Enabling On-Device Self-Supervised Contrastive Learning with Selective Data Contrast

Yawen Wu
Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, USA
yawen.wu@pitt.edu

Zhepeng Wang
Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, USA
zhepeng.wang@pitt.edu

Dewen Zeng
Computer Science and Engineering
University of Notre Dame
Notre Dame, USA
dzeng2@nd.edu

Yiyu Shi
Computer Science and Engineering
University of Notre Dame
Notre Dame, USA
yshi4@nd.edu

Jingtong Hu
Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, USA
jthu@pitt.edu

2021 58th ACM/IEEE Design Automation Conference (DAC) | 978-1-6654-3274-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DAC18074.2021.9586228

***Abstract*—After a model is deployed on edge devices, it is desirable for these devices to learn from unlabeled data to continuously improve accuracy. Contrastive learning has demonstrated its great potential in learning from unlabeled data. However, the online input data are usually none independent and identically distributed (non-iid) and edge devices' storages are usually too limited to store enough representative data from different data classes. We propose a framework to automatically select the most representative data from the unlabeled input stream, which only requires a small data buffer for dynamic learning. Experiments show that accuracy and learning speed are greatly improved.**

***Index Terms*—On-Device Learning, Contrastive Learning, Self-Supervised Learning**

I. Introduction

Deep learning models have been widely deployed on the edge and mobile devices to accomplish different tasks, such as robots for search and rescue [1] and UAVs for wildfire surveillance [2]. Traditionally, a model is pre-trained in high-performance servers and then deployed in these devices without further training. However, it is often desirable for these devices to learn from real-world input data (e.g. images captured by a camera) either based on a pre-trained model or totally from scratch when deployed to an unknown [3]. In this way, the model on robots or UAVs can adapt to new environments [4].

While it is feasible to send a few data to servers for labeling, it is prohibitive to send all these new data due to the requirement of expert knowledge, data privacy, communication cost, and latency concerns [5]. Thus, different from conventional training on servers by using fully labeled datasets, it is also desirable to learn from new streaming data in-situ with as few labels as possible.

Contrastive learning, as an effective self-supervised learning approach [6], can learn visual representations from unlabeled data to improve the feature extractor (convolutional layers) in the model. After contrastive learning, the classifier (fully connected layers) can be trained on top of the improved feature extractor by using few labeled data to achieve improved classification performance. Contrastive learning is conventionally conducted by using a large dataset, which is completely collected before the training starts. In the learning process, each mini-batch is randomly sampled from the whole dataset to update the model [7]. On edge platforms such as robots and UAVs, the data are collected by sensors such as cameras and continuously fed into the device. While it is theoretically possible to store the constantly generated massive unlabeled data on the device and employ contrastive learning, both the storage and energy overhead associated with writing and reading these data from storage devices (e.g. Flash memory) can be prohibitive in practice.

To learn from the unlabeled data stream without accumulating a large dataset, a small data buffer can be used to form each mini-batch for training. Existing contrastive learning frameworks [6], [7] assume that each mini-batch is independent and identically distributed (iid) by sampling uniformly at random from all the classes (i.e. each class has representative data in this mini-batch). However, it is challenging to maintain the most representative data in the buffer such that learning from this buffer will efficiently reach an accurate model due to the following two reasons. *First*, the streaming data collected on edge devices are usually temporally correlated [8] and result in a correlation within each mini-batch. This is because a long sequence of data in the temporally correlated stream can be in the same class [9]. For example, in wildlife monitoring, goats from a group can appear in adjacent images captured by a continuous monitoring camera [10] at some time, while zebras can appear in adjacent images at another time. *Second*, there is no easy way to select representative data for each class from the non-iid streaming data due to the fact that the streaming data are *unlabeled*. If labels were available for all the data, we could easily select representative data for each class [9] based on all the labels even if the streaming data is non-iid. Without addressing this challenge, directly learning from these temporally correlated non-iid mini-batches will result in slow learning speed and poor learned representations.

To improve the accuracy and expedite the learning process, it is essential to maintain a data buffer filled with representative data from the streaming data. To achieve this goal, this paper defines a contrast score, which is computed by the similarity between the features of a data and its flipped view. The contrast score of each data measures the quality of feature representation encoded by the model. Based on the contrast score, we propose a data replacement policy to maintain a representative data buffer. Data with a low quality of encoded representation by the model is more valuable for learning since they have not been effectively learned. These data will be maintained in the buffer for further learning. On the other hand, data with a high quality of representations have been effectively learned, and they will be dropped to save places for more valuable data. After contrastive learning effectively learns from the unlabeled data and improve the feature extractor, the classifier needs to be updated as well. Since training the classifier without any labels does not generate meaningful accuracy, we will send as few as 1% of the data to the server for labeling to improve the classifier and overall accuracy.

