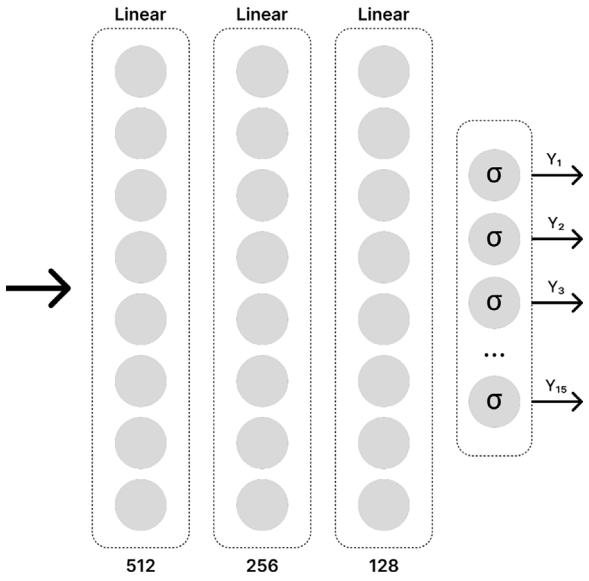


Fig. 2. The multi-label architecture. Below the number of neurons in each layer. Each linear layer has a dropout rate of 50% as regularization and ReLU activation function except for the output layer which has independent sigmoid activation for each neuron. In this case 15 labels has to be predicted and so 15 independent sigmoid outputs.

labels in the output $\mathcal{A}_{out}$ remains the same for the new deployment of the target machine, while the input and output distributions change according to the current task. Indeed, even if the machines are the same type, they may differ from the already deployed ones regarding settings, recipes, and the behavior of human operators (Gentner et al., 2021). We use replay-based approaches from CL, according to schema depicted in Fig. 3, to make a Multi-Label classifier learn to perform alarm forecasting on new machines without forgetting the old ones.

3.3. Optimizing Class Distribution in Memory (OCDM)

Before introducing the novel approach to address the Continual multi-label classification problem, we recall some notions about the OCDM method. OCDM formulates the memory update mechanism as an optimization problem. This greedy algorithm, detailed in Alg. 3, provides a solution with a linear complexity in the number of tasks.

The core of the algorithm is the procedure to update the memory when a new batch of data $\mathcal{B}_t$ of size b_t coming from the task $t \in \mathcal{T}$ arrives, as depicted in the Algorithm 1. Given that the memory $\mathcal{M}$ has fixed size M, for each task the goal of OCDM is to select M samples to save in the memory among the $M + b_t$ available ones. This can be represented as an optimization problem:

$$\min_{\Omega} Dist(\mathbf{p}_{\Omega}, \mathbf{p}) \quad \text{SUBJECT TO} \quad \Omega \subseteq \mathcal{M} \cup \mathcal{B}_t, \quad |\Omega| = M \tag{2}$$

where $\mathbf{p}$ represents the target distribution i.e. the ideal optimal solution, while, $\mathbf{p}_{\Omega}$ represents the distribution of the labels produced from the samples of the set Ω. The $Dist(\cdot,\cdot)$ function used to measure the difference between the two distributions is the Kullback–Leibler (KL) Distance. The target distribution proposed in Liang and Li (2022) is defined as follows:

$$p_i = \frac{(n_i)^{\rho}}{\sum_{j=1}^{C}(n_j)^{\rho}} \tag{3}$$

where n_i is the frequency of a class i and ρ is the allocation power. Using $\rho = 0$ the samples are saved in memory $\mathcal{M}$ in order to have equally distributed classes.

We propose the pseudocode for the OCDM algorithm in three parts, where a description of each one is shown below:

- Alg. 1 (*Memory Update*): This is the main function of OCDM. It takes as input a list of samples in memory and a list of new potential samples to be added in memory. Then it returns the updated memory with the final distribution as closest as possible to the uniform distribution. This is done by iteratively deleting the elements more harmful.
- Alg. 2 *(OCDM_task_update)*: The Alg.*Memory Update* is iteratively executed multiple times, by processing the entire dataset divided into batches of samples. The final result is the updated memory to be used during the training of future tasks.
- Alg. 3 *(OCDM)*: The Alg. *OCDM_task_update* is repeated multiple times on the entire stream of tasks.

Algorithm 1: Memory Update (MU)

Input: Memory $\mathcal{M}$, b

`/* Given a memory` $\mathcal{M}$`, it will delete` b `elements from the memory */`

for $k \in [1, 2, ..., b]$ **do**
  $i = arg\min_{j \in \mathcal{M}} Dist(\mathbf{p}_{\mathcal{M}\setminus\{j\}}, \mathbf{p})$
  $\mathcal{M} \leftarrow \mathcal{M} \setminus \{i\}$
end
return $\mathcal{M}$

Algorithm 2: OCDM_task_update

Input: dataset $\mathcal{D}_t$ of task $t \in \mathcal{T}$, memory $\mathcal{M}$, total size of memory M

for $B_t \in \mathcal{D}_t$ **do**
  if $|\mathcal{M}| \leq M$ **then**
    diff $\leftarrow |\mathcal{M}| - M$
    $V_t \leftarrow$ select randomly $min(|B_t|, \text{diff})$ samples from B_t
    $B_t \leftarrow B_t \setminus V_t$
    $\mathcal{M} \leftarrow \mathcal{M} \cup V_t$
  end
  if $|B_t| > 0$ **then**
    $\Omega \leftarrow \mathcal{M} \cup B_t$
    $\mathcal{M} \leftarrow Memory\_Update(\Omega, |B_t|)$
  end
end

Algorithm 3: Optimizing Class Distribution in Memory (OCDM)

Input: task stream $\mathcal{T}$, total size of memory M

$\mathcal{M} \leftarrow \{\}$ `/* Initialize the memory */`
for $t \in \mathcal{T}$ **do**
  $D_t \leftarrow$ Get Dataset $D_t = \{X_t, Y_t\}$ of task t
  $OCDM\_task\_update(D_t, \mathcal{M}, M)$
end

3.4. Rehearsal strategy: Balanced among tasks OCDM

The OCDM approach was proposed considering the CIL scenario, where a new set of labels arrives at each new task (Van de Ven and Tolias, 2019). As stated above, in our case, we are considering the DIL scenario, where we always have the same labels with potentially new frequencies.

