Enabling Memory-Efficient On-Device Learning via Dataset Condensation

Gelei Xu[1], Ningzhi Tang[1], Jun Xia[1], Ruiyang Qin[1], Wei Jin[2], Yiyu Shi[1]
[1]University of Notre Dame, [2]Emory University
{gxu4, ntang, jxia4, rqin, yshi4}@nd.edu, wei.jin@emory.edu

Abstract—**Upon deployment to edge devices, it is often desirable for a model to further learn from streaming data to improve accuracy. However, learning from such data is challenging because it is typically unlabeled, non-independent and identically distributed (non-i.i.d), and only seen once, which can lead to potential catastrophic forgetting. A common strategy to mitigate this issue is to maintain a small data buffer on the edge device to select and retain the most representative data for rehearsal. However, the selection process leads to significant information loss since most data is either never stored or quickly discarded. This paper proposes a framework that addresses this issue by condensing incoming data into informative synthetic samples. Specifically, to effectively handle unlabeled incoming data, we propose a pseudo-labeling technique designed for on-device learning environments. We also develop a dataset condensation technique tailored for on-device learning scenarios, which is significantly faster compared to previous methods. To counteract the effects of noisy labels during the condensation process, we further utilize a feature discrimination objective to improve the purity of class data. Experimental results indicate substantial improvements over existing methods, especially under strict buffer limitations. For instance, with a buffer capacity of just one sample per class, our method achieves a 56.7% relative increase in accuracy compared to the best existing baseline on the CORe50 dataset.**

I. Introduction

Deep learning models have seen extensive application in edge devices, such as robots used in search operations and UAVs for wildfire surveillance [1]–[3]. These models are typically pre-trained on high-performance servers and are then learning on devices in unfamiliar environments, thus continually improving their generalization performance. However, in real-world scenarios, *on-device learning* faces the challenge of processing non-i.i.d. data that arrives in dynamic streams with temporal correlations. Take an autonomous vehicle, for instance: its camera might first capture consecutive video frames of a car in motion, followed by a series of traffic signs. Adding to the complexity, due to the limited storage capacities of edge devices, data streams cannot be permanently stored and some of them can be seen only once. Under these conditions, on-device learning can result in an issue known as catastrophic forgetting [4]–[6], where the model loses previously acquired knowledge when learning newly arrived data.

To mitigate catastrophic forgetting in on-device learning, replay-based strategies are commonly employed by maintaining a limited-size buffer that stores a selection of past samples [7]–[10].This buffer is used for rehearsal; as new data arrives, it is partially refreshed by replacing some older samples with representative new ones. The model is then trained on this evolving buffer, helping it maintain knowledge from earlier data and enhance its overall performance. However, despite their effectiveness, these strategies come with serious limitations. First, individual samples typically contain a low density of information, and the extremely limited storage capacity of the buffer further restricts the total amount of information it can hold. Second, the buffer must continually remove old samples to make room for new samples, many of which still hold critical value. As a result, vital information is often lost over time. These challenges raise the question: *How can we enhance the information density of the buffer while also preserving the information from old data when learning newly arrived data?*

Building upon the latest breakthroughs in dataset condensation [11]–[13], we propose a novel solution to buffer maintenance: condensing incoming data into the buffer without discarding existing samples. Dataset condensation techniques aim to distill a large dataset into a smaller synthetic one, preserving sufficient information for effective model training [11]. For example, a model can achieve 97.4% test accuracy on MNIST using just 100 synthetic samples, compared to 99.6% with the full 60,000-image set [12]. Employing such techniques means the stored images are not limited to just the incoming data. Instead, it provides the flexibility to condense the representative features of incoming data into the existing buffer. Consequently, this approach can enhance the buffer's information density and offers a powerful strategy to mitigate catastrophic forgetting.

Challenges in On-Device Learning. While dataset condensation offers a potential solution for on-device learning, its direct application is non-trivial as it presents several challenges. First, the effectiveness of existing dataset condensation methods relies on the availability of label information. Many current approaches assume labels are readily accessible [14]–[16], but this is often unrealistic in on-device contexts, where labeling data in real-time after sensor capture is impractical. Second, even when data can be labeled and condensed into the buffer, label noise can significantly degrade the quality of the condensed data, thereby reducing the learning performance of the model. Third, dataset condensation methods were originally designed for offline learning settings. These methods typically use bi-level optimization, which requires multiple iterations of model updates before the synthetic data is updated. While effective, this iterative approach is time-consuming and computationally demanding, which poses significant challenges for implementation in on-device learning scenarios [17].

To address these challenges, we propose a framework, DECO[1], for on-device learning that updates the buffer's synthetic data through an efficient dataset condensation technique.

[1]DECO stands for on-**D**evice **E**fficient **CO**ndensation.

When new data arrives, it is initially assigned pseudo-labels and filtered through majority voting. It is then efficiently condensed into the buffer using several acceleration techniques. Additionally, to mitigate the effects of inaccurate labels and improve the quality of the synthetic data, a feature discrimination module is used to enhance class purity within the buffer. Our results demonstrate significant improvements over existing methods. The key contributions of our approach are as follows:

(a) To our knowledge, we are the first to develop dataset condensation techniques for on-device learning challenges.
(b) We introduce a simple yet effective labeling technique for streaming data in on-device learning environments. Additionally, we design a feature discrimination approach to mitigate the negative impact of incorrect pseudo-labels.
(c) We optimize the algorithm to significantly speed up the condensation process, making it feasible for on-device settings without compromising accuracy.
(d) Experiments show that our method greatly outperforms existing methods, especially under strict buffer limitations.

