ENABLING EFFICIENT ON-DEVICE SELF-SUPERVISED CONTRASTIVE LEARNING BY DATA SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

This work aims to enable efficient on-device contrastive learning from input streaming data after a model is deployed on edge devices such as robots or unmanned aerial vehicles (UAVs) so that they can adapt to a dynamic new environment for higher accuracy. On the other hand, such data usually does not have any labels, calling for unsupervised learning. Most recently, contrastive learning has demonstrated its great potential in learning visual representation from unlabeled data. However, directly applying it to streaming data requires storing a large dataset on-the-fly, which will quickly drain edge devices' storage resources. In this paper, we propose a framework to automatically select the most representative data from unlabeled input stream on-the-fly, which only requires the use of a small data buffer for dynamic learning. What is more, considering the fact that the data are not independent and identically distributed (iid) as in the traditional training process, we score new data as they come in by measuring the quality of their representations without requiring any label information, based on which the data in the buffer will be updated. Extensive experiments show that the learning speed and accuracy are greatly improved compared with approaches without data selection.

1 INTRODUCTION

Deep learning models have been widely deployed on the edge and mobile devices to accomplish different tasks, such as robots (Yang et al., 2016) for search and rescue (Shabbir & Anwer, 2018) and UAVs for wildfire surveillance (Samaras et al., 2019) and counting cars (Ammour et al., 2017). To achieve high accuracy for new inputs, the model needs to adapt to real-world data by learning from the input stream, such as images captured by a camera (Pinto et al., 2016). Using the real-world streaming data, on-device training has the potential to update the model in-situ for higher accuracy. In this way, the model on robots and UAVs can quickly adapt to new environments (She et al., 2020; Sünderhauf et al., 2018). However, it is often inconvenient or even infeasible to send these data to servers for labeling due to both data privacy, communication cost, and latency concerns (Bonawitz et al., 2019). Thus, different from conventional training on servers by using labeled datasets, on-device training has to learn from *unlabeled* streaming data in-situ.

Contrastive learning, as an effective self-supervised learning approach (He et al., 2020), can learn visual representations from unlabeled data to improve the model. 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 (Chen et al., 2020). However, on edge platforms such as robots and UAVs, the data are collected by sensors such as cameras and fed into the device one-by-one. Storing all the data from the never-ending input stream to form a large dataset is prohibitive since it will quickly drain the device storage. For example, a mobile GPU platform NVIDIA Jetson (Nvidia, 2020), which is typically used for edge devices, has 32GB data storage. Accumulating images from a camera into the storage at 30 frame-per-second (FPS) will drain the storage in less than 3 hours. Therefore, it is necessary to use the unlabeled data for training on-the-fly without accumulating a large dataset.

To learn from the unlabeled data stream without accumulating a large dataset, a small data buffer (i.e. the same size as one mini-batch) can be used to form each mini-batch for training. However, there are two challenges in maintaining the most representative data in the buffer such that learning from this buffer will efficiently reach an accurate model. The *first challenge* is that the streaming data

cannot be assumed to be independent and identically distributed (iid). Directly learning from these non-iid data will result in slow learning speed and poor learned representations. The streaming data collected on edge devices are temporally correlated (Orhan et al., 2020) and result in a correlation within each mini-batch. A long sequence of data in the temporally correlated stream can be in the same class (Hayes et al., 2019; Knights et al., 2020). For example, in wildlife monitoring (Dyo et al., 2012), goats from a group can appear in adjacent images captured by a continuous monitoring camera (Huynh et al., 2017) at some time, while zebras can appear in adjacent images at another time. Since typical contrastive learning frameworks (Chen et al., 2020; He et al., 2020) implicitly assume each mini-batch is sampled uniformly at random from the whole dataset, using the temporally correlated mini-batch will compromise this assumption and result in slow learning speed and poor learned representations. The *second challenge* is that the streaming data are *unlabeled* and we cannot employ class information for data selection. If labels were available, we could maintain an exemplar set for each class (Rebuffi et al., 2017; Hayes et al., 2019). Without labels, we need to select data by investigating the data itself, instead of relying on the attached labeling information.

To expedite the learning process and improve the learned representations, it is essential to maintain a data buffer filled with representative data from the streaming data. To this end, we propose a data replacement policy by scoring the newly arrived data and old data without leveraging labeling information. The score of each data measures the quality of feature representation encoded by the model without using label information. 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.