The primary limitation of the current state-of-the-art algorithm is its focus solely on maintaining label balance, disregarding the variations among tasks. In other words, while OCDM ensures balance over time from a label perspective, it overlooks the need for balance across tasks. Consequently, in a Domain Incremental scenario, there is a possibility that if memory optimization is solely based on label balance, the algorithm may discard all data from an entire task to favor samples from another task that improve label balance. For example, when

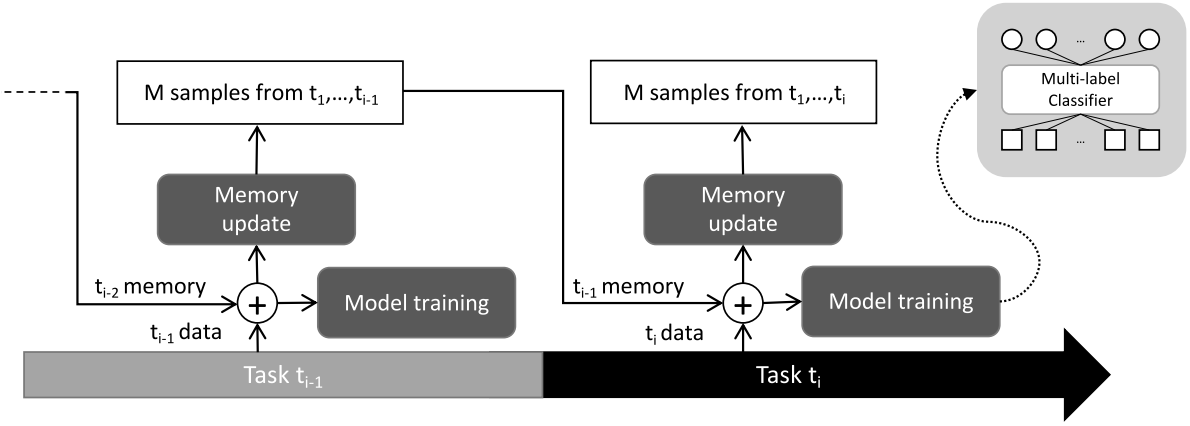


Fig. 3. Rehearsal-based approach to deal with multi-label classification in a CL fashion.

a new task with frequent labels is introduced, the algorithm may choose not to add any samples from this task to the memory, as it would compromise balance. As a result, OCDM does not guarantee the retention of all previously seen tasks in memory, preventing the model from fully incorporating and adapting to them. This limitation hampers the model's ability to retain memory of entire tasks and effectively adapt to them. Additionally, a secondary limitation is that despite the algorithm's improved computational speed compared to previous state-of-the-art methods, it remains computationally expensive for many real-world applications. Therefore, addressing these limitations is crucial for improving performance and enhancing its practical applicability.

Algorithm 4: BAT-OCDM

Input: task stream $\mathcal{T}$, total memory size M
Constraint: $\sum_{t \in T} |\mathcal{M}_t| = M$

$N \leftarrow 0$
for $t \in \mathcal{T}$ **do**
  /* We are going to assign a part of the memory to the new task and select the elements to keep in memory. */
  $\mathcal{D}_t \leftarrow$ Get Dataset $\mathcal{D}_t = \{X_t, Y_t\}$ of task t
  $\mathcal{M}_t \leftarrow \{\}$
  $N \leftarrow N + 1$
  $m \leftarrow \frac{M}{N}$
  $OCDM\_task\_update(\mathcal{D}_t, \mathcal{M}_t, m)$
  /* We update the memories of old tasks, removing some elements, to give space in memory to the new task. */
  diff $\leftarrow \frac{M}{N-1} - \frac{M}{N}$
  if $N \geq 2$ **then**
    for $t_{old} \in \mathcal{T}_{1 \dots N-1}$ **do**
      $\mathcal{M}_{t_{old}} \leftarrow Memory\_Update(\mathcal{M}_{t_{old}}, \text{diff})$
    end
  end
end

To address these limitations, we propose to use a separated memory for each task. Through this approach, described in Alg. 4, balance is performed on both labels and tasks, hence the name Balance Among Tasks OCDM (BAT-OCDM). BAT-OCDM requires two steps to handle the memory:

1. **Step 1**: Given the data of a new task, BAT-OCDM selects a subset of samples to be included in the memory, using the same procedure of OCDM to update the memory.
2. **Step 2**: Since the total size of memory has to remain fixed, BAT-OCDM disregards some of the samples belonging to the memories of old tasks through the $Memory\_Update$ (MU) procedure detailed in Alg. 1.

We consider the schema of Fig. 3 for the proposed approach, where memory will be divided among several old tasks.

Algorithm 5: Dataset-based OCDM

Input: task stream $\mathcal{T}$, total size of memory M

$\mathcal{M} \leftarrow \{\}$ /* Initialize the memory */
for $t \in \mathcal{T}$ **do**
  $\mathcal{D}_t \leftarrow$ Get Dataset $\mathcal{D}_t = \{X_t, Y_t\}$ of task t
  $\mathcal{M} \leftarrow \mathcal{M} \cup \mathcal{D}_t$
  if $|\mathcal{M}| \geq M$ **then**
    $\Omega \leftarrow \mathcal{D}_t \cup \mathcal{M}$
    $diff \leftarrow min(|\Omega| - M, M)$
    $\mathcal{M} \leftarrow Memory\_Update(\mathcal{M}, diff)$
  end
end

We provide the pseudocode of BAT-OCDM in Alg. 4. The main points executed for each task can be described as follows:

- For the new task we calculate the new memory size dedicated to that specific task, which will be the total memory divided by the number of tasks.
- Then we execute the Alg. *OCDM_task_update* (discussed before) to obtain the memory associated to the new task (i.e. Step 1 of above)
- Then for each old task, we identify the number (identified as *diff* in the pseudocode) of elements that need to be deleted from the old memories. Then for each old task we execute the Alg. *Memory Update* with a total number of elements to be deleted equal to *diff*.

3.4.1. Computational complexity

Next, we compare BAT-OCDM and OCDM in terms of computational complexity. We also consider Dataset-based OCDM, a version of OCDM where, instead of sequential batches, all the data coming from the task are used for the memory update. This should improve performance since the original batch-based memory update policy could ignore some samples. Thus it may lead to a less uniform label distribution in the final memory. On the other hand, in this way, the update can no longer be performed online as in the original OCDM. The pseudocode of Dataset-based OCDM is provided in Alg. 5. The pseudocode is very simple, it iterates on the stream of tasks. Then for each task, it calls the already defined Alg. *Memory Update*, the main difference between the call of *Memory Update* in OCDM and in Dataset-based OCDM is the number of elements required to be deleted at once. As said in Dataset-based OCDM

Table 2
Notation used to show the complexity of the algorithms.

Notation	Description
D	Size of the dataset associated to a task
T	Number of tasks used in the tests
M	Size of the memory used

we want to delete a number of elements such that the final memory reach size $\mathcal{M}$.

Table 3 provides a summary of the computational complexity associated with the aforementioned strategies. For more comprehensive information, additional details can be found in the supplementary material (https://github.com/dallepezze/bat-ocdm) .

As for the notation, as indicated in Table 2, we denote the set of tasks as $\mathcal{T} = \{t_1, \dots, t_T\}$. For simplicity, we assume that each task corresponds to D samples and the memory $\mathcal{M}$ has a fixed size M.