This work was supported in part by NSF CNS-2007274.

In summary, the main contributions of the paper include:

- **Self-supervised on-device learning framework.** We propose a framework to form mini-batches of training data for self-supervised contrastive learning on-the-fly from the unlabeled input stream. It only uses a small data buffer and eliminates the necessity of storing all the streaming data into the device.
- **Contrast scoring for data selection.** We propose a data replacement policy by contrast scoring to maintain the most representative data in the buffer for on-device contrastive learning. Labels are not needed in the data replacement process, and the selected data will generate large gradients that benefit the learning most.
- **Lazy scoring for reduced computation overhead.** We propose a lazy scoring strategy to reduce the runtime overhead of data scoring. The data scores are updated every several iterations instead of every iteration to save computation.

Experimental results on multiple datasets including CIFAR-10, CIFAR-100, SVHN, ImageNet-20, ImageNet-50, and ImageNet-100 show that the proposed framework achieves significantly higher accuracy than the state-of-the-art (SOTA) techniques and greatly improves the learning speed. With 1% labeled data on the CIFAR-10 dataset, the proposed framework achieves 28.36% higher accuracy than using the 1% labeled data for direct supervised learning. The proposed contrast scoring based data selection achieves 13.9% higher accuracy than the SOTA data selection approach [11]. Meanwhile, the proposed approach achieves 2.67x faster learning than the baseline when the same accuracy is achieved.

II. Background and Related Work

A. Background of Contrastive Learning

Contrastive learning is a self-supervised approach to learn an encoder (feature extractor) for extracting visual representations from the input image. In this work, we employ the contrastive learning approach from [7] since it performs on par with its supervised counterpart. For an input image x, its representation vector h is obtained by $h = f(x)$, where $f(\cdot)$ is the backbone of a deep learning model (i.e. convolutional layers). To boost the performance of learned representation, a project head $g(\cdot)$ is used to map the data representation to the latent space as a vector $z = g(h) = g(f(x))$ where contrastive loss is applied. To create a positive pair (z_i, z_{i^+}), one input x is augmented twice as (x_i, x_{i^+}) and then fed into the encoder to get representation vectors $(h_i, h_{i^+}) = (f(x_i), f(x_{i^+}))$, which are further projected by $g(\cdot)$ and normalized as (z_i, z_{i^+}). Then for each positive pair (z_i, z_{i^+}) in one mini-batch, the contrastive loss is applied to compute the loss ℓ_{i,i^+} as follows:

$$\ell_{i,i^+} = -\log \frac{\exp(z_i \cdot z_{i^+}/\tau)}{\exp(z_i \cdot z_{i^+}/\tau) + \sum_{i^-} \exp(z_i \cdot z_{i^-}/\tau)} \quad (1)$$

where z_{i^-} is the representation vector of other data (serving as negatives to contrast with) in the same mini-batch, and τ is the temperature. Minimizing $\sum \ell_{i,i^+}$ in one mini-batch by iteratively updating the model will learn an encoder to generate representations.

B. Related Work

Contrastive Visual Representation Learning. [6], [7] employ contrastive loss for representation learning and achieve high accuracy on classification and segmentation tasks. [8], [12] use the temporal correlations in the streaming data to improve representation learning. However, all these works assume that the whole training dataset is available in the learning process, and each mini-batch can be formed by sampling from the dataset. Each mini-batch consists of independent and identically distributed (iid) data. But when learning from the streaming data, which cannot be assumed to be iid on edge devices, the data is collected sequentially as it is. Besides, random sampling from the entire input stream to create iid mini-batches is infeasible since it requires storing all the data. Therefore, an approach to form mini-batches on-the-fly while including the most representative data in each mini-batch is needed to enable efficient and accurate on-device contrastive learning.

Data Selection in Streaming and Continual Learning. There are several supervised streaming and continual learning models that can learn from a stream of data [13]. To overcome the problem of catastrophic forgetting of previously seen data, a data buffer is usually needed to store previous data for rehearsal [9], [13], [14]. The main drawback of these approaches is that *data labels* are needed to maintain the buffer. However, labeling all the data in the streaming is prohibitive or even infeasible on edge devices. Therefore, existing methods cannot be applied directly to contrastive learning and an effective data selection approach that works on *unlabeled* data is needed.

III. Self-Supervised On-Device Learning by Selective Data Contrast

This paper proposes a framework to efficiently learn data representations from the unlabeled input stream on-the-fly without accumulating a large dataset due to storage limitations on edge devices. To maintain the most representative data in the buffer such that learning from these data will benefit the model most, we propose a data replacement policy based on *Contrast Score* by measuring the quality of representation for each data without using labels. Data with low quality of representations have not been effectively learned by the model and will be maintained in the buffer for further learning, while data with high quality of representations will be dropped. The contrast scoring is supported by the theoretical analysis that data with higher scores will generate larger gradients and accelerate the learning process.