II. Background and Related Work

A. On-Device Learning

On-device learning learns from an ongoing data stream on devices with limited memory resources [18]–[21]. In this process, data may be presented in a non-i.i.d. manner, and each data sample is seen only once [22]. Due to the continuously changing distribution of data streams, models often face the challenge of catastrophic forgetting. Replay-based strategies have proven effective in mitigating these issues by storing a selected subset of samples for later review. [23] proposes maintaining a set of representative examples per class, chosen to closely represent the class averages in the feature space. [24] focus on preserving previous knowledge by aligning new predictions with past logits. [16] leverage contrastive learning features to evaluate the significance of individual samples. Additionally, [25] integrates a nearest-mean classifier with an efficient reservoir sampling approach to enhance learning continuity. Although these methods can be effective by selecting the most representative samples to store in the buffer, the information density of the buffer remains low due to the limited information each sample contains. Furthermore, the buffer must continuously remove old samples to accommodate new incoming data. Consequently, over time, a substantial amount of information is lost, as the majority of data is not fully utilized before being replaced. To address this challenge, it is crucial to develop a method that can increase the information density of the buffer while preserving the information from old data when new data arrives.

B. Dataset Condensation

Dataset condensation is designed to create a smaller, synthetic dataset from a larger training dataset. Training a model on this condensed dataset aims to achieve results comparable to those obtained from the original dataset. This concept was originally proposed by [11], framing the condensation process as a learning-to-learn problem. Subsequent studies have focused on matching selected knowledge during network training, such as feature distributions [13], gradients, and training trajectories [26]. Building on this foundation, several approaches have been developed with subtle variations [27]–[29]. While several condensation techniques are available, in this paper, we focus on *gradient matching* due to its intuitive approach and strong performance [12]. Note that similar to gradient matching, other dataset condensation techniques are also designed for offline use and face the same challenges such as lack of labels and high computational costs. Therefore, the method proposed in our paper can be flexibly adapted to other dataset condensation techniques as well.

The core idea of gradient matching is to replicate the training trajectory of the original dataset. Specifically, this technique aims to minimize the difference between the model's gradients with respect to the real and synthetic data at each training epoch. By doing so, the model θ optimized on the synthetic dataset will closely match those obtained from the original dataset. Let θ_t denote the model parameters at the t-th epoch, $\mathcal{S}$ as the synthetic dataset, and $\mathcal{R}$ as the original dataset. The gradient matching process can be formulated as follows:

$$\arg\min_{\mathcal{S}} \sum_{t=0}^{T-1} \mathcal{D}(\nabla_\theta \mathcal{L}_{\theta_t}(\mathcal{R}), \nabla_\theta \mathcal{L}_{\theta_t}(\mathcal{S})), \quad \text{s.t.} \quad \theta_{t+1} = \text{opt}_\theta(\theta_t, \mathcal{S}), \tag{1}$$

where $\mathcal{D}$ is the distance metric, T represents the total number of training epochs, and opt_θ is the optimization algorithm applied to the loss function $\mathcal{L}_\theta$.

III. Methodology

We first introduce some basic notations used in this paper. After deploying the model θ on the edge device, it continuously learns from the input data stream $\mathcal{I}$ received on the fly. Note that the data instances within $\mathcal{I}$ are not labeled, yet they have potential category candidates $\mathcal{C}$. We maintain a limited-size data buffer on the edge device to store the condensed dataset $\mathcal{S} = \{(\mathbf{x}'_i, y'_i)\}$. The synthetic data instances $(\mathbf{x}'_i, y'_i)$ are evenly distributed across classes to ensure class balance. That is, for each class $c \in \mathcal{C}$, the number of instances $|\{(\mathbf{x}'_i, y'_i)|(\mathbf{x}'_i, y'_i) \in \mathcal{S}, y'_i = c\}|$ equals $|\mathcal{S}|/|\mathcal{C}|$. Our goal is to enable the deployed model θ to learn from unlabeled input streaming data $\mathcal{I}$ while minimizing the forgetting of previously acquired knowledge [30]. Therefore, the overall workflow of on-device learning through dataset condensation can be summarized in two stages: (1) storing knowledge from the input data stream $\mathcal{I}$ into a condensed dataset $\mathcal{S}$ through gradient matching technique, and (2) the deployed model θ continually learns from $\mathcal{S}$ after several rounds of condensation.

A. Overview of the Framework

Fig. 1 provides the framework for DECO. The model is pre-trained on a small amount of labeled data before deployment on the edge device, and the buffer is initialized with data that are condensed using such labeled data in offline settings. As the on-device learning begins and new segments of streaming data arrive, the system assigns pseudo-labels and filters

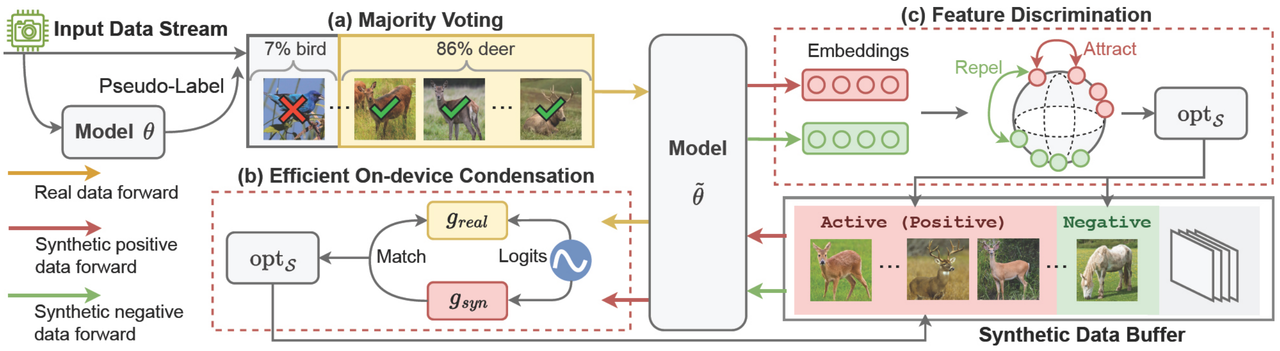


Fig. 1: Overview of DECO. The process begins by labeling and filtering the incoming unlabeled data stream through (a) *majority voting*. Subsequently, the limited-size synthetic data buffer is updated through (b) *efficient on-device condensation* and (c) *feature discrimination* among the synthetic data within the buffer.