The column *Task i* provides the complexity of the i−th memory update. For BAT-OCDM the complexity becomes smaller as i increases. In particular, the overall complexity, shown in column *Total*, is logarithmic in the number of tasks T for BAT-OCDM. For OCDM and Dataset-based OCDM instead, the complexity is linear in T.

The last column shows the complexity under the assumption that $M \leq D$, i.e. $M = c \cdot D$ where $c \in [0, 1]$. In this case, the complexity for all methods is quadratic with regard to the dataset size D. The results on computational complexity do not hold only for the DIL scenario, but also for the CIL scenario studied in the original paper proposing OCDM, showing the same speed improvement.

4. Tests

In order to understand how the models were tested, under what circumstances, and an analysis of the performance obtained, we go into different practical issues in this section. The previously discussed methods are put into practice and evaluated on a dataset taken from a real-world scenario. First, we look at the measures employed to assess the efficiency and efficacy of the methodologies mentioned. The setup of the tests, including the settings and experimental conditions used, is discussed. We also provide a summary of the continual learning approaches that were taken into account for the study, emphasizing their unique attributes and methodologies. As a result, we carefully examine the test results while concentrating on three key factors: performance, label balance in the memory, and task balance. By examining these different facets, we aim to provide a comprehensive and insightful analysis.

4.1. Metrics

Following the convention of multi-label classification (Wang et al., 2016; Ge et al., 2018) we are going to use the macro f1 score to evaluate the performance of the model. Let s be the macro f_1 score, i.e. the average of the f_1 scores for each label: $s = \sum_{i=1}^{L} f_1(y_i, \hat{y}_i)$. Let $s_{i,j}$ be the performance of the model on the test set of task j after training the model on task i. To measure performance in the CL setting, we introduce the following metrics:

Average macro f1 The average macro f1 score $S_T \in [0, 1]$ at task T is defined as:

$$S_T = \frac{1}{T} \sum_{j=1}^{T} s_{T,j} \tag{4}$$

Average Forgetting $F_T \in [-1, 1]$, the average forgetting measure at task T, is defined as:

$$F_T = \frac{1}{T-1} \sum_{j=1}^{T-1} \max_{l \in \{1,\dots,T-1\}} \frac{s_{l,j} - s_{T,j}}{s_{l,j}}. \tag{5}$$

With respect to the original definition used in Chaudhry et al. (2018), we are scaling respect to the maximum macro f1 score, as done in Kim et al. (2020); this is done to compare the forgetting among labels with very different scores. Notice that the closer the metric F_T is to 1, the higher the forgetting is.

4.2. Tests setup

We test the proposed approach on a publicly available real industrial dataset originating from monitoring dairy products packaging equipment (Tosato et al., 2020). In the tests, we consider monitoring 14 packaging machines deployed in different plants worldwide as different tasks to learn in a CL fashion. The splitting of data in train and test is based on time; in the test there will be only samples belonging to the future of the machine.

To obtain the design matrix to train the CL-based multi-label classifier, we draw inspiration from Dalle Pezze et al. (2021), so we consider input windows having a length of 1720 minutes, and output windows of 480 minutes. As described in Section 3.1, we represent the input windows using normalized alarm counts.

Fig. 4 shows in detail the frequencies of the labels and the number of samples for each task. Based on label frequencies, we split them into three groups to better assess the performance:

- high-freq: labels {4,6,7,13};
- medium-freq: labels {0,3,5,8,14};
- low-freq: labels {1,2,9,10,11,12}.

The aspects of software, training specifications, and simulation environment are as follows. Software-wise, Python 3.8 and the deep learning library PyTorch were utilized to implement the model and the continual learning framework. Adam is the optimizer taken into consideration during the model-training process, with a batch size of 32, a learning rate set to 0.001 and a number of epochs equals to 20 for each task. With regards to the loss function's hyperparameters were used the same optimal values found in Dalle Pezze et al. (2021) i.e. beta equal to 0.8 and gamma set to 0.2. The model training was carried out on a GPU Titan X in terms of hardware. Finally, the hyperparameter ρ used in OCDM and BAT-OCDM is set to zero, as suggested in the original paper. All the experimental settings-related details are available at https://github.com/dallepezze/bat-ocdm for reproducibility.

4.3. Considered continual learning approaches

All the memory management strategies can be schemed according to the framework shown in Fig. 3. To assess BAT-OCDM, we compare it not only with OCDM (Alg. 3) and Dataset-based OCDM (Alg. 5), but also with the following approaches:

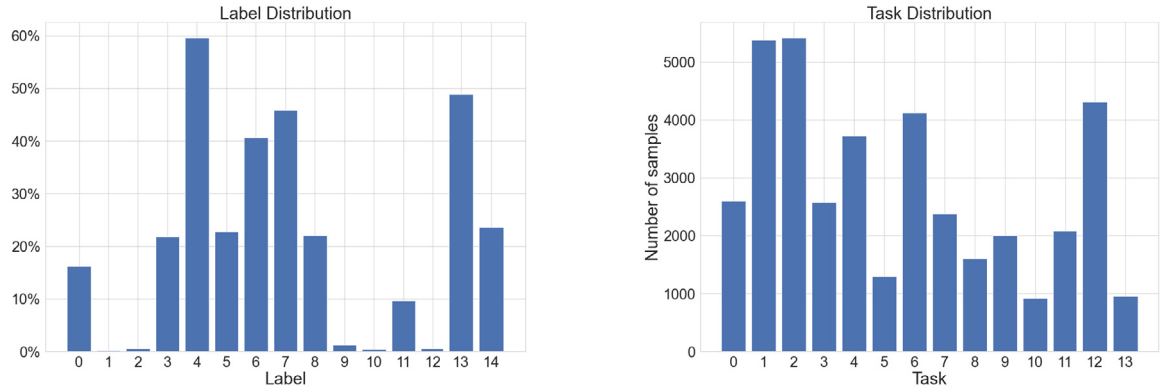
- Finetune: At each task t_i, the classifier is trained from scratch considering only the data from task t_i. There is no countermeasure against forgetting, so we consider this approach as a lower bound.
- Cumulative: At each task t_i, the classifier is trained from scratch based on all the data seen so far (i.e. from task t_1 to t_i), so there is no limit to memory size.
- Task-based Random: At each task t_i, an equal number of samples is selected at random.
- Reservoir Sampling: This stream-oriented approach from Online Learning selects samples at random at a fixed rate. Thus, it is expected to include more samples for tasks whose dataset size is bigger.

The code for the BAT-OCDM approach is shared publicly[1] as well the dataset used (Tosato et al., 2020).

[1] https://github.com/dallepezze/bat-ocdm

Table 3
Computational complexity of the proposed approaches.