In this section, we will first present the framework overview in Section III-A. Then we will introduce the proposed contrast scoring for data selection in Section III-B. After that, we will theoretically analyze the effectiveness of contrast scoring in Section III-C. Finally, we will introduce lazy scoring to reduce the runtime overhead of contrast scoring in Section III-D.

A. Framework Overview

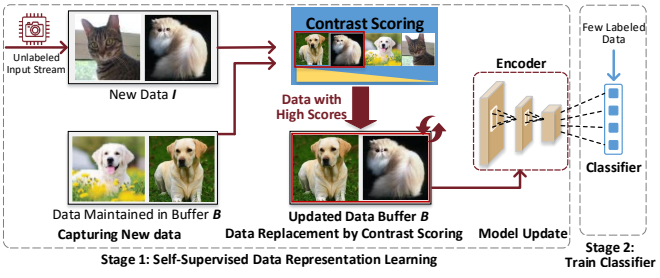


Fig. 1: Overview of on-device contrastive learning framework. The encoder is first trained by contrastive learning with data selected from unlabeled streaming data by contrast scoring, and then the classifier is trained by few (e.g. 1%) labeled data.

As shown in Fig. 1, the proposed framework has two stages. The first stage learns an encoder (i.e. convolutional layers) by self-supervised contrastive learning to generate data representations (i.e. low-dimensional vectors) from the high-dimensional unlabeled inputs (e.g. images). The second stage learns a classifier by using few (e.g. 1%) labeled data on top of the learned representations.

In stage 1, the proposed framework consumes the input streaming data on-the-fly to update the model for improved representation. We only use a small data buffer B (i.e. the same size as one mini-batch) to maintain the most representative data. When a segment of new input I arrives, both the new data in I and the data in the buffer B will be scored to find the most representative data. While any size of I can be used, for simplicity we assume I has the size as B by setting $\text{size}(I) = \text{size}(B)$. Then the data with the highest scores in $B \cup I$ will be selected and put into B. In this way, the data replacement process always maintains the most representative data among the new and the old ones. After each iteration of data replacement, the data preserved in the data buffer B will serve as one mini-batch for updating the model once. The detailed data replacement policy will be described in the next subsection.

B. Data Replacement By Contrast Scoring

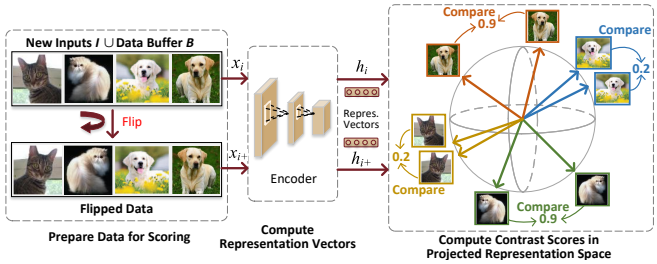


Fig. 2: Contrast scoring for data replacement. The original and flipped inputs are fed into the encoder to generate representation vectors, which are projected to vectors in the unit sphere to compute scores.

Contrast Scoring. For each input x_i, the *contrast scoring* function $S(x_i)$ aims to measure the quality of the representation vector $h_i = f(x_i)$ generated by the base encoder $f(\cdot)$. Intuitively, if the representation of x_i is not good, x_i will be valuable data for updating the base encoder since it can still learn from x_i to improve its capability of encoding x_i. To achieve this, as shown in Fig. 2, for each image x_i from the input stream and the buffer, we generate another view x_{i+} by horizontal flipping. Then we feed both x_i and x_{i+} into the encoder and generate the representation vectors h_i and h_{i+} for these two views. Ideally, if the encoder has learned to generate effective representations of x_i, h_i and h_{i+} will be identical or very similar. After that, based on h_i and h_{i+}, the score for x_i is computed by the contrast scoring function $S(\cdot)$.

The contrast scoring function $S(\cdot)$ is defined as:

$$\begin{aligned} S(x_i) = dissim(x_i, x_{i+}) &= 1 - similarity(z_i, z_{i+}) \\ &= 1 - z_i^T z_{i+}, \quad x_i \in \{B \cup I\} \end{aligned} \tag{2}$$

$$\text{where } z_i = g(h_i)/\|g(h_i)\|_{\ell_2},\ z_{i+} = g(h_{i+})/\|g(h_{i+})\|_{\ell_2} \tag{3}$$

where h_i and h_{i+} are the representation vectors generated by the base encoder $f(\cdot)$ as $h_i = f(x_i)$ and $h_{i+} = f(x_{i+})$, taking data x_i and its horizontally flipped view x_{i+} as inputs, respectively. z_i and z_{i+} are ℓ_2-normalized vectors from the projection head $g(\cdot)$ to enforce $\|z_i\|_{\ell_2} = \|z_{i+}\|_{\ell_2} = 1$. In this way, the dot product $z_i^T z_{i+}$ is in the range [-1,1], and $S(x_i)$ is non-negative and in the range [0,2].