In summary, the main contributions of the paper include:

- We propose a framework to form mini-batches of training data for 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.
- We propose a data replacement policy by data scoring to maintain the most representative data in the buffer for on-device contrastive learning. Labels are not needed in the data replacement process.
- We propose a lazy scoring strategy to reduce the cost of data scoring. The data scores are updated every several iterations instead of in 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 greatly improves the learning curve and achieves higher accuracy, and substantially outperforms heuristic baselines.

2 Background and Related Work

2.1 Background of Contrastive Learning

Contrastive learning is a self-supervised approach to learn an encoder for extracting visual representations from the input image. In this work, we employ the contrastive learning approach from (Chen et al., 2020) since it performs on par with its supervised counterpart. For an input image x_i, its representation vector h_i is obtained by $h_i = f(x_i)$, where $f(\cdot)$ is the backbone of a deep learning model. To boost the performance of learned representation, a project head $g(\cdot)$ is used to map the data representation to the latent space where contrastive loss is applied. To create a positive pair, x_i is augmented twice. Then for each x_i in one mini-batch, the contrastive loss is applied to its two augmentations to compute the loss $\mathcal{L}$. Minimizing $\mathcal{L}$ by iteratively updating the model will learn an encoder to generate representations.

2.2 Related Work

Contrastive Visual Representation Learning. (Wu et al., 2018; Oord et al., 2018; He et al., 2020; Chen et al., 2020) employ contrastive loss for representation learning and achieve high accuracy on classification and segmentation tasks. (Knights et al., 2020; Orhan et al., 2020) use the temporal

correlations in the streaming data to improve the 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 (Aljundi et al., 2019). To overcome the problem of catastrophic forgetting of previously seen data (Kirkpatrick et al., 2017), a data buffer is usually needed to store previous data for rehearsal (Rebuffi et al., 2017; Hayes et al., 2019; Aljundi et al., 2019; Lopez-Paz & Ranzato, 2017; Borsos et al., 2020). 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 inconvenient and 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.

3 METHODS

This framework aims 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 achieve this, the first *challenge* is that the streaming data cannot be assumed to be independent and identically distributed (iid) since the data is sequentially collected on edge devices as it is. Directly learning from the non-iid data will result in slow learning speed and poor learned representations. To tackle this challenge, the proposed framework forms each mini-batch by maintaining a small buffer with data selected from the input stream. The buffer is filled with the most representative data and learning from this buffer will benefit the model most. The buffer has the same size as one mini-batch and data in the buffer serves directly as one mini-batch after each buffer update.

The second *challenge* is that the streaming data are *unlabeled* and we cannot use class information for data selection. To tackle this challenge, we propose a data replacement policy named *Contrast Scoring* 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.

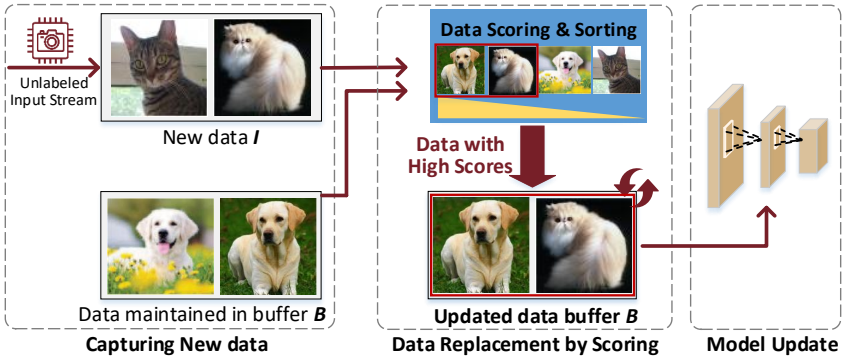


Figure 1: Overview of on-device contrastive learning framework.

3.1 FRAMEWORK OVERVIEW

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. As shown in Figure 1, 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 is the size as B by setting size(I) = size(B). Then the data in buffer B with the lowest scores will be replaced by the new data in I with the highest scores. 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.