	Task i	Total	Assuming $M = c \cdot D$
OCDM	$O(D \cdot M)$	$O(T \cdot D \cdot M)$	$O(c \cdot T \cdot D^2)$
Dataset-based OCDM	$O(D \cdot M + \frac{D^2}{2})$	$O(T \cdot [D \cdot M + \frac{D^2}{2}])$	O $(\frac{2c+1}{2} \cdot T \cdot D^2)$
BAT-OCDM	$O(\frac{D \cdot M}{i} + \frac{M^2}{i-1})$	$O((\ln T + 1) \cdot M \cdot (D + M))$	$O((c + c^2) \cdot (\ln T + 1) \cdot D^2)$



(a) Label frequency (b) Number of samples per task

Fig. 4. Experimental dataset: Labels and tasks statistics.

4.4. Tests results

We divide alarm codes into three categories based on frequency to more thoroughly assess how the label frequency influences model performance. As stated, we then independently calculate the performance measures for low, medium, and high-frequency alerts. The computation time (measured in seconds) needed for the memory update for each suggested strategy is also evaluated in addition to the average macro f1 score and the forgetting measure.

4.4.1. Performance

The results are displayed in Table 4. There is no significant difference among the two approaches based on the random selection of the samples. It is clear from the results that task-based Random and Reservoir Sampling appear to work well on labels with high and medium frequencies. Instead, outcomes are unsatisfactory when low-frequency labels are considered because of the poor representation of these labels in the memory. Unbalanced label distribution is one of the difficulties with multi-label since it makes it harder for the model to concentrate on the rare labels and results in poor end performance. In addition, the best choice of samples to keep in memory also makes the multi-label in continual learning more challenging (with respect to the classic multi-label setting) because it must come up with a good strategy to keep a portion of samples from previous tasks in memory while attempting to have a balanced label distribution that gives an equal representation to all labels. The OCDM method, in contrast, produces good performance for low-frequency labels. These labels critically impact the equipment, as evidenced in Dalle Pezze et al. (2021). However, this result comes with the trade-off of less effective performance on labels with high frequency. Instead, there is no relevant performance difference for classifying medium-frequency labels.

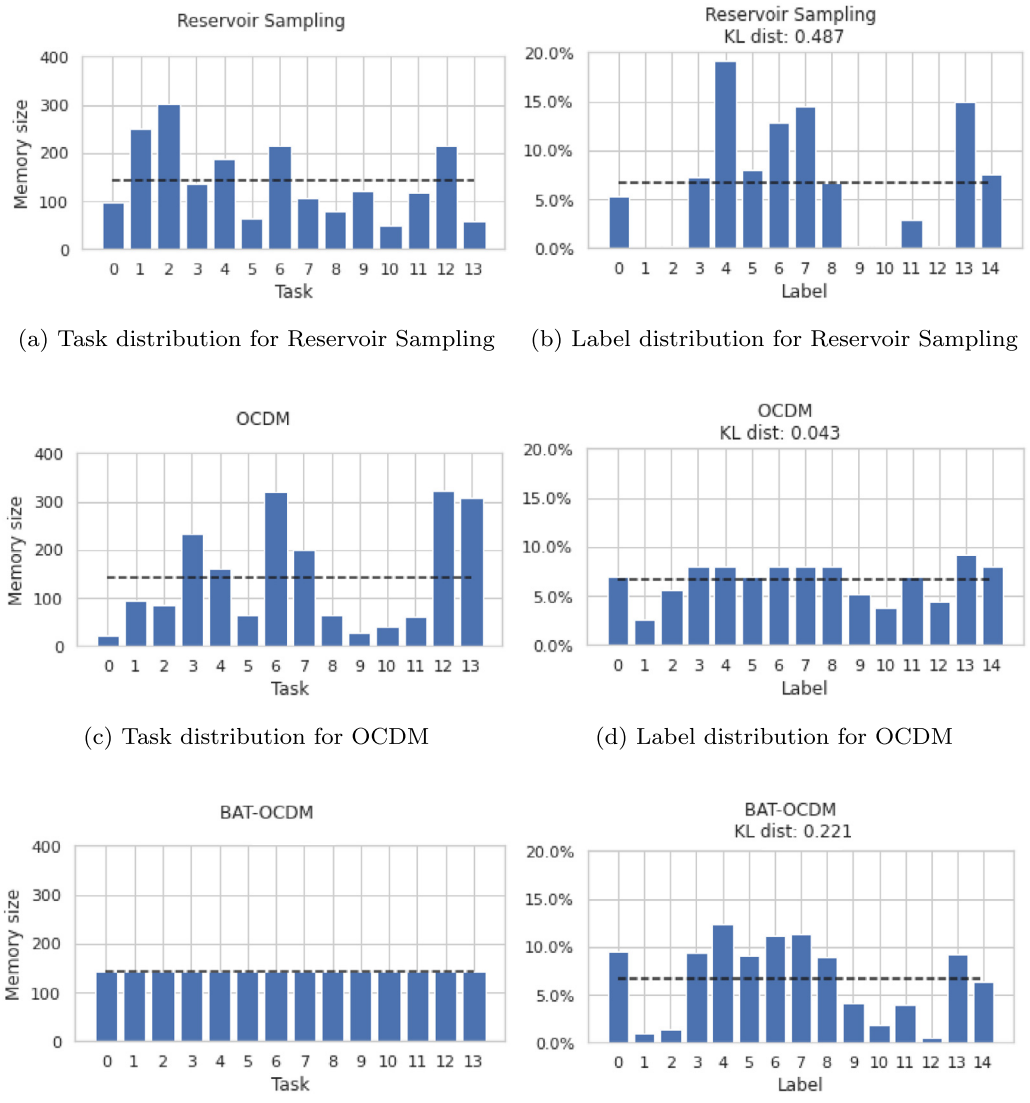
As mentioned above, a possible explanation for this could be that some tasks are not stored in memory. Indeed, the algorithm focuses only on balance among labels and not among tasks. This will be explained in more detail in Section 4.4.2 and in Fig. 4(b) with the task distribution in memory, where it can be seen the very low balance among tasks. For this reason, BAT-OCDM shows superior performance with respect to OCDM for low-freq, medium-freq, and high-freq labels. In particular, the drop in performance for higher frequency labels observed for OCDM is much lower. As for Dataset-based OCDM, even if it simultaneously uses all data from a task to find a better label distribution in memory, no significant difference appears in the performance. Instead, as anticipated by the complexity analysis, the time required to update the memory is much higher.

In multi-label classification, a common challenge is achieving good performance for labels with low frequency. This challenge arises due to the inherent class imbalance between frequent labels and rare labels. As a result, the performance on low-frequency labels often suffers, and the model struggles to make accurate predictions for these less common labels. Improving the performance of low-frequency labels is of significant importance in many real-world applications. These labels might represent critical or rare events, and correctly identifying them is essential for accurate decision-making or understanding the data. However, addressing this challenge and achieving notable improvements in the performance of low-frequency labels is particularly demanding in the multi-label setting. Multi-label classification already poses complexities with the presence of multiple labels and their correlations. The imbalance among frequent and rare labels exacerbates these challenges, making it difficult for models to learn and generalize effectively. Thus, obtaining improvements in the performance of low-frequency labels in multi-label classification is not only significant but also highly challenging.