The contrast scoring function Eq.(2) measures the dissimilarity between the projected representation vectors of an image x_i and its horizontal flip x_{i+}, where *a higher score means a larger dissimilarity*. Essentially, the representation of one image needs to be invariant to image transformations [15], and the representations of x_i and x_{i+} need to be as similar as possible. Since a higher score represents a larger dissimilarity and less invariance, input x_i with a higher score is more valuable for updating the base encoder because the base encoder still cannot generate sufficiently good representations of it. By updating the encoder with x_i using the contrastive loss [7], which aims to maximize the similarity of two strongly augmented views of x_i, the score of x_i in Eq.(2) will decrease and x_i will have a lower probability of being selected into the next mini-batch in Eq.(4). In this way, more valuable data to update the base encoder will have a higher probability of being selected into the next mini-batch and others are more likely to be dropped. A detailed analysis of the effectiveness of contrast scoring will be provided in Section III-C.

Contrast Score Design Principle. Contrast scoring is a metric to represent the capability of the base encoder in generating the representation $h_i = f(x_i)$ for x_i. Thus, it should only relate to the image itself and the encoder. In Fig. 2, when generating a pair of inputs (x_i, x_{i+}) to $S(\cdot)$ from an image x_i, we find it crucial to avoid any randomness (e.g. random crop) and *only apply the weak data augmentation* (i.e. horizontal flipping) to generate x_{i+}. The reason is that this weak augmentation is deterministic and provides consistent inputs to $S(\cdot)$. In this way, the score $S(\cdot)$ is deterministic to x_i and is consistent in different runs of $S(\cdot)$.

Contrast Score Based Data Selection. At iteration t, the goal is to form the next mini-batch B_{t+1} by selecting the most informative data from I_t and B_t, such that learning from B_{t+1} will benefit the model most. To achieve this, we apply the contrast scoring function $S(\cdot)$ to both the data already in the buffer B_t and new data I_t. B_{t+1} is formed by selecting the data with the highest contrast scores in $B_t \cup I_t$:

$$B_{t+1} = \{x_i | x_i \in B_t \cup I_t, i \in topN(\{S(x_i)\}_{i=1}^{2N})\} \tag{4}$$

where $topN()$ returns the indices of x_i with the top N scores. In this way, the most representative data is maintained in the buffer by using the proposed contrast scoring.

C. Effectiveness of Contrast Score

The proposed contrast scoring effectively selects data that can generate large gradients, which benefits the learning most. To understand this, for each data x_i in one mini-batch, the gradient of contrastive loss $\ell_{i,i+}$ in Eq.(1) with respect to the representation vector z_i is computed as:

$$\frac{\partial \ell_{i,i+}}{\partial z_i} = -\frac{1}{\tau}\left((1 - p_{z_{i+}}) \cdot z_i - \sum_{z_{i-}} p_{z_{i-}} \cdot z_{i-} \right), \tag{5}$$

$$\text{where } p_z = \frac{\exp\left(z_i{}^T z/\tau\right)}{\sum_{z_j \in \{z_{i+}, z_{i-}\}} \exp\left(z_i{}^T z_j/\tau\right)}, \quad p_z \in \{p_{z_{i+}}, p_{z_{i-}}\} \tag{6}$$

p_z is the probability distribution generated by applying the softmax function to the similarity $z_i{}^T z_j$ between z_i and each $z_j \in \{z_{i+}, z_{i-}\}$ in the mini-batch. For $z = z_{i+}$, $p_{z_{i+}}$ is the matching probability of z_i with its positive pair z_{i+}. Similarly, for $z = z_{i-}$, $p_{z_{i-}}$ is the matching probability of z_i with a negative pair z_{i-} (i.e. the representation vector of other data in the same mini-batch).

A data with a small contrast score $S(x_i)$ generates a near-zero gradient and contributes almost nothing to the learning process. On the other hand, a data x_i with a high contrast score $S(x_i)$ in Eq.(2) corresponds to a large gradient in Eq.(5), which contributes much to the learning process. To understand this, we analyze the relationship between the contrast score in Eq.(2) and the gradient in Eq.(5) in two cases:

Case 1: A data with a small contrast score generates a near-zero gradient. A small contrast score in Eq.(2) corresponds to a large similarity between z_i and z_{i+}. Therefore, the value of dot product $z_i{}^T z_{i+}$ will be large and dominate the elements in the softmax function in Eq.(6). As a result, $p_{z_{i+}}$ will be large and near 1. Since $p_{z_i^+} + \sum_{z \in z_{i-}} p_z = 1$ as a property of the softmax function, the

values of all $p_{z_{i-}}$ will be small and near 0. In this way, $1 - p_{z_{i+}}$ as well as all $p_{z_{i-}}$ will be near 0, and the gradient $\frac{\partial \ell_{i,i+}}{\partial z_i}$ will be near 0. Using the near-zero gradient to perform one gradient descent step $w \leftarrow w - \eta \frac{\partial \ell_{i,i+}}{\partial z_i}$ does not contribute to the learning since there is almost no change in the weight.