3.2 Data Replacement By Contrast Scoring

The goal of data replacement is to select the most informative data from the newly arrived data I and the data in the buffer B, which will be put back to the buffer B to form the next mini-batch. At iteration t, we denote the set of new data I as $\mathcal{I}_t$ and the set of data in buffer B as $\mathcal{B}_t$. The goal is to form $\mathcal{B}_{t+1}$ by selecting data from $\mathcal{I}_t$ and $\mathcal{B}_t$. Formally, we have

$$\mathcal{B}_{t+1} \subset \mathcal{B}_t \cup \mathcal{I}_t, \quad |\mathcal{B}_{t+1}| = |\mathcal{B}_t| = N \tag{1}$$

where N is the buffer size and the batch size. To maximize the improvement of representation by updating the model with $\mathcal{B}_{t+1}$, we design a contrast scoring function $S(\cdot)$ that generates a score for each input x_i in $\mathcal{B}_t \cup \mathcal{I}_t$ to select data for $\mathcal{B}_{t+1}$. By using the contrast scoring function $S(\cdot)$, the next mini-batch is formed as:

$$\mathcal{B}_{t+1} = \{x_i | x_i \in \mathcal{B}_t \cup \mathcal{I}_t, i \in topN(\{S(x_i)\}_{i=1}^{2N})\} \tag{2}$$

where $topN()$ returns the indices of x_i with the top N scores.

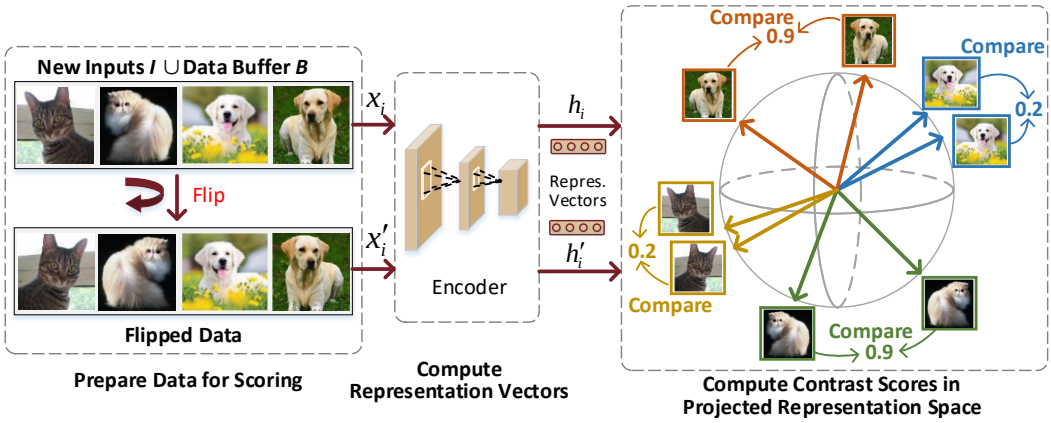


Figure 2: Contrast scoring for data replacement. The original and flipped inputs are fed into the encoder to generate representation vectors, which are projected to the unit sphere to compute scores.

Contrast Scoring for Data Replacement. For each input x_i, The *contrast scoring* function $S(x_i)$ aims to measure the quality of 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 Figure 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:

$$S(x_i) = dissim(x_i, x_i') = 1 - similarity(z_i, z_i') = 1 - z_i^T z_i', \quad x_i \in \mathcal{B}_t \cup \mathcal{I}_t \tag{3}$$

$$z_i = g(h_i)/\|g(h_i)\|_{\ell 2}, \quad z_i' = g(h_i')/\|g(h_i')\|_{\ell 2} \tag{4}$$

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.(3) measures the dissimilarity between the projected representations 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 (Ji et al., 2019). 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 in a sense that the base encoder still cannot generate sufficiently good representations of it. By updating the encoder with x_i using the contrastive loss (Chen et al., 2020), which aims to maximize the similarity of two strongly augmented views of x_i (Khosla et al., 2020), the score of x_i in Eq.(3) will decrease and x_i will have a lower probability of being selected into the next mini-batch in Eq.(2). 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.

Weak Augmentation is Important for Contrast Scoring. 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 Figure 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 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 $S(\cdot)$. If strong data augmentation such as random crop, random color distortion, and random Gaussian blur were used here, the score will change due to the randomness introduced in these augmentation techniques. Involving the randomness into the inputs to the contrast scoring function will generate biased scores, in which the scores will mainly reflect the randomness. In that case, the score will not be an objective evaluation of the encoder's capability.