the data using (a) *majority voting* (Section III-B). Next, the data of active classes is condensed to these corresponding synthetic samples through (b) *efficient on-device condensation* (Section III-C). Specifically, we update the buffer's synthetic samples by matching their gradients with the gradients of labeled incoming data. To reduce the impact of mislabeling, (c) *feature discrimination* is applied to enhance the purity of the classes (Section III-D). The model is updated every β step to continuously learn from the input data stream. This iterative process aims to enhance the performance of the on-device model to real-world data after deployment.

B. Majority Voting based Pseudo-Label Assignment

To ensure an effective dataset condensation process, it is crucial to initially assign labels to the input streaming data with an acceptable level of noise. A natural idea to obtain the label for unlabeled data involves self-training [31], [32]. This method generates pseudo-labels by using the predictions of the existing model. By continuously applying these pseudo-labels, the model can progressively learn from the unlabeled data, thereby refining the accuracy of subsequent pseudo-labels. Self-training relies on fine-tuning an already optimized model, which demands high initial accuracy. However, in on-device learning scenarios, even though the model is typically pre-trained on high-performance servers, its initial accuracy may not suffice when deployed at the edge, potentially leading to the generation of low-quality pseudo-labels. This uncertainty can even degrade model accuracy when trained with these imprecise labels. Thus, a critical challenge emerges: How to ensure relatively high predictive accuracy of the pseudo-labels?

To enhance the accuracy of pseudo-labels, we recognize the temporal correlation in the input streaming data. When the model learns to identify a specific object from multiple views, it would receive many temporally sequential examples of the same class. Therefore, we can infer that the data received within a certain timeframe are more likely to belong to the same class. For instance, if a majority of images receive the pseudo-label "deer" within a brief period, it strongly suggests the presence of a deer at that time. Conversely, if only a few images are labeled as "truck" or "boat" among those identified as "deer", there is a high likelihood that these images are mislabeled.

Building on this inference, we observe that in a segment of the data stream, the frequency of predictions for each class can help verify the accuracy of pseudo-labels. Thus, a simple yet effective majority voting method is proposed to filter out the samples with low-confidence pseudo-labels. Suppose $\mathcal{I}_t = \{\mathbf{x}_i\}$ represents the t-th segment of input data stream from $\mathcal{I}$, where $\mathbf{x}_i$ is the i-th unlabeled instance in $\mathcal{I}_t$. After the model assigns pseudo-labels $\hat{y}_i$ to each instance $\mathbf{x}_i$ within the segment, a sliding window is used to monitor the frequency of predictions. For simplicity, we set the size of the sliding window equal to the size of the segment, denoted as $|\mathcal{I}_t|$. We then count the occurrences of each pseudo-label $\hat{y}_i$ for every class $c \in \mathcal{C}$ within this window. The classes are considered "active" in the current window, identified as $\mathcal{C}_t^A$, if the ratio of the count of their pseudo-labels to the size of the segment exceeds a predetermined threshold m:

$$\mathcal{C}_t^A = \{c \in \mathcal{C} | \sum_{i=1}^{|\mathcal{I}_t|} \mathbb{1}\left(\hat{y}_i = c\right) / |\mathcal{I}_t| > m\}. \tag{2}$$

After the counting process is completed, the active classes $\mathcal{C}_t^A$ are identified within the current sliding window. This identification implies that the model updates only the synthetic data corresponding to the labels in $\mathcal{C}_t^A$ within this segment. We finally obtain the active data instances in $\mathcal{I}_t$ along with their pseudo-labels and the active synthetic subset, denoted as:

$$\begin{aligned} \mathcal{I}_t^A &= \{(\mathbf{x}_i, \hat{y}_i) | \mathbf{x}_i \in \mathcal{I}_t, \hat{y}_i \in \mathcal{C}_t\}, \\ \mathcal{S}_t^A &= \{(\mathbf{x}'_i, y'_i) | (\mathbf{x}'_i, y'_i) \in \mathcal{S}, y'_i \in \mathcal{C}_t^A\}. \end{aligned} \tag{3}$$

C. Efficient On-Device Dataset Condensation

After assigning pseudo-labels to the data stream and performing majority voting, gradient matching is then used to condense the data into the buffer. We employ a confidence-weighted cross-entropy loss as the model's learning objective for matching. Denote $\mathcal{X}$ and $\mathcal{Y}$ as the general image set and label set, respectively. The loss function is defined as follows:

$$\mathcal{L}_\theta(\mathcal{X}, \mathcal{Y}) = -\sum_{i=1}^{|\mathcal{X}|} w_i \sum_{c \in \mathcal{C}} y_{i,c} \log p_\theta(\mathbf{x}_i)_c, \tag{4}$$

where $y_{i,c}$ equals 1 if $y_i = c$, and 0 otherwise. The weight w_i is assigned a value of 1 for synthetic data. For real data, it is set to the confidence score $p_\theta(\mathbf{x}i)\hat{y}_i$ to prioritize higher confidence labels for a more accurate classification.