Case 2: A data with a high contrast score generates a large gradient. When the contrast score is high, z_i and z_{i+} are dissimilar to each other. By applying the same reasoning as case 1, $1 - p_{z_{i+}}$ and all $p_{z_{i-}}$ will be near 1, and the gradient $\frac{\partial \ell_{i,i+}}{\partial z_i}$ will be large, which significantly contributes to the learning process.

Therefore, by using the proposed contrast score, trivial data that only generate near-zero gradients will be dropped while important data that can generate large gradients will be maintained in the buffer for learning.

D. Lazy Scoring

Computing the scores for new data and data in the buffer requires feeding these data into the base encoder to generate the representations. This computation incurs additional time overhead. To minimize the overhead, we propose lazy scoring, in which part of the data scores can be reused to reduce computation.

We made the following two observations as the foundation of lazy scoring. *First*, during each iteration of data replacement, most of the data (i.e. about 90%) in the buffer are preserved while most of the new data are directly dropped. Therefore, by reusing contrast scores of data in the buffer, a large portion of the computation in scoring can be reduced. *Second*, the score $S(x_i)$ of data x_i only slightly changes across several adjacent iterations. This is because the score of data x_i only depends on itself and the base encoder $f(\cdot)$. x_i remains constant and $f(\cdot)$ is slowly updated across iterations. Therefore, $S(x_i)$ is only slowly updated following the pace of $f(\cdot)$, and the score $S(x_i)$ computed iterations ago still provides meaningful information of x_i.

To achieve lazy scoring, as long as data x_i remains in buffer B, its score is updated every T iterations instead of in every iteration. More specifically, for each x_i in B, we track its age $age(x_i)$ in the number of iterations since it was placed in B. When performing scoring, we separate data in B into two subsets, in which one needs scoring while the other does not. The subset of data that needs scoring is denoted as:

$$B'_t = \{x_i \mid x \in B_t \text{ and } age(x_i) \ mod \ T = 0\} \tag{7}$$

When scoring data in B, the scores are updated as:

$$S_t(x_i) = \begin{cases} dissim(x_i, x_{i+}), & x_i \in B'_t \\ S_{t-1}(x_i), & \text{otherwise} \end{cases} \tag{8}$$

In the above equation, if x_i needs scoring, its score is computed by Eq.(2). Otherwise, its score in the last iteration is copied to save computation. By lazying scoring, the computation overhead of contrast scoring is effectively reduced to about $\frac{1}{T}$ of that without lazy scoring, while the accuracy is preserved.

IV. Experiments

In this section, we first evaluate the *accuracy* with different labeling ratios. Then, we evaluate the *learning speed* of the proposed framework. After that, we evaluate the *reduced computation overhead* by lazy scoring. Finally, we evaluate the impact of *buffer size*.

A. Experimental Setup

Datasets and Evaluation Protocols. We use multiple datasets, including CIFAR-10, CIFAR-100 [16], SVHN [17], ImageNet-20/50/100 [18] to evaluate the proposed approaches. To perform classification, the encoder is first trained by the proposed approaches to generate data representations. As we mentioned before, training a classifier without any labels does not generate meaningful accuracy. Therefore, we train a classifier with 1%, 10%, or 100% labeled data on the learned encoder.

Strength of Temporal Correlation (STC). We use the metric Strength of Temporal Correlation (STC) to represent the temporal correlation of the input stream. STC represents how many consecutive data in the input stream are from the same class until a class change happens [9]. A larger STC represents a stronger temporal correlation.

Default Training Setting. We use ResNet-18 as the base encoder. We train the encoder with the contrastive loss [7] by the Adam optimizer. While the proposed approaches can be applied to both training *from scratch* and *fine-tuning* a pre-trained model, to avoid any bias in the pre-trained model on any approach to compare with, we train *from scratch*. Unless otherwise specified, the batch size is 256 with the weight decay 0.0001. For subsets of ImageNet, the learning rate is 0.0004, the temperature τ is 0.07, and STC is 100. The model is trained for 300 epochs for ImageNet-20/50 and 100 epochs for ImageNet-100. For CIFAR-10, CIFAR-100, and SVHN, the learning rate is 0.0001, the temperature τ is 0.5, and the model is trained for 500 epochs with STC 500. For all datasets, the classifier is trained for 500 epochs with Adam optimizer and learning rate 0.0003. The lazy scoring is disabled by default to have a fair comparison of different data replacement approaches. The results are averaged over three runs on 2 Nvidia V100 GPUs with different random seeds.