The data replacement policy by contrast scoring is capable of maintaining a core set of data collected from different time, which increases the data diversity in the buffer. This provides the model with the opportunity of seeing data across a long time span within one mini-batch, instead of always seeing the mini-batches of highly temporally correlated data. Data from different times in the input stream have a higher probability of being in different classes (Knights et al., 2020), and data from different classes provide better negative samples for the contrastive loss (Khosla et al., 2020). In this way, the learning process will be expedited.

3.3 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. Second, the score $S(x_i)$ only slightly changes across several adjacent iterations. This is because the score of one 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 set $\mathcal{B}$, 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 $\mathcal{B}$ into two subsets, in which one needs scoring while the other does not. The subset of data that needs scoring is denoted as:

$$\mathcal{B}_t' = \{x_i \mid x \in \mathcal{B}_t \text{ and } age(x_i) \bmod T = 0\} \tag{5}$$

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

$$S_t(x_i) = \begin{cases} dissim(x_i, x_i'), & x_i \in \mathcal{B}_t' \\ S_{t-1}(x_i), & \text{otherwise} \end{cases} \tag{6}$$

In the above equation, if x_i needs scoring, its score is computed by Eq.(3). Otherwise, its score in the last iteration is copied to save computation.

4 EXPERIMENTS

In this section, we first evaluate the learning curve of our proposed framework. Then, we perform ablation studies on the impact of buffer size, strength of temporal correlation (STC), and lazy scoring interval. Kindly note that learning from unlabeled streaming data is a challenging task (Borsos et al., 2020; Aljundi et al., 2019), and we do not aim to beat the state-of-the-art methods that do not use streaming data. Instead, we aim to achieve fast on-device learning from streaming such that new knowledge can be quickly learned (Hayes et al., 2019).

4.1 EXPERIMENTAL SETUP

Datasets and Evaluation Protocols. We use multiple datasets, including CIFAR-10, CIFAR-100 (Krizhevsky et al., 2009), SVHN (Netzer et al., 2011), ImageNet-20, ImageNet-50, and ImageNet-100 to evaluate the proposed approaches. ImageNet-20, ImageNet-50, and ImageNet-100 (Van Gansbeke et al., 2020) are subsets of the ILSVRC-2012 dataset (Russakovsky et al., 2015) with 20, 50, and 100 randomly sampled classes. Kindly note that learning from streaming is a challenging task and existing works (Aljundi et al., 2019; Lopez-Paz & Ranzato, 2017; Nguyen et al., 2017) are usually evaluated on *labeled small-scale* datasets such as variants of MNIST or CIFAR datasets. Therefore, our evaluation on *unlabeled streaming* from subsets of *large-scale* ImageNet can be considered as a more challenging task. To evaluate the learned representations, we employ two widely used evaluation protocols, including linear evaluation (Khosla et al., 2020; Kolesnikov et al., 2019; Chen et al., 2020; Bachman et al., 2019) and K-nearest neighbor (KNN) classifiers (Caron et al., 2020; Asano et al., 2019; Wu et al., 2018) on frozen features. For linear evaluation, we replace the projection head with a linear classifier. When training, the encoder is kept frozen and the linear layer is trained with standard cross-entropy loss. We report the top-1 accuracy on the held-out test/validation part of the datasets. KNN classifier is also applied to the output of the encoder without projection. We report the top-1 accuracy with 200 nearest neighbors. Training details can be found in Appendix A.1.

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, and a larger STC represents a stronger temporal correlation.