The vanilla gradient matching framework described in Eq. (1) is a two-level optimization problem, requiring updates to synthetic images $\mathcal{S}$ in the outer loop and optimization of network parameters θ_t in the inner loop. However, this nested optimization process is computationally expensive and time-consuming for on-device learning, necessitating a more simplified approach. Drawing theoretical inspiration from [33], it is noted that the outer loop plays a more crucial role than the inner loop. Empirical observations further support this refinement: (1) there is a significant reduction in gradient matching loss immediately following model initialization, and (2) using multiple randomized models for a single step of gradient matching yields significantly better results than using one model across multiple steps. Based on these findings, we introduce a simplified one-step gradient matching strategy to speed up the condensation process. This approach focuses exclusively on matching the gradients during the first epoch after model initialization while bypassing the training trajectory. Thus, the objective function simplifies by omitting $\sum_{t=0}^{T-1}$ in Eq. (1), which reduces constraints on the model's training trajectories. The initial randomized model parameters are denoted as $\tilde{\theta}$. $\mathcal{X}_t$ and $\hat{\mathcal{Y}}_t$ represent these sets for active incoming data $\mathcal{I}_t^A$, while $\mathcal{X}'_t$ and $\mathcal{Y}'_t$ represent the sets for the active synthetic data $\mathcal{S}_t^A$. The new objective for efficient gradient matching is:

$$\arg\min_{\mathcal{X}'_t} \mathcal{D}(\nabla_{\tilde{\theta}}\mathcal{L}_{\tilde{\theta}}(\mathcal{X}'_t, \mathcal{Y}'_t), \nabla_{\tilde{\theta}}\mathcal{L}_{\tilde{\theta}}(\mathcal{X}_t, \hat{\mathcal{Y}}_t)). \tag{5}$$

Furthermore, computing the value of Eq. (5) requires a second-order derivative of $\mathcal{D}$ with respect to $\mathcal{X}'_t$, which is computationally expensive. For simplicity, we denote $g_{\text{syn}} = \nabla_{\tilde{\theta}}\mathcal{L}_{\tilde{\theta}}(\mathcal{X}'_t, \mathcal{Y}'_t)$, $g_{\text{real}} = \nabla_{\tilde{\theta}}\mathcal{L}_{\tilde{\theta}}(\mathcal{X}_t, \hat{\mathcal{Y}}_t)$. Applying the chain rule to compute the gradient of the synthetic data yields:

$$\begin{aligned}\nabla_{\mathcal{X}'_t}\mathcal{D}(g_{\text{syn}}, g_{\text{real}}) &= \nabla_{g_{\text{syn}}}\mathcal{D}(g_{\text{syn}}, g_{\text{real}}) \cdot \nabla_{\mathcal{X}'_t} g_{\text{syn}} \\ &= \nabla_{g_{\text{syn}}}\mathcal{D}(g_{\text{syn}}, g_{\text{real}}) \cdot \nabla_{\mathcal{X}'_t}\nabla_{\tilde{\theta}}\mathcal{L}_{\tilde{\theta}}(\mathcal{X}'_t, \mathcal{Y}'_t).\end{aligned} \tag{6}$$

This process is computationally demanding due to the costly matrix-vector product. Fortunately, the complexity can be substantially reduced using finite difference approximation. With ϵ as a small scalar[2] and $\tilde{\theta}^{\pm} = \tilde{\theta} \pm \epsilon \cdot \nabla_{g_{\text{syn}}}\mathcal{D}(g_{\text{syn}}, g_{\text{real}})$ we have:

$$\nabla_{\mathcal{X}'_t}\mathcal{D}(g_{\text{syn}}, g_{\text{real}}) \approx \frac{1}{2\epsilon}\left(\nabla_{\mathcal{X}'_t}\mathcal{L}_{\tilde{\theta}^+}(\mathcal{X}'_t, \mathcal{Y}'_t) - \nabla_{\mathcal{X}'_t}\mathcal{L}_{\tilde{\theta}^-}(\mathcal{X}'_t, \mathcal{Y}'_t)\right). \tag{7}$$

This approximation requires a total of five forward-backward passes to compute the gradient of the synthetic data $\mathcal{X}'_t$ with respect to the distance metric $\mathcal{D}$: the computation of $g_{\text{syn}}, g_{\text{real}}$, $\nabla_{g_{\text{syn}}}\mathcal{D}(g_{\text{syn}}, g_{\text{real}})$, and two additional terms in Eq. (7). The computational efficiency is further enhanced as both the time and space complexity of our method are reduced from $\mathcal{O}(|\tilde{\theta}| \cdot |\mathcal{X}'_t|)$ to $\mathcal{O}(|\tilde{\theta}| + |\mathcal{X}'_t|)$.

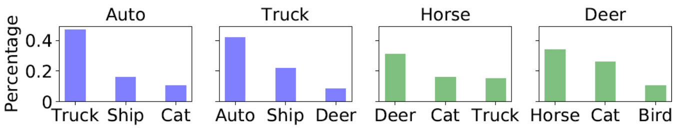


Fig. 2: Top 3 classes most frequently misclassified in CIFAR-10 for selected classes.

D. Enhancing Class Purity through Feature Discrimination

During the condensation process, images are grouped into their corresponding classes based on pseudo-labels. However, issues arise when these pseudo-labels are incorrect. Fig. 2 shows the top three classes most frequently misclassified within the CIFAR-10 dataset [35], with the y-axis representing their respective proportions of all misclassifications. Notably, the most frequently misclassified classes often share similar features with the actual class, suggesting that incorrect pseudo-labels are likely to result in images being erroneously grouped with other similar classes. Over time, this misclassification may cause the synthetic data representing similar classes increasingly indistinguishable, thereby reducing the quality of the buffer.