Baselines. We first compare the proposed framework with supervised learning using 1% or 10% labeled data. Then, we compare the proposed contrast scoring with four data selection baselines which select data from *unlabeled streaming*. The first two baselines are popular and effective strategies for maintaining exemplars in continual learning while not requiring labels. *Random replacement* is a variant of reservoir sampling [19] and is recently used for continual learning [9]. It selects data uniformly at random from new data and data in the buffer to form the new data buffer. *FIFO replacement* is also recently employed for continual learning [9]. It replaces the oldest data in the buffer with new data. While not requiring labeling information and seemingly simple, these two approaches have demonstrated superior performance in maintaining data for continual learning compared with approaches that rely on exact labels [20]. The next two baselines are SOTA approaches to select data for efficient training and improving accuracy. *Selective-Backprop* [11] selects data with the largest losses for training. *K-Center* is a SOTA active learning approach [21], which selects the most representative data by performing k-center clustering in the features space. For conciseness, in the following figures and tables, we use ***Contrast Scoring*** to represent the proposed approaches, and use ***Random Replace***, ***FIFO Replace***, ***Selective-BP***, and ***K-Center*** to represent the baselines.

B. Improved Accuracy with Different Labeling Ratios

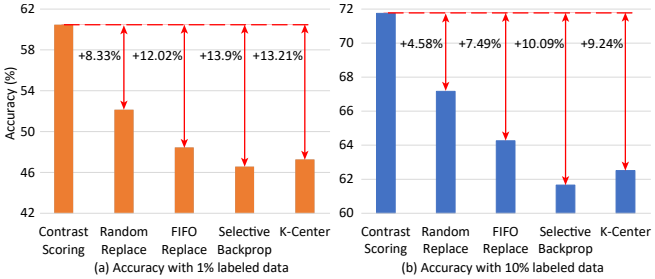


Fig. 3: Accuracy on CIFAR-10 with 1% and 10% labeled data.

We first compare the proposed framework with supervised learning using 1% or 10% labeled data. The supervised learning achieves the

accuracy of 32.11% and 40.53%, which are 28.36% and 31.22% lower than the proposed approaches. Therefore, supervised learning is not a practical option, and we will focus on evaluating the accuracy of the proposed framework with different data selection approaches.

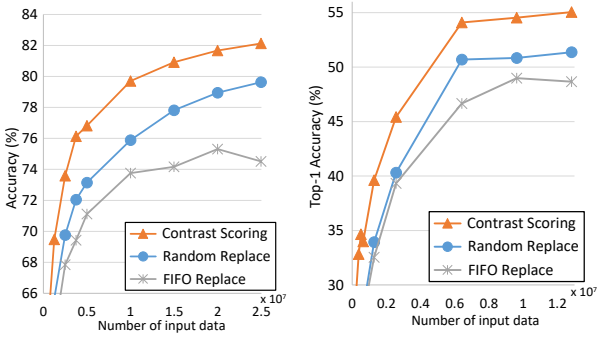
We compare the proposed contrast scoring with other data selection approaches in terms of accuracy by first performing contrastive learning on unlabeled data with different approaches, and then learning the classifier with different ratios of labeled data (i.e. 1%, 10%).

The proposed data selection approach by contrast scoring substantially outperforms the SOTA baselines. The accuracy with different labeling ratios (i.e. 1%, 10%) on CIFAR-10 is shown in Fig. 3, in which the contrastive learning is performed for 100 epochs without labels before training the classifier. First, with 1% and 10% labeled data for learning the classifier, the proposed Contrast Scoring achieves the accuracy of 60.47% and 71.75%, and outperforms other four approaches by {8.33%, 12.02%, 13.9%, 13.21%} and {4.58%, 7.49%, 10.09%, 9.24%}, respectively. Second, with fewer labels (i.e. 1% vs. 10%), the proposed contrast scoring outperforms each baseline by a larger margin. This is because with fewer labels, the quality of learned representation becomes more important, and the proposed framework learns better representations than the baselines. Different from this, the proposed Contrast Scoring selects data that benefit contrastive learning the most.

The results show that the most competitive baselines are the two seemingly simple, yet surprisingly effective approaches *Random Replace* and *FIFO Replace*. These results match the results in [14], where a random replacement policy outperforms elaborately designed approaches.

C. Learning Curve: Improved Learning Speed and Accuracy

We evaluate the learning curve of the proposed approaches and baselines on CIFAR-10, ImageNet-20, ImageNet-50, ImageNet-100, SVHN, and CIFAR-100 datasets. The learning curve represents how fast the model learns representations from the new inputs. Since we aim to evaluate the contrastive learning process by different data selection approaches, to avoid the influence of different label ratios in training the classifier, in the following evaluations, we will use 100% labeled data to train the classifier after contrastive learning and only compare with the two most competitive baselines.