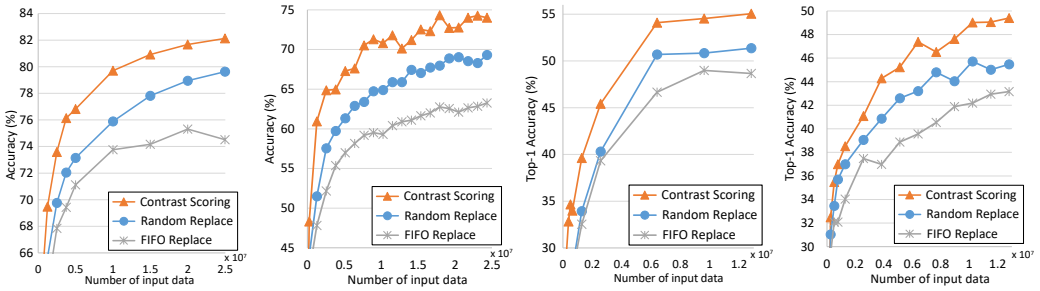
Baselines. Due to the lack of literature on data selection from unlabeled streaming for representation learning, we compare the proposed data replacement policy with two heuristic baselines. *Random replacement* selects data uniformly at random from new data and data in the buffer to form the new data buffer. *FIFO replacement* always replaces the oldest data in the buffer with new data.

For conciseness, in the following figures and tables, we use ***Contrast Scoring*** to represent the proposed approaches, and use ***Random Replace*** and ***FIFO Replace*** to represent the two baselines.

4.2 MAIN RESULTS: LEARNING CURVE

We evaluate the learning curve of the proposed approaches and baselines on CIFAR-10 and ImageNet-100 datasets. Additional results on ImageNet-20, ImageNet-50, SVHN, and CIFAR-100 datasets can be found in Appendix A.2. The learning curve represents how fast the model learns representations from the new inputs. The x-axis is the number of seen inputs and the y-axis is the accuracy following the linear or KNN classification protocol.

Learning Curve on CIFAR-10. The proposed data replacement policy quickly learns data representations and achieves a significantly faster learning curve and a higher accuracy than the baselines. The learning curves on CIFAR-10 with linear and KNN classification protocols are shown in Figure 3 (a) and (b), respectively. Under the linear evaluation, the accuracy of the proposed approaches quickly increases to 76.1% with 3.74M seen data, which is 2.67$\times$ 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. Similar results can be observed under the KNN protocol. Besides, the proposed approaches achieve a much higher final accuracy than the baselines. Under the linear evaluation, 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. Kindly note that while the standard training approach (e.g. random sampling from the whole dataset to form each mini-batch)



(a) Linear classification. (b) KNN classification. (c) Linear classification. (d) KNN classification.

Figure 3: Learning curve on CIFAR-10 (a), (b) and ImageNet-100 (c), (d) datasets. The learned representations by the proposed data replacement with scoring substantially outperform the baselines under two evaluation protocols.

can achieve a higher accuracy of 91.7%, it is infeasible in the challenging streaming task since the data comes in sequentially and the whole dataset is not available for random sampling. Under the KNN protocol, the proposed approaches and the baselines achieve 74.01%, 69.32%, and 63.26%, respectively.

Learning Curve on ImageNet-100. We further evaluate the proposed approaches on the ImageNet-100 dataset. As shown in Figure 3 (c) and (d), the proposed approaches achieve a consistently faster learning curve than the baselines. Under the linear evaluation protocol, the proposed approaches achieve 55.05% top-1 accuracy and outperforms the baselines by 3.69% and 6.39%, respectively. Under the KNN protocol, the proposed approaches achieve 49.40% top-1 accuracy and outperforms the baselines by 3.92% and 6.24%, respectively.

4.3 Ablation Study

Buffer Size. 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 1×10^{-5}, 3×10^{-5}, 5×10^{-5}, and 1×10^{-4}, roughly following a learning rate $\propto \sqrt{\text{batch size}}$ scaling scheme (Krizhevsky, 2014). We train the models for 100 epochs for efficient and fair ablations.

Table 1: Top-1 accuracy on CIFAR-10 dataset with different buffer sizes.

Buffer Size	Method	Linear Eval.	KNN Eval.
8	Contrast Scoring	69.38	53.66
	Random Replace	66.71 (-2.67)	48.08 (-5.58)
	FIFO Replace	65.91 (-3.47)	45.83 (-7.83)
32	Contrast Scoring	73.26	59.47
	Random Replace	70.65 (-2.61)	54.10 (-5.37)
	FIFO Replace	70.80(-2.46)	54.72 (-4.75)
128	Contrast Scoring	73.97	64.59
	Random Replace	71.28 (-2.69)	58.09 (**-6.50**)
	FIFO Replace	70.65 (-3.32)	54.39 (**-10.2**)
256	Contrast Scoring	**76.06**	**66.64**
	Random Replace	72.75(**-3.31**)	61.07 (-5.57)
	FIFO Replace	70.53 (**-5.53**)	57.06 (-9.58)

The proposed approaches consistently outperform the baselines under different buffer sizes. As shown in Table 1, under different buffer sizes, the accuracy by the proposed approaches maintains a clear margin over the baselines. Besides, under the linear evaluation, the margin becomes larger when the buffer size increases to 256. 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 (Chen et al., 2020).