In summary, our novel approach, BAT-OCDM, achieves comparable results to the state-of-the-art OCDM on low-frequency and medium-frequency labels. However, we demonstrate significant improvements in performance for high-frequency labels, reaching levels that are nearly comparable to those obtained with Reservoir Sampling (RS). Therefore, we are able to obtain optimal performance for the challenging low-frequency labels while at the same having a very low compromise with the high-freq labels.

4.4.2. Balance among labels

We can see the label distributions for each technique in Fig. 5 on the right side. In order we have for each row: RS, OCDM, and BAT-OCDM techniques. Each plot also shows the KL distance to indicate how far apart the final distribution is from the desired distribution defined in Eq. (3). We can observe that the worst algorithm, Reservoir Sampling (RS), obtains a label distribution that is very unbalanced since the samples are collected randomly, and therefore the unbalance present in the original data of the task remains. Instead, we can note how OCDM



(a) Task distribution for Reservoir Sampling

(b) Label distribution for Reservoir Sampling

(c) Task distribution for OCDM

(d) Label distribution for OCDM

(e) Task distribution for BAT-OCDM

(f) Label distribution for BAT-OCDM

Fig. 5. At top RS, at center OCDM, and at bottom BAT-OCDM. At left, it is shown the final distribution of the number of samples for each task in memory varying the approach. On the x-axis of the plots are represented the task IDs, and on the y-axis, the number of samples for each machine is an absolute value. The dashed line in each plot represents the average number of samples among all the machines. Instead, at right, it is shown the final distribution of the labels of the samples in memory varying the approach. In the title for each plot is indicated the name of the approach and KL distance with respect to the target uniform distribution: the larger the value, the less balanced the labels.

approximates better the target distribution, i.e. it is the algorithm that allows better balancing of the labels of samples in memory. More in detail, it is noticeable how the KL distance, that we use here to measure the unbalancement, is roughly ten times larger in RS compared to OCDM. Indeed, in terms of performance, this results in very low results for low-freq labels, which are the most sensitive to how the memory is handled. Regarding BAT-OCDM, we have to consider that there is a trade-off between the balance of labels and the balance of tasks. This implies that the uniformity of label distribution will be inferior to OCDM, though also better than RS. However, as shown before in the performance table, this factor does not translate into inferior results since it also considers the balance of tasks that OCDM ignores.

4.4.3. Balance among tasks

The task distribution in memory achieved by the RS, OCDM, and BAT-OCDM approaches is shown in Fig. 5. By design, BAT-OCDM achieves a uniform allocation of the tasks (Fig. 5(e)). Task-based Random has the same characteristic because it maintains a separate memory for every task. For the Reservoir Sampling technique, the amount of samples in memory for each task is proportional to the dataset size of that task, as shown in Figs. 4(b) and 5(a). This is because random selection occurs at a fixed rate during the stream of samples corresponding to the various tasks. OCDM and Dataset-based OCDM have a similar distribution shown in Fig. 5(c). We can observe that for OCDM some tasks are almost absent from the memory, with the risk of being forgotten more quickly than others.

As mentioned previously, upon comparing the figures depicting the balance of labels and tasks with the corresponding performance results, it becomes evident that the trade-off offered by BAT-OCDM presents a superior choice over OCDM.

4.4.4. Computation times

In Table 4, we can examine the computing time to handle the memory for the RS, OCDM, and BAT-OCDM approaches. Compared to OCDM, BAT-OCDM requires four times less computing time to update memory. This outcome is consistent with the algorithm's anticipated theoretical complexity, which is logarithmic in the number of tasks T, as opposed to the original approach's linear in T. Dataset-based OCDM

Table 4
The table contains the performance for each approach. In each cell are showed two metrics defined like in Section 4.1. Above is the Average macro f1 S_T and below the Average Forgetting F_T. Based on the column, these metrics are calculated on a different set of labels. *Low*, *Medium*, and *High* are label sets grouped by the frequency of the labels, while *Total* consider all the labels together.

Approach	Total	Low	Medium	High	Time (s)
Finetune	0.19 (0.27)	0.04 (0.14)	0.21 (0.39)	0.38 (0.33)	–
Cumulative	0.39 –	0.17 –	0.42 –	0.7 –	–
Task-based Random	0.37 (0.11)	0.14 (0.09)	**0.4** (0.18)	0.69 (0.05)	0.3
Reservoir Sampling (RS)	0.37 (0.1)	0.14 (0.1)	**0.4** (0.16)	**0.69** (0.04)	0.3
OCDM	0.32 (0.2)	**0.2** (0.04)	0.32 (0.31)	0.49 (0.31)	2607.9
Dataset-based OCDM	0.32 0.2	**0.2** **0.04**	0.33 0.31	0.49 0.31	4702.3
BAT-OCDM	**0.38** (0.09)	**0.2** (0.04)	**0.4** (0.15)	0.64 (0.085)	**624.4**

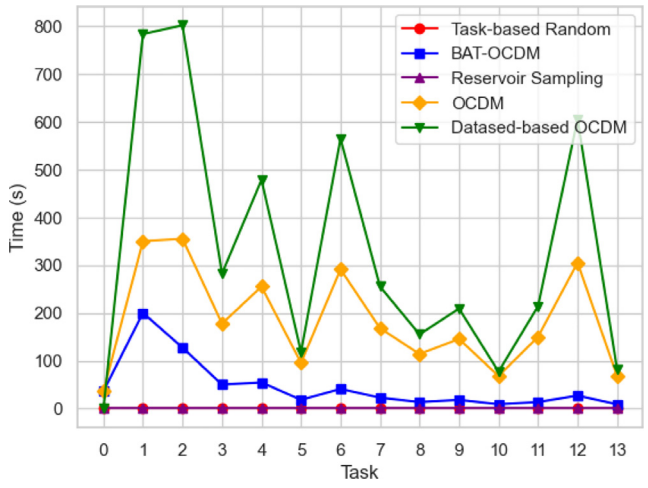


Fig. 6. The computation time of each technique to handle the selection of the samples to keep in memory and remove from it. It shows the time required for each task. On the y-axis is represented the time in seconds, and on x-axis is the current task.

is slower than OCDM because it takes into account all of the samples from a task at once. In practice, it is twice as slow as OCDM.

Additionally, the computation time (in seconds) for each task for all strategies is shown in Fig. 6 For OCDM, we can observe that the time needed for each task is proportional to the size of the task's dataset. In other words, the larger the task's dataset, the longer it will take. When using dataset-based OCDM, we can observe that a task takes twice as long to complete as when using OCDM. Finally, it appears that the BAT-OCDM computation time gradually reduces over time, which is consistent with the algorithm's logarithmic nature.

These aspects hold significant importance, particularly in practical applications, as our approach not only outperforms the state-of-the-art OCDM in terms of results but also exhibits significantly reduced computation times. This is not merely demonstrated through ablation studies, but is also supported theoretically by showcasing that our algorithm has logarithmic complexity with respect to the number of tasks, as proven in the appendix.