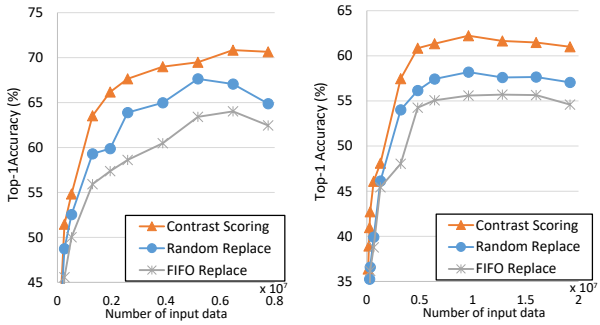


(a) CIFAR-10. (b) ImageNet-100.

Fig. 4: Learning curve on CIFAR-10 and ImageNet-100 datasets. The learned representations by the proposed data replacement with scoring substantially outperform the baselines under two evaluation protocols.

Learning Curve on CIFAR-10. The proposed data replacement policy quickly learns data representations and achieves a significantly faster learning speed and a higher accuracy than the baselines. The learning curve on CIFAR-10 is shown in Fig. 4 (a). The x-axis is the number of seen inputs and the y-axis is the accuracy. The accuracy of the proposed approaches quickly increases to 76.1% with 3.74M seen data, which is 2.67× faster than the random replacement policy that needs 9.98M data to achieve similar accuracy. The FIFO replacement policy cannot achieve this accuracy even with 25M data. Besides, the proposed approaches achieve a much higher final accuracy than the baselines. The proposed approaches achieve a final accuracy of 82.13%, while the random and FIFO replacement policies only achieve 79.63% and 74.51%, respectively.

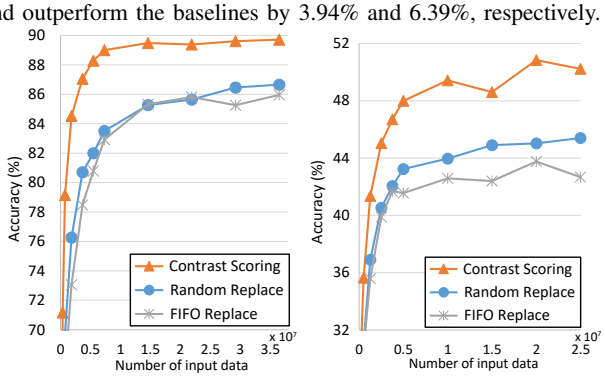
Learning Curve on ImageNet-100. We further evaluate the proposed approaches on the ImageNet-100 dataset. While this dataset is a subset of the large-scale ImageNet dataset, it still features high-resolution images and is challenging for the stream setting. As shown in Fig. 4 (b), the proposed approaches achieve a consistently faster learning speed than the baselines. The proposed approaches achieve 55.05% top-1 accuracy and outperform the baselines by 3.69% and 6.39%, respectively.



(a) ImageNet-20. (b) ImageNet-50.

Fig. 5: Learning curve on ImageNet-20 and ImageNet-50 dataset.

Learning Curve on ImageNet-20 and ImageNet-50. We evaluate the proposed approaches on the ImageNet-20 and ImageNet-50 dataset. As shown in Fig. 5, the proposed approaches achieve a significantly faster learning speed and higher accuracy than the baselines. On ImageNet-20, the proposed approaches achieve 70.64% top-1 accuracy and outperform two baselines by 5.76% and 8.19%, respectively. On ImageNet-50, the proposed approaches achieve 60.99% top-1 accuracy and outperform the baselines by 3.94% and 6.39%, respectively.



(a) SVHN. (b) CIFAR-100.

Fig. 6: Learning curve on SVHN and CIFAR-100 datasets.

Learning Curve on SVHN and CIFAR-100. We evaluate the learning curve on the SVHN and CIFAR-100 datasets, and the results are shown in Fig. 6. The learning curve of the proposed approaches substantially outperforms the baselines.

D. The Impacts of Lazy Scoring

We also evaluate the impact of lazy scoring on the accuracy, runtime overhead, and average percent of re-scored data in the buffer in each training iteration. The model is trained on the CIFAR-10 dataset with buffer size 256 and STC 500.

TABLE I: Top-1 accuracy, average re-scoring percent, and batch time (relative to that without scoring) on CIFAR-10 with different lazy scoring intervals.