To improve feature discrimination between classes within the buffer, we incorporated principles from contrastive learning [36]. This approach clusters data from the same classes more closely together while seperating data from different classes, enhancing class distinctiveness. Specifically, for each active sample in $\mathcal{S}_t^A$, we found its corresponding index i in $\mathcal{S}$, i.e., $(\mathbf{x}'_i, y'_i) \in \mathcal{S}$. Similarly, the set of indices of all current active samples in $\mathcal{S}$ is denoted as A. All samples of class y'_i except for the sample itself are considered positive samples, with their indices denoted by $P(i) = \{j|(\mathbf{x}_j, y_j) \in \mathcal{S}, y'_j = y'_i, j \neq i\}$. A class different from y_i is randomly selected as the negative class, denoted by $c_i^{\text{neg}} \in \mathcal{C}, c_i^{\text{neg}} \neq y'_i$. All samples of class c_i^{neg} are considered negative samples, with their indices represented by $N(i) = \{j|(\mathbf{x}_j, y_j) \in \mathcal{S}, y'_j = c_i^{\text{neg}}\}$. Assuming f_θ is the encoder of the deployed model θ, we denote $z'_i = f_\theta(\mathbf{x}'_i)$ as the feature representation of $\mathbf{x}'_i \in \mathcal{S}$. The feature discrimination loss is defined as follows:

$$\mathcal{L}_{\text{disc},\mathcal{S}} = \sum_{i\in A}\frac{-1}{|P(i)|}\sum_{p\in P(i)}\log\frac{\exp(z'_i \cdot z'_p/\tau)}{\sum_{n\in N(i)}\exp(z'_i \cdot z'_n/\tau)}, \tag{8}$$

where $\cdot$ denotes the inner (dot) product, and τ denotes the temperature. By doing this, the model learns closely aligned representations for all samples from the same class, while pushing apart representations from different classes.

E. Overall Optimization

Eq. (9) below represents the overall optimization for $\mathcal{S}$.

$$\text{opt}_{\mathcal{S}}\left(\nabla_{\mathcal{S}}\mathcal{D}(g_{\text{syn}}, g_{\text{real}}) + \alpha \cdot \nabla_{\mathcal{S}}\mathcal{L}_{\text{disc},\mathcal{S}}\right). \tag{9}$$

Here, α is a weighting factor that balances the gradient matching loss with the feature discrimination loss. $\text{opt}_{\mathcal{S}}$ is an optimizer to update the condensed dataset $\mathcal{S}$. After every β

[2]We used $\epsilon = 0.01/||\nabla_{g_{\text{syn}}}\mathcal{D}(g_{\text{syn}}, g_{\text{real}})||_2$ in our experiments, as suggested in previous work [34], and found it to be sufficiently accurate.

segments streaming data, $\mathcal{S}$ is used to update the deployed model θ with opt_θ. The whole process is shown in Algorithm 1.

Algorithm 1 The proposed DECO algorithm

```
Input: input data stream $\mathcal{I}$, initial model parameters $\theta_0$, iteration number $L$, model optimizer $\text{opt}_\theta$, synthetic data optimizer $\text{opt}_\mathcal{S}$, hyper-parameters $\tau, \alpha$, $\beta$
Output: condensed dataset $\mathcal{S}$, updated model parameters $\theta$
  Deploy pre-trained model $\theta \leftarrow \theta_0$
  Initialize condensed dataset $\mathcal{S}$ in buffer
  for $\mathcal{I}_t \in \mathcal{I}$ do
    Assign pseudo-labels $\hat{y}_i$ for each $\mathbf{x}_i \in \mathcal{I}_t$
    Identify active classes $\mathcal{C}_t^A$ from majority voting   ▷ Eq. (2)
    Filter active data $\mathcal{I}_t^A, \mathcal{S}_t^A$   ▷ Eq. (3)
    for $l \leftarrow 1$ to $L$ do
      Randomize initial model parameters $\tilde{\theta}$
      Compute model gradients $g_{\text{syn}}, g_{\text{real}}$
      Perform gradient matching for $\nabla_{\mathcal{X}_t'}\mathcal{D}$   ▷ Eq. (7)
      Compute feature discrimination loss $\mathcal{L}_{\text{disc},\mathcal{S}}$   ▷ Eq. (8)
      Update condensed dataset $\mathcal{S}$ with $\text{opt}_\mathcal{S}$   ▷ Eq. (9)
    end for
    if $t \ \% \ \beta = 0$ then
      Update deployed model $\theta \leftarrow \text{opt}_\theta(\theta, \mathcal{S})$
    end if
  end for
```

IV. EXPERIMENTS

A. Experimental Setups

1) Datasets: We utilize two types of datasets for our experiments. The first type includes two streaming learning datasets iCub World 1.0 (iCub1) [37] and CORe50 [38], both specifically designed to test an algorithm's ability to learn from near real-time videos with temporal dependencies. iCub1 comprises household objects categorized into 10 classes, while CORe50 features similar objects distributed across 10 classes observed in 11 different environments. To evaluate our framework's performance on datasets with more classes and higher resolution images, we also selected two widely used datasets for dataset condensation experiments: CIFAR-100 [35], which contains 100 classes, and ImageNet-10 [39], featuring cropped high-resolution images (224x224 pixels). To emulate the temporal correlation present in incoming data streams for these image datasets, we employ the Strength of Temporal Correlation (STC) metric [22]. STC quantifies the number of consecutive data in the input stream that belong to the same class before a class transition occurs; a higher STC value indicates a more consistent class presence over time. We set STC to 500 for CIFAR-100 and 100 for ImageNet-10, based on the settings from previous work [16]. To initialize the simulation, the models are pre-trained using 1% labeled data for iCub1, CORe50, and ImageNet-10, and 10% labeled data for CIFAR-100 due to its greater number of classes.