Table 2: Top-1 accuracy on CIFAR-10 dataset with different strengths of temporal correlations.

Methods	Strength of Temporal Correlation (STC) 40	100	400	800	3000	5000
Contrast Scoring	79.64	78.93	76.43	75.74	73.82	71.61
Random Replace	79.36(-0.28)	78.12(-0.81)	74.18(-2.25)	72.35(-3.39)	69.55(**-4.27**)	69.37(-2.24)
FIFO Replace	78.01(-1.63)	74.76(-4.17)	71.45(-4.98)	70.43(**-5.31**)	69.58(-4.24)	69.17(-2.44)

Strength of Temporal Correlation (STC). We evaluate the impact of the strength of temporal correlations (STC) in the input stream. The model is trained on the CIFAR-10 dataset for 100 epochs with a fixed buffer size 256. The top-1 accuracy is reported using the linear evaluation protocol.

With a larger STC in the input stream, the proposed approaches outperform the baselines by a larger margin. As shown in Table 2, when STC increases from 40 to 3000, the performance margin over two baselines increases from 0.28 to 4.27 and from 1.63 to 4.24, respectively. When STC is small, the input stream is similar to the data stream that is randomly sampled from the whole dataset in conventional training. Therefore, the baselines can maintain a data buffer similar to conventional mini-batches and achieve fair performance. When STC becomes large, temporal correlation in the inputs becomes strong, and it is more essential to select data for training instead of naively following the input sequence. With a large STC, the proposed approaches can still maintain the most representative data in the buffer and achieve high accuracy, while the baselines cannot.

Table 3: 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. We evaluate the impact of lazy scoring on the accuracy, average percent of re-scored data in the buffer in each training iteration, and batch time relative to the approach without scoring. The lazy scoring interval is in $\{4, 20, 50, 100, 200\}$ iterations. The model is trained on the CIFAR-10 dataset with buffer size 256 and STC 500. The learned representation is evaluated by the linear evaluation protocol.

Lazy scoring reduces the additional computation for scoring during training and reduces the batch time. As shown in Table 3, when lazy scoring interval T in Eq.(5) increases, the average re-scoring percent and the relative batch time 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 (He et al., 2020). 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.

5 Conclusion

This work aims to achieve efficient 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 scoring each data. Experimental results on multiple datasets show that the proposed approaches achieve superior learning curves and accuracy compared with baselines.

REFERENCES

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, pp. 11816–11825, 2019. 3, 6

Nassim Ammour, Haikel Alhichri, Yakoub Bazi, Bilel Benjdira, Naif Alajlan, and Mansour Zuair. Deep learning approach for car detection in uav imagery. *Remote Sensing*, 9(4):312, 2017. 1

Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv preprint arXiv:1911.05371*, 2019. 6

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*, pp. 15535–15545, 2019. 6

Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046*, 2019. 1

Zalán Borsos, Mojmír Mutnỳ, and Andreas Krause. Coresets via bilevel optimization for continual learning and streaming. *arXiv preprint arXiv:2006.03875*, 2020. 3, 6

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020. 6

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 1, 2, 5, 6, 8, 11

Vladimir Dyo, Stephen A Ellwood, David W Macdonald, Andrew Markham, Niki Trigoni, Ricklef Wohlers, Cecilia Mascolo, Bence Pásztor, Salvatore Scellato, and Kharsim Yousef. Wildsensing: Design and deployment of a sustainable sensor network for wildlife monitoring. *ACM Transactions on Sensor Networks (TOSN)*, 8(4):1–33, 2012. 2

Tyler L Hayes, Nathan D Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9769–9776. IEEE, 2019. 2, 3, 6

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 11

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020. 1, 2, 8

Loc N Huynh, Youngki Lee, and Rajesh Krishna 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*, pp. 82–95, 2017. 2

Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9865–9874, 2019. 5

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020. 5, 6

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017. 3

Joshua Knights, Anthony Vanderkop, Daniel Ward, Olivia Mackenzie-Ross, and Peyman Moghadam. Temporally coherent embeddings for self-supervised video representation learning. *arXiv preprint arXiv:2004.02753*, 2020. 2, 5

Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1920–1929, 2019. 6

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014. 7

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in neural information processing systems*, pp. 6467–6476, 2017. 3, 6

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 6

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017. 6

Nvidia. High performance ai at the edge: Nvidia jetson tx2, 2020. URL https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/. 1

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2

A Emin Orhan, Vaibhav V Gupta, and Brenden M Lake. Self-supervised learning through the eyes of a child. *arXiv preprint arXiv:2007.16189*, 2020. 2

Lerrel Pinto, Dhiraj Gandhi, Yuanfeng Han, Yong-Lae Park, and Abhinav Gupta. The curious robot: Learning visual representations via physical interactions. In *European Conference on Computer Vision*, pp. 3–18. Springer, 2016. 1

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017. 2, 3

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015. 6

Stamatios Samaras, Eleni Diamantidou, Dimitrios Ataloglou, Nikos Sakellariou, Anastasios Vafeiadis, Vasilis Magoulianitis, Antonios Lalas, Anastasios Dimou, Dimitrios Zarpalas, Konstantinos Votis, et al. Deep learning on multi sensor data for counter uav applications—a systematic review. *Sensors*, 19(22):4837, 2019. 1

Jahanzaib Shabbir and Tarique Anwer. A survey of deep learning techniques for mobile robot applications. *arXiv preprint arXiv:1803.07608*, 2018. 1

Qi She, Fan Feng, Xinyue Hao, Qihan Yang, Chuanlin Lan, Vincenzo Lomonaco, Xuesong Shi, Zhengwei Wang, Yao Guo, Yimin 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)*, pp. 4767–4773. IEEE, 2020. 1

Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018. 1

Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *European Conference on Computer Vision (ECCV)*, 2020. 6

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3733–3742, 2018. 2, 6

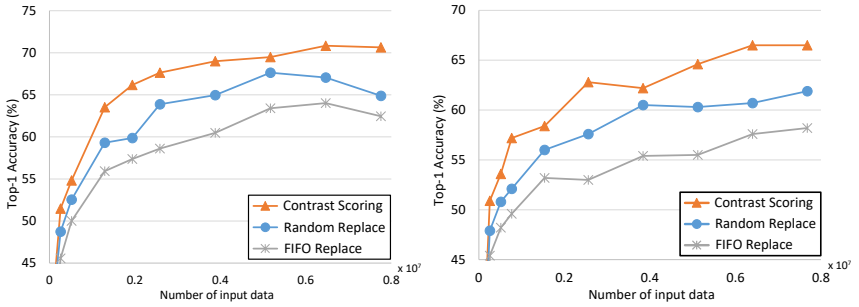
Pin-Chu Yang, Kazuma Sasaki, Kanata Suzuki, Kei Kase, Shigeki Sugano, and Tetsuya Ogata. Repeatable folding task by humanoid robot worker using deep learning. *IEEE Robotics and Automation Letters*, 2(2):397–403, 2016. 1

A Appendix

A.1 Default Training Setting

We use ResNet-18 as the base encoder and a 2-layer MLP projection head to project the representations from the encoder to a 128-dimensional latent space. For subsets of ImageNet, we use ResNet-18 with the standard architecture (He et al., 2016). For datasets CIFAR-10, CIFAR-100, and SVHN with small images, following (Chen et al., 2020), we replace the first 7×7 Conv of stride 2 with 3×3 Conv of stride 1 and remove the first max pooling layer in ResNet-18. For data augmentation, we use random cropping, resizing, horizontal flipping, and color distortions (color jittering and color dropping). We train the encoder from scratch with the SimCLR (Chen et al., 2020) objective and use Adam optimizer. Unless otherwise specified, the batch size is 256 with 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 linear 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 with different random seeds. All experiments are conducted on 2 Nvidia V100 GPUs.