Lazy Scoring Interval	Disabled	4	20	50	100	200
Accuracy (%)	76.06	77.04 (+0.98)	77.18 (+1.12)	**77.23** (**+1.17**)	76.38 (+0.32)	74.22 (-1.84)
Re-scoring Pct. (%)	100.0	21.78	4.31	1.71	0.89	0.44
Relative Batch Time	1.478	1.312	1.232	1.199	1.191	1.172

Lazy scoring effectively reduces the additional computation for scoring during training and reduces the batch time. As shown in Table I, when lazy scoring interval T in Eq.(7) increases, the average re-scoring percent and the relative batch time (runtime overhead) are effectively reduced. When lazy scoring is not used, each training step of our method is 47.8% slower than the baselines (without scoring). When lazy scoring is employed with interval 50, each training step is only about 19.9% slower than the baselines. Besides, lazy scoring slightly increases the final accuracy by up to 1.17%. We conjecture that the increased accuracy is because the lazy scoring performs similarly to the momentum encoder in [6]. The score computed multiple iterations ago serves as a momentum score. This slowly updated score brings more information from the past and benefits the data selection.

E. Improved Accuracy With Different Buffer Sizes

We evaluate the impact of buffer size on the performance of the proposed approaches. The model is trained on the CIFAR-10 dataset. The buffer size is in $\{8, 32, 128, 256\}$. The corresponding learning rate is scaled to $\{1, 3, 5, 10\}\times 10^{-5}$, roughly following a learning rate $\propto \sqrt{\text{batch size}}$ scaling scheme.

TABLE II: Accuracy on CIFAR-10 dataset with different buffer sizes.

Buffer Size	Method	Accuracy
8	Contrast Scoring	69.38
	Random Replace	66.71 (-2.67)
	FIFO Replace	65.91 (-3.47)
32	Contrast Scoring	73.26
	Random Replace	70.65 (-2.61)
	FIFO Replace	70.80(-2.46)
128	Contrast Scoring	73.97
	Random Replace	71.28 (-2.69)
	FIFO Replace	70.65 (-3.32)
256	Contrast Scoring	**76.06**
	Random Replace	72.75(**-3.31**)
	FIFO Replace	70.53 (**-5.53**)

The proposed approaches consistently outperform the baselines under different buffer sizes. As shown in Table II, under different buffer sizes, the accuracy by the proposed approaches maintains a clear margin over the baselines. Besides, the margin becomes larger as the buffer size increases. This is because a larger buffer size provides the framework a better opportunity to select more informative data, and the proposed approaches can leverage this opportunity to maintain more representative data in the buffer for learning, while the baselines cannot. Also, all the approaches achieve higher accuracy when the buffer size becomes larger. This is because a larger buffer size provides a larger batch size, and contrastive learning naturally benefits from a large batch size since it provides more negative samples [7].

V. Conclusion

This work aims to enable on-device contrastive learning from input streaming data. We propose a framework to maintain a small data buffer filled with the most representative data for learning. To achieve the data selection without requiring labels, we propose a data replacement policy by contrast scoring. Experimental results on multiple datasets show that the proposed approaches achieve superior learning speed and accuracy compared with SOTA baselines.

References

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

[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] L. Pinto, D. Gandhi, Y. Han, Y.-L. Park, and A. Gupta, "The curious robot: Learning visual representations via physical interactions," in *European Conference on Computer Vision*. Springer, 2016, pp. 3–18.

[4] Q. She, F. Feng, X. Hao, Q. Yang, C. Lan, V. Lomonaco, X. Shi, Z. Wang, Y. Guo, Y. Zhang *et al.*, "Openloris-object: A robotic vision dataset and benchmark for lifelong deep learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4767–4773.

[5] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, H. B. McMahan *et al.*, "Towards federated learning at scale: System design," *arXiv preprint arXiv:1902.01046*, 2019.

[6] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9729–9738.

[7] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv preprint arXiv:2002.05709*, 2020.

[8] A. E. Orhan, V. V. Gupta, and B. M. Lake, "Self-supervised learning through the eyes of a child," *arXiv preprint arXiv:2007.16189*, 2020.

[9] 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.

[10] L. N. Huynh, Y. Lee, and R. K. Balan, "Deepmon: Mobile gpu-based deep learning framework for continuous vision applications," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, 2017, pp. 82–95.

[11] 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.

[12] J. Knights, A. Vanderkop, D. Ward, O. Mackenzie-Ross, and P. Moghadam, "Temporally coherent embeddings for self-supervised video representation learning," *arXiv preprint arXiv:2004.02753*, 2020.

[13] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio, "Gradient based sample selection for online continual learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 816–11 825.

[14] Z. Borsos, M. Mutnỳ, and A. Krause, "Coresets via bilevel optimization for continual learning and streaming," *arXiv preprint arXiv:2006.03875*, 2020.

[15] X. Ji, J. F. Henriques, and A. Vedaldi, "Invariant information clustering for unsupervised image classification and segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9865–9874.

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

[17] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

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

[20] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato, "Continual learning with tiny episodic memories," 2019.

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