5. Conclusions and future works

As far as we know, this study is the first to address multi-label classification in the context of domain incremental learning. For multi-label classification, earlier Continual Learning methods have focused on the Class Incremental Scenario, in which additional labels are introduced into the task sequence. Instead, in Domain Incremental Learning, the set of labels remains the same while their distribution changes over time. To overcome this problem, we present BAT-OCDM, an effective replay-based method of managing memory updates. Furthermore, the proposed procedure exhibits higher performance than the simple adaptation of the previous techniques to the Domain Incremental Learning scenario, especially in the presence of class imbalance.

We validate the proposed approach on a real-world industrial Alarm Forecasting task stemming from the monitoring of packaging equipment. Obtained results suggest the efficacy of the proposed methodology, especially on low-freq labels. Moreover, the complexity of BAT-OCDM is logarithmic in the number of tasks. Therefore, the suggested method is more effective than earlier ones with linear complexity. Given the efficiency of BAT-OCDM, implementation on the Edge is a viable perspective. Moreover, this would be a step towards the Tiny ML paradigm, which is becoming increasingly popular, also in the scenario of the Industrial Internet of Things (Han and Siebert, 2022).

It could be remarkable to see other real-world applications of the multi-label setting in continual learning. For instance, it might be interesting to test the studied approaches on domains like medical images. Indeed, it is common to have multiple labels (i.e., multiple diseases associated with the same patient) and it might benefit from a scenario of continual learning (for instance, new diseases are discovered over time) that handles the imbalance among the diseases. On the more methodological side, the replay family of strategies is the primary focus of multi-label approaches in continual learning. However, approaches from other families (i.e. regularization-based and architectural-based) might benefit from a specific adaptation of the multilabel setting. Furthermore, imbalance within labels, imbalance between labels, and imbalance among label-sets are three different sorts of imbalanced problems that frequently arise in multi-label classification. The imbalance with labels is the primary concern of the studies in the CL context. Therefore, a complementary attention to the imbalance of the label-sets may help to further improve the performance. Other possible future research directions include validating BAT-OCDM performance in the Continual Incremental Learning scenario. Though the proposed algorithm allows for a significant reduction in the computation costs, the total complexity with regard to the dataset dimension is still quadratic. Therefore, we also envision further investigation of more efficient approaches.

CRediT authorship contribution statement

Davide Dalle Pezze: Conceptualization, Methodology, Software, Formal analysis, Resources, Writing – original draft, Writing – review editing, Visualization, Supervision. **Denis Deronjic:** Conceptualization, Methodology, Software, Validation, Resources, Writing – original draft, Visualization. **Chiara Masiero:** Conceptualization, Methodology, Investigation, Writing – original draft, Writing – review editing, Visualization, Supervision. **Diego Tosato:** Conceptualization, Investigation, Data curation. **Alessandro Beghi:** Supervision, Project administration, Funding acquisition. **Gian Antonio Susto:** Supervision, Writing – review editing, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Gian Antonio Susto reports financial support was provided by European Union. Gian Antonio Susto is associated editor of Engineering Applications of Artificial Intelligence.

Table A.5
Description of different symbols used in the appendix.

Symbol	Description
min	The min operator represents the minimum value between two numbers
$\cup$	It represents the set union operator. It is used to combine two sets into a single set that contains all the elements from both sets
$\in$	It is used to indicate that an element belongs to a particular set.
O	In computer science is used to describe the upper bound or worst-case scenario of the time or space complexity of an algorithm.
{}	It indicates an empty set
ln	It represents the natural logarithm function

Data availability

The dataset used is available at https://data.mendeley.com/datasets/4nhx2x67cd while the used code is available at https://github.com/dallepezze/bat-ocdm.

Appendix. Computational complexity

Below, we are going to show the calculations that led to the results on the complexity of the algorithms present in the table of the complexity of algorithms.

The analysis will be performed for the original approach OCDM, the Dataset-based OCDM and our method BAT-OCDM.

We are going to proof that our algorithm has a complexity wrt the number of tasks T that is logarithmic, while the complexity of OCDM and Dataset-based OCDM is lineart wrt T.

While, in the original paper is showed only the complexity for batch, we insight further and analyze the complexity for the training of an entire task and the training on all tasks.

We are going to analyze as first, the original approach OCDM. In the following section, we are going to assume that each task has the same dataset size D and the memory $\mathcal{M}$ has a fixed size M (see Table A.5).

A.1. OCDM

Studying the complexity for task, for each task we are going to iterate on the dataset $\mathcal{D}$ $\frac{D}{b}$ times, assuming that each batch will have size b.

In the original approach was proved that the complexity for the algorithm $Memory\_Update$(MU) using a memory of size M and removing b elements i.e. $MU(M,b)$ is:

$$MU\,(M,b) = O\left(b \cdot M - \frac{b\cdot(b-1)}{2}\right) \tag{A.1}$$

OCDM algorithm use Memory_Update to update the memory using a batch B of size b. In other words the algorithm MU given a set of M+b elements will remove b elements from it. The complexity of this is:

$$MU\,(M+b,b) \overset{(A.1)}{=} O\left(b\cdot(M+b) - \frac{b\cdot(b-1)}{2}\right) \tag{A.2}$$

to update a memory of size M removing b elements. The complexity of algorithm OCDM for a task consist in $\frac{D}{b}$ iterations over the MU method as indicated in (A.2), which correspond to:

$$\begin{aligned} OCDM_t\,(D,M) &= O\left(\frac{D}{b}\cdot[b\cdot(M+b) - \frac{b\cdot(b-1)}{2}]\right) \\ &= O\left(D\cdot[M+\frac{b+1}{2}]\right) \overset{M\gg b}{=} O\,(D\cdot M) \end{aligned} \tag{A.3}$$

Where $OCDM_t(D,M)$ is the complexity of the algorithm OCDM for a single task, assuming the number of the samples of the task as D and the memory size used as M.