2) Baselines: Our approach is compared against five baseline methods including heuristics and state-of-the-art methods for on-device learning with limited buffer size (Random [9], FIFO [22], Selective-BP [40], [41], K-Center [42], [43] and GSS-Greedy [10], [44]). *Random* selects a random subset for the new data buffer. *FIFO*, a method widely used in continual learning, replaces the oldest buffer data with new entries. *Selective-BP* selects data with lower prediction confidence as determined by the model for storage in the buffer. *K-Center* selects the centers that minimize the largest distance between a sample and its nearest center. *GSS-Greedy* evaluates the similarity of buffer data, prioritizing the replacement of similar items with distinct new data.

3) Implementation Details: We use ConvNet [45] as the backbone model for all experiments, and employ SGD with momentum as the optimizer. A batch size of 128 is used along with a weight decay of 5×10^{-4}. The iteration number L is set to 10, the filtering threshold m to 0.4, the scalar temperature factor τ in the contrastive loss to 0.07, and the weight factor $alpha$ in the loss function to 0.1. Cosine similarity is used as the distance metric $\mathcal{D}$ for gradient matching. The learning rate is 1e-3 for iCub, CORe50, CIFAR-100, and 1e-4 for ImageNet-10. For all datasets, the training interval β is set to 10, and the model is trained for 200 epochs on the condensed dataset for each update. Following the common practice in dataset condensation, we use and report the Images per Class (IpC) metric to determine the number of synthetic images per class. The buffer size can be derived by multiplying the IpC value by the number of classes. For each experiment, we conduct five trials with different random seeds.

B. Experimental Results

1) Classification Performance: We report the average end accuracy and variance of DECO compared to baselines across varying labeled ratios in Table I. We present the results for different IpC values in the buffer, specifically {1, 5, 10, 50}. Experiment results show that DECO consistently outperforms the baseline methods, showing greater benefits with smaller IpC values. For instance, DECO exceeds the top baseline by 28.5% on iCub1, 56.7% on CORe50, 55.8% on CIFAR-100, and 38.6% on ImageNet-10 with IpC=1. This highlights DECO's capability to effectively utilize new data in scenarios with extremely limited memory resources. Additionally, DECO shows markedly lower variance than baselines, benefiting from its ability to integrate new data without discarding old, thus minimizing uncertainty associated with frequent buffer updates.

2) Training Curve: In this section, we analyze the training efficiency of DECO on the CORe50 and ImageNet-10 datasets with an IpC of 10 by comparing its learning curves against the two most competitive baselines, FIFO and Selective-BP. The learning curves in Fig. 3 illustrate the speed at which our model learns from new data, with the x-axis representing the number of processed inputs and the y-axis showing model accuracy. Our findings indicate that DECO consistently outperforms the baselines. Notably, on both datasets, our method achieves the baselines' final accuracy using just a quarter of the data. By the end of the training period, DECO surpasses the best baseline by over 8% on CORe50 and 6% on ImageNet-10. Additionally, DECO's learning curve is smoother across all datasets compared to those of the baseline methods, demonstrating enhanced robustness throughout the learning process.

TABLE I: Comparison of final average accuracy. "Improvement" indicates the increase over the best baseline. "Upper Bound" indicates the average end accuracy achievable with an unlimited buffer size.

	IpC	Random	FIFO	Selective-BP	K-Center	GSS-Greedy	DECO (Ours)	Improvement	Upper Bound
iCub1	1	29.89±1.33	30.50±1.05	31.22±1.17	31.01±1.09	30.28±1.15	**38.37±0.20**	22.9%↑	62.13%
	5	33.70±1.90	34.41±1.38	34.85±1.19	33.97±1.70	34.27±1.62	**44.79±0.19**	28.5%↑	
	10	39.53±1.97	40.77±1.88	41.20±1.58	40.23±1.66	40.67±0.88	**50.11±0.34**	21.6%↑	
	50	42.69±1.51	44.90±1.73	45.78±1.32	44.50±1.29	44.04±1.81	**55.45±0.28**	21.1%↑	
CORe50	1	16.97±1.90	18.88±0.62	19.05±0.78	17.80±0.69	18.52±1.10	**29.84±0.26**	56.7%↑	88.71%
	5	29.19±1.81	31.90±0.75	32.18±1.22	31.50±0.45	31.56±0.70	**40.38±0.33**	25.5%↑	
	10	38.86±0.80	39.47±1.02	38.25±0.68	38.88±0.73	39.33±0.70	**48.04±0.16**	21.7%↑	
	50	50.36±0.66	52.89±0.82	53.44±0.40	51.76±1.02	52.68±0.83	**78.55±0.11**	46.9%↑	
CIFAR-100	1	10.15±0.32	10.07±0.22	12.01±0.28	12.13±0.23	14.16±0.48	**22.06±0.05**	55.8%↑	46.83%
	5	19.00±0.35	19.50±0.26	18.45±0.32	18.33±0.40	19.72±0.16	**27.23±0.08**	38.1%↑	
	10	20.72±0.38	21.59±0.42	21.71±0.18	20.60±0.21	21.65±0.36	**29.01±0.15**	33.6%↑	
	50	28.51±0.53	31.32±0.33	29.76±0.35	29.55±0.25	29.72±0.20	**32.13±0.12**	2.6%↑	
ImageNet-10	1	16.23±2.16	15.10±1.78	17.03±1.90	17.47±1.35	18.62±1.22	**25.80±0.25**	38.6%↑	39.60%
	5	17.96±3.77	19.22±2.38	20.24±2.91	20.48±1.25	20.34±1.90	**28.29±0.19**	38.1%↑	
	10	20.67±2.10	22.48±0.98	22.99±1.39	22.38±1.30	21.28±1.45	**29.58±0.09**	28.7%↑	
	50	22.55±1.89	23.01±2.21	23.51±0.99	23.35±0.59	22.83±1.20	**31.55±0.20**	34.2%↑	