A.2 Learning Curve on Various Datasets

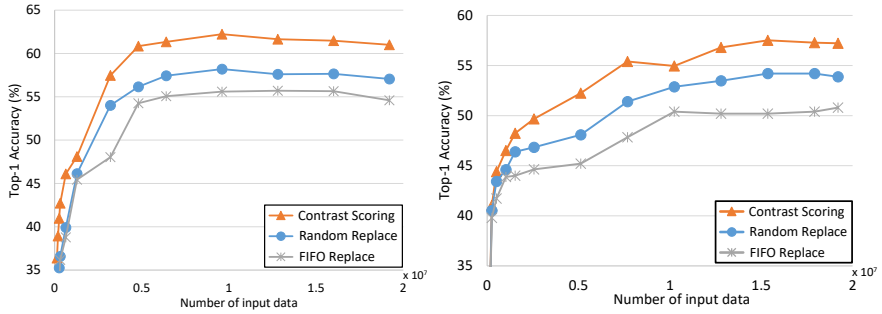


(a) Linear classification. (b) KNN classification.

Figure 4: Learning curve on ImageNet-20 dataset.

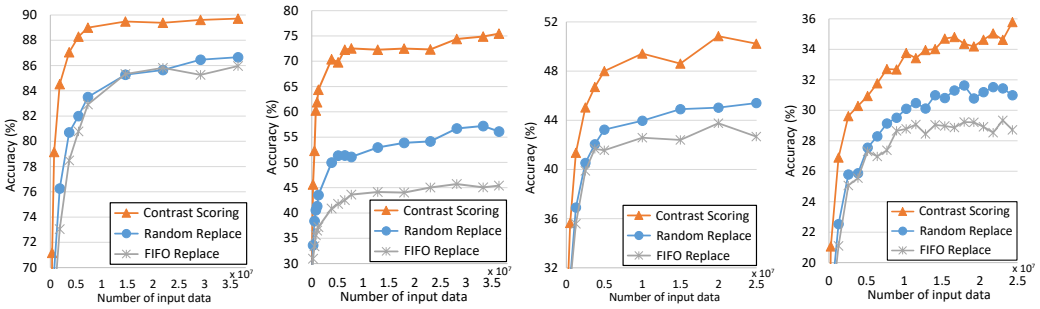
Learning Curve on ImageNet-20. We evaluate the proposed approaches on the ImageNet-20 dataset. While this dataset is a subset of the large-scale ImageNet dataset, it still features high-resolution images and is challenging for the stream setting. As shown in Figure 4, the proposed data selection approach quickly learns representations and substantially outperforms the baselines. Under the linear evaluation protocol, the proposed approaches achieve 70.64% top-1 accuracy and outperforms two baselines by 5.76% and 8.19%, respectively. Similar results are observed under the KNN evaluation protocol, in which the proposed approaches and two baselines achieve 66.50%, 61.90%, and 58.20% top-1 accuracy.

Learning Curve on ImageNet-50. We evaluate the proposed approaches on the ImageNet-50 dataset. As shown in Figure 5, the proposed approaches achieve a significantly faster learning curve than the baselines. Under the linear evaluation protocol, the proposed approaches achieve 60.99% top-1 accuracy and outperforms the baselines by 3.94% and 6.39%, respectively. Under the KNN protocol, the proposed approaches achieve 57.23% top-1 accuracy and outperforms the baselines by 3.35% and 6.43%, respectively.



(a) Linear classification. (b) KNN classification.

Figure 5: Learning curve on ImageNet-50 dataset.



(a) Linear classification. (b) KNN classification. (c) Linear classification. (d) KNN classification.

Figure 6: Learning curve on SVHN (a), (b) and CIFAR-100 (c), (d) 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 Figure 6. The learning curve of the proposed approaches substantially outperform the baselines. For the results on the SVHN dataset shown in Figure 6 (a) and (b), under the linear evaluation protocol, the proposed approaches achieve 89.71% accuracy, while the random and FIFO replacement baselines only achieve 86.66% and 85.96%, respectively. Under the KNN protocol, the proposed approaches and the baselines achieve 75.46%, 56.10%, and 45.41%, respectively. For the CIFAR-100 dataset shown in Figure 6 (c) and (d), under the linear evaluation protocol, the proposed approaches and the baselines achieve 50.22%, 45.40%, and 42.68% top-1 accuracy, respectively. Under the KNN protocol, the proposed approaches and the baselines achieve 35.79%, 30.99% and, 28.71% top-1 accuracy, respectively.