Since the entire training correspond to iterate on T tasks, we obtain as overall performance for OCDM:

$$OCDM(D,M) = \sum_{t\in\mathcal{T}} OCDM_t(D,M) = O(T\cdot D\cdot M) \tag{A.4}$$

A.2. Dataset-based OCDM

Though the update per batch is essential for the Online Continual Learning(OCL) setting, we are obtain suboptimal solutions respect to find the optimal distribution using all dataset D at once. Since we are evaluating the DIL scenario where the data D of a task is received all together we also study the performance of MU(M,D) which correspond to the of the variant *Dataset-based OCDM* (Db-OCDM), while the original will remain OCDM.

$$\begin{aligned} Db-OCDM\,(D,M)_t &= MU\,(D+M,D) \\ &\overset{(A.1)}{=} O\left(D\cdot(M+D) - \frac{D\cdot(D-1)}{2}\right) \\ &\overset{D\gg 1}{=} O\left(M\cdot D + \frac{D^2}{2}\right) \end{aligned} \tag{A.5}$$

Since the entire training correspond to iterate on T tasks, we obtain as overall performance for Dataset-based:

$$Db-OCDM\,(D,M) = O\left(T\cdot[D\cdot M + \frac{D^2}{2}]\right) \tag{A.6}$$

A.3. BAT-OCDM

Below we show the complexity obtain considering our approach *BAT-OCDM*. In this case, the complexity is splitted in two subprocesses. The first one consist during the Task i to select $\frac{M}{i}$ samples from the data of the new task. Therefore the complexity of first part is equivalent to OCDM per task i.e. $OCDM_t(D,\frac{M}{i})$ will be $O(D\cdot\frac{M}{i})$.

$$OCDM_t\left(D,\frac{M}{i}\right) \overset{(A.3)}{=} O\left(D\cdot\frac{M}{i}\right) \tag{A.7}$$

The memory of an old task must be reduced from $\frac{M}{i-1}$ to $\frac{M}{i}$, therefore eliminating $\frac{M}{i\cdot(i-1)}$ samples.

To do this the complete complexity is $O(\frac{M^2}{i-1})$. In fact, we are going to perform $MU(\frac{M}{i-1},\frac{M}{i\cdot(i-1)})$ i-1 times during task i(assuming tasks start from 1 to T included). Therefore, for the second part we have:

$$\begin{aligned} &(i-1)\cdot MU\left(\frac{M}{i-1},\frac{M}{i\cdot(i-1)}\right) \\ &\overset{(A.1)}{=} (i-1)\cdot O\left(\frac{M}{i\cdot(i-1)}\cdot\frac{M}{i-1} - \frac{M^2}{2\cdot i^2\cdot(i-1)^2}\right) \\ &= O\left(\frac{M^2}{i\cdot(i-1)} - \frac{M^2}{2i^2\cdot(i-1)}\right) \\ &= O\left(\frac{M^2}{i\cdot(i-1)}\cdot(1-\frac{1}{2\cdot(i)})\right) \\ &= O\left(\frac{M^2}{i\cdot(i-1)}\right) \\ &= O\left(\frac{M^2}{i-1}\right) \end{aligned} \tag{A.8}$$

Therefore, we have the total complexity for task ith of algorithm BAT-OCDM is:

$$BAT-OCDM_t\,(D,M,i) = (A.8)+(A.7) = O\left(\frac{D\cdot M}{i} + \frac{M^2}{i-1}\right) \tag{A.9}$$

Where if i=1 then the second member is 0 since during i=1 there are not old tasks to update. To evaluate the complexity of the entire training

the calculations are trivial for OCDM and Dataset-based OCDM because it is enough to multiply the complexity by task for the value T, which is the total number of tasks. In the case for BAT-OCDM is a little more tricky because the task complexity depends by task i. To calculate is necessary to consider the following inequality:

$$\sum_{i=1}^{n} \frac{1}{i} \leq \ln n + 1 \tag{A.10}$$

Therefore, for the first part:

$$O\left(\sum_{i=1}^{T} D \cdot M \cdot \frac{1}{i}\right) \overset{(A.10)}{=} O\left(D \cdot M \cdot (\ln T + 1)\right) \tag{A.11}$$

In the same way we have for the second part:

$$O\left(\sum_{i=2}^{T} \frac{M^2}{i-1}\right) \overset{(k=i-1)}{=} O\left(\sum_{k=1}^{T-1} \frac{M^2}{k}\right) \overset{(A.10)}{=} \\ \overset{(A.10)}{=} O\left(M^2 \cdot [\ln(T-1)+1]\right) = O\left(M^2 \cdot [\ln(T)+1]\right) \tag{A.12}$$

In total we have:

$$BAT - OCDM\,(D, M) = (A.11)+(A.12) \\ = O\left((\ln T + 1) \cdot (D \cdot M + M^2)\right) \tag{A.13}$$

References

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., Tuytelaars, T., 2018. Memory aware synapses: Learning what (not) to forget. In: Proceedings of the European Conference on Computer Vision. ECCV, pp. 139–154.

Bajic, B., Cosic, I., Lazarevic, M., Sremcev, N., Rikalovic, A., 2018. Machine Learning Techniques for Smart Manufacturing: Applications and Challenges in Industry 4.0, Vol. 29. Department of Industrial Engineering and Management Novi Sad, Serbia.

Borlea, I.-D., Precup, R.-E., Borlea, A.-B., 2022. Improvement of K-means cluster quality by post processing resulted clusters. Procedia Comput. Sci. 199, 63–70.

Buzzega, P., Boschini, M., Porrello, A., Calderara, S., 2021. Rethinking experience replay: A bag of tricks for continual learning. In: 2020 25th International Conference on Pattern Recognition. ICPR, IEEE, pp. 2180–2187.

Cai, S., Palazoglu, A., Zhang, L., Hu, J., 2019. Process alarm prediction using deep learning and word embedding methods. ISA Trans. 85, 274–283.

Charte, F., Rivera, A.J., del Jesus, M.J., Herrera, F., 2019. Dealing with difficult minority labels in imbalanced mutilabel data sets. Neurocomputing 326, 39–53.

Chaudhry, A., Dokania, P.K., Ajanthan, T., Torr, P.H., 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In: Proceedings of the European Conference on Computer Vision. ECCV, pp. 532–547.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M., 2019. On tiny episodic memories in continual learning. arXiv preprint arXiv:1902.10486.

Dalle Pezze, D., Masiero, C., Tosato, D., Beghi, A., Susto, G.A., 2021. FORMULA: A deep learning approach for rare alarms predictions in industrial equipment. IEEE Trans. Autom. Sci. Eng. 19 (3), 1491–1502.

De Lange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T., 2021. A continual learning survey: Defying forgetting in classification tasks. IEEE Trans. Pattern Anal. Mach. Intell. 44 (7), 3366–3385.

Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., Tuytelaars, T., 2021. A continual learning survey: Defying forgetting in classification tasks. IEEE Trans. Pattern Anal. Mach. Intell..

Domingues, R., Filippone, M., Michiardi, P., Zouaoui, J., 2018. A comparative evaluation of outlier detection algorithms: Experiments and analyses. Pattern Recognit. 74, 406–421.

Du, M., Li, F., Zheng, G., Srikumar, V., 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1285–1298.

Essien, A., Giannetti, C., 2020. A deep learning model for smart manufacturing using convolutional LSTM neural network autoencoders. IEEE Trans. Ind. Inf. 16 (9), 6069–6078.

Fan, S.-K.S., Hsu, C.-Y., Tsai, D.-M., He, F., Cheng, C.-C., 2020. Data-driven approach for fault detection and diagnostic in semiconductor manufacturing. IEEE Trans. Autom. Sci. Eng. 17 (4), 1925–1936.

Ge, W., Yang, S., Yu, Y., 2018. Multi-evidence filtering and fusion for multi-label classification, object detection and semantic segmentation based on weakly supervised learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1277–1286.