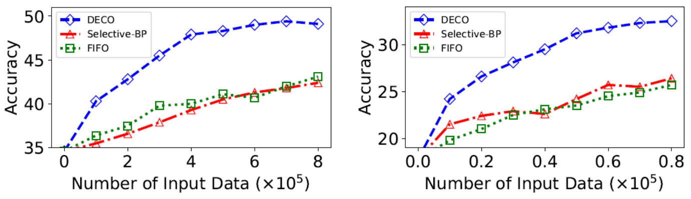


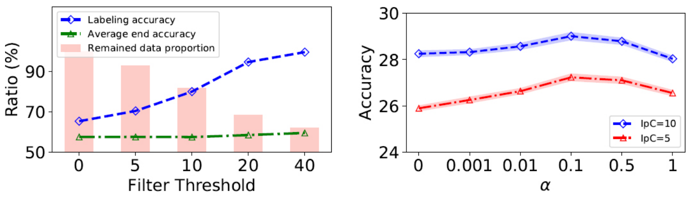
Fig. 3: Learning curves on CORe50 (left) and ImageNet-10 (right) show the average accuracy to the amount of input data.

3) Time Complexity Analysis: Given the superior performance of our method compared to selection-based baselines, we extended our comparison to include other condensation methods DC [12], DSA [27], and DM [13]. We evaluated total execution time and accuracy on the CORe50 dataset, with results presented in Table II. DECO achieves a significant speed improvement of approximately 10x compared to both DC and DSA. While DM is an accelerated condensation method with a speed marginally greater than that of DECO, DECO offers markedly better accuracy. These findings highlight the effectiveness of our speed enhancements in the condensation algorithm, achieved with minimal impact on accuracy.

TABLE II: Comparison of execution time.

Method	IpC=1		IpC=5		IpC=10		IpC=50	
	Time	Acc	Time	Acc	Time	Acc	Time	Acc
DC	437.2	30.1	783.5	40.4	1112.0	47.6	1683.9	77.9
DSA	448.7	28.0	679.2	43.2	1023.5	45.8	1628.1	75.2
DM	30.6	25.1	70.3	34.5	102.5	39.4	423.6	58.5
DECO	50.9	29.8	89.2	40.4	128.3	48.0	519.3	78.6

4) Analysis of Filter Threshold m: Fig. 4a demonstrates the impact of the filter threshold m for majority voting on three key metrics: the percentage of data retained after filtering, the accuracy of the generated pseudo-labels, and the overall model accuracy. The x-axis represents the filter threshold, while the y-axis shows both accuracy and retained data proportion. An increase in the filter threshold leads to a decrease in data



(a) Filter threshold m

(b) Weighting factor α

Fig. 4: (a) Impact of filter thresholds m on pseudo-labeling accuracy and classification performance. (b) Effects of loss weighting factor α on average end accuracy.

retention but enhances the accuracy of the remaining pseudo-labels, demonstrating a trade-off between data quantity and label quality. The highest training accuracy is achieved at a filter threshold of 40, suggesting that label accuracy is more crucial than data volume for effective training.

5) Analysis of Weighting Factor α: Fig. 4b presents a parameter analysis of the feature discrimination loss term, $\alpha = 0.1$, with values range of $\{0, 0.001, 0.01, 0.1, 0.5, 1\}$ on CIFAR-100 for IpC at 5 and 10. Results show that classification accuracy improves as α increases from 0 to 0.1, identifying $\alpha = 0.1$ as the optimal setting. Similar experiments on other datasets consistently confirm $\alpha = 0.1$ as the most effective setting, demonstrating its robustness across various datasets and minimizing the need for extensive parameter tuning.

V. CONCLUSION

This study enhances the learning process in edge-device environments using dataset condensation techniques. A framework DECO is introduced to manage a limited-size data buffer of synthetic data. Initially, pseudo-labels are assigned to streaming data via majority voting. An efficient gradient matching technique then condenses this data by class. Additionally, a feature discrimination objective is incorporated to mitigate labeling noise and enhance data quality. Our results demonstrate significantly improved performance compared to existing methods.

References

[1] J. Xia, T. Liu, Z. Ling, T. Wang, X. Fu, and M. Chen, "Pervasivefl: Pervasive federated learning for heterogeneous iot systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4100–4111, 2022.

[2] S. Samaras, E. Diamantidou, D. Ataloglou, N. Sakellariou, A. Vafeiadis, V. Magoulianitis, A. Lalas, A. Dimou, D. Zarpalas, K. Votis *et al.*, "Deep learning on multi sensor data for counter uav applications—a systematic review," *Sensors*, vol. 19, no. 22, p. 4837, 2019.

[3] J. Shabbir and T. Anwer, "A survey of deep learning techniques for mobile robot applications," *arXiv preprint arXiv:1803.07608*, 2018.

[4] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[5] R. Kemker, M. McClure, A. Abitino, T. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[6] E. L. Aleixo, J. G. Colonna, M. Cristo, and E. Fernandes, "Catastrophic forgetting in deep learning: A comprehensive taxonomy," *arXiv preprint arXiv:2312.10549*, 2023.

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

[8] S. Aggarwal, K. Binici, and T. Mitra, "Chameleon: Dual memory replay for online continual learning on edge devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[9] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software (TOMS)*, vol. 11, no. 1, pp. 37–57, 1985.

[10] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," *Advances in neural information processing systems*, vol. 32, 2019.

[11] T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset distillation," *arXiv preprint arXiv:1811.10959*, 2018.

[12] B. Zhao, K. R. Mopuri, and H. Bilen, "Dataset condensation with gradient matching," *arXiv preprint arXiv:2006.05929*, 2020.

[13] B. Zhao and H. Bilen, "Dataset condensation with distribution matching," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2023, pp. 6514–6523.

[14] Y. Yang, G. Li, and R. Marculescu, "Efficient on-device training via gradient filtering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 3811–3820.

[15] R. Qin, D. Liu, Z. Yan, Z. Tan, Z. Pan, Z. Jia, M. Jiang, A. Abbasi, J. Xiong, and Y. Shi, "Empirical guidelines for deploying llms onto resource-constrained edge devices," *arXiv preprint arXiv:2406.03777*, 2024.

[16] Y. Wu, Z. Wang, D. Zeng, Y. Shi, and J. Hu, "Enabling on-device self-supervised contrastive learning with selective data contrast. in 2021 58th acm/ieee design automation conference (dac)," 2021.

[17] J. Xia, Y. Zhang, and Y. Shi, "Towards energy-aware federated learning via marl: A dual-selection approach for model and client," *arXiv preprint arXiv:2405.08183*, 2024.

[18] M. Sangermano, A. Carta, A. Cossu, and D. Bacciu, "Sample condensation in online continual learning," in *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2022, pp. 01–08.

[19] R. Qin, Y. Hu, Z. Yan, J. Xiong, A. Abbasi, and Y. Shi, "Fl-nas: Towards fairness of nas for resource constrained devices via large language models," in *2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2024, pp. 429–434.

[20] R. Qin, J. Xia, Z. Jia, M. Jiang, A. Abbasi, P. Zhou, J. Hu, and Y. Shi, "Enabling on-device large language model personalization with self-supervised data selection and synthesis," *arXiv preprint arXiv:2311.12275*, 2023.

[21] R. Qin, Z. Yan, D. Zeng, Z. Jia, D. Liu, J. Liu, Z. Zheng, N. Cao, K. Ni, J. Xiong *et al.*, "Robust implementation of retrieval-augmented generation on edge-based computing-in-memory architectures," *arXiv preprint arXiv:2405.04700*, 2024.

[22] T. L. Hayes, N. D. Cahill, and C. Kanan, "Memory efficient experience replay for streaming learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9769–9776.

[23] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

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

[25] M. De Lange and T. Tuytelaars, "Continual prototype evolution: Learning online from non-stationary data streams," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 8250–8259.

[26] G. Cazenavette, T. Wang, A. Torralba, A. A. Efros, and J.-Y. Zhu, "Dataset distillation by matching training trajectories," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4750–4759.

[27] B. Zhao and H. Bilen, "Dataset condensation with differentiable siamese augmentation," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 674–12 685.

[28] S. Lee, S. Chun, S. Jung, S. Yun, and S. Yoon, "Dataset condensation with contrastive signals," in *International Conference on Machine Learning*. PMLR, 2022, pp. 12 352–12 364.

[29] J.-H. Kim, J. Kim, S. J. Oh, S. Yun, H. Song, J. Jeong, J.-W. Ha, and H. O. Song, "Dataset condensation via efficient synthetic-data parameterization," in *International Conference on Machine Learning*. PMLR, 2022, pp. 11 102–11 118.

[30] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[31] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 687–10 698.

[32] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le, "Rethinking pre-training and self-training," *Advances in neural information processing systems*, vol. 33, pp. 3833–3845, 2020.

[33] W. Jin, X. Tang, H. Jiang, Z. Li, D. Zhang, J. Tang, and B. Yin, "Condensing graphs via one-step gradient matching," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 720–730.

[34] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[35] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[36] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," *Advances in neural information processing systems*, vol. 33, pp. 18 661–18 673, 2020.

[37] S. Fanello, C. Ciliberto, M. Santoro, L. Natale, G. Metta, L. Rosasco, and F. Odone, "icub world: Friendly robots help building good vision data-sets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 700–705.

[38] V. Lomonaco and D. Maltoni, "Core50: a new dataset and benchmark for continuous object recognition," in *Conference on robot learning*. PMLR, 2017, pp. 17–26.

[39] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer, 2020, pp. 776–794.

[40] A. H. Jiang, D. L.-K. Wong, G. Zhou, D. G. Andersen, J. Dean, G. R. Ganger, G. Joshi, M. Kaminksy, M. Kozuch, Z. C. Lipton *et al.*, "Accelerating deep learning by focusing on the biggest losers," *arXiv preprint arXiv:1910.00762*, 2019.

[41] K. Killamsetty, S. Durga, G. Ramakrishnan, A. De, and R. Iyer, "Grad-match: Gradient matching based data subset selection for efficient deep model training," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5464–5474.

[42] X. Lu, L. Liu, L. Nie, X. Chang, and H. Zhang, "Semantic-driven interpretable deep multi-modal hashing for large-scale multimedia retrieval," *IEEE Transactions on Multimedia*, vol. 23, pp. 4541–4554, 2020.

[43] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

[44] D. Ha, M. Kim, and C. Y. Jeong, "Online continual learning in acoustic scene classification: An empirical study," *Sensors*, vol. 23, no. 15, p. 6893, 2023.

[45] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4367–4375.