Gentner, N., Carletti, M., Kyek, A., Susto, G.A., Yang, Y., 2021. DBAM: Making virtual metrology/soft sensing with time series data scalable through deep learning. Control Eng. Pract. 116, 104914.

Goodfellow, I.J., Mirza, M., Xiao, D., Courville, A., Bengio, Y., 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. arXiv preprint arXiv:1312.6211.

Han, H., Siebert, J., 2022. TinyML: A systematic review and synthesis of existing research. In: 2022 International Conference on Artificial Intelligence in Information and Communication. ICAIIC, pp. 269–274. http://dx.doi.org/10.1109/ICAIIC54071.2022.9722636.

Kim, C.D., Jeong, J., Kim, G., 2020. Imbalanced continual learning with partitioning reservoir sampling. In: European Conference on Computer Vision. Springer, pp. 411–428.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al., 2017. Overcoming catastrophic forgetting in neural networks. Proc. Natl. Acad. Sci. 114 (13), 3521–3526.

Langarica, S., Rüffelmacher, C., Núñez, F., 2019. An industrial internet application for real-time fault diagnosis in industrial motors. IEEE Trans. Autom. Sci. Eng. 17 (1), 284–295.

Li, W., Li, H., Gu, S., Chen, T., 2020. Process fault diagnosis with model-and knowledge-based approaches: Advances and opportunities. Control Eng. Pract. 105, 104637.

Li, H., Parikh, D., He, Q., Qian, B., Li, Z., Fang, D., Hampapur, A., 2014. Improving rail network velocity: A machine learning approach to predictive maintenance. Transp. Res. C 45, 17–26.

Liang, Y.-S., Li, W.-J., 2022. Optimizing class distribution in memory for multi-label continual learning. URL https://openreview.net/forum?id=HavXnq6KyT3.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P., 2017. Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2980–2988.

Lopez-Paz, D., Ranzato, M., 2017. Gradient episodic memory for continual learning. Adv. Neural Inf. Process. Syst. 30.

Ma, X., Si, Y., Qin, Y., Wang, Y., 2022. Fault detection for dynamic processes based on recursive innovational component statistical analysis. IEEE Trans. Autom. Sci. Eng..

Madjarov, G., Kocev, D., Gjorgjevikj, D., Džeroski, S., 2012. An extensive experimental comparison of methods for multi-label learning. Pattern Recognit. 45 (9), 3084–3104. http://dx.doi.org/10.1016/j.patcog.2012.03.004, URL http://www.sciencedirect.com/science/article/pii/S0031320312001203, Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA'2011).

Maggipinto, M., Beghi, A., Susto, G.A., 2022. A deep convolutional autoencoder-based approach for anomaly detection with industrial, non-images, 2-dimensional data: A semiconductor manufacturing case study. IEEE Trans. Autom. Sci. Eng..

Masarczyk, W., Wawrzyński, P., Marczak, D., Deja, K., Trzciński, T., 2022. Logarithmic continual learning. arXiv preprint arXiv:2201.06534.

Parisi, G.I., Kemker, R., Part, J.L., Kanan, C., Wermter, S., 2019. Continual lifelong learning with neural networks: A review. Neural Netw. 113, 54–71.

Pellegrini, L., Graffieti, G., Lomonaco, V., Maltoni, D., 2020. Latent replay for real-time continual learning. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems. IROS, IEEE, pp. 10203–10209.

Puruncajas, B., Vidal, Y., Tutivén, C., 2020. Damage detection and diagnosis for offshore wind foundations. In: ICINCO. pp. 181–187.

Read, J., Pfahringer, B., Holmes, G., Frank, E., 2009. Classifier chains for multi-label classification. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, pp. 254–269.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., Lampert, C.H., 2017. icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2001–2010.

Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T., Wayne, G., 2019. Experience replay for continual learning. Adv. Neural Inf. Process. Syst. 32.

Rusu, A.A., Rabinowitz, N.C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., Hadsell, R., 2016. Progressive neural networks. arXiv preprint arXiv:1606.04671.

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., Valaee, S., 2017. Recent advances in recurrent neural networks. arXiv preprint arXiv:1801.01078.

Susto, G.A., Schirru, A., Pampuri, S., Beghi, A., De Nicolao, G., 2018. A hidden-Gamma model-based filtering and prediction approach for monotonic health factors in manufacturing. Control Eng. Pract. 74, 84–94.

Tosato, D., Dalle Pezze, D., Masiero, C., Beghi, A., Susto, G.A., 2020. Alarm logs in packaging industry. http://dx.doi.org/10.17632/4nhx2x67cd.1.

Van de Ven, G.M., Tolias, A.S., 2019. Three scenarios for continual learning. arXiv preprint arXiv:1904.07734.

Van Horn, G., Perona, P., 2017. The devil is in the tails: Fine-grained classification in the wild. arXiv preprint arXiv:1709.01450.

Verma, A., Meenpal, T., Acharya, B., 2022. Computational cost reduction of convolution neural networks by insignificant filter removal. Sci. Technol. 25 (2), 150–165.

Villalobos, K., Suykens, J., Illarramendi, A., 2020. A flexible alarm prediction system for smart manufacturing scenarios following a forecaster–analyzer approach. J. Intell. Manuf. 1–22.

Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., Xu, W., 2016. CNN-RNN: A unified framework for multi-label image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2285–2294.

Xu, Y., Wang, J., Yu, Y., 2019. Alarm event prediction from historical alarm flood sequences based on Bayesian estimators. IEEE Trans. Autom. Sci. Eng. 17 (2), 1070–1075.

Yen, T.-F., Oprea, A., Onarlioglu, K., Leetham, T., Robertson, W., Juels, A., Kirda, E., 2013. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In: Proceedings of the 29th Annual Computer Security Applications Conference. pp. 199–208.

Yoon, J., Yang, E., Lee, J., Hwang, S.J., 2017. Lifelong learning with dynamically expandable networks. arXiv preprint arXiv:1708.01547.

Yuan, X., Qi, S., Wang, Y., Xia, H., 2020. A dynamic CNN for nonlinear dynamic feature learning in soft sensor modeling of industrial process data. Control Eng. Pract. 104, 104614.

Zenke, F., Poole, B., Ganguli, S., 2017. Continual learning through synaptic intelligence. In: International Conference on Machine Learning. PMLR, pp. 3987–3995.

Zhang, M.-L., Zhou, Z.-H., 2013. A review on multi-label learning algorithms. IEEE Trans. Knowl. Data Eng. 26 (8), 1819–1837.

Zhang, J., Zhou, D., Chen, M., Hong, X., 2022. Continual learning for multimode dynamic process monitoring with applications to an ultra-supercritical thermal power plant. IEEE Trans. Autom. Sci. Eng..

Zhu, J., Wang, C., Li, C., Gao, X., Zhao, J., 2016. Dynamic alarm prediction for critical alarms using a probabilistic model. Chin. J. Chem. Eng. 24 (7), 881–885.